Υπολογιστική Άλγεβρα με το Mathematica

Περιεχόμενα

Κεφάλαιο 1: Βασικοί Αλγόριθμοι της Υπολογιστικής Άλγεβρας

Το μεγάλο πλεονέκτημα της Υπολογιστικής Άλγεβρας είναι πως χρησιμοποιώντας σύμβολα και ακεραίους απείρου ακριβείας μας απάλλαξε από τα σφάλματα των αριθμητικών υπολογισμών και μας έδωσε την δυνατότητα να συγκεντρωνόμαστε στην ανάπτυξη των αλγορίθμων.

Όπως ξέρουμε η Άλγεβρα αρχίζει με αριθμούς και οι υπολογιστές δουλεύουν πάνω σε πληροφορίες που εισάγονται σε αυτόν. Έτσι το πρώτο θέμα που πρέπει να εξετάσουμε είναι πως εισάγωνται οι αριθμοί στον υπολογιστή σαν πληροφορίες.

Οι πληροφορίες αποθηκεύονται στη μνήμη του υπολογιστή, σε "κομμάτια" που λέγονται "λέξεις" (words). Οι σύγχρονοι υπολογιστές χρησημοποιούν λέξεις με 32 ή 64 δυαδικά ψηφία (δ.ψ. ή bits) και στην συζήτηση αυτή θα υποθέτουμε ότι έχουμε υπολογιστή με 64 δ.ψ., εκτός και αν ορίσουμε διαφορετικά. Έτσι, μια λέξη του υπολογιστή περιέχει έναν ακέραιο **απλής ακριβείας** (single precision integer) στο διάστημα $[0, 2^{64}-1]$.

Στην Υπολογιστική Άλγεβρα οι αριθμοί που δεν ανήκουν στο διάστημα $[0, 2^{64}-1]$, (όπου $2^{64} \approx 1.845*10^{19}$), παριστάνονται από ακεραίους **πολλαπλής** (ή **απείρου**) **ακριβείας** (multiprecision or infinite precision integers). Το μόνο μειονέκτημα από την παράσταση αυτή είναι ότι οι πράξεις στους ακεραίους εκτελούνται από το λογισμικό (software) με συνέπεια να είναι σχετικά αργές.

Σε αντίθεση, στην Αριθμητική Ανάλυση οι πραγματικοί αριθμοί προσεγγίζονται από τους αριθμούς κινητής υποδιαστολής (floating-point numbers, Παράρτημα Β) όπου οι αριθμητικές πράξεις γίνονται γρήγορα γιατί εκτελούνται από το hardware και μεγάλη έμφαση δίνεται στον περιορισμό του σφάλματος των υπολογισμών.

Σκοπός του κεφαλαίου αυτού είναι να παρουσιάσει την παράσταση και τους βασικούς αριθμητικούς αλγορίθμους (που μάθαμε στο σχολείο) των ακεραίων πολλαπλής ακριβείας και των πολυωνύμων. Αυτά αποτελουν την βάση της Υπολογιστικής Άλγεβρας. Στα παραρτήματα του κεφαλαίου γίνεται μια

εισαγωγή στην ασυμπτωτική ανάλυση των αλγορίθμων (Παράρτημα Α), *στους αριθμούς κινητής υποδιαστολής που χρησιμοποιούνται στην* Αριθμητική Ανάλυση *για την προσέγγιση των πραγματικών αριθμών* (Παράρτημα Β), *και στις θεμελιώδεις έννοιες της* Άλγεβρας (Παράρτημα Γ).

1.1 Παράσταση και πρόσθεση ακεραίων πολλαπλής ακριβείας

Υπενθυμίζουμε πως μια λέξη μηχανής περιέχει έναν ακέραιο απλής ακριβείας στο διάστημα $[0, 2^{64}-1]$. Οι ακέραιοι που είναι εκτός του διαστήματος $[0, 2^{64}-1]$ παρίστανται από μια λίστα (ή ένα διάνυσμα) από λέξεις 64-δ.ψ., όπου η πρώτη λέξη παρέχει πληροφορίες για το πρόσημο και το μήκος της λίστας. Για να ακριβολογούμε, θεωρούμε την 2^{64}-κή (ως προς την βάση 2^{64}) παράσταση του μη-μηδενικού ακεραίου

$$a = (-1)^s \sum_{0 \le i \le n} a_i \, 2^{64\, i},$$

όπου $s \in \{0, 1\}$, $0 \le n + 1 < 2^{63}$, και τα $a_i \in \{0, ..., 2^{64}-1\}$ για όλα τα i είναι τα ψηφία του a ως προς την βάση 2^{64}. Το ψηφίο a_n ονομάζεται και **κύριο** ή **πιο σημαντικό ψηφίο** (leading or most significant digit). Ο ακέραιος a κωδικοποιείται με την λίστα

$$\{s \cdot 2^{63} + n + 1, a_0, a_1, \ldots, a_n\}$$

από λεξεις 64-δ.ψ., όπου το πιο σημαντικό ψηφίο είναι τελευταίο. Αυτή η διάταξη επιλέγεται γιατί οι περισσότερες αριθμητικές πράξεις αρχίζουν με τα λιγότερο σημαντικά ψηφία των αριθμών. Έτσι, για παράδειγμα, αν το $s = 1$ και $n = 5$ έχουμε έναν αρνητικό αριθμό που παρίσταται από μια λίστα με 6 ψηφία ως προς την βάση 2^{64}. Στην περίπτωση αυτή, αξίζει να δούμε ότι το πρώτο στοιχείο της λίστας, δηλαδή η έκφραση $s \cdot 2^{63} + n + 1$ που παρέχει πληροφορίες για το πρόσημο και το μήκος της λίστας, παρίσταται σε δυαδική μορφή σαν:

```
BaseForm[ s 2^63 + n + 1, 2] /. {s → 1, n → 5}
```

$$10000000000000000000000000000000000000000000000000000000000000110_2$$

Η παράσταση $\{s \cdot 2^{63} + n + 1, a_0, a_1, \ldots, a_n\}$ μπορεί να γίνει μοναδική αν για $a \neq 0$ απαιτήσουμε το κύριο ψηφίο να είναι και αυτό διάφορο του μηδενός, $a_n \neq 0$. Την παράσταση αυτή ονομάζουμε **κανονική παράσταση** (standard representation) του

a. Έτσι η κανονική παράσταση του -1 είναι $\{2^{63} + 1, 1\}$, ενώ για την παράσταση του 0 ορίζουμε την λίστα $\{0\}$, με μοναδικό στοιχείο το 0.

Οι ακέραιοι που μπορούν να παρασταθούν με την κανονική παράσταση σε έναν υπολογιστή με λέξεις 64-δ.ψ. είναι στο διάστημα $[-2^{64\cdot 2^{63}} + 1,\ 2^{64\cdot 2^{63}} - 1]$, όπου ο εκθέτης προκύπτει από την αντικατάσταση του i με το 2^{63} που είναι και η μέγιστη τιμή του—στην έκφραση $a = (-1)^s \sum_{0\le i\le n} a_i\, 2^{64\, i}$. Για να αποθηκευθεί κάθε ένα από τα όρια απαιτεί $2^{63}+1$ λέξεις μνήμης. Ο μεγαλύτερος αριθμός θα γέμιζε περίπου 10 δισεκατομύρια δισκέττες των 8 GB, και έτσι τα όρια αυτά είναι ικανοποιητικά για όλους τους σκοπούς μας.

Με το β-**μήκος** $\lambda_\beta(a)$ ενός μη μηδενικού ακεραίου $a \in \mathbb{Z}$, εννοούμε τον αριθμό των ψηφίων του a στην β-κή του παράσταση (ο συμβολισμός των συνόλων που χρησιμοποιούμε βρίσκεται στο Παράρτημα Γ). Επειδή στην συζήτησή μας $\beta = 2^{64}$, μπορούμε να παραλείπουμε τον δείκτη και να γράφουμε απλά $\lambda(a)$. Αν $\lfloor\cdot\rfloor$ δηλώνει την στρογγύλευση ενός αριθμού προς τον μικρότερο ακέραιο (έτσι ώστε $\lfloor 1.8\rfloor = 1$ και $\lfloor -1.8\rfloor = -2$), τότε ισχύει

$$\lambda(a) = \lfloor \log_{2^{64}} |a| \rfloor + 1 = \lfloor \tfrac{\log_2 |a|}{64} \rfloor + 1.$$

Έτσι $\lambda(a) + 1 = n + 2$ είναι ο αριθμός των λέξεων που χρησιμοποιούνται στην κανονική παράσταση $\{s\cdot 2^{63} + n + 1,\ a_0,\ a_1,\ \ldots,\ a_n\}$ του a. Με την βοήθεια του συμβολισμού του μεγάλου "O" (big-Oh notation) έχουμε και την απλούστερη έκφραση $O(\log_2|a|)$ που μας δίνει τον αριθμό των λέξεων που χρειάζονται στην παράσταση του a—χωρίς να φαίνεται η σταθερά $\frac{1}{64}$.

Ας δούμε τωρα πως θα προσθέσουμε δύο ακεραίους πολλαπλής ακρίβειας $a = (-1)^s \sum_{0\le i\le n} a_i\, 2^{64\, i}$ και $b = (-1)^s \sum_{0\le i\le m} b_i\, 2^{64\, i}$, με $s \in \{0, 1\}$. Προσέξτε ότι τα ανεστραμμένα ψηφία a_i και b_i θεωρούνται ακέραιοι απλής ακριβείας.

Υποθέτουμε πως ο υπολογιστής με τον οποίο εργαζόμαστε έχει εντολή για την πρόσθεση δύο ακεραίων a_i, b_i απλής ακριβείας. Το αποτέλεσμα αυτής της εντολής είναι μια λέξη c_i με 64-δ.ψ. και η σημαία του κρατουμένου (carry flag) $\gamma \in \{0, 1\}$, ένα ειδικό δ.ψ. στην λέξη κατάστασης του επεξεργαστή (status word) που υποδεικνύει αν το αποτέλεσμα υπερβαίνει το 2^{64} ή όχι. Για να μπορούμε να

εκτελούμε πρόσθεση ακεραίων πολλαπλής ακρίβειας, αυτή την σημαία του κρατουμένου την κάνουμε το τρίτο στοιχείο (input) *που δίνεται στην εντολή πρόσθεσης*. Θέτοντας $\gamma_0 = 0$, *έχουμε για κάθε ζεύγος ψηφίων*:

$$a_i + b_i + \gamma_i = \gamma_{i+1} \cdot 2^{64} + c_i,$$

όπου γ_i *είναι το κρατούμενο από το προηγούμενο "στάδιο". Έτσι, το άθροισμα των ακεραίων θα είναι* $c = a + b$, *η παράσταση του οποίου είναι*

$$c = (-1)^s \sum_{0 \le i \le k} (a_i + b_i)\, 2^{64\, i},$$

όπου $k = \max\{n, m\}$, *και αν, για παράδειγμα*, m ≤ n, *τότε θέτουμε* $b_{m+1} = \ldots = b_n = 0$. *Με άλλα λόγια, μπορούμε να υποθέσουμε πως* m = n *και η πρόσθεση γίνεται όπως ακριβώς την μάθαμε στο σχολείο—ξεκινώντας από τα λιγώτερο σημαντικά ψηφία και προχωρώντας προς τα περισσότερο σημαντικά. Όπως και στην σχολική πρόσθεση έτσι και εδώ, αν* $c_i = a_i + b_i \ge 2^{64}$, *τότε το ψηφίο γίνεται* $c_i = c_i - 2^{64}$ *και το κρατούμενο πρέπει να προστεθεί στο άθροισμα του επόμενου ζεύγους ψηφίων. Ακολουθεί ο αλγόριθμος για την πρόσθεση δύο ακεραίων απείρου ακριβείας.*

Αλγόριθμος: Πρόσθεση ακεραίων πολλαπλής ακριβείας ως προς την βάση *β*

Είσοδος: Τα *αντεστραμμένα ψηφία δύο ακεραίων πολλαπλής ακριβείας* $a = (-1)^s \sum_{0 \le i \le n} a_i\, \beta^i$ *και* $b = (-1)^s \sum_{0 \le i \le n} b_i\, \beta^i$, *όχι κατ' ανάγκη κανονικής μορφής, με* $s \in \{0, 1\}$, *και η βάση* β.

Έξοδος: Τα *αντεστραμμένα ψηφία του ακέραιου πολλαπλής ακριβείας* $c = (-1)^s \sum_{0 \le i \le n+1} c_i\, \beta^i$, *έτσι ώστε* $c = a + b$.

==========

1. $\gamma_0 \longleftarrow 0$

2. **for i** = 0, …, *n* **do** {$c_i \longleftarrow a_i + b_i + \gamma_i$; $\gamma_{i+1} \longleftarrow 0$; **if** $c_i \ge \beta$ **then** {$c_i \longleftarrow c_i - \beta$; $\gamma_{i+1} \longleftarrow 1$}}

3. $c_{n+1} \longleftarrow \gamma_{n+1}$; **return** $(-1)^s \sum_{0 \le i \le n+1} c_i\, \beta^i$

==========

Διευκρίνιση: Για να εφαρμοστεί ο αλγόριθμος αυτός καθώς και άλλοι που ακολουθούν στο κεφάλαιο αυτό, χρειάζεται μια προεργασία (preprocessing). Η προεργασία αυτή θα μπορούσε να παρουσιαστεί σαν τμήμα του αλγορίθμου, αλλά επιλέξαμε να μήν ακολουθήσουμε αυτήν την τακτική για να δίνουμε μεγαλύτερη έμφαση στην κυρίως διαδικασία. Η προεργασία παρουσιάζεται στα παραδείγματα.

Ακολουθεί ο αλγόριθμος αυτός προγραμματισμένος στο *Mathematica.* Οι μεταβλητές *aR* και *bR* είναι οι ανεστραμμένες (Reverse) λίστες παράστασης των ακεραίων *a* και *b* που θέλουμε να προσθέσουμε. (Οι πληροφορίες για το πρόσημο και το μήκος της λίστας που βρίσκονται στην πρώτη λέξη της παράστασης των *a* και *b* επεξεργάζονται εκτός του αλγορίθμου.) Η βάση β ορίζεται με προκαθορισμένη τιμή (default value) 2^{64} ώστε να είναι προαιρετική η γραφή της όταν χρησιμοποιούμε αυτήν την τιμή.

```
addLongIntegers[aR_List, bR_List, beta_: 2^64] :=
 Module[{carry = 0, a = aR, b = bR, listR = {}, temp},
  For[i = 1, i < Length[a] + 1, i++, temp = a[[i]] + b[[i]] + carry;
   carry = 0; If[temp < beta, (*then*) listR = Append[listR, temp],
    (*else*) listR = Append[listR, temp - beta]; carry = 1]];
  listR]
```

Έστω λοιπόν ότι θέλουμε να προσθέσουμε τους ακεραίους

```
a = 12345678987654321012345678987654321001 23;
b = 32100123456789876543210123456789876543 21;
```

τους οποίου θεωρούμε πολλαπλής ακρίβειας με βάση $\beta = 2^{64}$. Παίρνοντας τον λογάριθμο ενός από αυτούς βλέπουμε πως και οι δύο παρίστανται σαν λίστες με 3 ψηφία

```
digitLength[n_Integer] := Floor[Log[2^64, n]] + 1

digitLength[a]

3
```

Υπολογίζουμε μία-μία τις λίστες με τα 3 ψηφία ως προς την βάση 2^{64} για κάθε έναν από τους ακεραίους αυτούς, τις αντιστρέφουμε και, αν χρειασθεί, τις κάνουμε του ιδίου μήκους ($a \geq b$):

```
aR = IntegerDigits[a, 2^64] // Reverse
```

```
{10075424478403138843, 11585827674988634732, 3}
```

```
bR = IntegerDigits[b, 2^64] // Reverse;
bR = Append[bR, Table[0, {digitLength[a] - digitLength[b]}]] // Flatten
```

```
{1696731548016309425, 7994421281136458287, 9}
```

Εισάγουμε τις ανεστραμμένες λίστες (χωρίς την τιμή του β) στην συνάρτηση και παίρνουμε το αποτέλεσμα της πρόσθεσης ανεστραμμένο.

```
cR = addLongIntegers[aR, bR]
```

```
{11772156026419448268, 1133504882415541403, 13}
```

Αντιστρέφουμε την λίστα και παίρνουμε το αποτέλεσμα στην γνωστή του μορφή.

```
c = FromDigits[ Reverse[cR], 2^64]
```

```
444458024444441975555558024444441975444 4
```

Είναι το ίδιο με αυτό που βρίσκει και το Mathematica.

```
c == a + b
```

```
True
```

Το ερώτημα είναι πόσο χρόνο παίρνει η εκτέλεση του αλγορίθμου αυτού. Έστω ότι η εκτέλεση της πρόσθεσης των ακεραίων απλής ακριβείας απαιτει k κύκλους μηχανής (machine cycles). Ο αριθμός αυτός εξαρτάται από τον επεξεργαστή που χρησιμοποιούμε, και ο χρόνος της κεντρικής μονάδας επεξεργασμού (CPU time) εξαρτάται από την ταχύτητα του επεξεργαστή. Έτσι ο χρόνος για να προστεθούν δύο ακέραιοι μήκους το πολύ n είναι kn κύκλοι μηχανής για να γίνουν οι προσθέσεις

απλής ακριβείας συν μερικούς κύκλους για ενέργειες όπως ο χειρισμός των δυαδικών ψηφίων των προσήμων, πρόσβαση στην μνήμη κτλ. Υποθέτουμε πως ο συνολικός χρόνος για τις διάφορες άλλες ενέργειες είναι της ίδιας τάξης μεγέθους με τον χρόνο για τις προσθέσεις ακεραίων απλής ακριβείας.

Επειδή θέλουμε να ανεξαρτοποιηθούμε από τις λεπτομέρειες του εκάστοτε υπολογιστή θα λέμε ότι ο χρόνος για την πρόσθεση των ακεραίων a και b πολλαπλής ακριβείας είναι $O(n)$, όπου $n = \max(\lambda(a), \lambda(b)) + 1$. Αυτό σημαίνει είτε ότι ο χρόνος εξαρτάται από το μήκος του μεγαλύτερου ακεραίου είτε ότι γίνονται περίπου $O(n)$ προσθέσεις απλής ακριβείας.

1.2 Παράσταση και πρόσθεση πολυωνύμων

Έστω ότι R είναι ένας μεταθετικός δακτύλιος (commutative ring), όπως ο $\mathbb{Z}$. Αυτό σημαίνει πως μπορούμε να εκτελέσουμε τις πράξεις της πρόσθεσης, αφαίρεσης και του πολλαπλασιασμού σύμφωνα με τους γνωστούς κανόνες. Αν επίσης μπορούμε να διαιρέσουμε με κάθε μη μηδενικό στοιχείο, όπως στους ρητούς $\mathbb{Q}$, τότε το R λέγεται σώμα.

Το πολυώνυμο $a \in R[x]$, όπου R είναι ένας μεταθετικός δακτύλιος (ring), αποτελείται από μία πεπερασμένη ακολουθία $\{a_0, a_1, \ldots, a_n\}$ στοιχείων του R, τους συντελεστές του a, με $n \in \mathbb{N}$, και γράφεται

$$a = a_0 + a_1 x + \cdots + a_n x^n = \sum_{i=0}^{n} a_i x^i .$$

Εμάς μας ενδιαφέρει η περίπτωση που ο δακτύλιος R είναι $\mathbb{Z}$ ή $\mathbb{Q}$, οπότε οι συντελεστές a_i είναι ακέραιοι πολλαπλής ακριβείας. (Ο κλασματικός αριθμός $r = \frac{n}{d}$ παρίσταται από την λίστα $\{n, d\}$, όπου n και d είναι ακέραιοι πολλαπλής ακρίβειας και πρώτοι μεταξύ τους.)

Αν $a_n \neq 0$, τότε ο **βαθμός** (degree) του πολυωνύμου είναι $n = \deg(a)$, και $a_n = \mathrm{lc}(a)$ είναι ο **κύριος συντελεστής** του (leading coefficient). Αν $\mathrm{lc}(a) = 1$, τότε λέμε πως το πολυώνυμο έχει μοναδιαίο κύριο συντελεστή (**monic** πολυώνυμο). Το πολυώνυμο a παρίσταται από την λίστα (ή διάνυσμα) $\{a_0, a_1, \ldots, a_n\}$ μήκους $n + 1$, το k-στό στοιχείο της οποίας είναι ο συντελεστής a_k. (Όπως ακριβώς και με τους ακεραίους, η λίστα θα μπορούσε να είναι $\{n, a_0, a_1, \ldots, a_n\}$, με πρώτο στοιχείο τον βαθμό του πολυωνύμου.)

Προσέξτε την ομοιότητα των παραστάσεων των πολυωνύμων με τις παραστάσεις των ακεραίων πολλαπλής ακριβείας. Εξ αιτίας της ομοιότητας αυτής, πολλοί αλγόριθμοι για τους ακεραίους εφαρμόζονται, με μικρές τροποποιήσεις, και στα πολυώνυμα. Ο αλγόριθμος για την πρόσθεση ακεραίων, για παράδειγμα, διαφέρει από τον αλγόριθμο για την πρόσθεση πολυωνύμων μόνον στο ότι δεν έχουμε

κρατούμενο. Στους ακεραίους το κρατούμενο, κατά κάποιον τρόπο, κρατάει τα πιο σημαντικά ψηφία "δέσμια" των λιγότερο σημαντικών ψηφίων.

Ας θεωρήσουμε λοιπόν τα πολυώνυμα

$$a = \sum_{i=0}^{n} a_i x^i \text{ και } b = \sum_{i=0}^{m} b_i x^i$$

στο $R[x]$ με $\deg(a) \geq \deg(b)$. Χωρίς βλάβη της γενικότητας μπορούμε και εδώ (όπως στους ακεραίους) να θεωρήσουμε πως $n = m$, οπότε το άθροισμά τους είναι

$$c = a + b = \sum_{i=0}^{n} (a_i + b_i) x^i = \sum_{i=0}^{n} c_i x^i .$$

Σημειώστε πως η πρόσθεση $c_i = a_i + b_i$ εκτελείται στο R.

Αλγόριθμος: Πρόσθεση πολυωνύμων

Είσοδος: Οι συντελεστές δύο πολυωνύμων $a = \sum_{i=0}^{n} a_i x^i$ και $b = \sum_{i=0}^{n} b_i x^i$ στον $R[x]$, όπου R είναι δακτύλιος.
Έξοδος: Οι συντελεστές του πολυωνύμου $c = \sum_{i=0}^{n} c_i x^i$ στον $R[x]$, έτσι ώστε $c = a + b$.

==========

1. **for i** = 0, …, n **do** $c_i \longleftarrow a_i + b_i$
2. **return** $c = \sum_{i=0}^{n} c_i x^i$

==========

Προσέξτε ότι το σύμβολο "+" έχει διπλή έννοια εδώ: σημαίνει και πρόσθεση απλής ακριβείας και πρόσθεση πολλαπλής ακριβείας (δηλαδή addLongIntegers[a_i, b_i]). Ακολουθεί ο αλγόριθμος αυτός προγραμματισμένος στο *Mathematica*. Οι μεταβλητές aR και bR είναι οι ανεστραμμένες (Reverse) λίστες των συντελεστών των πολυωνύμων a και b που θέλουμε να προσθέσουμε.

```
addPolynomials[aR_List, bR_List] :=
 Module[{a = aR, b = bR, listR = {}},
  For[i = 1, i < Length[a] + 1, i++,
   listR = Append[listR, (a[[i]] + b[[i]])]];
  listR]
```

Έστω λοιπόν ότι θέλουμε να προσθέσουμε τα πολυώνυμα

```
a = x^3 - 7 x + 7;
b = 3 x^2 - 7;
```

Υπολογίζουμε την αντεστραμμένη λίστα των συντελεστών για κάθε ένα από αυτά τα πολυώνυμα, και κατόπιν (αν χρειάζεται) κάνουμε τις λίστες του ιδίου μήκους:

```
aR = CoefficientList[a, x]
```

```
{7, -7, 0, 1}
```

```
bR = CoefficientList[b, x];
bR = Append[bR, Table[0, {Exponent[a, x] - Exponent[b, x]}]] // Flatten
```

```
{-7, 0, 3, 0}
```

Εισάγουμε τις ανεστραμμένες λίστες των συντελεστών στην συνάρτηση και παίρνουμε τους συντελεστές του αθροίσματος ανεστραμμένους.

```
cR = addPolynomials[aR, bR]
```

```
{0, -7, 3, 1}
```

Από τους συντελεστές παίρνουμε το αποτέλεσμα στην γνωστή του μορφή

```
c = Fold[#1 x + #2 &, 0, Reverse[cR]] // Expand
```

```
-7 x + 3 x^2 + x^3
```

Είναι το ίδιο με αυτό που βρίσκει και το Mathematica.

```
c == a + b
```

```
True
```

Παρά το γεγονός ότι στο σχολείο πρώτα μαθαίνουμε για τους ακεραίους, βλέπουμε πως ο αλγόριθμος πρόσθεσης πολυωνύμων είναι απλούστερος από τον

αντίστοιχο αλγόριθμο για τους ακεραίους πολλαπλής ακριβείας. Το ίδιο ισχύει και για άλλους αλγορίθμους, όπως ο πολλαπλασιασμός, διαίρεση κτλ.

Αναλύοντας τον χρόνο υπολογισμού του αλγορίθμου για την πρόσθεση δύο πολυωνύμων του ιδίου βαθμού n βλέπουμε πως χρειάζονται το πολύ $n + 1$ ή $O(n)$ προσθέσεις στον δακτύλιο R. Αν δε $R = \mathbb{Z}$, και B ένα πάνω φράγμα στις απόλυτες τιμές των συντελεστών και των δύο πολυωνύμων, τότε "+" = addLongIntegers[a_i, b_i] και ο χρόνος υπολογισμού του αλγορίθμου γίνεται $O(n \log B)$.

Εν γένει για το πολυώνυμο $a = \sum_{i=0}^{n} a_i x^i$ ορίζουμε την ***μέγιστη*** νορμ (max ή subinfinity norm) ως $||a||_\infty =$
$\max_{0 \le i \le n} \{|a_i|\}$, την ***αθροιστική*** νορμ (sum ή subone norm) ως $||a||_1 = \sum_{i=0}^{n} |a_i|$ και την **Ευκλείδεια** νορμ (Euclidean norm) ως $||a||_2 = (\sum_{i=0}^{n} a_i^2)^{1/2}$. Ισχύουν οι σχέσεις:

$$||a||_\infty \le ||a||_2 \le \sqrt{n}\, ||a||_\infty \text{ και } ||a||_2 \le ||a||_1 \le n||a||_\infty .$$

Με αυτές τις έννοιες το πάνω φράγμα στις απόλυτες τιμές των συντελεστών και των δύο πολυωνύμων ορίζεται σαν $B = \max\{||a||_\infty, ||b||_\infty\}$.

Γενικά για κάθε $q > 0$ έχουμε την ***q-νορμ*** $||\cdot||_q$ στον διανυσματικό χώρο $\mathbb{C}^n$ που ορίζεται ως

$$||a||_q = (\sum_{i=1}^{n} |\, a_i \,|^q)^{1/q}$$

όπου $a = (a_1, a_2, \ldots, a_n) \in \mathbb{C}^n$.

Όλες οι νορμ ικανοποιούν

- $||a||_q \ge 0$, και η ισότης ισχύει μόνον εάν και μόνον εάν $a = 0$,
- $||a+b||_q \le ||a||_q + ||b||_q$, η τριγωνική ανισότητα,
- $||\lambda a||_q = |\lambda| \cdot ||a||_q$.

1.3 Πολλαπλασιασμός

Δοθέντων δύο πολυωνύμων $a = \sum_{i=0}^{n} a_i x^i$ και $b = \sum_{i=0}^{m} b_i x^i$ στον $R[x]$, το γινόμενό τους είναι $c = a \cdot b = \sum_{k=0}^{n+m} c_k x^k$, όπου για $0 \le k \le n+m$ οι συντελεστές είναι

$$c_k = \sum_{\substack{0 \le i \le n \\ 0 \le j \le m \\ i+j=k}} a_i b_j$$

Η έκφραση για τους συντελεστές c_k γίνεται κατανοητή με ένα παράδειγμα

```
a = a0 + a1 x + a2 x^2 + a3 x^3; b = b0 + b1 x + b2 x^2; a b // Expand; Collect[%, x]
```

```
a0 b0 + (a1 b0 + a0 b1) x + (a2 b0 + a1 b1 + a0 b2) x^2 +
 (a3 b0 + a2 b1 + a1 b2) x^3 + (a3 b1 + a2 b2) x^4 + a3 b2 x^5
```

Υπάρχουν διάφοροι τρόποι υπολογισμού των συντελεστών. Ακολουθεί ο αλγόριθμος για τον πολλαπλασιασμό δύο πολυωνύμων που μάθαμε στο σχολείο.

Αλγόριθμος: Πολλαπλασιασμός πολυωνύμων

Είσοδος: Οι συντελεστές δύο πολυωνύμων $a = \sum_{i=0}^{n} a_i x^i$ και $b = \sum_{i=0}^{m} b_i x^i$ στον $R[x]$, όπου R είναι (μεταθετικός) δακτύλιος.
Έξοδος: Το πολυώνυμο $c = a \cdot b$ στον $R[x]$.

==========

1. **for i** = 0, …, n **do** $p_i \longleftarrow a_i x^i \cdot b$
2. **return** $c = \sum_{i=0}^{n} p_i$

==========

Τα πολυώνυμα $a_i x^i \cdot b$ υπολογίζονται με ξεχωριστό πρόγραμμα (το αφίνουμε για άσκηση) που πολλαπλασιάζει ένα πολυώνυμο b με τους ακεραίους a_i απλής ή πολλαπλής ακριβείας. Οι συντελεστές που προκύπτουν μετατοπίζονται κατά i θέσεις. (Η μεταβλητή x δεν χρησιμοποιείται πουθενά. Είναι απλά ένας κατάλληλος τρόπος για να γράφουμε πολυώνυμα.) Έτσι, τα σύμβολα της πρόσθεσης και του πολλαπλασιασμού έχουν διπλή έννοια εδώ: σημαίνουν

πρόσθεση / πολλαπλασιασμό τόσο απλής όσο και πολλαπλής ακριβείας (δηλαδή addLongIntegers[a_i, b_i] / multiplyLongIntegers[a_i, b_i]). Ο πολλαπλασιασμός ακεραίων πολλαπλής ακριβείας ορίζεται παρακάτω.

Προσέξτε ότι ο αλγόριθμος απαιτεί την αποθήκευση των ενδιάμεσων πολυωνύμων p_i. Αυτό είναι ένα μειονέκτημα το οποίο όμως εύκολα παρακάμπτεται όπως φαίνεται στο παρακάτω πρόγραμμα στο Mathematica. Παίρνουμε τους συντελεστές του πρώτου πολυωνύμου a (μεταβλητή aR) ενώ το δεύτερο πολυώνυμο b το αφίνουμε όπως είναι.

```
multiplyPolynomials[aR_List, b_] := Module[{a = aR, p = 0},
  For[i = 1, i < Length[a] + 1, i++, p = p + a[[i]] x^(i-1) b];
  c = Expand[p]]
```

Έτσι για παράδειγμα έχουμε

```
a = x^3 - 7 x + 7;
b = 3 x^2 - 7;

aR = CoefficientList[a, x]
```

```
{7, -7, 0, 1}
```

```
c = multiplyPolynomials[aR, b]
```

```
-49 + 49 x + 21 x^2 - 28 x^3 + 3 x^5
```

```
c == (a b // Expand)
```

```
True
```

Ας δούμε τώρα τον χρόνο υπολογισμού του αλγορίθμου. Κατ΄αρχάς παρατηρούμε πως κάθε ένας από τους $n + 1$ συντελεστές του a πολλαπλασιάζεται με κάθε έναν από τους $m + 1$ συντελεστές του b. Έτσι ο συνολικός αριθμός των πολλαπλασιασμών είναι

$$(n + 1)(m + 1).$$

Κατόπιν τα γινόμενα αυτά μοιράζονται σε $m + n+1$ αθροίσματα για να σχηματίσουν τους ισάριθμους συντελεστές. Επειδή όταν προσθέτουμε α αριθμούς εκτελούνται α-1 προσθέσεις, έπεται πως ο ολικός αριθμός των προσθέσεων είναι

$$(n + 1)(m + 1) - (n + m + 1) = nm.$$

Έτσι το συνολικό κόστος πολλαπλασιασμού δύο πολυωνύμων είναι $(n + 1)(m + 1) + nm = 2nm + n + m + 1 \le 2nm + 2n + 2m + 2 = 2(n + 1)(m + 1)$ πράξεις στον δακτύλιο R. Μετρώντας μόνον τις πράξεις στον δακτύλιο R βλέπουμε πως δύο πολυώνυμα μπορούν να πολλαπλασιασθούν με $2n^2 + 2n+ 1$ πράξεις ή με $2n^2 + O(n)$ πράξεις ή με $O(n^2)$ πράξεις ή σε **τετραγωνικό χρόνο**.

Αν δε $R = \mathbb{Z}$, και $B = \max\{||a||_\infty, ||b||_\infty\}$, η μέγιστη νορμ, τότε "+" γίνεται addLongIntegers[a_i, b_i], το "·" γίνεται multiplyLongIntegers[a_i, b_i] και ο χρόνος υπολογισμού του γινομένου δύο πολυωνύμων γίνεται $O(n^2 \log^2 B)$, όπου $\log^2 B = (\log B)^2$. Σημειώνουμε πως αργότερα θα συναντήσουμε σχεδόν γραμμικό αλγόριθμο πολλαπλασιασμού δύο πολυωνύμων.

Περνάμε τώρα στους ακεραίους. Το γινόμενο δύο ακεραίων απλής ακριβείας a και b στο διάστημα από 0 μέχρι $2^{64} - 1$ έχει "διπλή" ακρίβεια: δηλαδή βρίσκεται στο διάστημα από 0 μέχρι $(2^{64} - 1)^2 = 2^{168} - 2^{65} + 1$. Υποθέτουμε πως ο επεξεργαστής μας έχει εντολή εκτέλεσης πολλαπλασιασμού που μας δίνει το γινόμενο σε δύο λέξεις c, d των 64-δ.ψ. έτσι ώστε $a \cdot b = d \cdot 2^{64} + c$. Με αυτές τις προϋποθέσεις έχουμε

Αλγόριθμος: Πολλαπλασιασμός ακεραίων πολλαπλής ακριβείας

Είσοδος: Δύο ακέραιοι πολλαπλής ακριβείας $a = (-1)^s \sum_{0 \le i \le n} a_i \beta^i$ και $b = (-1)^t \sum_{0 \le i \le m} b_i \beta^i$, όχι κατ' ανάγκη κανονικής μορφής, με $s, t \in \{0, 1\}$, και η βάση β.
Έξοδος: Τα πολλαπλής ακριβείας γινόμενο $c = a \cdot b$.
==========

1. for $i = 0, \ldots, n$ **do** $p_i \longleftarrow a_i \beta^i \cdot b$

2. return $c = (-1)^{s+t} \sum_{i=0}^{n} p_i$

==========

Για τον υπολογισμό των ακεραίων $a_i \beta^i \cdot b$ χρειάζεται ένα ακόμα ξεχωριστό πρόγραμμα (το αφίνουμε για άσκηση) που πολλαπλασιάζει έναν ακέραιο πολλαπλής ακριβείας b επί τους ακεραίους a_i απλής ακριβείας. Προσέξτε ότι και εδώ ο αλγόριθμος απαιτεί την αποθήκευση των ενδιάμεσων ακεραίων p_i. Αυτό είναι ένα μειονέκτημα το οποίο όμως επίσης εύκολα παρακάμπτεται όπως και στην περίπτωση των πολυωνύμων. Στο παρακάτω πρόγραμμα στο Mathematica παίρνουμε τους αντεστραμμένους συντελεστές του πρώτου ακεραίου a (μεταβλητή aR) ενώ τον δεύτερο ακέραιο b τον αφίνουμε όπως είναι.

```
multiplyLongIntegers[aR_List, b_, beta_: 2^64] :=
 Module[{a = aR, p = 0},
  For[i = 1, i < Length[a] + 1, i++, p = p + a[[i]] beta^(i-1) b];
  c = p]
```

Έτσι για παράδειγμα έχουμε

```
a = 12345678987654321012345678987654321001 23;
b = 32100123456789876543210123456789876543 21;

aR = IntegerDigits[a, 2^64] // Reverse

{10075424478403138843, 11585827674988634732, 3}

c = multiplyLongIntegers[aR, b]

396297819661600366590501447157396738737888882685547631062350555189 7
 512585581483

c == a b

True
```

Ο χρόνος υπολογισμού του γινομένου δύο ακεραίων πολλαπλής ακριβείας προφανώς είναι $O(nm)$, όπου $n = \lambda(a)$ και $m = \lambda(b)$. Σημειώνουμε πως αργότερα θα συναντήσουμε σχεδόν γραμμικό αλγόριθμο πολλαπλασιασμού δύο ακεραίων.

1.4 Διαίρεση με υπόλοιπο

Η διαίρεση είναι λίγο πιο δύσκολη. Όταν διαιρούμε τον ακέραιο a με τον ακέραιο $b \neq 0$ αυτό που πραγματικά ενδιαφερόμαστε είναι να βρούμε δύο ακεραίους, το **πηλίκο** q και το **υπόλοιπο** r, έστι ώστε να ισχύει

$$a = qb + r, \ 0 \leq r < |b|.$$

Αυτή είναι η γνωστή ***διαίρεση με υπόλοιπο*** ή **Ευκλείδεια διαίρεση**, και το σύνολο των ακεραίων $\mathbb{Z}$ λέγεται Ευκλείδεια περιοχή. Συνηθίζεται ο συμβολισμός $q = \text{quo}(a, b)$ και $r = \text{mod}(a, b)$. Σημειώστε πως το q είναι ο μικρότερος ακέραιος που ικανοποιεί την Ευκλείδια σχέση. Στο Mathematica υπάρχουν οι συναρτήσεις (για ακεραίους) Quotient[] και Mod[] που υπολογίζουν το πηλίκο και το υπόλοιπο αντίστοιχα.

Αν πάλι $a, b \in R[x]$, με $b \neq 0$, τότε ενδιαφερόμαστε για $q, r \in R[x]$ έστι ώστε να ισχύει

$$a = qb + r, \ \deg(r) < \deg(b).$$

Επαναλαμβάνουμε πως $a, b \in R[x]$, σημαίνει πως όλοι οι συντελεστές $a_i, b_i \in R$. Έτσι, όταν $R = \mathbb{Z}$, τότε τα πολυώνυμα q και r, δεν υπάρχουν πάντα: για παράδειγμα, δεν μπορούμε να διαιρέσουμε με υπόλοιπο το $x^3 - 7x + 7$ με το $3x^2 - 7$ στο $\mathbb{Z}[x]$. Όπως θα δούμε αργότερα το πρόβλημα λύνεται με ***ψευτο-διαίρεση***. Προς το παρόν θα υποθέσουμε πως ο κύριος συντελεστής $\text{lc}(b)$ του b είναι μια ***μονάδα*** (unit) στο R, έτσι ώστε να έχει αντίστροφο $\alpha \in R$ με $\text{lc}(b)\alpha = 1$ (βλέπε το παράρτημα περί δακτυλίων). Στην περίπτωση που $R = \mathbb{Z}$ οι επιτρεπτές τιμές του κύριου συντελεστή είναι $\text{lc}(b) = \pm 1$, ενώ όταν ο δακτύλιος R είναι ένα σώμα, η διαίρεση με υπόλοιπο είναι δυνατή για οποιδήποτε μη μηδενικό πολυώνυμο. Σημειώστε πως όταν $\text{lc}(b)$ του b είναι μια ***μονάδα*** (unit) στον R, το πηλίκο και το υπόλοιπο υπολογίζονται μονοσήμαντα (αποδείξτε το).

Εφαρμόζοντας στο $\mathbb{Z}[x]$ την σχολική μέθοδο διαίρεσης πολυωνύμων με υπόλοιπο για $a = x^6 - 7x + 7$ και $b = x^2 - 7$ το πηλίκο είναι $q = x^4 + 7x^2 + 49$, και το υπόλοιπο $r = -7x + 350$.

Οι συντελεστές του πηλίκου υπολογίζονται ένας-ένας. Αρχίζοντας με τον κύριο συντελεστή και προχωρώντας προς την σταθερά κάθε ένας συντελεστής του πηλίκου $q = x^4 + 7x^2 + 49$ υπολογίζεται εξισώνοντάς τον με τον αντίστοιχο συντελεστή του "τρέχοντος" υπολοίπου. Το πρώτο "τρέχον" υπόλοιπο είναι το $a = x^6 - 7x + 7$, από το οποίο, στην συνέχεια, αφαιρείται ένα κατάλληλο πολλαπλάσιο του $b = x^2 - 7$, κ.ο.κ. Το τελικό υπόλοιπο είναι $r = -7x + 350$ και για $q \neq 0$ ο βαθμός του πηλίκου είναι $\deg(a) - \deg(b)$.

Αλγόριθμος: Διαίρεση πολυωνύμων με υπόλοιπο

Είσοδος: Τα πολυώνυμα $a = \sum_{i=0}^{n} a_i x^i$ και $b = \sum_{i=0}^{m} b_i x^i$ στον $R[x]$, όπου R είναι (μεταθετικός) δακτύλιος με μονάδα. Ισχύει $n \geq m \geq 0$ και ο κύριος συντελεστής b_m είναι μονάδα.
Έξοδος: Τα πολυώνυμα q και $r \in R[x]$ έτσι ώστε να ισχύει $a = qb + r$ και $\deg(r) < m$.

==========

1. $r \longleftarrow a$
2. **for** $i = n - m, n - m - 1, \ldots, 0$ **do**
   { **if** $\deg(r) = m + i$ **then** { $q_i \longleftarrow \frac{\mathrm{lc}(r)}{b_m}$; $r \longleftarrow r - q_i x^i \cdot b$ }
   else $q_i \longleftarrow 0$ }
3. **return** $q = \sum_{i=0}^{n-m} q_i x^i$ και r

==========

Όπως στον πολλαπλασιασμό των πολυωνύμων έτσι και εδώ τα πολυώνυμα $q_i x^i \cdot b$ υπολογίζονται με ξεχωριστό πρόγραμμα που πολλαπλασιάζει ένα πολυώνυμο b με τους ακεραίους q_i απλής ή πολλαπλής ακριβείας (βλέπε τις ασκήσεις). Οι συντελεστές που προκύπτουν μετατοπίζονται κατά i θέσεις. (Υπενθυμίζουμε πως η μεταβλητή x δεν χρησιμοποιείται πουθενά. Είναι απλά ένας κατάλληλος τρόπος για να γράφουμε πολυώνυμα.) Έτσι, τα σύμβολα της αφαίρεσης και του

πολλαπλασιασμού έχουν διπλή έννοια εδώ: σημαίνουν αφαίρεση / πολλαπλασιασμό τόσο απλής όσο και πολλαπλής ακριβείας.

Στο παρακάτω πρόγραμμα για την διαίρεση πολυωνύμων με υπόλοιπο, οι συντελεστές του πηλίκου q_i συγκεντρώνονται στην λίστα qCoeff από την οποία κατόπιν με την βοήθεια του Fold[] σχηματίζεται το πολυώνυμο.

```
dividePolynomials[a_, b_] := Module[
  {n = Exponent[a, x], m = Exponent[b, x], r = a, qCoeff = {}, i, q},
  i = n - m;
  While[i ≥ 0, If[Exponent[r, x] == m + i,
    (*then*) q = Last[CoefficientList[r, x]];
    qCoeff = Prepend[qCoeff, q]; r = r - q x^i b,
    (*else*)q = 0; qCoeff = Prepend[qCoeff, q]];
   i = i - 1];
  q = Fold[#1 x + #2 &, 0, Reverse[qCoeff]];
  {Expand[q], Expand[r]}]
```

Έτσι για παράδειγμα έχουμε

```
a = x^6 - 7 x + 7;
b = x^2 - 7;

{q, r} = dividePolynomials[a, b]
```

```
{49 + 7 x^2 + x^4, 350 - 7 x}
```

και το αποτέλεσμα είναι το ίδιο με αυτό που υπολογίζουμε με το Mathematica

```
Print["q = ", PolynomialQuotient[a, b, x],
  ";   r = ", PolynomialRemainder[a, b, x]];
```

```
q = 49 + 7 x^2 + x^4;    r = 350 - 7 x
```

Όσον αφορά το κόστος (ή τον χρόνο) μιας εκτέλεσης του βρόγχου "for" (για $\deg(r) = m + i$) αυτό(ς) είναι μια διαίρεση, m πολλαπλασιασμοί και m αφαιρέσεις (προσθέσεις) στον δακτύλιο R. Για το συνολικό κόστος, μετρώντας μόνον τις πράξεις στον δακτύλιο R, βλέπουμε πως δύο πολυώνυμα μπορούν να διαιρεθούν με

$$(2m + 1)(n - m + 1) = (2\deg(b) + 1)(\deg(q) + 1) = O(n^2)$$

πράξεις ή σε τετραγωνικό χρόνο.

Αν δε $R = \mathbb{Z}$, και $B = \max\{||a||_\infty, ||b||_\infty\}$, η μέγιστη νορμ, τότε ο χρόνος διαίρεσης δύο πολυωνύμων με υπόλοιπο γίνεται $O(n^2 \log^2 B)$, όπου $\log^2 B = (\log B)^2$, δηλαδή είναι ίδιος με τον χρόνο πολλαπλασιασμού δύο πολυωνύμων.

Περνάμε τώρα στην διαίρεση με υπόλοιπο ακεραίων που είναι αρκετά δύσκολη να προγραμματισθεί. Η πράξη αυτή διδάσκεται στην τετάρτη Δημοτικού ως εξής: ο μαθητής ή η μαθήτρια πρέπει να διαιρέσει έναν διψήφιο αριθμό με έναν μονοψήφιο για να σχηματίσει ένα **δοκιμαστικό πηλίκο** (trial quotient)—το οποίο είναι τις περισσότερες φορές πολύ μεγάλο. Έτσι, αν για παράδειγμα η μαθήτρια έχει να διαιρέσει το 144 με το 29, πρώτα διαιρεί το 14 με το 2 και σχηματίζει το δοκιμαστικό πηλίκο 7. Αυτό αποδεικνύεται πως είναι μεγάλο και έτσι παίρνει το 6, και μετά το 5, και τέλος το 4 για να γράψει $144 = 4 \cdot 29 + 28$.

Ο αριθμός των δοκιμαστικών πηλίκων μπορει να ελαττωθεί με μια προκαταρκτική ***κανονικοποίηση*** που προκύπτει πολλαπλασιάζοντας τις σχέσεις $a = qb + r$, $0 \le r < |b|$, επί k

$$ka = q(kb) + kr, \ 0 \le kr < |kb|.$$

Σκοπός της κανονικοποίησης είναι να επιτευχθεί η ανισότητα $b_{n-1} \ge \frac{\beta}{2}$ όπου b_{n-1} είναι το πιο σημαντικό ψηφίο του διαιρέτη ως προς την β-κή του παράσταση.

Έχοντας όμως πολλαπλασιάσει τους a και b με το k, το μεν πηλίκο παραμένει αναλλοίωτο το υπόλοιπο όμως πολλαπλασιάζεται επί k. Αυτό σημαίνει πως για να βρούμε το πραγματικό υπόλοιπο με κανονικοποίηση πρέπει να διαιρέσουμε με k αυτό που προκύπτει από την διαίρεση.

Για το παραπάνω παράδειγμα θέλουμε ένα μονοψήφιο k ώστε $29 \cdot k$ να παραμένει διψήφιο αλλά το πρώτο ψηφίο του να είναι τουλάχιστον 5 (με βάση $\beta = 10$). Η επιλογή $k = \lfloor \frac{\beta}{b_{n-1}+1} \rfloor = \lfloor \frac{10}{2+1} \rfloor = 3$ (όπου $\lfloor \cdot \rfloor$ δηλώνει στρογγύλευση προς τον μικρότερο ακέραιο) εκπληρεί τον σκοπό μας διότι

$$29 \cdot 3 = 87 \quad και \quad 144 \cdot 3 = 432.$$

(Η απόδειξη της επιλογής του k αφίνεται για τις ασκήσεις.) Όπως βλέπουμε τώρα το πρώτο δοκιμαστικό πηλίκο είναι $5 = \text{quo}(43, 8)$ που είναι μόνο κατά μία μονάδα μεγαλύτερο από το πραγματικό.

Το γεγονός αυτό δεν είναι τυχαίο και αποδεικνύουμε το εξής

Θεώρημα:
Αν $a = \sum_{0 \le i \le n} a_i \beta^i$ και $b = \sum_{0 \le i \le n-1} b_i \beta^i$, είναι δύο θετικοί ακέραιο που διαφέρουν κατά ένα ψηφίο (οπότε $b \le a < \beta \cdot b$), ισχύει $b_{n-1} \ge \frac{\beta}{2}$ και ορίσουμε το δοκιμαστικό πηλίκο σαν

$$q_t = \beta - 1 \quad αν \quad b_{n-1} = a_n, \quad ή$$

$$q_t = \lfloor \frac{a_n \beta + a_{n-1}}{b_{n-1}} \rfloor \quad αν \quad b_{n-1} > a_n,$$

τότε ικανοποιείται η ανισότητα $q_t - 2 \le q < q_t$, όπου q είναι το πραγματικό πηλίκο. Έχουμε δηλαδή ότι q_t ισούται με q ή με $q + 1$ ή με $q + 2$.

Απόδειξη:
Για να αποδείξουμε τον ισχυρισμό πρέπει να δείξουμε δύο πράγματα: α. πως $q_t \ge q$, και β. πως αν q_t και q δεν είναι ίσα τότε η διαφορά τους είναι το πολύ 2.

α. $q_t \ge q$. Αυτή η ανισότητα αληθεύει όταν $q_t = \beta - 1$, έτσι υποθέτουμε πως $q_t = \lfloor \frac{a_n \beta + a_{n-1}}{b_{n-1}} \rfloor$. Η τιμή αυτή του q_t συνεπάγεται $q_t b_{n-1} \ge a_n \beta + a_{n-1} \ge a_n \beta + a_{n-1} - b_{n-1} + 1$ και έχουμε:

$$a - q_t b \le a - q_t b_{n-1} \beta^{n-1}$$
$$(επειδή \quad q_t b_{n-1} \ge a_n \beta + a_{n-1} - b_{n-1} + 1)$$

$$\leq a_n \beta^n + \cdots + a_0 - a_n \beta^n - a_{n-1} \beta^{n-1} + b_{n-1} \beta^{n-1} - \beta^{n-1}$$
$$= a_{n-2} \beta^{n-2} + \cdots + a_0 - \beta^{n-1} + b_{n-1} \beta^{n-1}$$
$$(\text{και επειδή } a_{n-2} \beta^{n-2} + \cdots + a_0 < \beta^{n-1})$$
$$< b_{n-1} \beta^{n-1} \leq b$$

και $q_t \geq q$ *από το γεγονός ότι* q *είναι ο μικρότερος ακέραιος που ικανοποιεί την ανισότητα* $a - qb < b$ (*όπου εδώ* $b > 0$ *και* $a - qb$ *είναι το υπόλοιπο*).

β. $\boldsymbol{q_t \leq q + 2}$. Ο *ισχυρισμός αυτός αποδεικνύεται με την εις άτοπο επαγωγή. Προσέξτε όμως πως εκτός από την ανισότητα* $a - qb < b$ *που χρησιμοποιήσαμε ισχύει και η*

$$q_t \leq \frac{a}{b} \leq \frac{a}{b_{n-1} \beta^{n-1}} < \frac{a}{b - \beta^{n-1}}.$$

Έστω *λοιπόν ότι* $q_t \geq q + 3$. Τότε *από την* $a - qb < b$ *έχουμε* $q > \frac{a}{b} - 1$ *και συνεπώς*

$$3 \leq q_t - q < \frac{a}{b - \beta^{n-1}} - \frac{a}{b} + 1 = \frac{a}{b}\left(\frac{\beta^{n-1}}{b - \beta^{n-1}}\right) + 1,$$

απ' όπου προκύπτει

$$\frac{a}{b} > 2\frac{(b - \beta^{n-1})}{\beta^{n-1}} \geq 2(b_{n-1} - 1).$$

Εξ *άλλου από τις ανισότητες* $\beta > q_t$ *και* $q_t \geq q + 3$ *έχουμε* $\beta - 4 \geq q_t - 3 \geq q = \lfloor \frac{a}{b} \rfloor \geq 2(b_{n-1} - 1)$ *απ' όπου προκύπτει το άτοπο* $b_{n-1} < \frac{\beta}{2}$. (Μετά *την προκαταρκτική κανονικοποίηση έχουμε* $b_{n-1} \geq \frac{\beta}{2}$.) //

Βλέπουμε *λοιπόν πως η "σχολική" διαίρεση ακεραίων με υπόλοιπο είναι αρκετά πολύπλοκη για να εφαρμοσθεί και για αλγοριθμικές λεπτομέρειες παραπέμπουμε στην βιβλιογραφία* (Flanders *και* Knuth). Αντί *αυτής παρουσιάζουμε μια* ***αναγωγική*** (recursive) *μορφή της διαίρεσης* (*θετικών ακεραίων*) *που βασίζεται στην αφαίρεση.*

Αλγόριθμος: *Διαίρεση θετικών ακεραίων πολλαπλής ακριβείας με υπόλοιπο*

Είσοδος: Δύο *θετικοί ακέραιοι πολλαπλής ακριβείας* $a = \sum_{0 \leq i \leq n} a_i \beta^i$ *και* $b = \sum_{0 \leq i \leq m} b_i \beta^i$, *όχι κατ' ανάγκη κανονικής μορφής, και η βάση* β (*που χρησιμοποιείται*

για την αφαίρεση ακεραίων πολλαπλής ακριβείας).

Έξοδος: Οι πολλαπλής ακριβείας ακέραιοι q και r έτσι ώστε $a = qb + r$, $0 \le r < |b|$.

==========

if $a < b$ **then** { $q \longleftarrow 0$; $r \longleftarrow a$ }

else { $q \longleftarrow 1 + \text{quo}(a - b, b)$; $r \longleftarrow \text{mod}(a - b, b)$ }

==========

Ο αλγόριθμος αυτός υλοποιείται με τις ακόλουθες δύο αναγωγικές συναρτήσεις.

```
myQuotient[a_ /; a ≥ 0, b_ /; b ≥ 0] :=
 Module[{q}, If[a < b, (*then*)q = 0, (*else*)1 + myQuotient[a - b, b]]]

myMod[a_ /; a ≥ 0, b_ /; b ≥ 0] :=
 Module[{r}, If[a < b, (*then*)r = a, (*else*)myMod[a - b, b]]]
```

Έτσι για σχολικό παράδειγμα έχουμε το αποτέλεσμα που περιμέναμε

```
{q, r} = {myQuotient[144, 29], myMod[144, 29]}
```

```
{4, 28}
```

Εύκολα αποδεικνύεται πως η διαίρεση ακεραίων με υπόλοιπο εκτελείται σε χρόνο $O(m(n - m + 1))$, όπου $n = \lambda(a)$, $m = \lambda(b)$ και $n - m + 1 = \lambda(q)$, το μήκος του πηλίκου. Δηλαδή σε όσο χρειάζεται να πολλαπλασιασθεί το b επί το q.

Παράρτημα Α: Ασυμπτωτική ανάλυση αλγορίθμων

Όταν προσπαθούμε να υπολογίσουμε τον χρόνο εκτέλεσης ενός αλγορίθμου μεταβάλλοντας το μέγεθος των παραμέτρων εισόδου δεν είναι δυνατόν να γνωρίζουμε την απάντηση ακριβώς—και για να είμαστε ειλικρινείς δεν μας ενδιαφέρει η απόλυτη ακριβεία. Αυτό που χρειαζόμαστε δεν είναι το ακριβές κόστος του αλγορίθμου, αλλά απλά ο "ρυθμός αύξησής" του για αυξάνουσες τιμές των παραμετρων εισόδου.

Οι συναρτήσεις της μεταβλητής n που συναντάμε στην πράξη έχουν διαφορετικούς "ασυμπτωτικούς ρυθμούς αύξησης" και πάνε στο άπειρο με διαφορετικές ταχύτητες. Αν λοιπόν η $f(n)$ αυξάνεται πιο αργά από την $g(n)$ λέμε ότι η $f(n)$ είναι ***ελεγχόμενη*** (dominated) από την $g(n)$ και γράφουμε

$$f(n) \prec g(n) \iff \lim_{n\to\infty} \tfrac{f(n)}{g(n)} = 0.$$

Η σχέση αυτή είναι μεταβατική: Αν $f(n) \prec g(n)$ και $g(n) \prec h(n)$ τότε $f(n) \prec h(n)$. Μπορούμε επίσης να γράψουμε και $g(n) \succ f(n)$ αν $f(n) \prec g(n)$. Προφανώς, για $a, b \in \mathbb{R}$, ισχύει

$$n^a \prec n^b \iff a < b.$$

Για τις συναρτήσεις που είναι διάφορες από τις πολυωνυμικές έχουμε την εξής ιεραρχία:

$$1 \prec \log\log n \prec \log n \prec n^\epsilon \prec n^c \prec n^{\log n} \prec c^n \prec n^n \prec c^{c^n}.$$

Προσέξτε πως, εκτός από το 1, όλες οι συναρτήσεις πάνε στο άπειρο. Έτσι αν θέλουμε να κατατάξουμε μια συνάρτηση σε αυτήν την ιεραρχία, το ζητούμενο δεν είναι αν πάει στο άπειρο αλλά πόσο γρήγορα πάει εκεί.

Υπάρχει επίσης και η σχέση

$$f(n) \sim g(n) \iff \lim_{n\to\infty} \tfrac{f(n)}{g(n)} = 1,$$

οπότε λέμε πως οι συναρτήσεις $f(n)$ και $g(n)$ είναι **αλληλοελεγχόμενες** (co-dominant) ή ότι η "$f(n)$ είναι **ασυμπτωτική** προς την $g(n)$." Εύκολα αποδεικνύεται πως αν $f(n)$ και $g(n)$ είναι δύο πολυώνυμα του ιδίου βαθμού, τότε αυτά είναι αλληλοελεγχόμενα.

Παρά την χρησιμότητα της έννοιας των ελεγχομένων συναρτήσεων, υπάρχουν μεινεκτήματα. Για παράδειγμα οι λεπτομέρειες δεν απαλοίφονται και το σύμβολό "$<$" δεν μπορεί να χρησιμοποιηθεί στο μέσο μιας έκφρασης.

Έτσι, είκοσι χρόνια αργότερα (1894) αναπτύχθηκε ο συμβολισμός του "μεγάλου O" (big-Oh notation). Λέμε, για παράδειγμα, ότι ο γνωστός αλγόριθμος πολλαπλασιασμού δύο $n \times n$ πινάκων πραγματικών αριθμών είναι "τάξης $O(n^3)$". Αυτό σημαίνει πως ο αριθμός $f(n)$ των πράξεων πολλαπλασιασμού πραγματικών αριθμών που χρειάζεται για να πολλαπλασιασθούν οι δύο πίνακες είναι μικρότερος του $c \cdot n^3$ για όλα τα $n \in \mathbb{N}$, και όπου $c \in \mathbb{R}$ είναι μια σταθερά για την οποία δεν ενδιαφερόμαστε. Αυτή η "προχειρότητα" παίρνει επίσημη μορφή στον ορισμό μεγάλου "O" που ακολουθεί.

Ορισμός: **α:** Μία μερική συνάρτηση $f : \mathbb{N} \to \mathbb{R}$, (δηλαδή μία συνάρτηση που δεν ορίζεται κατ' ανάγκη για όλα τα $n \in \mathbb{N}$) λέγεται **τελικά θετική** (eventually positive) αν υπάρχει μια σταθερά $n_0 \in \mathbb{N}$ τέτοια ώστε η $f(n)$ ορίζεται και είναι γνήσια θετική για όλα τα $n > n_0$.
β: Υποθέτουμε ότι η συνάρτηση $g : \mathbb{N} \to \mathbb{R}$, είναι τελικά θετική. Τότε $O(g)$ είναι το σύνολο των τελικά θετικών συναρτήσεων $f : \mathbb{N} \to \mathbb{R}$ για τις οποίες υπάρχουν σταθερές $n_0 \in \mathbb{N}$ και $c \in \mathbb{R}_{>0}$ έτσι ώστε οι $f(n)$ και $g(n)$ ορίζονται και ισχύει $f(n) \leq c \cdot g(n)$ για όλα τα $n > n_0$.

Η ομορφιά του ορισμού αυτού είναι ότι αγνοεί δευτερεύουσες πληροφορίες και μας επιτρέπει να προσέχουμε μόνο τα εμφανεί χαρακτηριστικά. Για παράδειγμα ξέρουμε πως το άθροισμα των n πρώτων τετραγώνων είναι

$$\sum_{i=1}^{n} i^2 = \tfrac{1}{6}\, n(1+n)\,(1+2\,n)$$

Επειδή για όλους τους ακεραίους $n \in \mathbb{N}$, $\frac{1}{3} n^3 + \frac{1}{2} n^2 + \frac{1}{6} n \leq \frac{1}{3} n^3 + \frac{1}{2} n^2 + \frac{1}{6} n \leq \frac{1}{3} n^3 + \frac{1}{2} n^3 + \frac{1}{6} n^3 = n^3$ το άθροισμα των n πρώτων τετραγώνων γράφεται σαν

$$\sum_{i=1}^{n} i^2 = O(n^3) \text{ ή } \sum_{i=1}^{n} i^2 \in O(n^3)$$

Ο δεύτερος τρόπος γραφής, με το "$\in$", είναι ο πιο σωστός αλλά κάνοντας κατάχρηση του συμβόλου "=" χρησιμοποιούμε τον πρώτο που είναι και ο συνηθέστερος στην βιβλιογραφία. Σε αυτήν την περίπτωση όμως πρέπει να προσέχουμε τις "μονοδρομικές" ιδιότητες του "=". Μπορούμε κάλλιστα να γράφουμε

$$\frac{1}{3} n^3 + \frac{1}{2} n^2 + \frac{1}{6} n = O(n^3)$$

αλλά ποτέ δεν πρέπει να αντιστρέψουμε τα μέλη της εξίσωσης. Αν κάνουμε την αντιστροφή μπορούμε να βγάλουμε εντελώς γελοία συμπεράσματα όπως $n^2 = n^3$ από τις εξισώσεις $n^2 = O(n^3)$ και $n^3 = O(n^3)$.

Συνεχίζοντας με το άθροισμα των n πρώτων τετραγώνων βλέπουμε πως ισχύει και η πιο λεπτομερής σχέση

$$\sum_{i=1}^{n} i^2 = \frac{1}{3} n^3 + O(n^2),$$

όπου το O είναι μέρος μόνον της δεξιάς έκφρασης. Από την άλλη μεριά όμως μπορούμε να αγνοήσουμε πληροφορίες και να γράψουμε

$$\sum_{i=1}^{n} i^2 = O(n^7).$$

Βλέπουμε δηλαδή πως τίποτα στον ορισμό του O δεν μας αναγκάζει να δώσουμε το καλλίτερο δυνατό πάνω όριο. Όμως, σε καμμία περίπτωση δεν μπορούμε να πούμε πως $\sum_{i=1}^{n} i^2 = O(n^2)$. Οι παρακάτω κανόνες έπονται άμεσα από τον ορισμό:

- $f(n) = O(f(n))$,
- $c \cdot O(f(n)) = O(f(n))$ για $c \in \mathbb{R}_{>0}$,
- $O(O(f(n))) = O(f(n))$,

- $O(f(n))O(g(n)) = O(f(n)g(n))$,
- $O(f(n)g(n)) = f(n)O(g(n))$.

Μπορούμε να χρησιμοποιούμε λογαρίθμους στον O-συμβολισμό χωρίς αναφορά στην βάση τους, γιατί $O(\log_2 n) = O(\log_{10} n) = O(\ln n)$. Έτσι, για ευκολία μας, μπορούμε να τους θεωρούμε ως προς μία σταθερή βάση, π.χ. το 2 ή το 10.

Εκεί που πρέπει να προσέχουμε είναι όταν ο O-συμβολισμός είναι στον εκθέτη γιατί στην περίπτωση αυτή οι σταθερές που κρύβονται στο O επηρεάζουν τον ρυθμό αύξησης. Έτσι ενώ ισχύει $e^{3n} = e^{O(n)}$ έχουμε $e^{3n} = (e^n)^3 \neq O(e^n)$ με συνέπεια $e^{O(n)} \neq O(e^n)$.

Παράρτημα Β: Αριθμητική κινητής υποδιαστολής — η βάση της Αριθμητικής Ανάλυσης

Οι υπολογιστές είναι μηχανές με πεπερασμένη μνήμη, που όπως αναφέραμε αποτελείται από λέξεις πεπερασμένου μήκους. Όταν λοιπόν έκτελούμε αριθμητικούς υπολογισμούς με την βοήθεια τέτοιων μηχανών, πρέπει να παραστήσουμε το άπειρο πλήθος των πραγματικών αριθμών μέσα στην πεπερασμένη μνήμη του υπολογιστή. Επειδή προφανώς μία απεικόνιση ένα-προς-ένα είναι αδύνατη, προσεγγίζουμε τους πραγματικούς αριθμούς με τους αριθμούς κινητής υποδιαστολής (floating-point numbers), που είναι και αυτοί ένα πεπερασμένο σύνολο.

Ένα σύνολο F αριθμών κινητής υποδιαστολής χαρακτηρίζεται από τέσσερις παραμέτρους: την βάση των αριθμών β, την ακρίβεια (precision) t, και το διάστημα $[L, U]$ μέσα στο οποίο κινείται ο εκθέτης. Προφανώς, οι παράμετροι β, t, L, U εξαρτώνται από τον υπολογιστή. Κάθε αριθμός κινητής υποδιαστολής $f \in F$ γράφεται ως εξής:

$$f = \pm\left(\frac{d_1}{\beta} + \frac{d_2}{\beta^2} + \cdots + \frac{d_t}{\beta^t}\right)\beta^e$$

όπου οι ακέραιοι d_i, i = 1, 2, …,t ικανοποιούν την ανισότητα $0 \le d_i \le \beta$-1 και ο εκθέτης την ανισότητα $L \le e \le U$. Αν απαιτήσουμε να είναι $d_1 \neq 0$ για όλους τους αριθμούς $f \in F, f \neq 0$, τότε έχουμε τους λεγόμενους ***κανονικοποιημένους αριθμούς κινητής υποδιαστολής*** (normalized floating point numbers). Για παράδειγμα, με β = 10, ο αριθμός 1234 γράφεται σαν $0.1234 * 10^4$ και αποθηκεύεται σε μια λέξη μνήμης σαν το διανυσμα {0, 4, 1234}, όπου το 0 δηλώνει το πρόσημο +, το 4 δηλώνει τον ***εκθέτη*** (exponent or characteristic) και το 1234 το ***δεκαδικό μέρος*** (δ.μ. ή mantissa).

Το στάνταρντ της IEEE απαιτεί όπως: $\beta = 2$, $t = 24$ και -126 ≤ e ≤ 127 για αριθμούς απλής ακριβείας και $\beta = 2$, $t = 53$ και -1022 ≤ e ≤ 1023 για αριθμούς διπλής ακριβείας. Στην περίπτωση απλής ακριβείας οι αριθμοί κινητής υποδιαστολής που

μπορούν να παρασταθούν στον υπολογιστή σύμφωνα με το στάνταρντ είναι: $\pm (d_1 d_2 d_3 \ldots d_{24}) 2^e$, $-126 \le e \le 127$, $\pm$ Άπειρο, και NaN (Not-a-Number).

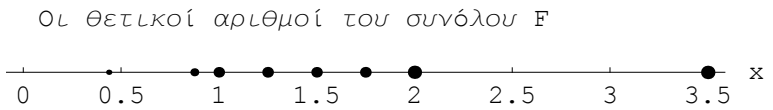
Σημειώνουμε πως για τους αριθμούς κινητής υποδιαστολής οι αριθμητικές πράξεις +, -, *, / εκτελούνται πολύ γρήγορα (hardware implemented). Αυτό γίνεται διότι στον υπολογιστή υπάρχουν ειδικά κυκλώματα που εκτελούν τις πράξεις αυτές χωρίς να χρειάζεται λογισμικό.

Αυτή η προσέγγιση όμως των πραγματικών αριθμών από τους αριθμούς κινητής υποδιαστολής δεν είναι και χωρίς προβλήματα. Προσέξτε ότι το σύνολο των αριθμών F οχι μόνον δεν είναι συνεχές αλλά ούτε και άπειρο. Υπάρχουν (μαζί με το 0) ακριβώς $2(\beta - 1)\beta^{t-1}(U - L + 1) + 1$ κανονικοποιημένοι αριθμοί κινητής υποδιαστολής στο F. Επιπλέον, οι αριθμοί αυτοί δεν έχουν την ίδια απόσταση μεταξύ τους σε όλο το διάστημα ορισμού τους, αλλά μόνον ανάμεσα σε διαδοχικές δυνάμεις του β.

Παράδειγμα 1. Ας θεωρήσουμε το σύνολο F οι παράμετροι του οποίου έχουν τις τιμές: $\beta = 2$, $t = 3$, $L = -1$, και $U = 2$. Το σύνολο αυτό αποτελείται από 33 κανονικοποιημένους αριθμούς κινητής υποδιαστολής. Επειδή οι 16 θετικοί αριθμοί είναι συμμετρικοί (γύρω από το μηδέν) με τούς αντίστοιχους 16 αρνητικούς παρουσιάζουμε μόνο τους πρώτους:

$$\left\{\frac{1}{4}, \frac{5}{16}, \frac{3}{8}, \frac{7}{16}, \frac{1}{2}, \frac{5}{8}, \frac{3}{4}, \frac{7}{8}, 1, \frac{5}{4}, \frac{3}{2}, \frac{7}{4}, 2, \frac{5}{2}, 3, \frac{7}{2}\right\}$$

Οι αριθμοί αυτοί παρίστανται ως $+\left(\frac{1}{2} + \frac{d_2}{4} + \frac{d_3}{8}\right) 2^e$ και υπολογίζονται από τον τύπο αυτό όταν $-1 \le e \le 2$, και για κάθε τιμή του e, $0 \le d_2, d_3 \le 1$. Η απεικόνισή τους φαίνεται στο ακόλουθο σχήμα:



Π*ροσέξτε πως η τιμή του* d_1 *είναι πάντα* 1 *επειδή πρόκειται για κανονικοποιημένους αριθμούς κινητής υποδιαστολής*. Ε*πιπλέον*, *σημειώστε πως το πλάτος των* 4 *πρώτων διαστημάτων* (*που αντιστοιχούν σε αριθμούς για* e = -1) *είναι* $\frac{1}{16}$, *το πλάτος των επόμενων* 4 *διαστημάτων* (e = 0) *είναι* $\frac{1}{8}$, *μετά το πλάτος γινεται* $\frac{1}{4}$ *και τέλος* $\frac{1}{2}$.

Ε*ξ αιτίας των ανωτέρω*, *είναι δυνατόν για* f_1, $f_2 \in F$ *το άθροισμά τους* f_1+f_2 *να μην ανήκει στο* F, *οπότε θα πρέπει να προσεγγισθεί—σε περίπτωση μη ισοαπόστασης—* *από τον πλησιέστερο αριθμό κινητής υποδιαστολής του* F. *Σε περίπτωση ισοαπόστασης η κατάσταση αντιμετωπίζεται με προσέγγιση* (*στρογγύλευση*) *είτε προς το άρτιο δεκαδικό μέρος* (round to even mantissa) *είτε προς το άπειρο* (round to infinity). (Υ*πάρχει βέβαια και η αποκοπή* (truncation) *η οποία εφαρμόζεται στις δυο περιπτώσεις*, *ισοαπόστασης* ή *μη*.) Το *λάθος που προκύπτει από την προσέγγιση ενός αριθμού από τον πλησιέστερο αριθμό του συνόλου* F *λέγεται* ***λάθος στρογγύλευσης*** (round-off error).

Ε*ν γένει*, *όταν κάνουμε αριθμητική κινητής υποδιαστολής δεν ισχύουν το προσεταιριστικό και επιμεριστικό αξίωμα της πρόσθεσης και του πολλαπλασιασμού*. Ε*τσι*, *το αποτέλεσμα μιας πράξης εξαρτάται*: *α*) *από την σειρά με την οποία εκτελούμε τις πράξεις και β*) *από τον τρόπο στρογγύλευσης που εφαρμόζουμε*. Α*υτό φαίνεται στην συνέχεια του παραδείγματός μας*.

Παράδειγμα 1—Συνέχεια. Έστω ότι στο σύνολο με τα 33 στοιχεία που ορίσαμε θέλουμε να υπολογίσουμε το άθροισμα $\frac{5}{4}+\frac{3}{8}+\frac{3}{8}$, όπου οι αριθμοί $\frac{5}{4}$ και $\frac{3}{8}$ ανήκουν στο σύνολο F και παρίστανται αντίστοιχα από τα διανύσματα {1, -2, 5} και {1, -4, 6}. Το άθροισμα μπορεί να υπολογισθεί είτε σαν $(\frac{5}{4}+\frac{3}{8})+\frac{3}{8}$ είτε σαν $\frac{5}{4}+(\frac{3}{8}+\frac{3}{8})$. Όπως βλέπουμε όμως παρακάτω, το αποτέλεσμα είναι το ίδιο μόνον όταν ο κανόνας στρογγύλευσης είναι προς το άρτιο δεκαδικό μέρος (round to even mantissa). Οι πράξεις μπορούν να γίνουν με το χέρι, αλλά προτιμούμε να τις κάνουμε με το Mathematica που έχει το ειδικό πρόγραμμα-προσομοιωτή:

```
Needs["NumericalMath`ComputerArithmetic`"]
```

Αφού φορτώσουμε το πρόγραμμα, καθορίζουμε τις παραμέτρους ώστε να έχουμε το σύνολο F με τα 33 στοιχεία. Ο κανόνας στρογγύλευσης είναι προς το άρτιο δεκαδικό μέρος (round to even mantissa).

```
SetArithmetic[3, 2, ExponentRange -> {-2, 2}]
```

```
{3, 2, RoundingRule → RoundToEven,
 ExponentRange → {-2, 2}, MixedMode → False, IdealDivide → False}
```

Κάνοντας τις πράξεις προκύπτει το ίδιο αποτέλεσμα:

```
{Print["(5/4 + 3/8) + 3/8 με στρογγύλευση
προς το άρτιο δ.μ. = ",
Normal[ComputerNumber[5/4+3/8]+ComputerNumber[
3/8]]],
    Print["5/4 + (3/8 + 3/8) με στρογγύλευση
προς το άρτιο δ.μ. = ",
Normal[ComputerNumber[5/4]+ComputerNumber[3/8+
3/8]]]};
```

```
(5/4 + 3/8) + 3/8 με στρογγύλευση προς το άρτιο δ.μ. = 2
```

```
5/4 + (3/8 + 3/8) με στρογγύλευση προς το άρτιο δ.μ. = 2
```

Αλλάζουμε τώρα τον κανόνα στρογγύλευσης και τον κάνουμε αποκοπή (truncation):

```
SetArithmetic[3, 2, ExponentRange -> {-2, 2},
RoundingRule -> Truncation]
```

```
{3, 2, RoundingRule → Truncation,
 ExponentRange → {-2, 2}, MixedMode → False, IdealDivide → False}
```

Κάνοντας τις πράξεις βλέπουμε ότι το αποτέλεσμα δεν είναι το ίδιο:

```
{Print["(5/4 + 3/8) + 3/8 με αποκοπή = ",
Normal[ComputerNumber[5/4+3/8]+ComputerNumber[
3/8]]],
    Print["5/4 + (3/8 + 3/8) με αποκοπή = ",
Normal[ComputerNumber[5/4]+ComputerNumber[3/8+
3/8]]]};
```

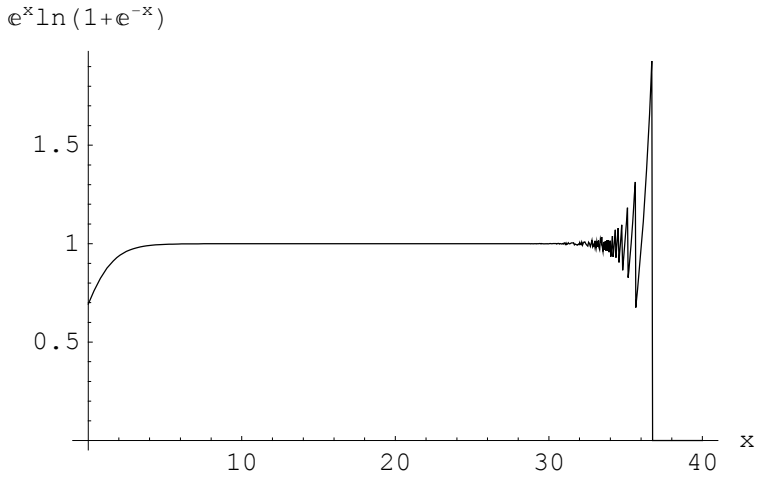
```
(5/4 + 3/8) + 3/8 με αποκοπή = 7/4
```

```
5/4 + (3/8 + 3/8) με αποκοπή = 2
```

Βλέπουμε λοιπόν πως το κύριο μέλημα στην Αριθμητική Ανάλυση είναι να προσέχουμε την σειρά των πράξεων και το σφάλμα στρογγύλευσης—αντί να εστιάζουμε την προσοχή μας στον εκάστοτε αλγόριθμο. Κλείνουμε το θέμα με ένα ακόμα παράδειγμα που τονίζει τα λάθη στρογγύλευσης.

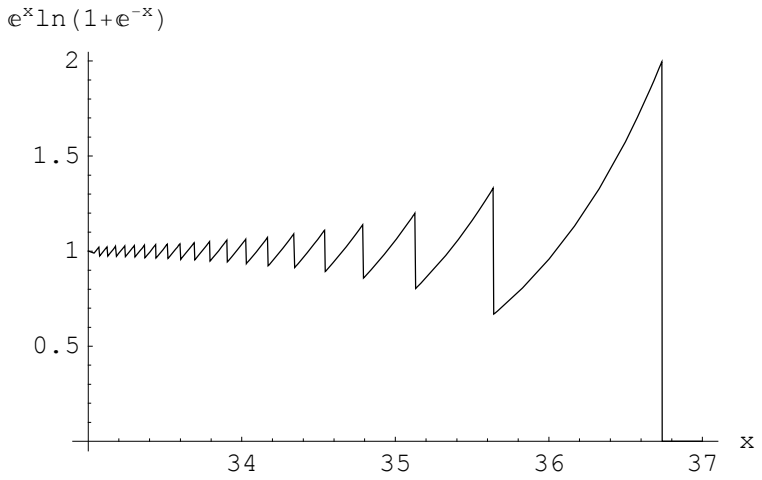
Παράδειγμα 2. Η συνάρτηση $e^x \ln(1 + e^{-x})$ είναι μαθηματικά καλά ορισμένη και το όριό της είναι 1 όταν το $x \to \infty$. Αριθμητικά όμως η συνάρτηση αυτή συμπεριφέρεται πολύ άσχημα, όπως φαίνεται από την γραφική παράστασή της:

```
Plot[E^x Log[1 + E^-x], {x, 0, 40}, PlotRange → All,
  AxesOrigin → {0, 0}, AxesLabel → {x, "e^x ln(1+e^-x)"}];
```



Βλέπουμε δηλαδή πώς αντί να αυξάνει σταθερά προς το 1 για αυξάνουσες τιμές του x, για $x \approx 30$ παρουσιάζεται ένας "θόρυβος" με αποτέλεσμα να έχουμε απόκλιση από την πραγματική τιμή 1, και για $x \geq 36.7368$ η τιμή της συνάρτησης γίνεται 0. Κάνοντας "ζουμ" στην περιοχή ενδιαφέροντος βλέπουμε πως πριν από την πτώση στο 0 υπάρχει μια συστηματική δομή από λωρίδες "τίγρη" (οι κάθετες γραμμές που συνδέουν τις λωρίδες μετράνε το ύψος του άλματος).

```
Plot[E^x Log[1 + E^-x], {x, 33, 37}, PlotRange → All,
  AxesOrigin → {33, 0}, AxesLabel → {x, "e^x ln(1+e^-x)"}];
```



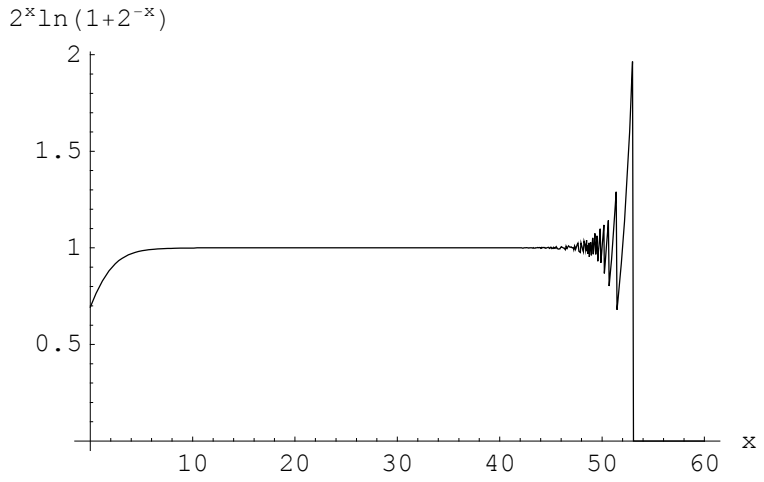
Οι *λωρίδες αυτές και το σημείο που η συνάρτηση πέφτει στο* 0 *είναι συνέπειες του* ϵ *του υπολογιστή* (machine epsilon) *και μπορούν πλήρως να προβλεφτούν*. Το ϵ *του υπολογιστή είναι το μικρότερο* ϵ *για το οποίο* $1 + \epsilon \neq 1$ *και μετράει τον μικρότερο αριθμό που δεν θα εξαφανισθεί εξ αιτίας της στρογγύλευσης*. (Στις *ασκήσεις δίνουμε ένα πρόγραμμα υπολογισμού του* ϵ *του υπολογιστή*.) Έτσι *ο αριθμός* $\frac{\epsilon}{2}$ *είναι ο μεγαλύτερος αριθμός που συμπεριφέρεται σαν* 0 *όταν προστεθεί στο* 1. Όταν *λοιπόν η συνάρτηση πέφτει στο* 0, *συμπεραίνουμε πως* e^{-x} = $\frac{\epsilon}{2}$ ή *ότι* $x = -\ln(\frac{\epsilon}{2})$. Πράγματι,

```
-Log[$MachineEpsilon/2]
```

```
36.7368
```

Μια *απλή τροποποίηση της συνάρτησης μας δίνει ένα πολύ χρήσιμο εργαλείο προσδιορισμού της ακρίβειας υπολογισμών* t. Δηλαδή *αν σχεδιάσουμε την συνάρτηση* $2^x \ln(1 + 2^{-x})$, *βλέπουμε πως η πτώση στο μηδέν γίνεται στην τιμή* $x = 53$, *που σημαίνει πως έχουμε ακρίβεια* 52 *δ.ψ*.

```
Plot[2^x Log[1 + 2^-x], {x, 0, 60}, PlotRange → All,
  AxesOrigin → {0, 0}, AxesLabel → {x, "2^x ln(1+2^-x)"}];
```



Πράγματι, *αυτο επιβεβαιώνεται παίρνοντας από το Mathematica την τιμή* $MachineEpsilon *σε δυαδική μορφή* (2^{-52}).

```
BaseForm[$MachineEpsilon, 2]
```

$1._2 \times 2^{-52}$

Παράρτημα Γ: Θεμελιώδεις έννοιες της Άλγεβρας

Στο παράρτημα αυτό παρουσιάζουμε τις θεμελιώδεις έννοιες της Άλγεβρας που θα χρειαστούμε στο βιβλίο αυτό. Συγκεκριμένα, θα ορίσουμε τις ομάδες, τους δακτύλιους, και τους δακτυλίους πολυωνύμων και επεκτάσεις σωμάτων. Όπως είναι φυσικό η παρουσίαση θα είναι σύντομη και χωρίς αποδείξεις αλλά θα περιλαμβάνει παραδείγματα. Εκτενή και άριστη κάλυψη των διαφόρων θεμάτων βρίσκεται στην βιβλιογραφία.

Συμβολισμός: *Θα χρησιμοποιήσουμε τον εξής συμβολισμό για μερικά κοινά σύνολα αριθμών*:

1. $\mathbb{Z} = \{0, \pm1, \pm2, \pm3, \ldots\}$ *δηλώνει τους* ***ακεραίους*** (integer numbers), *από την Γερμανική λέξη* Zahlen.

2. $\mathbb{Q} = \{\frac{a}{b}, a, b \in \mathbb{Z}, b \neq 0\}$ *δηλώνει τους* ***ρητούς*** *η* ***κλασματικούς*** *αριθμούς* (rational numbers).

3. $\mathbb{R} = \{$*όλα τα δεκαδικά αναπτύγματα* $\pm d_1 d_2 \ldots d_n.\delta_1 \delta_2 \ldots\}$ *δηλώνει τους* ***πραγματικούς*** *αριθμούς* (real numbers).

4. $\mathbb{C} = \{a + bi \mid a, b \in \mathbb{R}, i^2 = -1\}$ *δηλώνει τους* ***μιγαδικούς*** *αριθμούς* (complex numbers).

5. $\mathbb{Z}_{>0}$, $\mathbb{Q}_{>0}$ *και* $\mathbb{R}_{>0}$ *δηλώνουν τα θετικά* (*μη μηδενικά*) *στοιχεία των συνόλων* $\mathbb{Z}$, $\mathbb{Q}$ *και* $\mathbb{R}$ *αντίστοιχα*.

6. $\mathbb{R}^+$, $\mathbb{R}^\times$ *δηλώνουν το σύνολο* $\mathbb{R}$ *εφοδιασμένο μόνο με την πρόσθεση* ή *τον πολλαπλασιασμό αντίστοιχα* (*ο ίδιος συμβολισμός ισχύει και για τα άλλα σύνολα*).

7. $\mathbb{Z}\backslash\{0\}$ (ή *αλλιώς* $\mathbb{Z}$ - $\{0\}$), $\mathbb{Q}\backslash\{0\}$, $\mathbb{R}\backslash\{0\}$, $\mathbb{C}\backslash\{0\}$ *δηλώνουν τα μη μηδενικά στοιχεία των συνόλων* $\mathbb{Z}$, $\mathbb{Q}$, $\mathbb{R}$ *και* $\mathbb{C}$ *αντίστοιχα*.

8. $\mathbb{Z}/n\mathbb{Z}$ (ή *αλλιώς* $\mathbb{Z}_n$) $= \{\overline{0}, \overline{1}, \overline{2}, \ldots\overline{n-1}\}$ $\forall n \in \mathbb{N}_{>0}$ $(= \mathbb{Z}_{>0})$ *δηλώνει τους* ***ακεραίους*** **modulo** ***n*** (ή ***ακεραίους*** **mod** ***n***). (*Για τον συμβολισμό* $\mathbb{Z}/n\mathbb{Z}$ *βλέπε δακτύλιο πηλίκων*.) *Κάθε ένα από τα στοιχεία του είναι της μορφής* $\overline{a}$ *όπου* $\overline{a} = \{a + kn \mid k \in \mathbb{Z}\} = \{a, a \pm n, a \pm 2n, a \pm 3n, \ldots\}$ *και ονομάζεται* ***τάξη υπολοίπων*** (residue class) ή ***τάξη ισοδυναμίας*** (congruency class).

Σχετικά με τους ακεραίους $\mathbb{Z}$, ισχύει η αρχή της **καλής διάταξης** (well ordering principle) βάσει της οποίας αν A είναι ένα μη κενό υποσύνολο του $\mathbb{Z}_{>0}$, υπάρχει κάποιο **ελάχιστο** στοιχείο (minimal element) $m \in A$ έτσι ώστε $m \le a \; \forall a \in A$.

Όσον αφορά τους ακεραίους mod n, $\forall \overline{a}, \overline{b} \in \mathbb{Z}_n$ η πρόσθεση και ο πολλαπλασιασμός ορίζονται ως εξής:

$$\overline{a} + \overline{b} = \overline{a + b} \text{ και } \overline{a} \cdot \overline{b} = \overline{a \cdot b}.$$

Αυτό σημαίνει πως για να βρούμε το άθροισμα (ή το γινόμενο) δύο στοιχείων $\overline{a}, \overline{b} \in \mathbb{Z}/n\mathbb{Z}$ παίρνουμε έναν οποιονδήποτε ακέραιο a, **αντιπρόσωπο** της τάξης $\overline{a}$, και έναν οποιονδήποτε ακέραιο b, αντιπρόσωπο της τάξης $\overline{b}$, προσθέτουμε (πολλαπλασιάζουμε) τους ακέραιους a και b όπως ξέρουμε στο $\mathbb{Z}$, και παίρνουμε την τάξη ισοδυναμίας που περιέχει το αποτέλεσμα. Οι πράξεις είναι ανεξάρτητες από τους αντιπροσώπους που επιλέγουμε. (Η διαδικασία αυτή είναι γνωστή από τις πράξεις κλασματικών αριθμών, όπου επίσης διαλέγουμε αντιπρόσωπο. Για παράδειγμα $\frac{1}{2} = \frac{2}{4} = \frac{3}{6} = \cdots$)

Παράδειγμα:
Έστω $n = 12$. Το σύνολο $\mathbb{Z}_{12}$ αποτελείται από 12 τάξεις υπολοίπων (αργότερα θα παραλείπουμε την γραμμή πάνω από τους αριθμούς)

$$\{\overline{0}, \overline{1}, \overline{2}, \ldots \overline{11}\}$$

που είναι τα 12 υπόλοιπα που είναι δυνατόν να προκύψουν από την διαίρεση ενός ακεραίου με 12. Κάθε τάξη αποτελείται από άπειρο αριθμό ακεραίων. Έτσι, η τάξη $\overline{3}$ αποτελείται από όλους τους ακέραιους (**ισοδύναμους** ή **ισοϋπόλοιπους** προς το 3 mod 12) που αφίνουν υπόλοιπο 3 όταν διαιρεθούν με το 12. Οποιοσδήποτε ακέραιος ισοδύναμος προς το 3 (όπως για παράδειγμα …-21, -9, 3, 15, 27, …) μπορεί να θεωρηθεί αντιπρόσωπος της τάξης.
Έστω λοιπόν τώρα ότι $\overline{a} = \overline{5}$, $\overline{b} = \overline{8}$ και θέλουμε να υπολογίσουμε $\overline{a} + \overline{b}$. Παίρνουμε σαν αντιπροσώπους τα 5 και 8 και έχουμε $\overline{a} + \overline{b} = \overline{5} + \overline{8} = \overline{13} = \overline{1}$ διότι το 13 και το 1 ανήκουν στην ίδια τάξη υπολοίπων mod 12. Αν παίρναμε για αντιπρόσωπο της τάξης $\overline{a}$ το 17 και για αντιπρόσωπο της τάξης $\overline{b}$ το -28 θα είχαμε

$\overline{a} + \overline{b} = \overline{(17 - 28)} = \overline{-11} = \overline{1}$, όπως και προηγούμενα. Το γινόμενο των δύο τάξεων είναι $\overline{a} \cdot \overline{b} = \overline{5} \cdot \overline{8} = \overline{40} = \overline{4}$, επίσης ανεξάρτητο από την επιλογή των αντιπροσώπων.

Μία απεικόνιση (συνάρτηση) $f : A \to B$ μεταξύ δύο συνόλων A και B είναι ***μονοσήμαντη και εντός*** (injective ή one-to-one) αν $f(a_1) = f(a_2)$ συνεπάγεται $a_1 = a_2$ $\forall a_1, a_2 \in A$. Η απεικόνιση f είναι ***μονοσήμαντη και επί*** (surjective ή onto) αν $\forall b \in$ B, $\exists a \in A$ έτσι ώστε $f(a) = b$. (Επειδή η συνάρτηση απεικονίζεται στο πεδίο τιμών της πρέπει να ορίσουμε το codomain για να δούμε αν είναι μονοσήμαντη και επί.) Τέλος η απεικόνιση f είναι ***αμφιμονοσήμαντη*** (bijective**)** αν είναι μονοσήμαντη εντός και επί.

Αν S είναι ένα σύνολο και g, f είναι δύο συναρτήσεις (απεικονίσεις) από το S στο S τότε η ***σύνθεση*** (composition) των συναρτήσεων είναι η απεικόνιση $g \circ f : S \to S$ βάσει της οποίας $S \ni s \to g(f(s)) \in S$.

Δοθέντων δύο συνόλων A και B το γινόμενό τους είναι το σύνολο $A \times B = \{(a, b) \mid a \in A, b \in B\}$.

Μία δυαδική σχέση $\sim$ στο σύνολο A (δηλαδή ένα υποσύνολο του $A \times A$) είναι μια σχέση ισοδυναμίας αν είναι:

- ***αυτοπαθής*** (reflexive), δηλαδή $a \sim a$, $\forall a \in A$,
- ***συμμετρική*** (symmetric), δηλαδή $a \sim b$, συνεπάγεται $b \sim a$ $\forall a,b \in A$, και
- ***μεταβατική*** (transitive), δηλαδή $a \sim b$ και $b \sim c$ συνεπάγεται $a \sim c$ $\forall a,b,c \in A$

- **Γ1: Ομάδες**

Η *ομάδα είναι μία αλγεβρική δομή με μία πράξη.*

Ορισμός:

α. Μια ***δυαδική πράξη*** (binary operation) $*$ *στο σύνολο* G *είναι μια συνάρτηση* $*$: $G \times G \to G$. *Για κάθε ζεύγος* $a, b \in G$ *γράφουμε* $a*b$ *αντί να γράφουμε* $*(a, b)$.

β. Μια *δυαδική πράξη είναι* ***προσεταιριστική*** (associative) *αν* $\forall a,b,c \in G$ *ισχύει* $a*(b*c) = (a*b)*c$.

γ. Αν $*$ *είναι μια δυαδική πράξη σε ένα σύνολο* G *λέμε πως τα στοιχεία* $a, b \in G$ ***μετατίθενται*** *αν* $a*b = b*a$. *Λέμε πως η πράξη* $*$ *και το σύνολο* G *είναι* ***μεταθετικά*** *αν* $\forall a,b \in G$ *ισχύει* $a*b = b*a$.

Παραδείγματα:

1. Οι *συνήθεις δυαδικές πράξεις της πρόσθεσης* (+) *και του πολλαπλασιασμού* (×) *είναι μεταθετικές στο* $\mathbb{Z}$. Το *ίδιο ισχύει και στα σύνολα* $\mathbb{Q}$, $\mathbb{R}$ ή $\mathbb{C}$.

2. Η *συνήθης δυαδική πράξη της αφαίρεσης* (-) *δεν είναι μεταθετική στο σύνολο* $\mathbb{Z}$. Επιπλέον, *στο σύνολο των θετικών ακεραίων* $\mathbb{Z}_{>0}$ (*καθώς επίσης και στα σύνολα* $\mathbb{Q}_{>0}$, *και* $\mathbb{R}_{>0}$) *η αφαίρεση δεν είναι καν δυαδική πράξη διότι για* $a, b \in \mathbb{Z}_{>0}$ *με* $a < b$, $a - b \notin \mathbb{Z}_{>0}$, *δηλαδή δεν υπάρχει η απεικόνιση* -: $\mathbb{Z}_{>0} \times \mathbb{Z}_{>0} \to \mathbb{Z}_{>0}$.

Ορισμός:

α. **Ομάδα** (group) *είναι ένα διατεγμένο ζεύγος* $(G, *)$ *όπου* G *είναι ένα μη κενό σύνολο μαζί με μια δυαδική πράξη* $*$: $G \times G \to G$ *που ικανοποιεί τις εξής ιδιότητες*:

- Προσεταιριστκότητα (Associativity): $\forall a,b,c \in G$ *ισχύει* $a*(b*c) = (a*b)*c$,
- Ταυτότητα (Identity): $\exists e \in G$ *έτσι ώστε* $a*e = e*a = a \ \ \forall a \in G$. (Η *ιδιότητα αυτή εξασφαλίζει πως το σύνολο* G *δεν είναι κενό*.) Το e *ονομάζεται* ***ταυτοτικό στοιχείο*** (identity) *του* G *και πολλές φορές θα το παριστάνουμε και με* **1**,
- Αντίστροφο (Inverse): $\forall a \in G \ \exists a^{-1} \in G$ *έτσι ώστε* $a*a^{-1} = a^{-1}*a = e$. Το *στοιχείο* a^{-1} *ονομάζεται* ***αντίστροφο*** *του* a.

β. Η *ομάδα* $(G, *)$ *λέγεται* ***αβελιανή*** (abelian) ή ***μεταθετική*** (commutative) *αν* $\forall a,b \in G$ *ισχύει* $a*b = b*a$.

Σημείωση:

Παρακάτω, για $\forall a,b \in G$ αντί να γράφουμε $a*b$ θα γράφουμε απλά ab και για ταυτοτικό στοιχείο της ομάδας θα θεωρούμε το 1.

Παραδείγματα:

1. Τα σύνολα $\mathbb{Z}^+$, $\mathbb{Q}^+$, $\mathbb{R}^+$ και $\mathbb{C}^+$ είναι ***προσθετικές*** ομάδες (additive groups) με $e = 0$ και $\forall a \quad a^{-1} = -a$.

2. Τα σύνολα $\mathbb{Q}^\times\backslash\{0\}$, $\mathbb{R}^\times\backslash\{0\}$, $\mathbb{C}^\times\backslash\{0\}$, $\mathbb{Q}^\times_{>0}$, $\mathbb{R}^\times_{>0}$ είναι ***πολλαπλασιαστικές*** ομάδες (multiplicative group) με $e = 1$ και $\forall a \quad a^{-1} = \frac{1}{a}$. Προσέξτε όμως πως $\mathbb{Z}^\times\backslash\{0\}$ δεν είναι ομάδα ως προς τον πολλαπλασιασμό διότι το 3, για παράδειγμα, δεν έχει αντίστροφο στο $\mathbb{Z}^\times\backslash\{0\}$.

Ορισμός:

Η ***τάξη*** κάθε ομάδας G είναι το πλήθος των στοιχείων της και συμβολίζεται με $|G|$ ή $\#G$ (***πληθικός*** αριθμός (cardinality) του G). Για τυχόν $g \in G$ ορίζουμε ως ***τάξη*** (order) του στοιχείου g τον μικρότερο θετικό ακέραιο m (αν υπάρχει) ώστε $g^m = 1$. Ο συμβολισμός είναι $\mathbf{ord}(g) = m$, και λέμε ότι το g είναι τάξης m. Αν δεν υπάρχει τέτοιο m, τότε λέμε πως η τάξη του g είναι άπειρη.

Παραδείγματα:

1. Ένα στοιχείο μιας ομάδας έχει τάξη 1 εάν και μόνο εάν είναι το ταυτοτικό στοιχείο.

2. Στις προσθετικές ομάδες $\mathbb{Z}^+$, $\mathbb{Q}^+$, $\mathbb{R}^+$ και $\mathbb{C}^+$ κάθε μη μηδενικό (μη ταυτοτικό) στοιχείο έχει άπειρη τάξη.

3. Στις πολλαπλασιαστικές ομάδες $\mathbb{Q}^\times\backslash\{0\}$ και $\mathbb{R}^\times\backslash\{0\}$ το στοιχείο -1 έχει τάξη 2 και όλα τα άλλα μη ταυτοτικά στοιχεία έχουν άπειρη τάξη.

Ορισμός:

Έστω μία ομάδα G. Το υποσύνολο H του G είναι μία ***υποομάδα*** (subgroup) της G αν το H είναι μη κενό και κλειστό ως προς τις πράξεις του πολλαπλασιασμού και της αντιστροφής, δηλαδή αν $\forall a,b \in H$ ισχύει $ab \in H$ και $a^{-1} \in H$. Αν H είναι μία υποομάδα της G γράφουμε $H \subseteq G$.

Παραδείγματα:

1. $\mathbb{Z}^+ \subseteq \mathbb{Q}^+$ και $\mathbb{Q}^+ \subseteq \mathbb{R}^+$.

2. Το σύνολο των άρτιων ακεραίων είναι υποομάδα της $\mathbb{Z}^+$.

3. Κάθε ομάδα G έχει δύο υποομάδες: $H = G$ και $H = \{1\}$.

Κριτήριο Υποομάδας:

Για να αποδείξουμε ότι ένα υποσύνολο H μιας ομάδας G είναι υποομάδα αρκεί να δείξουμε πως H είναι μη κενό, και $\forall x,y \in H$, $xy^{-1} \in H$.

Ένα υποσύνολο S του G "παράγει" (generates) την $\langle S \rangle \subseteq G$ που αποτελείται από όλα τα πεπερασμένα γινόμενα στοιχείων του S και τα αντίστροφά τους. $\langle S \rangle$ είναι η μικρότερη υποομάδα του G που περιέχει το σύνολο S. Αν $S = \{g_1, g_2, \ldots, g_s\}$ είναι πεπερασμένο τότε γράφουμε και $\langle g_1, g_2, \ldots, g_s \rangle$ αντί $\langle \{ g_1, g_2, \ldots, g_s \} \rangle$.

Ορισμός:

Αν υπάρχει ένα στοιχείο $g \in G$ που παράγει όλη την ομάδα $G = \langle g \rangle$ τότε η ομάδα G λέγεται ***κυκλική*** (cyclic) και το στοιχείο g λέγεται ***γεννήτρια*** (generator) της G.

Τώρα μπορούμε ισοδύναμα να πούμε πως **ord**$(g) = m$, αν η τάξη της κυκλικής υποομάδας που παράγει το g είναι m. Αυτό σημαίνει $g^m = 1$ όπως ορίσαμε προηγούμενα.

Παραδείγματα:

1. Αν G είναι η προσθετική ομάδα $\mathbb{Z}_{12}^+ = \{\overline{0}, \overline{1}, \overline{2}, \ldots \overline{11}\}$ τότε το σύνολο $S = \{\overline{3}, \overline{8}\}$ παράγει την ομάδα G. Για παράδειγμα $1 = 9 + 4 \bmod 12$ όπου $9 = 3 + 3 + 3 \bmod 12$ και $4 = 8 + 8 \bmod 12$, κ.ο.κ. Το G όμως παράγεται και από το $S = \{\overline{1}\}$, με μία μόνο γεννήτρια. Επίσης κάθε ένα από τα στοιχεία $\overline{5}, \overline{7}$ και $\overline{11}$ παράγει την $\mathbb{Z}_{12}^+$.

2. Η προσθετική ομάδα $\mathbb{R}^+$ δεν έχει πεπερασμένο σύνολο γεννητριών.

3. Οι ρίζες της μονάδας τάξης n αποτελούν μία πολλαπλασιαστική ομάδα. Για $n = 8$ έχουμε την εξής ομάδα

```
Map[ComplexExpand, Solve[x^8 - 1 == 0], 3]
```

$$\left\{\{x \to -1\}, \{x \to -i\}, \{x \to i\}, \{x \to 1\}, \left\{x \to -\frac{1+i}{\sqrt{2}}\right\}, \left\{x \to \frac{1+i}{\sqrt{2}}\right\}, \left\{x \to \frac{1-i}{\sqrt{2}}\right\}, \left\{x \to -\frac{1-i}{\sqrt{2}}\right\}\right\}$$

Τις *ρίζες αυτές τις παριστάνουμε και ως* $\omega_k = e^{2\pi i k/8}$, $k = 0, 1, \ldots, 7$,

```
Table[ComplexExpand[ω_k = e^(2 π i k / 8)], {k, 0, 7}]
```

$$\left\{1, \frac{1+i}{\sqrt{2}}, i, -\frac{1-i}{\sqrt{2}}, -1, -\frac{1+i}{\sqrt{2}}, -i, \frac{1-i}{\sqrt{2}}\right\}$$

και βλέπουμε πως η ρίζα ω_1 *είναι γεννήτρια της κυκλικής ομάδας* (*βρήτε τις άλλες*).

```
Table[ComplexExpand[ω_1^k], {k, 0, 7}]
```

$$\left\{1, \frac{1+i}{\sqrt{2}}, i, -\frac{1-i}{\sqrt{2}}, -1, -\frac{1+i}{\sqrt{2}}, -i, \frac{1-i}{\sqrt{2}}\right\}$$

Ορισμός:

Το *σύνολο των* $n!$ *διατάξεων* (permutations) *των στοιχείων* $\{1, 2, \ldots, n\}$, $n \in \mathbb{Z}_{>1}$, *αποτελεί την* ***συμμετρική*** *ομάδα* S_n (symmetric group) *με πράξη την σύνθεση συναρτήσεων*. Εν *γένει*, *μία διάταξη ενός συνόλου* A *είναι μια αμφιμονοσήμαντη απεικόνιση από το* A *στο* A.

Παραδείγμα:

Ας *εξετάσουμε την συμμετρική ομάδα* S_3 *που περιέχει έξι* (3!) *στοιχεία*, *τις έξι διατάξεις των στοιχείων* $\{1, 2, 3\}$. *Για να περιγράψουμε την ομάδα διαλέγουμε δύο συγκεκριμένες διατάξεις* $\sigma = \{2, 3, 1\}$ (*κυκλική αναδιάταξη*) *και* $\tau = \{2, 1, 3\}$ (*αναδιάταξη των δύο πρώτων*) *που τις παριστάνουμε με μορφή* ***πινάκων διάταξης*** (permutation matrix)

$$\sigma = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix}; \quad \tau = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix};$$

Ο *πίνακας διάταξης P είναι ένας πίνακας με την ιδιότητα ότι αν πολλαπλασιάσουμε με αυτόν από τα αριστερά έναν άλλο πίνακα M, επέρχεται μια αναδιάταξη στις σειρές του M. Έτσι για παράδειγμα έχουμε*

```
σ.{1, 2, 3}
```

```
{2, 3, 1}
```

```
τ.{1, 2, 3}
```

```
{2, 1, 3}
```

Στην περίπτωση αυτή η σύνθεση συναρτήσεων αντιστοιχεί στον πολλαπλασιασμός πινάκων. Με την βοήθεια των σ και τ οι έξι διατάξεις των στοιχείων $\{1, 2, 3\}$ *είναι*

$$\{1, \sigma, \sigma^2, \tau, \sigma\tau, \sigma^2\tau\} = \{\sigma^i \tau^j \mid 0 \le i \le 2, 0 \le j \le 1\}$$

όπου $1 = \{1, 2, 3\} \to \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$. *Εύκολα επιβαιώνονται οι σχέσεις* $\sigma^3 = 1$, $\tau^2 = 1$ *και* $\tau\sigma = \sigma^2\tau$

```
MatrixPower[σ, 3] == IdentityMatrix[3]
```

```
True
```

```
MatrixPower[τ, 2] == IdentityMatrix[3]
```

```
True
```

```
τ.σ == MatrixPower[σ, 2].τ
```

```
True
```

Προσέξτε πως η τρίτη σχέση $\tau\sigma = \sigma^2\tau$ *χρησιμοποιείται σε μια έκφραση για να φέρουμε το τ δεξιά του σ και μετά οι δύο πρώτες μειώνουν τους εκθέτες.*

Ορισμός:

Για *κάθε υποομάδα* $H \subseteq G$ *και κάθε* $g \in G$ *ορίζουμε ως το αριστερό* (ή *δεξί*) ***σύμπλοκο*** (coset) *της* H *στην* G *το εξής σύνολο*

$$gH = \{gh \mid h \in H\} \quad (\text{ή } Hg = \{hg \mid h \in H\}).$$

Τα *σύμπλοκα είναι* ***τάξεις ισοδυναμίας*** *ως προς την σχέση ισοδυναμίας*

$$\forall a,b \in G \;\; a \equiv b \;\; \text{εάν} \;\; b = ah \;\; \text{για κάποιο } h \in H.$$

Όπως *είναι γνωστό από την βιβλιογρφία*, *οι* ***σχέσεις ισοδυναμίας*** (equivalence relations) *χωρίζουν* (partition) *τα στοιχεία του συνόλου σε τάξεις ισοδυναμίας και έτσι τα σύμπλοκα χωρίζουν την ομάδα σε τάξεις ισοδυναμίας*.

Ο *συμβολισμός* aH *υποδηλώνει ένα συγκεκριμένο υποσύνολο της ομάδας* G. Το *σύμπλοκο* aH *είναι το μοναδικό που περιέχει το* a *και έτσι* $aH = bH$ *αν και μόνον εάν* $a = b$. Προφανώς *κάθε σύμπλοκο περιέχει τόσα στοιχεία όσα και η υποομάδα* H.

Παράδειγμα:

Ας *μείνουμε στην συμμετρική ομάδα* S_3 *που περιέχει τα έξι* (3!) *στοιχεία*, $S_3 = \{1, \sigma, \sigma^2, \tau, \sigma\tau, \sigma^2\tau\} = \{\sigma^i \tau^j \mid 0 \le i \le 2,\ 0 \le j \le 1\}$. Το *στοιχείο* $\sigma\tau$ *έχει τάξη* 2, *δηλαδή* $(\sigma\tau)^2 = 1$, *και επομένως παράγει την κυκλική ημιομάδα* $H = \{1, \sigma\tau\}$ *τάξης* 2, *δηλαδή* $\#H = 2$.

```
MatrixPower[σ.τ, 2] == IdentityMatrix[3]

True
```

Για *να βρούμε τα αριστερά σύμπλοκα της ημιομάδας* H *στην* S_3 *δοκιμάζουμε και τα έξι υποψήφια σύμπλοκα* $1H = \{1, \sigma\tau\}$, $\sigma H = \{\sigma, \sigma^2\tau\}$, $\sigma^2 H = \{\sigma^2, \sigma^3\tau\}$, $\tau H = \{\tau, \tau\sigma\tau\}$, $\sigma\tau H = \{\sigma\tau, \sigma\tau\sigma\tau\}$, $\sigma^2\tau H = \{\sigma^2\tau, \sigma^2\tau\sigma\tau\}$.

Τα σύμπλοκα αυτά όμως πρέπει να τα φέρουμε στην μορφή $\{1, \sigma, \sigma^2, \tau, \sigma\tau, \sigma^2\tau\} = \{\sigma^i \tau^j \mid 0 \le i \le 2,\ 0 \le j \le 1\}$ *χρησιμοποιώντας τις σχέσεις* $\sigma^3 = 1$, $\tau^2 = 1$ *και* $\tau\sigma =$

$\sigma^2\tau$ *και να διαγράψουμε τα επαναλαμβανόμενα*. Τα *δύο πρώτα δεν μπορούν να τροποποιηθούν*. Έτσι *έχουμε τα πρώτα*

2 *αριστερά σύμπλοκα της H*:

$$1H = \{1, \sigma\tau\},\ \sigma H = \{\sigma, \sigma^2\tau\},$$

Το *τρίτο* (*πιθανό*) *σύμπλοκο είναι* $\sigma^2 H = \{\sigma^2, \sigma^3\tau\}$. *Όμως* $\sigma^3\tau = \tau$ (*βάσει της* $\sigma^3 = 1$) *και έτσι* $\sigma^2 H = \{\sigma^2, \tau\}$ *το οποίο προσθέτουμε στην λίστα των σύμπλοκων επειδή είναι διαφορετικό από τα προηγούμενα*

3 *αριστερά σύμπλοκα της H*:

$$1H = \{1, \sigma\tau\},\ \sigma H = \{\sigma, \sigma^2\tau\},\ \sigma^2 H = \{\sigma^2, \tau\}$$

Το *τέταρτο* (*πιθανό*) *σύμπλοκο είναι* $\tau H = \{\tau, \tau\sigma\tau\}$ = (*επειδή* $\tau\sigma = \sigma^2\tau$) = $\{\tau, \sigma^2\tau^2\}$ = (*επειδή* $\tau^2 = 1$) = $\{\tau, \sigma^2\}$ = $\sigma^2 H$ *που υπάρχει ήδη στην λίστα*.

Το *πέμπτο* (*πιθανό*) *σύμπλοκο είναι* $\sigma\tau H = \{\sigma\tau, \sigma\tau\sigma\tau\}$ = (*επειδή* $\tau\sigma = \sigma^2\tau$) = $\{\sigma\tau, \sigma^3\tau^2\}$ = (*επειδή* $\sigma^3 = 1$ *και* $\tau^2 = 1$) = $\{\sigma\tau, 1\}$ = $1H$ *το οποίο έχουμε ήδη συμπεριλάβει στην λίστα*.

Το *έκτο* (*πιθανό*) *σύμπλοκο είναι* $\sigma^2\tau H = \{\sigma^2\tau, \sigma^2\tau\sigma\tau\}$ = (*επειδή* $\tau\sigma = \sigma^2\tau$) = $\{\sigma^2\tau, \sigma^2\sigma^2\tau\tau\}$ = (*επειδή* $\sigma^3 = 1$) = $\{\sigma^2\tau, \sigma\tau^2\}$ = $\{\sigma^2\tau, \sigma\}$ = σH *που είναι ήδη στην λίστα*.

Άρα *τα* ***αριστερά*** *σύμπλοκα της H στην* S_3 *που χωρίζουν την* S_3 *σε* 3 *τάξεις ισοδυναμίας είναι*:

3 *αριστερά σύμπλοκα της H*:

$$1H = \sigma\tau H = \{1, \sigma\tau\},\ \sigma H = \sigma^2\tau H = \{\sigma, \sigma^2\tau\},\ \sigma^2 H = \tau H = \{\sigma^2, \tau\}$$

Προσοχή! Τα *αριστερά σύμπλοκα δεν είναι κατ' ανάγκη τα ίδια με τα δεξιά σύμπλοκα*. Έτσι *τα δεξιά σύμπλοκα της υποομάδας* $H = \{1, \sigma\tau\}$ *στην* S_3 *είναι*

3 *δεξιά σύμπλοκα της H*:

$$H = H\sigma\tau = \{1, \sigma\tau\},\ H\sigma = H\tau = \{\sigma, \tau\},\ H\sigma^2 = H\sigma^2\tau = \{\sigma^2, \sigma^2\tau\}$$

και χωρίζουν την σε διαφορετικές τάξεις ισοδυναμίας. Εξαίρεση αποτελούν οι ***κανονικές*** (normal) *υποομάδες για τις οποίες* $gH = Hg \ \forall g \in G$. (*δηλαδή κάθε αριστερό σύμπλοκο είναι επίσης και δεξί σύμπλοκο*).

Ορισμός:
Το *σύνολο των αριστερών σύμπλοκων μιας υποομάδας H της G συμβολίζεται με* $G : H$. Ο *πληθικός αριθμός του συνόλου αυτού λέγεται* ***δείκτης*** (index) *της H στην G και συμβολίζεται με* $|G : H|$ ή $\#(G : H)$.

Έτσι *στο παράδειγμά μας ο δείκτης είναι* 3. Αν *η ομάδα G περιέχει άπειρα στοιχεία, ο δείκτης μπορεί να είναι επίσης άπειρος*.

Επειδή *η ομάδα G είναι η ένωση όλων των σύμπλοκων της υποομάδας H και επειδή τα σύμπλοκα δεν έχουν κοινά στοιχεία έχουμε τον τύπο*

$$|G| = |H| \cdot |G : H| \quad \text{ή} \quad \#G = \#H \cdot \#(G : H)$$

Στο *προηγούμενο παράδειγμά μας έχουμε* $6 = 2 \cdot 3$. Ένα από *τα επακόλουθα είναι και το περίφημο*

Θεώρημα του Lagrange: Αν *G είναι μια πεπερασμένη ομάδα, και H μια υποομάδα της G, τότε η τάξη της H διαιρεί την τάξη της G και ο αριθμός των σύμπλοκων* (*ο δείκτης*) *της H στην G είναι*

$$|G : H| = \frac{|G|}{|H|}$$

Μια *πολύ σημαντική συνέπεια του θεωρήματος του* Lagrange *είναι ότι η τάξη ενός στοιχείου* $g \in G$ *διαιρεί την τάξη της ομάδας και συγκεκριμένα* $g^{|G|} = 1 \ \forall g \in G$.

Ορισμός:
Έστω *ότι G και H είναι δύο πολλαπλασιαστικές ομάδες*. Η *απεικόνιση* $\varphi : G \to H$ *λέγεται* ***ομοιομορφισμός*** *ομάδων* (group homomorphism) *αν* $\forall g_1, g_2 \in G$ *ισχύει* $\varphi(g_1 g_2) = \varphi(g_1)\varphi(g_2)$.

Προσέξτε πως το γινόμενο $g_1 g_2$ στα αριστερά εκτελείται στην ομάδα G ενώ το γινόμενο $\varphi(g_1)\varphi(g_2)$ στα δεξιά εκτελείται στην ομάδα H. Δηλαδή, μία απεικόνιση φ είναι ομοιομορφισμός ομάδων αν δεν αλλάζει τις δομές των ομάδων στα πεδία ορισμού της, G και H (domain και codomain).

Ένας αμφιμονοσήμαντος ομοιομορφισμός ομάδων $\varphi : G \to H$ λέγεται ***ισομορφισμός*** (group isomorphism) μεταξύ των ομάδων G και H. Οι ομάδες λέγονται ***ισομορφικές*** και συμβολίζονται με $G \cong H$. Με απλά λόγια οι ισομορφικές ομάδες G και H είναι η ίδια ομάδα μόνο που τα στοιχεία και οι πράξεις γράφονται διαφορετικά στο G και στο H. (Αυτό δικαιολογεί την απόφασή μας να θεωρούμε ότι η πράξη σε όλες τις ομάδες είναι $\cdot$, διότι αλλάζοντας το σύμβολο της πράξης δεν αλλάζει ο τύπος ισομορφισμού.) Ένας ομοιομορφισμός ομάδων $\chi : G \to H$ είναι ισομορφισμός εάν και μόνον εάν υπάρχει ένας ομοιομορφισμός ομάδων $\psi : H \to G$ έτσι ώστε $\chi \circ \psi = 1_H$, το ταυτοτικό στοιχείο του H και $\psi \circ \chi = 1_G$, το ταυτοτικό στοιχείο του G και όπου $\circ$ είναι η σύνθεση συναρτήσεων.

Παράδειγμα:
Θεωρούμε την προσθετική ομάδα $\mathbb{R}^+$, την πολλαπλασιαστική ομάδα $\mathbb{R}^\times_{>0}$ και την εκθετική απεικόνιση $\exp : \mathbb{R}^+ \to \mathbb{R}^\times_{>0}$, που ορίζεται ως $\exp(x) = e^x$. Η απεικόνιση exp είναι ισομορφισμός διότι έχει την αντίστροφη συνάρτηση $\log_e$ και διατηρεί τις πράξεις των ομάδων αφού $e^{x+y} = e^x e^y$.

Ορισμός:
Εάν $\varphi : G \to H$ είναι ομοιομορφισμός ομάδων τότε ο ***πυρήνας*** (kernel) του φ, είναι το σύνολο

$$\ker(\varphi) = \{g \in G \mid \varphi(g) = 1_H\} \subseteq G,$$

που είναι ***κανονική*** υποομάδα της G και 1_H είναι το ταυτοτικό στοιχείο της H. (Μια υποομάδα N μιας ομάδας G είναι κανονική εάν και μόνον εάν N είναι ο πυρήνας κάποιου ομοιομορφισμού.) Η ***εικόνα*** (image) του φ είναι το σύνολο

$$\varphi(G) = \{\varphi(g) \mid g \in G\} \subseteq H,$$

υποομάδα της H. Ο *ομοιομορφισμός* φ *είναι μονοσήμαντος και εντός εάν και μόνον εάν* $\ker(\varphi) = \{1_G\}$, *η ταυτοτική υποομάδα της* G.

Επειδή $K = \ker(\varphi)$ *είναι κανονική υποομάδα της* G *μπορούμε να σχηματίσουμε το σύνολο* $G : K$ *των σύμπλοκων της* K *στην* G. Το *σύνολο* $G : K$ *είναι επίσης ομάδα που λέγεται* ***ομάδα πηλίκων*** (quotient group ή factor group) *της* G modulo K *και συμβολίζεται με* G/K. Η (*καλά ορισμένη*) *πράξη της ομάδας αυτής ορίζεται από* $(gK)(g^*K) = (gg^*)K, \forall g,g^* \in G$.

Παράδειγμα:
Ας *επανέλθουμε στην συμμετρική ομάδα* S_3 *με τα έξι στοιχεία*, $\{1, \sigma, \sigma^2, \tau, \sigma\tau, \sigma^2\tau\}$ *με τις σχέσεις* $\sigma^3 = 1$, $\tau^2 = 1$ *και* $\tau\sigma = \sigma^2\tau$, *και την κυκλική ημιομάδα της* $H = \{1, \sigma\tau\}$. Η *ομάδα πηλίκων της* S_3 modulo H *είναι* $S_3/H = \{1H, \sigma H, \sigma^2 H\}$. Προσέξτε *πως* $(\sigma H)\cdot(\sigma^2 H) = \sigma\sigma^2 H = \sigma^3 H = 1H$.

Το (*πρώτο*) ***θεώρημα ομοιομορφισμού ομάδων*** *δηλώνει πως* $G/\ker(\varphi) \cong \varphi(G)$ *από το οποίο συνεπάγεται πως* $|G : \ker(\varphi)| = |\varphi(G)|$ ή *αλλιώς* $|G| = |\ker(\varphi)|\cdot|\varphi(G)|$.

Αν G *και* H *είναι δύο ομάδες*, *μια νέα ομάδα* $G \times H$, *το* ***γινόμενό*** *τους* (direct product) *σχηματίζεται θέτοντας* $G \times H = \{(g, h) \mid g \in G, h \in H\}$ *και ορίζοντας τον πολλαπλασιασμό ως* $(g_1, h_1)\cdot(g_2, h_2) = (g_1 g_2, h_1 h_2)$ $\forall g_1, g_2 \in G$ *και* $\forall h_1, h_2 \in H$.

■ **Γ2: Δακτύλιοι**

Ο *δακτύλιος είναι μία αλγεβρική δομή με δύο πράξεις*.

Ορισμός:

1. Ένας ***δακτύλιος*** (ring) *είναι ένα σύνολο* R *με δύο δυαδικές πράξεις* $+, \cdot : R \times R \to R$ *με τις εξής ιδιότητες*

▪ R *με* $+$, *είναι αβελιανή* (*μεταθετική*) *ομάδα με ταυτοτικό στοιχείο* 0,

▪ *η πράξη* $\cdot$ *είναι προσεταιριστική*,

▪ *ισχύει η* ***επιμεριστική*** *ιδιότητα* (distributive law): $\forall a,b,c \in R\ a\cdot(b + c) = (a\cdot b) + (a\cdot c)$ *και* $(a + b)\cdot c = (a\cdot c) + (b\cdot c)$.

2. Ο *δακτύλιος* R *είναι* ***μεταθετικός*** *αν η πράξη* $\cdot$ *είναι μεταθετική*.

3. Λέμε *ότι ο δακτύλιος* R *έχει ταυτοτικό στοιχείο* 1 *ως προς τον πολλαπλασιασμό, αν υπάρχει ένα στοιχείο* $1 \in R$ *ώστε* $1\cdot a = a\cdot 1 = a, \forall a \in R$.

***Σημείωση*:** Εκτός *και αν αναφέρουμε το αντίθετο, οι δακτύλιοι με τους οποίους θα ασχοληθούμε θα είναι μεταθετικοί με* 1.

Παραδείγματα:

1. Γ*νωστά παραδείγματα μεταθετικών δακτυλίων με ταυτοτικό στοιχείο είναι* $\mathbb{Z}$, $\mathbb{Q}$, $\mathbb{R}$, $\mathbb{C}$ *με τις συνήθεις πράξεις πρόσθεσης και πολλαπλασιασμού, και* $\forall n \in \mathbb{N}_{>0}$ *το σύνολο* $\mathbb{Z}/n\mathbb{Z}$ *με πρόσθεση και πολλαπλασιασμό* modulo n.

2. Κλασσικό *παράδειγμα μη μεταθετικού δακτυλίου με ταυτοτικό στοιχείο είναι το σύνολο* $M_n(\mathbb{R})$ ή $\mathbb{R}^{n\times n}$, *των* $n\times n$ *πινάκων με στοιχεία πραγματικούς αριθμούς και πράξεις την πρόσθεση και πολλαπλασιασμό πινάκων*.

3. Παράδειγμα *μεταθετικού δακτυλίου χωρίς ταυτοτικό στοιχείο είναι ο δακτύλιος* $2\mathbb{Z}$ *των αρτίων ακεραίων με πρόσθεση και πολλαπλασιασμό ακεραίων*.

4. Τέλος *μια σημαντική τάξη δακτυλίων είναι οι δακτύλιοι συναρτήσεων*. Έστω X *ένα μη κενό σύνολο και* A *ένας δακτύλιος*. Το *σύνολο* R *των συναρτήσεων* $f : X \to A$ *αποτελεί δακτύλιο με τις συνήθεις πράξεις πρόσθεσης και πολλαπλασιασμού συναρτήσεων σε σημείο* x: $(f + g)(x) = f(x) + g(x)$ *και* $(fg)(x) = f(x)g(x)$. Κάθε *αξίωμα των δακτυλίων στο* R *έπεται από το αντίστοιχο αξιώμα στο* A. Ο *δακτύλιος* R *είναι μεταθετικός εάν και μόνον εάν ο* A *είναι μεταθετικός και ο* R *έχει*

ταυτοτικό στοιχείο 1 (*την σταθερά συνάρτηση* 1 *στο X*) *εάν και μόνον εάν ο A έχει* 1.

Ε*να παράδειγμα συνόλου που δεν είναι δακτύλιος είναι το* $\mathbb{Z}\backslash\{6\}$ (ή $\mathbb{Z}$ - {6}) *με πρόσθεση και πολλαπλασιασμό ακεραίων*. Η *αιτία είναι πως για* $\forall a,b \in \mathbb{Z}\backslash\{6\}$ *δεν είμαστε σίγουροι πως* $a + b \in \mathbb{Z}\backslash\{6\}$. Γ*ια παράδειγμα* $4 + 2 = 6$. Σ*ε αυτές τις περιπτώσεις λέμε πως το σύνολο* $\mathbb{Z}\backslash\{6\}$ *δεν είναι* ***κλειστό*** (closed) *ως προς την πρόσθεση*. (Τ*ο ίδιο συμβαίνει και με τον πολλαπλασιασμό*.) Α*υτό σημαίνει πως το* $\mathbb{Z}\backslash\{6\}$ *είναι υποσύνολο του μεγαλυτέρου συνόλου* $\mathbb{Z}$ *μέσα στο οποίο οι πράξεις αυτές είναι καλά ορισμένες*.

Ορισμός:
Ένας *δακτύλιος R με ταυτοτικό στοιχείο* 1, *όπου* $1 \neq 0$, *λέγεται* ***δακτύλιος διαίρεσης*** (division ring) *αν* $\forall a \in R,\ a \neq 0,\ \exists b \in R$ *έτσι ώστε* $ab = ba = 1$ (*b είναι το πολλαπλασιαστικό αντίστροφο του a*). Ένας *μεταθετικός δακτύλιος διαίρεσης λέγεται* ***σώμα*** (field).

Παραδείγματα:
Γ*νωστά παραδείγματα σωμάτων είναι τα σύνολα* (*με άπειρο πλήθος στοιχείων*) $\mathbb{Q}$, $\mathbb{R}$, $\mathbb{C}$ *με τις γνωστές πράξεις πρόσθεσης και πολλαπλασιασμού*. Ι*σχύει* $\mathbb{Q} \subseteq \mathbb{R} \subseteq \mathbb{C}$.

Ο *αριθμός των στοιχείων σε ένα σώμα* (ή *δακτύλιο*) *λέγεται η* ***τάξη*** *του* (order). Τ*α σώματα στο προηγούμενο παράδειγμα έχουν άπειρη τάξη*. Υ*πάρχουν όμως και σώματα με πεπερασμένη τάξη*. Α*νάμεσα σε αυτά είναι τα σώματα* $\mathbb{Z}_p$, *όπου p είναι πρώτος*. Η *ύπαρξη αντιστρόφου για κάθε μη μηδενικό στοιχείο* $a \in \mathbb{Z}_p$, $0 \leq a < p$, *αποδεικνύεται με την βοήθεια του επεκταμένου* Ε*υκλείδειου αλγόριθμου* (extended Euclidean algorithm, *συζήτηση περί αυτού στα επόμενα*) *που υπολογίζει* $s, t \in \mathbb{Z}$ *έτσι ώστε* $1 = sa + tp \equiv sa \bmod p$.

Αξίζει να τονίσουμε πως ένας δακτύλιος R με πολλαπλασιασμό δεν είναι οπωσδήποτε ομάδα. Και τούτο διότι το πολλαπλασιαστικό αντίστροφο ίσως να μην υπάρχει για μερικά από τα στοιχεία του R. Αντίθετα, ένα σώμα R είναι ομάδα τόσο με την πρόσθεση όσο και με τον πολλαπλασιασμό.

Ορισμός:

Έστω R δακτύλιος.

1. Ένα μη μηδενικό στοιχείο $a \in R$ λέγεται ***μηδενοδιαιρέτης*** (zero divisor) αν υπάρχει μη μηδενικό στοιχείο $b \in R$ έτσι ώστε $ab = 0$ ή $ba = 0$. (Το 0 είναι μηδενοδιαιρέτης κάθε δακτυλίου.)

2. Υποθέτουμε πως το R έχει ένα ταυτοτικό στοιχείο $1 \neq 0$. Ένα στοιχείο $u \in R$ λέγεται ***μονάδα*** (unit) στο R αν υπάρχει κάποιο $v \in R$ έτσι ώστε $uv = vu = 1$ (δηλαδή $v = u^{-1}$). Το σύνολο των μονάδων στο R συμβολίζεται με $R^\times$, και είναι ομάδα ως προς τον πολλαπλασιασμό. (Οι μονάδες είναι εκείνα τα στοιχεία που έχουν *νορμ* 1.)

3. Μπορούμε τώρα να πούμε πως ***σώμα*** είναι ένας μεταθετικός δακτύλιος F με ταυτοτικό στοιχείο $1 \neq 0$ στον οποίο κάθε μη μηδενικό στοιχείο είναι μονάδα, δηλαδή $F^\times = F\backslash\{0\}$. (Το σώμα είναι η "μικρότερη" μαθηματική δομή μέσα στην οποία μπορούμε να εκτελέσουμε $+$, $-$, $\times$ και $\div$ με μη μηδενικά στοιχεία.)

***Άσκηση*:**

Με την εις άτοπο απαγωγή αποδείξτε πως ένας μηδενοδιαιρέτης ποτέ δεν μπορεί να είναι μονάδα.

Παραδείγματα:

1. Ο δακτύλιος $\mathbb{Z}$ των ακεραίων δεν έχει μηδενοδιαιρέτες και οι μονοι ακέραιοι που είναι μονάδες είναι ± 1, δηλαδή $\mathbb{Z}^\times = \{+1, -1\}$.

2. Για $n \geq 2$ στο σύνολο $\mathbb{Z}/n\mathbb{Z}$ τα στοιχεία $\overline{u}$ για τα οποία u και n είναι πρώτοι μεταξύ τους είναι μονάδες. Έτσι για τον $\mathbb{Z}/6\mathbb{Z} = \{\overline{0}, \overline{1}, \overline{2}, \overline{3}, \overline{4}, \overline{5}\}$ οι μονάδες είναι $(\mathbb{Z}/6\mathbb{Z})^\times = \{\overline{1}, \overline{5}\}$. Τα υπόλοιπα στοιχεία του είναι μηδενοδιαιρέτες διότι $\overline{2}\cdot\overline{3} = \overline{6} = \overline{0}$ και $\overline{3}\cdot\overline{4} = \overline{12} = \overline{0}$.

Οι *δακτύλιοι που έχουν τα ίδια χαρακτηριστικά με τους ακεραίους έχουν ξεχωριστή ονομασία*:

Ορισμός:
Ένας *μεταθετικός δακτύλιος με ταυτοτικό στοιχείο* $1 \neq 0$ *λέγεται* ***ακέραια περιοχή*** (integral domain) *αν δεν έχει μηδενοδιαιρέτες*.

Η *απουσία μηδενοδιαιρετών από έναν δακτύλιο* R "*ενεργοποιεί*" *κατά κάποιον τρόπο την ιδιότητα της* ***απολοιφής*** (cancellation). *Έτσι*, *αν* $a, b, c \in R$, R *ακέραια περιοχή και* $ab = ac$ *τότε* ή $a = 0$, ή $b = c$ (*επειδή* $ab = ac \Rightarrow a(b - c) = 0$ *και* R *είναι ακέραια περιοχή*).

Θεώρημα:
Κάθε ***πεπερασμένη*** *ακεραία περιοχή* R *είναι* ***σώμα*** (finite field).

Απόδειξη: *Έστω* $\{r_1, r_2, \ldots, r_n\}$ *τα μη μηδενικά στοιχεία του* R. *Για τυχόν μη μηδενικό* $a \in R$ *θεωρούμε το* $\{ar_1, ar_2, \ldots, ar_n\}$. *Επειδή ο* R *είναι ακεραία περιοχή κάθε* $ar_i \neq 0$. *Επιπλέον*, $ar_i \neq ar_j$ *για* $i \neq j$, *αλλιώς θα είχαμε* $r_i = r_j$ *λόγω της απαλοιφής που ισχύει*. *Άρα η απεικόνιση* $R \ni r \to a\,r \in R$ *είναι μονοσήμαντη και επί*. *Συνεπώς υπάρχει κάποιο* $r_k \in R$ *ώστε* $ar_k = 1$ *που σημαίνει πως το* a *είναι μονάδα*. //

Παράδειγμα:
Θεωρούμε *τα σύνολα* $\mathbb{Z}/p\mathbb{Z}$, *όπου* p *πρώτος αριθμός*, *και έστω το* $\mathbb{Z}/5\mathbb{Z} = \{\overline{0}, \overline{1}, \overline{2}, \overline{3}, \overline{4}\}$. *Αυτό είναι ένα πεπερασμένο σώμα γιατί όλα τα μη μηδενικά στοιχεία του έχουν αντίστροφο* (*βρήτε τους*).

Ορισμός:
Ως ***χαρακτηριστική*** (characteristic) char(F) *ενός σώματος* F (ή *ενός δακτυλίου*) *ορίζεται ο ελάχιστος αριθμός φορών που πρέπει να προστεθεί το ταυτοτικό στοιχείο στον εαυτό του για να προκύψει* 0.

Παράδειγμα:

Η χαρακτηριστική του σώματος $\mathbb{Z}/p\mathbb{Z}$, όπου p πρώτος αριθμός, είναι p, ενώ η χαρακτηριστική της ακεραίας περιοχής $\mathbb{Z}$ και των σωμάτων $\mathbb{Q}$, $\mathbb{R}$ και $\mathbb{C}$ είναι 0.

Ορισμός:

Ένας ***υποδακτύλιος*** (subring) του δακτυλίου R είναι μια υποομάδα του R κλειστή ως προς τον πολλαπλασιασμό.

Παράδειγμα:

Ο δακτύλιος $\mathbb{Z}$ είναι υποδακτύλιος του $\mathbb{Q}$ και ο $\mathbb{Q}$ είναι υποδακτύλιος του $\mathbb{R}$. (Η ιδιότητα "είναι υποδακτύλιος" είναι μεταβατική.)

Σε αναλογία με το κριτήριο για υποομάδες, έχουμε

Κριτήριο Υποδακτυλίου:

Για να αποδείξουμε ότι ένα υποσύνολο ενός δακτυλιδίου R είναι υποδακτύλιος αρκεί να δείξουμε πως είναι μη κενό, και κλειστό ως προς την αφαίρεση και τον πολλαπλασιασμό.

Ορισμός:

α. Ένας ***ομοιομορφισμός δακτυλίων*** (ring homomorphism) ή απλά ***ομοιομορφισμός*** από τον δακτύλιο R στον δακτύλιο S είναι μια απεικόνιση $\varphi : R \to S$ έτσι ώστε $\varphi(r_1 + r_2) = \varphi(r_1) + \varphi(r_2)$, $\varphi(r_1 r_2) = \varphi(r_1)\varphi(r_2)$ $\forall r_1, r_2 \in R$, $\varphi(0_R) = 0_S$ και $\varphi(1_R) = 1_S$.

β. Ο ***πυρήνας*** (kernel) του ομοιομορφισμού δακτυλίων φ, είναι το σύνολο $\ker(\varphi) = \{r \in R \mid \varphi(r) = 0_S\}$, υποδακτύλιος του R (δηλαδή ο πυρήνας του φ θεωρείται ομοιομορφισμός προσθετικών ομάδων). Ο πυρήνας είναι κλειστός ως προς τον πολλαπλασιασμό με στοιχεία του R.

γ. Η ***εικόνα*** (image) του ομοιομορφισμού δακτυλίων φ είναι το σύνολο $\mathrm{im}(\varphi) = \varphi(R) = \{\varphi(r) \mid r \in R\}$, υποδακτύλιος του S.

δ. Ένας αμφιμονοσήμαντος ομοιομορφισμός δακτυλίων λέγεται ***ισομορφισμός*** (ring isomorphism) ενώ οι δακτύλιοι R και S ***ισομορφικοί***: $R \cong S$. Οι ισομορφικοί

δακτύλιοι, όπως *και οι ισομορφικές ομάδες*, *θεωρούνται* ότι *είναι ουσιαστικά η ίδια δομή*.

Παραδείγματα:

1. Η *απεικόνιση* $\varphi : \mathbb{Z} \to \mathbb{Z}/2\mathbb{Z}$ *που ορίζεται στέλνοντας έναν άρτιο ακέραιο στο* 0 *και έναν περιττό στο* 1 *είναι ένας ομοιομορφισμός δακτυλίων*. Η *απεικόνιση είναι προσθετική και πολλαπλασιαστική διότι το άθροισμα / γινόμενο δύο περιττών ακεραίων είναι περιττό*, *ενώ το άθροισμα / γινόμενο δύο άρτιων ακεραίων είναι άρτιο* (*το γινόμενο άρτιου επι περιττό είναι επίσης άρτιο*). Ο *πυρήνας της* φ *είναι η τάξη ισοδυναμίας* $\overline{0}$, *το σύνολο των άρτιων ακεραίων*, *ενώ η τάξη ισοδυναμίας* $\overline{1}$ *είναι το σύνολο των περιττών ακεραίων*. Στο *Mathematica ο ομοιομορφισμός* φ *είναι*:

```
Clear[φ]; φ[x_Integer] := Mod[x, 2]
```

2. Οι *απεικονίσεις* $\varphi_n : \mathbb{Z} \to \mathbb{Z}$ *για* $n \in \mathbb{Z}$ *οριζόμενες από* $\varphi_n(x) = nx$ ***δεν*** *είναι εν γένει ομοιομορφισμοί δακτυλίων διότι* $\varphi_n(xy) = nxy$ *ενώ* $\varphi_n(x)\varphi_n(y) = nxny = n^2xy$. Οι $\varphi_n(x)$ *είναι ομοιομορφισμοί δακτυλίων μόνον όταν* $n^2 = n$, *που σημαίνει πως* $n = 0, 1$. Όμως, *στις προσθετικές ομάδες* $\forall n \in \mathbb{Z}$ *οι* $\varphi_n(x)$ *είναι ομοιομορφισμοί ομάδων*. Στους *δακτύλιους λοιπόν πρέπει και οι δύο πράξεις να διατηρούνται*.

Οπως *αναφέραμε στην συζήτηση περί ομάδων*, *αν* $\varphi : G \to H$ *είναι ένας ομοιομορφισμός ομάδων τότε μπορούμε να σχηματίσουμε την ομάδα πηλίκων* G/K *των σύμπλοκων της* K *στην* G, όπου $K = \ker(\varphi)$ *κανονική υποομάδα*, *και είδαμε πως* $G/\ker(\varphi) \cong \varphi(G)$. Ανάλογο *αποτέλεσμα ισχύει και για ομοιομορφισμό δακτυλίων*, όπου *τον ρόλο της κανονικής υποομάδας παίζει το* ***ιδεώδες*** (ideal) *που ορίζουμε παρακάτω*.

Έστω ένας ομοιομορφισμός δακτυλίων $\varphi : R \to S$ με πυρήνα I. Επειδή οι δακτύλιοι R και S είναι αβελιανές ομάδες ως προς την πρόσθεση, η απεικόνιση φ είναι ομοιομορφισμός ομάδων. Αν λοιπόν $R \ni r$, $\varphi(r) = a \in S$ τότε η τάξη ισοδυναμίας του a ως προς φ είναι το σύμπλοκο $r + I$. Αυτές οι ταξεις ισοδυναμίας έχουν την δομή ενός δακτυλίου ισομορφικού προς την εικόνα της φ με πράξεις που ορίζονται ως εξής:

$$(r + I) + (s + I) = (r + s) + I$$
$$(r + I) \times (s + I) = (rs) + I$$

Το σύνολο αυτών των σύμπλοκων είναι ένας δακτύλιος και λέγεται ο ***δακτύλιος πηλίκων*** (quotient ring ή factor ring) του R modulo I και συμβολίζεται με R/I, όπου $I = \ker(\varphi)$.

Ορισμός:

Το υποσύνολο I του μεταθετικού δακτυλίου R ονομάζεται ***ιδεώδες*** (ideal), χωρίς τον προσδιορισμό αριστερό ή δεξιό (λόγω της μεταθετικότητας του R), αν ικανοποιεί τις συνθήκες:

- $\forall a,b \in I \quad a + b \in I$
- $\forall a \in I, r \in R \quad ar \in I.$

Παράδειγμα:

Για κάθε $n \in \mathbb{Z}$, το σύνολο $n\mathbb{Z} = \{nr, r \in \mathbb{Z}\}$ είναι ένα ιδεώδες του $\mathbb{Z}$ και αυτά είναι τα μόνα ιδεώδη του $\mathbb{Z}$ επειδή είναι και οι μόνες υποομάδες του $\mathbb{Z}$. Ο αντίστοιχος δακτύλιος πηλίκων είναι $\mathbb{Z}/n\mathbb{Z}$ (που εξηγεί και τον συμβολισμό των ακεραίων modulo n). Για $n = 6$, $6\mathbb{Z} = \{6r, r \in \mathbb{Z}\} = \{\ldots, -12, -6, 0, 6, 12, \ldots\}$ και τα στοιχεία του $\mathbb{Z}/6\mathbb{Z}$ είναι τα σύμπλοκα $\{\overline{0}, \overline{1}, \ldots, \overline{5}\}$. Για να προσθέσουμε (πολλαπλασιάσουμε) στον δακτύλιο πηλίκων απλά διαλέγουμε δύο τυχαίους αντιπροσώπους των σύμπλοκων, τους προσθέτουμε (πολλαπλασιάζουμε) στους ακεραίους $\mathbb{Z}$ και για απάντηση παίρνουμε το σύμπλοκο στο οποίο ανήκει το άθροισμα (γινόμενο). Έτσι $\overline{3} + \overline{4} = \overline{7}$ και επειδή $\overline{7} = \overline{1}$ γράφουμε $\overline{3} + \overline{4} = \overline{1}$ ή $3 + 4 \equiv 1 \bmod 6$. Όμοια $\overline{3}\cdot\overline{4} = \overline{12} = \overline{0}$ ή $3\cdot 4 \equiv 0 \bmod 6$.

Η *απεικόνιση* $\mathbb{Z} \to \mathbb{Z}/n\mathbb{Z}$ *είναι ομοιομορφισμός δακτυλίων και ονομάζεται* ***υπόλοιπο*** **mod n** (reduction mod n).

Ορισμός:

Γ*ενικά όταν ένα ιδεώδες παράγεται απο τα στοιχεία* $a_1,\ldots, a_s$ *γράφουμε* $\langle a_1,\ldots, a_s\rangle = a_1R + \cdots + a_sR = \{a_1r_1 + \cdots + a_sr_s \mid r_1, \ldots, r_s \in \mathrm{R}\}$ *και λέμε ότι τα στοιχεία αυτά είναι μια* ***βάση*** (basis) *του ιδεώδους* Πολλοί *συγγραφείς χρησιμοποιούν τον συμβολισμό* $(a_1,\ldots, a_s)$. Ε*ιδική περίπτωση είναι το σύνολο* $\langle a\rangle = aR = \{ar \mid r \in \mathrm{R}\}$ *που ονομάζεται* ***κύριο*** *ιδεώδες* (principal ideal) *και είναι το αντίστοιχο των κυκλικών υποομάδων μιας ομάδας*. Ένα *στοιχείο* $b \in \mathrm{R}$ *ανήκει στο ιδεώδες* $\langle a\rangle$ *εάν και μόνον εάν* $b = ar$ *για κάποιο* $r \in \mathrm{R}$, *δηλαδή εάν και μόνον εάν το b είναι πολλαπλάσιο του a* (ή *το a διαιρεί το b στο R*). Επίσης $b \in \langle a\rangle$ *εάν και μόνον εάν* $\langle b\rangle \subseteq \langle a\rangle$.

Παράδειγμα:

Γ*ια κάθε n στο* $\mathbb{Z}$ *έχουμε* $n\mathbb{Z} = \mathbb{Z}n = \langle n\rangle = \langle -n\rangle$, *όλα κύρια ιδεώδη του* $\mathbb{Z}$. Δ*εν υπάρχουν άλλα ιδεώδη στο* $\mathbb{Z}$. Γ*ια θετικούς ακεραίους m και n,* $\langle n\rangle = n\mathbb{Z} \subseteq m\mathbb{Z} = (m)$ *εάν και μόνον εάν το m διαιρεί το n στον* $\mathbb{Z}$. Επιπλέον, *το ιδεώδες που παράγεται από δύο μη μηδενικούς ακεραίους m και n είναι το κύριο ιδεώδες που παράγεται από τον* ***μέγιστο κοινό διαιρέτη*** *τους d*: $\langle m, n\rangle = \langle d\rangle$. Δ*ύο αριθμοί είναι πρώτοι μεταξύ τους αν* $\langle m, n\rangle = \langle 1\rangle$.

Π*ροσέξτε πως ο παλαιότερος συμβολισμός για τα ιδεώδη* $(m, n) = (d)$ *ταυτίζεται με τον παλαιότερο συμβολισμό για τον μέγιστο κοινό διαιρέτη* (m, n) *των m, n.* Σ*ε αντίθεση, ο σύγχρονος συμβολισμός για τον μέγιστο κοινό διαιρέτη είναι* $\gcd(m, n)$.

Έ*στω ότι* $I \subseteq R$ *είναι ένα ιδεώδες, και* $r, s \in R$. Α*ν* $r - s \in I$ *λέμε πως τα* r *και* s *είναι* ***ισοδύναμα*** **modulo *I*** (congruent modulo I) *και γράφουμε* $r \equiv s \bmod I$. Έ*τσι για* $R = \mathbb{Z}$, *και* $I = 6\mathbb{Z}$, *έχουμε* $9 \equiv 3 \bmod 6\mathbb{Z}$ ή *απλούστερα* $9 \equiv 3 \bmod 6$. Α*ν* $a, b \in \mathbb{Z}$, *γράφουμε* $a \mid b$ ("*το a διαιρεί το b*") *αν* $\exists r \in \mathbb{Z}$ *ώστε* $ar = b$, *αλλιώς* $a \nmid b$. (Μ*ερικοί συγγραφείς συμβολίζουν το "a διαιρεί το b" με* $a \setminus b$.)

Γ*ια* $r \in R$, *το σύνολο* $r \bmod I = r + I = \{r + a \mid a \in I\} \subseteq R$ *είναι η* ***τάξη υπολοίπων*** **modulo *I*** (residue class modulo I) ή *το σύμπλοκο του ιδεώδους I*. Π*ροσοχή στην*

διαφορά μεταξύ της σχέσης ισοδυναμίας modulo I, όπως $9 \equiv 3 \bmod 6\mathbb{Z}$, που το "mod" ανήκει στο ≡, και της τάξης υπολοίπων modulo I, όπως $3 \bmod 6\mathbb{Z}$. Έχουμε τις εξής σχέσεις ισοδυναμίας:

$$r \bmod I = s \bmod I \Leftrightarrow r - s \in I \Leftrightarrow r \equiv s \bmod I$$

$\forall r, s \in R$. Το σύνολο $R/I = \{r \bmod I \mid r \in R\}$ όλων των τάξεων υπολοίπων modulo I είναι πάλι δακτύλιος, με πράξεις $(r \bmod I) + (s \bmod I) = (r + s) \bmod I$ και $(r \bmod I)\cdot(s \bmod I) = (rs) \bmod I$, και ονομάζεται δακτύλιος ***τάξης υπολοίπων του*** R **modulo** I (residue class ring).

Ο ***κανονικός ομοιομορφισμός δακτυλίων*** (canonical ring homomorphism) $\varphi : R \to R/I$ απεικονίζει $\forall r \in R$ στην τάξη υπολοίπων $r \bmod I$. Για $R = \mathbb{Z}$, και $I = 6\mathbb{Z}$ η τάξη υπολοίπων γίνεται $R/I = \mathbb{Z}/6\mathbb{Z} = \{0 \bmod 6\mathbb{Z}, 1 \bmod 6\mathbb{Z}, \ldots, 5 \bmod 6\mathbb{Z}\}$ και $\varphi(9) = 9 \bmod 6\mathbb{Z} = 3 \bmod 6\mathbb{Z}$. Πιο απλά θα γράφουμε $\mathbb{Z}/6\mathbb{Z} = \{0, 1, \ldots, 5\} = \mathbb{Z}_6$.

Όπως και στις ομάδες, υπάρχει και στους δακτυλίους το ***θεώρημα ομοιομορφισμού***. Αν R και S είναι δακτύλιοι, $\varphi : R \to S$ είναι ομοιομορφισμός και $I = \ker(\varphi) = \{r \in R \mid \varphi(r) = 0_S\} \subseteq R$ ο πυρήνας του φ, τότε το I είναι ιδεώδες και $R/\ker(\varphi) \cong \operatorname{im}(\varphi)$ όπου $\operatorname{im}(\varphi) = \varphi(R) = \{\varphi(r) \mid r \in R\} \subseteq S$ είναι η εικόνα του φ.

Για τον δακτύλιο των ακεραίων έχουμε ακόμα δύο ενδιαφέρουσες ιδιότητες.

- Διαίρεση: $\forall a, b \in \mathbb{Z}, b \neq 0, \exists! q, r \in \mathbb{Z}$ (μοναδικά!) ώστε $a = qb + r$ και $0 \le r < |b|$.
- Παραγοντοποίηση (unique factorization): Κάθε ακέραιος μεγαλύτερος από το 1 έχει μια (βασικά) μοναδική παραγοντοποίηση σε γινόμενο πρώτων αριθμών.

Οι ιδιότητες αυτές επεκτείνονται και σε άλλους δακτύλιους. Αρχίζοντας με την διαίρεση, βλέπουμε πως χρειάζεται μια συνάρτηση που να παίζει τον ρόλο της απόλυτης τιμής στους ακεραίους, δηλαδή να "μετράει μεγέθη στον R".

Ορισμός:

Μια ακέραια περιοχή R λέγεται **Ευκλείδεια περιοχή** (Euclidean domain) αν υπάρχει μια **Ευκλείδεια συνάρτηση** (Euclidean function) ή ***συνάρτηση μεγέθους*** (size function) $\sigma : R\backslash\{0\} \to \mathbb{Z}_{\ge 0}$ από τα μη μηδενικά στοιχεία του R στους μη

αρνητικούς ακεραίους, έτσι ώστε $\forall a,b \in R$, $b \neq 0$, $\exists q,r \in R$ (*όχι κατ' ανάγκη μοναδικά*) *ώστε* $a = qb + r$ *και είτε* $r = 0$ *είτε* $\sigma(r) < \sigma(b)$. *Το* q *λέγεται* ***πηλίκο*** *και το* r ***υπόλοιπο***. (*Ανάλογα με την περίσταση η Ευκλείδια συνάρτηση* σ *θα παρίσταται και με διαφορετικά γράμματα, όπως με το γράμμα* n *παρακάτω.*)

Παραδείγματα:

1. *Κάθε σώμα είναι Ευκλείδεια περιοχή με Ευκλείδεια συνάρτηση που ικανοποιεί* $\sigma(a) = 0$, *για όλα τα* a. *Αυτό ισχύει επειδή* $\forall a,b$, $b \neq 0$, *έχουμε* $a = qb + 0$, *με* $q = ab^{-1}$.

2. *Οι ακέραιοι* $\mathbb{Z}$ *είναι Ευκλείδεια περιοχή με* $\sigma(a) = |a|$, *την γνωστή απόλυτη τιμή. Η μοναδικότητα του πηλίκου και υπολοίπου εξασφαλίζεται από την έκφραση* $0 \leq r < |b|$.

Αν $b>0$, *το* a *δεν είναι πολλαπλάσιο του* b *και απαιτούμε απλά* $r < b$, *τότε υπάρχουν πάντα 2 επιλογές για τα* q, r: $5 = 3\cdot2 - 1$ *και* $5 = 2\cdot2 + 1$. (*Τα πηλίκα διαφέρουν κατά* 1 *και τα υπόλοιπα κατά* $b = 2$.)

Αν $b<0$, *το* a *δεν είναι πολλαπλάσιο του* b *και απαιτούμε απλά* $r < b$, *τότε υπάρχουν άπειρα* q, r, *όπως φαίνεται διαιρώντας το* 5 *με το* -2. *Στην περίπτωση αυτή έχουμε* $r < -2$ *οπότε* $5 = (-4)\cdot(-2) - 3$ *αλλά και* $5 = (-5)\cdot(-2) - 5$ *και* $5 = (-6)\cdot(-2) - 7$, *κοκ.*

Ένας ***μέγιστος κοινός διαιρέτης*** ή *μκδ*, (greatest common divisor or gcd) *δύο στοιχείων* a, b *σε μια ακέραια περιοχή* R *είναι ένα στοιχείο* $c \in R$ *τέτοιο ώστε* $c \mid a$ *και* $c \mid b$ *και κάθε άλλος κοινός διαιρέτης των* a, b *διαιρεί το* c. *Όπως έχουμε δει ο συμβολισμός είναι* $c = (a, b)$ ή $c = \gcd(a, b)$. *Δύο στοιχεία* a, $b \in R$ *είναι* ***πρώτα μεταξύ*** *τους* (relatively prime) *αν* $\gcd(a, b) = 1$ *και συμβολίζονται* $a \perp b$. *Οι μέγιστοι κοινοί διαιρέτες δεν είναι μοναδικοί. Στους ακέραιους,* ±2 *είναι μκδ των* 6 *και* 8. *Όπως θα δούμε, ο Ευκλείδειος αλγόριθμος και ο Κινεζικός αλγόριθμος υπολοίπων δουλεύουν μέσα σε Ευκλείδειες περιοχές που περιέχονται στις ακέραιες περιοχές.*

Αξίζει να σημειωθεί πως σε μια Ευκλείδεια περιοχή κάθε ιδεώδες I είναι *κύριο ιδεώδες*, δηλαδή $I = \langle d \rangle$, όπου d είναι ένα μη μηδενικό στοιχείο του I με την μικρότερη τιμή της Ευκλείδειας συνάρτησης.

Στην συνέχεια, όσον αφορά την παραγοντοποίηση χρειαζόμαστε ακόμα μερικές έννοιες για την ακέραια περιοχή R—εκτός από την *μονάδα* που έχουμε ήδη ορίσει.

Ένα στοιχείο $a \in R$ λέγεται **εταιρικό** (associate) του στοιχείου $b \in R$, αν υπάρχει μια μονάδα u έτσι ώστε $a = ub$ και συμβολίζεται με $a \sim b$. Η σχέση αυτή είναι σχέση *ισοδυναμίας*. Για παράδειγμα, στο $\mathbb{Z}$ με μονάδες $\{+1, -1\}$ τα στοιχεία $+2$ και -2 είναι εταιρικά. Αν ορίσουμε μια *νορμ* $n : R \to \mathbb{Z}$, τότε για τα εταιρικά στοιχεία $a \sim b$ ισχύει: $n(a) = n(b)$, δηλαδή τα εταιρικά στοιχεία έχουν την ίδια νορμ. Στο $\mathbb{Z}$, με $n(\cdot) = |\cdot|$, την απόλυτη τιμή, έχουμε $|+2| = |-2|$.

Ορισμός:
Ένα μη μηδενικό στοιχείο p μιας ακέραιας περιοχής R, λέγεται ***παραγοντοποιήσιμο*** (reducible) αν υπάρχουν μη μονάδες $a, b \in R$ έτσι ώστε $p = ab$ αλλιώς το λέγεται ***μη παραγοντοποιήσιμο*** (irreducible). Ένα μη μηδενικό στοιχείο $p \in R$, p διάφορο της μονάδας, λέγεται ***πρώτο*** (prime) αν $p \mid (ab)$ συνεπάγεται $p \mid a$ ή $p \mid b$ $\forall a,b \in R$. Η ακέραια περιοχή R λέγεται ***περιοχή μοναδικής παραγοντοποίησης*** (unique factorization domain, ufd) αν κάθε μη μηδενικό στοιχείο και διάφορο της μονάδας μπορεί να γραφτεί σαν το γινόμενο μη παραγοντοποιήσιμων στοιχείων κατά έναν και μοναδικό τρόπο, εκτός από την σειρά των στοιχείων και τον πολλαπλασιασμό τους με μονάδες.

Παράδειγμα:
Το $R = \mathbb{Z}$ είναι μια περιοχή μοναδικής παραγοντοποίησης και $30 = 2\cdot3\cdot5 = 3\cdot5\cdot2 = (-2)\cdot3\cdot(-5)$.

Προσοχή:
Μέγιστοι κοινοί διαιρέτες (μκδ) δεν υπάρχουν κατ' ανάγκη σε όλους τους δακτύλιους. Πάντα όμως υπάρχουν σε περιοχές μοναδικής παραγοντοποίησης. Ομοίως οι έννοιες "πρώτο" και "μη παραγοντοποιήσιμο" δεν είναι κατ' ανάγκη

ισοδύναμες για κάθε δακτύλιο. Σε ακέραιες περιοχές τα πρώτα στοιχεία είναι πάντα μη παραγοντοποιήσιμα το αντίστροφο όμως ισχύει μόνο για δακτύλιους με μκδ, όπως οι περιοχές μοναδικής παραγοντοποίησης. Χρήσιμα παραδείγματα παρέχονται από τους δακτύλιους αλγεβρικών ακεραίων.

Ορισμός:
Ένα στοιχείο $a \in \mathbb{C}$ λέγεται ***αλγεβρικός ακέραιος*** (algebraic integer) *αν είναι η ρίζα ενός πολυωνύμου με συντελεστές από τον* $\mathbb{Z}$ *και μοναδιαίο κύριο συντελεστή.*

Οι *αλγεβρικοί ακέραιοι είναι γενικεύσεις των συνήθων ακεραίων* $\mathbb{Z}$ *που είναι ρίζες πρωτοβάθμιων πολυωνύμων με συντελεστές από το* $\mathbb{Z}$ *και παρέχουν χρήσιμα παραδείγματα.*

Θεωρούμε *τα "φανταστικά τετραγωνικά" σώματα* (imaginary quadratic fields) $\mathbb{Q}(\sqrt{d}) = \mathbb{Q} + \mathbb{Q}\sqrt{d} = \{b + c\sqrt{d} \mid b,c \in \mathbb{Q}\} \subseteq \mathbb{C}$, *δηλαδή τα σώματα των ρητών αριθμών μαζί με το* $\sqrt{d}$, *όπου* $d \in \mathbb{Z}$ *είναι αρνητικός και έχει μόνον απλούς παράγοντες* (squarefree). *Τότε* $R = \mathcal{O}_d$, *ο* ***δακτύλιος των αλγεβρικών ακεραίων*** *στο* $\mathbb{Q}(\sqrt{d})$, *ισούται με*

$$\mathcal{O}_d = \mathbb{Z}[\sqrt{d}\,] = \mathbb{Z} + \mathbb{Z}\sqrt{d} \quad \text{για } d \equiv 2, 3 \bmod 4$$

ή

$$\mathcal{O}_d = \mathbb{Z}[\tfrac{1+\sqrt{d}}{2}\,] = \mathbb{Z} + \mathbb{Z}\tfrac{1+\sqrt{d}}{2} \quad \text{για } d \equiv 1 \bmod 4$$

Ο *δακτύλιος των* ***μιγαδικών ακεραίων*** (Gaussian integers) $R = \mathbb{Z}[i]$ *είναι ειδική περίπτωση με* $d = -1$. *Στον δακτύλιο αυτό έχουμε την νορμ* $n : R \to \mathbb{Z}$ *που ορίζεται με* $n(a) = a\overline{a} = |a|^2 = b^2 + c^2$, *όπου* $\overline{a}$ *είναι ο συζυγής μιγαδικός του* $a = b + ci$, *με* $b, c \in R$.

Ο Gauss *απέδειξε πως ο δακτύλιος* $\mathcal{O}_d$ *είναι περιοχή μοναδικής παραγοντοποίησης για* $d = -1, -2, -3, -7, -11, -19, -43, -67, -163$, *και υπέθεσε πως δεν υπάρχουν άλλες τιμές. Η υπόθεση απεδείχθη* 150 *χρόνια αργότερα.*

Επίσης είναι γνωστό πως μόνο για τις τιμές d = -1, -2, -3, -7, -11, ο δακτύλιος $\mathcal{O}_d$ είναι Ευκλείδεια περιοχή. Δηλαδή ο δακτύλιος $\mathcal{O}_{-19}$ είναι περιοχή μοναδικής παραγοντοποίησης αλλά δεν είναι Ευκλείδεια περιοχή.

Παράδειγμα (κλασσικό 1863, του Dirichlet):
Ο δακτύλιος των αλγεβρικών ακεραίων $\mathcal{O}_{-5}$ δεν είναι περιοχή μοναδικής παραγοντοποίησης. Σε αυτόν τον δακτύλιο έχουμε

$$(1 + \sqrt{-5})\cdot(1 - \sqrt{-5}) = 6 = 2\cdot 3$$

δηλαδή έχουμε δύο διαφορετικές παραγοντοποιήσεις του 6, όπου τα στοιχεία $1 \pm \sqrt{-5}$, 2 και 3 είναι μη παραγοντοποιήσιμα (irreducible) και τα $1 \pm \sqrt{-5}$ δεν είναι εταιρικά με τα 2 ή 3.

Απόδειξη: Έστω ότι το στοιχείο $1 + \sqrt{-5} = bc$, για κάποια $b, c \in R$. Τότε από την πολλαπλασιαστικότητα της νορμ έχουμε $6 = n(1 + \sqrt{-5}) = n(b)n(c)$. Οι μονάδες του $\mathcal{O}_{-5}$ είναι τα στοιχεία εκείνα με νορμ 1, δηλαδή +1, -1 $\in \mathcal{O}_{-5}$. Αλλά $n(\alpha + \beta\sqrt{-5}) = \alpha^2 + 5\beta^2$ που είναι ισοδύναμο με 0, 1, ή 4 mod 5 $\forall \alpha,\beta \in \mathbb{Z}$. Έτσι $n(b)$ δεν μπορεί ποτέ να ισούται με 2 ή 3 και έτσι ή το b ή το c είναι μονάδαν και το $1 + \sqrt{-5}$ δεν παραγοντοποιείται.
Για να δείξουμε πως τα $1 \pm \sqrt{-5}$ δεν είναι εταιρικά με τα 2 ή 3 παρατηρούμε πως οι νορμ $n(2) = 4$ και $n(3) = 9$ είναι διαφορετικές από την νορμ $n(1 + \sqrt{-5}) = 6$ και όπως ξέρουμε τα εταιρικά στοιχεία έχουν την ίδια νορμ. //

Το παραπάνω παράδειγμα δείχνει πως τα μη παραγοντοποιήσιμα στοιχεία δεν είναι κατ' ανάγκη πρώτα σε όλες τις περιοχές. Στον δακτύλιο $\mathcal{O}_{-5}$ το 2 διαιρεί το $(1 + \sqrt{-5})\cdot(1 - \sqrt{-5})$, αλλά δεν είναι πρώτο στοιχείο επειδή δεν διαιρεί κανέναν παράγοντα. Επιπλέον τα στοιχεία 6 και $2 + 2\sqrt{-5}$ δεν έχουν μκδ.

Συνοψίζοντας έχουμε την εξής διάταξη γνήσιας περιεκτικότητας δακτυλίων:

σώματα $\subset$ Ευκλείδειες περιοχές $\subset$ περιοχές μοναδικής παραγοντοποίησης $\subset$ ακέραιες περιοχές $\subset$ μεταθετικοί δακτύλιοι $\subset$ δακτύλιοι,

όπου κάθε σύνολο γνησίως περιέχεται στο επόμενό του. Επιπλέον *ισχύουν και οι εξής ισοδυναμίες για μία ακεραία περιοχή R*:

α. *R είναι περιοχή μοναδικής παραγοντοποίησης*.

β. Κάθε *μη μηδενικό και μη μοναδιαίο στοιχείο στο R μπορεί να γραφτεί σαν γινόμενο πρώτων* (prime) *στοιχείων*.

γ. Κάθε *μη μηδενικό και μη μοναδιαίο στοιχείο στο R μπορεί να γραφτεί σαν γινόμενο μη παραγοντοποιήσιμων* (irreducible) *στοιχείων και κάθε μη παραγοντοποιήσιμο στοιχείο στο R είναι πρώτο*.

δ. Κάθε *δύο μη μηδενικά στοιχεία στο R έχουν μκδ στο R*.

- **Γ3: Δακτύλιοι πολυωνύμων και επεκτάσεις σωμάτων**

Έστω ότι R είναι ένας μεταθετικός δακτύλιος με ταυτοτικό στοιχείο. Ο ***δακτύλιος των πολυωνύμων*** μιας μεταβλητής με συντελεστές από το R (ring of polynomials over R) είναι το σύνολο S των διανυσμάτων $a = \{a_0, a_1, \ldots\}$, $a_i \in R$, όπου μόνον ένας πεπερασμένος αριθμός από τα a_i είναι διάφορα του μηδενός. Η πρόσθεση και ο πολλαπλασιασμός πολυωνύμων ορίζεται αντίστοιχα ως $\{a_0, a_1, \ldots\}+\{b_0, b_1, \ldots\} = \{a_0 + b_0, a_1 + b_1, \ldots\}$ και $\{a_0, a_1, \ldots\}\cdot\{b_0, b_1, \ldots\} = \{c_0, c_1, \ldots\}$, όπου $c_n = \sum_{i=0}^{n} a_i\, b_{n-i}$.

Ο ***βαθμός***, $\deg(a)$, ενός μη μηδενικού πολυωνύμου a, είναι ο μεγαλύτερος ακέραιος n έτσι ώστε $a_n \neq 0$, και $a_n = \mathrm{lc}(a)$ είναι ο ***κύριος συντελεστής*** του (leading coefficient). Αν $\mathrm{lc}(a) = 1$, τότε λέμε πως το πολυώνυμο έχει μοναδιαίο κύριο συντελεστή (είναι **monic**). Το πολυώνυμο $\{a_0, a_1, \ldots\}$ παρίσταται ως $a_n x^n + a_{n-1} x^{n-1} + \cdots + a_0$, όπου $a_k = 0$ για όλα $k > n$, όπου η μεταβλητή x χρησιμοποιείται απλά για να δηλώνει την θέση των συντελεστών. Το σύνολο S παρίσταται ως $R[x]$ και είναι επίσης μεταθετικός δακτύλιος (όπως προκύπτει από τον ορισμό του πολλαπλασιασμού) με ταυτοτικό στοιχείο εκείνο του R. Προσέξτε πως ο δακτύλιος R εμφανίζεται στον $R[x]$ σαν τα ***σταθερά πολυώνυμα*** (βαθμού 0).

Ο δακτύλιος από τον οποίο παίρνουμε τους συντελεστές παίζει σημαντικότατο ρόλο στην συμπεριφορά των πολυωνύμων.

Παράδειγμα:
Ας θεωρήσουμε τον δακτύλιο πολυωνύμων $\mathbb{Z}/2\mathbb{Z}[x]$, δηλαδή τα πολυώνυμα με μεταβλητή x και συντελεστές στον δακτύλιο $\mathbb{Z}/2\mathbb{Z}$. Στον δακτύλιο αυτό το πολυώνυμο $x^2 + 1$ είναι παραγοντοποιήσιμο γιατί είναι τέλειο τετράγωνο: $(x + 1)^2 = x^2 + 2x + 1 = x^2 + 1$. Όμως στον δακτύλιο πολυωνύμων $\mathbb{Z}[x]$ το $x^2 + 1$ δεν είναι παραγοντοποιήσιμο.

Ο δακτύλιος των ***δυναμοσειρών*** ορίζεται κατά τον ίδιο τρόπο αλλά χωρίς τον περιορισμό "ένας πεπερασμένος αριθμός από τα a_i είναι διάφορα του μηδενός" και συμβολίζεται ως $R[[x]]$.

Τα πολυώνυμα $R[x][y]$ με δύο μεταβλητές (bivariate polynomials) μπορεί να θεωρηθούν ότι είναι πολυώνυμα της μιας μεταβλητής y με συντελεστές από $R[x]$. Αντίστοιχα, μπορεί να θεωρηθούν ότι είναι πολυώνυμα της μιας μεταβλητής x με συντελεστές από $R[y]$. Έτσι συμβολίζονται ως $R[x,y]$ κάνοντας εμφανή την συμμετρία αυτή. Πολυώνυμα με περισσότερες μεταβλητές (multivariate polynomials) συμβολίζονται ως $R[x_1,...,x_n]$.

Ο βαθμός ενός πολυώνυμου $a \neq 0$, με πολλές μεταβλητές, ως προς την μεταβλητή x_i συμβολίζεται ως $\deg_{x_i}(a)$. Ο **ολικός βαθμός** του όρου $x_1^{d_1} \cdots x_n^{d_n}$ είναι $d_1 + \cdots + d_n$ και ο ολικός βαθμός του a είναι ο μέγιστος ολικός βαθμός των όρων του.

Αν R είναι ένας μεταθετικός δακτύλιος ή μια ακέραια περιοχή, τότε το ίδιο είναι και $R[x]$ και οι μονάδες του $R[x]$ είναι αυτές του R. Το περίφημο θεώρημα του Γκάους (Gauss) μας λέει πως αν R είναι μια περιοχή μοναδικής παραγοντοποίησης, τότε το ίδιο είναι και $R[x]$. Όταν όμως R είναι μια Ευκλείδια περιοχή, $R[x]$ δεν είναι Ευκλείδια περιοχή.

Παράδειγμα:
Όπως ξέρουμε $\mathbb{Z}$ είναι Ευκλείδια περιοχή, αλλά έχουμε επίσης δει πως δεν μπορούμε να διαιρέσουμε με υπόλοιπο το $x^3 - 7x + 7$ με το $3x^2 - 7$ στο $\mathbb{Z}[x]$. Συνεπώς, $\mathbb{Z}[x]$ δεν είναι Ευκλείδια περιοχή. Γενικά στον $R[x]$, όπου R Ευκλείδια περιοχή, η διαίρεση είναι εφικτή μόνον όταν ο κύριος συντελεστής του διαιρέτη είναι μονάδα στο R (ή όταν R είναι σώμα).

Ισχύουν οι εξής ιδιότητες για πολυώνυμα με συντελεστές από την ακέραια περιοχή R με 1:
- $\forall c \in R$: έχουμε $a(c) = 0$ εάν και μόνον εάν $(x - c) \mid a$.
- αν $a \neq 0$, τότε a έχει το πολύ $\deg(a)$ ρίζες στο R.

Η δεύτερη ιδιότητα δεν ισχύει σε όλους τους δακτύλιους. Έτσι για παράδειγμα το πολυώνυμο πρώτου βαθμού $a = x \in \mathbb{Z}/4\mathbb{Z}[x]$ έχει δύο ρίζες 0 και 4.

Για τυχόν $m \in \mathbb{Z}$, ο κανονικός ομοιομορφισμός δακτυλίων (canonical ring homomorphism) $\varphi : \mathbb{Z} \to \mathbb{Z}/m\mathbb{Z}$ μπορεί να εφαρμοσθεί σε κάθε έναν από τους συντελεστές

ενός πολυωνύμου. Αυτό συνεπάγεται τον αντίστοιχο ομοιομορφισμό $\varphi : \mathbb{Z}[x] \to \mathbb{Z}/m\mathbb{Z}[x]$ που συμβολίζεται επίσης με φ. Ο πυρήνας αυτού του ομοιομορφισμού είναι $\ker(\varphi) = m\mathbb{Z}[x]$, το ιδεώδες των πολυωνύμων με όλους τους συντελεστές πολλαπλάσια του m.

Για τα πολυώνυμα υπάρχει και ο **ομοιομορφισμός διατίμησης ή εκτίμησης** (evaluation homomorphism) $\varepsilon : R[x] \to R$, που για κάθε σταθερό στοιχείο του δακτυλίου, $c \in R$ και $R[x] \ni f$ μας δίνει: $\varepsilon(f) = f(c) \in \mathrm{R}$, την διατίμηση στο c (την τιμή του f στο c). Ο πυρήνας αυτού του ομοιομορφισμού είναι $\ker(\varepsilon) = \{R[x] \ni f \mid f(c) = 0\}$. Το σύνολο αυτό όμως, βάσει των ιδιοτήτων που προαναφέραμε, είναι το ιδεώδες $\langle x - c\rangle$, και έτσι $\ker(\varepsilon) = \langle x - c\rangle$. Από το θεώρημα ομοιομορφισμού δακτυλίων έχουμε $R[x]/\ker(\varepsilon) \cong \mathrm{im}(\varepsilon)$. Εδώ όμως για την εικόνα της ε ισχύει $\mathrm{im}(\varepsilon) = R$ διότι η ε είναι μονοσήμαντη και επί αφού για κάθε $\rho \in R$ το σταθερό πολυώνυμο $f(x) = \rho$ παίρνει την τιμή ρ με διατίμηση στο c. Άρα $R[x]/\langle x - c\rangle \cong R$.

Ο ομοιομορφισμός διατίμησης είναι ειδική περίπτωση (για $m = x - c$) του κανονικού ομοιομορφισμού $R[x] \to R[x]/\langle m\rangle$, για $\forall m \in R[x]$. Στην γενική περίπτωση, αν m είναι μη σταθερό πολυώνυμο με κύριο συντελεστή 1, τότε τα πολυώνυμα $f \in R[x]$ βαθμού μικρότερου του $\deg(m)$ αποτελούν ένα σύστημα **αντιπροσώπων** για το ιδεώδες $\langle m\rangle$. Οι αντιπρόσωποι αυτοί αποτελούν τον δακτύλιο πηλίκων $R[x]/\langle m\rangle$ με πρόσθεση και πολλαπλασιασμό modulo m.

Οι ομοιομορφισμοί είναι **μεταθετικοί** με τις πολυωνυμικές εκφράσεις. Δηλαδή, γενικά αν R και S είναι δακτύλιοι, $\varphi : R \to S$ είναι ομοιομορφισμός δακτυλίων και $f \in R[x_1, \ldots, x_n]$ είναι ένα πολυώνυμο πολλών μεταβλητών τότε για τα στοιχεία $c_1, \ldots, c_n \in R$ ισχύει

$$\varphi(f(c_1, \ldots, c_n)) = \varphi(f)(\varphi(c_1), \ldots, \varphi(c_n)).$$

Παράδειγμα:

Έστω $\varphi : \mathbb{Z} \to \mathbb{Z}/6\mathbb{Z}$ ο ομοιομορφισμός δακτυλίων και $f(x, y) = 15x + 28y$ η πολυωνυμική έκφραση. Τότε $f(2, 3) = 114$, και $\varphi(f(2, 3)) = 0$. Το ίδιο αποτέλεσμα όμως παίρνουμε και εργαζόμενοι με το δεξί σκέλος της εξίσωσης $\varphi(f)(\varphi(2), \varphi(3)) = 3\varphi(2) + 4\varphi(3) = 6 + 12 = 18 = 0$ στον $\mathbb{Z}/6\mathbb{Z}$.

Στο *Mathematica* ο ομοιομορφισμός φ και η πολυωνυμική έκφραση f είναι (*αντίστοιχα*):

```
Clear[φ, f]; φ[x_Integer] := Mod[x, 6]; f[x_, y_] := 15 x + 28 y;
```

Η *έκφραση* $\varphi(f(c_1, \ldots, c_n))$ *υπολογίζεται ως*:

```
φ[ f[x, y]] /. {x → 8, y → 10}
```

```
4
```

ενώ η έκφραση $\varphi(f)(\varphi(c_1), \ldots, \varphi(c_n))$ *υπολογίζεται ως*:

```
φ[f[x, y]] /. {x → φ[8], y → φ[10]}
```

```
4
```

Το *αποτέλεσμα είναι το ίδιο και στις δύο περιπτώσεις και μας λέει πως οι ομοιομορφισμοί είναι* ***μεταθετικοί*** *και με τις τάξεις ισοδυναμίας*. Δηλαδή, *γενικά αν* I *είναι ένα ιδεώδες του* R, $\varphi : R \to S$ *είναι ομοιομορφισμός δακτυλίων και* $f \in R[x_1, \ldots, x_n]$ *είναι ένα πολυώνυμο πολλών μεταβλητών τότε για τα στοιχεία* $c_1, \ldots, c_n, \overline{c_1}, \ldots, \overline{c_n} \in R$ *που ικανοποιούν τις σχέσεις* $c_i \equiv \overline{c_i} \bmod I$ *για* $0 \le i \le n$ *ισχύει*: $f(c_1, \ldots, c_n) \equiv f(\overline{c_1}, \ldots, \overline{c_n}) \bmod I$. Αυτό *αποτελεί και την βάση της* ***αριθμητικής υπολοίπων*** (modular arithmetic) *που θα εξετάσουμε αργότερα*.

Ορισμός:
Αν R *είναι ακέραια περιοχή τότε* $F = \{\frac{a}{b} \mid a,b \in R, b \neq 0\}$ *είναι το* ***σώμα πηλίκων*** (field of fractions ή quotient field) *του* R.

Παραδείγματα:

1. Αν R είναι σώμα, τότε το σώμα πηλίκων του είναι το ίδιο το R.

2. Οι ακέραιοι $\mathbb{Z}$ είναι μια ακέραια περιοχή της οποίας το σώμα πηλίκων είναι $\mathbb{Q}$, οι κλασματικοί αριθμοί.

3. Ο υποδακτύλιος $2\mathbb{Z}$ του $\mathbb{Z}$ δεν έχει ταυτοτικό στοιχείο (ούτε και μηδενοδιαιρέτες). Το σώμα πηλίκων του είναι επίσης $\mathbb{Q}$. (Προσέξτε πως "εμφανίστηκε" το ταυτοτικό στοιχείο.)

4. Όπως έχουμε δει αν R είναι ακέραια περιοχή, το ίδιο είναι και ο δακτύλιος πολυωνύμων $R[x]$. Το σώμα πηλίκων του $R[x]$ είναι το σώμα των ***κλασματικών συναρτήσεων*** ως προς x. Τα στοιχεία αυτού του σώματος είναι της μορφής $\frac{p(x)}{q(x)}$, όπου $p(x)$ και $q(x)$ είναι πολυώνυμα με συντελεστές από το R και $q(x)$ είναι μη μηδενικό πολυώνυμο. Τα πολυώνυμα $p(x)$ και $q(x)$ μπορεί να είναι σταθερά πολυώνυμα και έτσι το σώμα των κλασματικών συναρτήσεων περιέχει το σώμα πηλίκων του R, δηλαδή το σύνολο $F = \{\frac{a}{b} \mid a,b \in R,\ b \neq 0\}$. Δηλαδή το σώμα πηλίκων του $R[x]$ είναι το ίδιο με το σώμα πηλίκων του $F[x]$ και συμβολίζεται με $F(x)$. Συνεπώς, αν $R = \mathbb{Z}$ τότε $F = \mathbb{Q}$, το σώμα πηλίκων του $\mathbb{Z}[x]$ είναι το ίδιο με το σώμα πηλίκων του $\mathbb{Q}[x]$ και συμβολίζεται με $\mathbb{Q}(x)$.

Ορισμός:

Αν ένα σώμα F περιέχεται σε κάποιο άλλο σώμα E τότε λέμε πως το E είναι το ***σώμα επέκτασης*** (extension field) του F και πως το F είναι ***υποσώμα*** (subfield) του E. Το σώμα F ονομάζεται και ***βάση*** (base field) της επέκτασης.

Παράδειγμα:

Το $\mathbb{C}$ είναι σώμα επέκτασης του $\mathbb{R}$, και το $\mathbb{R}$ είναι σώμα επέκτασης του $\mathbb{Q}$.

Ορισμός:

Ένα στοιχείο $\alpha \in E$ λέμε πως είναι ***αλγεβρικό*** (algebraic) στο F αν το α είναι ρίζα ενός μη μηδενικού πολυωνύμου $f(x) \in F[x]$ (ή αν το διανυσματικό πεδίο που παράγεται από $1, \alpha, \alpha^2, \ldots$ είναι πεπερασμένης διάστασης). Στοιχεία που δεν είναι αλγεβρικά λέγονται ***υπερβατικά*** (transcendental). Αν όλα τα στοιχεία του E

είναι αλγεβρικά στο F, τότε λέμε πως το E είναι ***αλγεβρική επέκταση*** (algebraic extension).

Παραδείγματα:

1. *Όλα τα στοιχεία ενός σώματος F είναι αλγεβρικά στο F.* Το *στοιχείο* $i = \sqrt{-1} \in \mathbb{C}$ *είναι αλγεβρικό στα* $\mathbb{Q}$ *και* $\mathbb{R}$ (*ρίζα του* $f(x) = x^2 + 1$).

2. Τα *στοιχεία* π *και* e *είναι υπερβατικά στο* $\mathbb{Q}$.

3. Το $\mathbb{C}$ *είναι αλγεβρική επέκταση του* $\mathbb{R}$ *αλλά όχι του* $\mathbb{Q}$.

Ορισμός:

Αν *το E είναι σώμα επέκτασης του F, τότε ο* ***βαθμός*** ή ***δείκτης*** *της επέκτασης* (degree ή index) $[E : F]$, *είναι η διάσταση του E σαν διανυσματικός χώρος στο F.* Δηλαδή $[E : F] = \dim_F(E)$. Η *επέκταση είναι* ***πεπερασμένη*** (finite) *αν* $[E : F]$ *είναι πεπερασμένος, αλλιώς είναι* ***άπειρη*** (infinite)—*για υπερβατικά στοιχεία.* Πεπερασμένες *επεκτάσεις είναι πάντοτε αλγεβρικές.* Για *τις πεπερασμένες επεκτάσεις* $F \subseteq E \subseteq K$, *ισχύει*: $[K : F] = [K : E]\cdot[E : F]$.

Αν $\alpha \in E$ *είναι αλγεβρικό στο F, τότε το σύνολο* $I = \{f(x) \in F[x] \mid f(\alpha) = 0\}$ *όλων των πολυωνύμων με συντελεστές από το F που έχουν το α σαν ρίζα είναι ένα ιδεώδες στο* $F[x]$. Επειδή F *είναι σώμα έπεται πως* $F[x]$ *είναι* Ευκλείδια *περιοχή.* Εμείς όμως *ξέρουμε πως στις* Ευκλείδιες *περιοχές κάθε ιδεώδες είναι κύριο.* Αυτό *συνεπάγεται πως υπάρχει ένα πολυώνυμο* m_α *με μοναδιαίο κύριο συντελεστή και με τον μικρότερο δυνατό εκθέτη ώστε* $I = \langle m_\alpha \rangle$. Το *πολυώνυμο* m_α *λέγεται το* ***ελάχιστο πολυώνυμο*** (minimal polynomial) *του α και είναι μη παραγοντοποιήσιμο* (*διότι αλλιώς θα υπήρχε πολυώνυμο μικρότερου βαθμού με ριζα το* α).

Αφού *το* m_α *παράγει το ιδεώδες I έπεται πως κάθε πολυώνυμο που έχει ρίζα το α διαιρείται από το* m_α, *που είναι και το μοναδικό πολυώνυμο με μοναδιαίο συντελεστή που έχει αυτή την ιδιότητα.* Ο *βαθμός του* m_α *είναι* $\deg(m_\alpha)$ *και αν* $F(\alpha) \subseteq E$ *είναι το μικρότερο σώμα που περιέχει το F και το α, τότε* $\deg(m_\alpha) = [F(\alpha) : F]$.

Παράδειγμα:
Αν $F = \mathbb{R}$, $E = \mathbb{C}$, $\alpha = i$ τότε $m_\alpha = m_i = x^2 + 1$, $\mathbb{R}(i) = \mathbb{C}$ και $\deg(m_i) = 2 = [\mathbb{C} : \mathbb{R}]$.

Μπορούμε να κατασκευάσουμε μια αλγεβρική επέκταση E του σώματος F παίρνοντας $E = F[x]/\langle f \rangle$ για κάποιο μη παραγοντοποιήσιμο πολυώνυμο f.

Θεώρημα (κατασκευαστικό):
Έστω το μη παραγοντοποιήσιμο πολυώνυμο βαθμού n, $f \in F[x]$, F σώμα, και θεωρούμε το σώμα πηλίκων $E = F[x]/\langle f \rangle$. Αν $\alpha = x \bmod f \in E$ (δηλαδή το α είναι ρίζα του f) τότε τα στοιχεία

$$1, \alpha, \alpha^2, \ldots, \alpha^{n-1}$$

ή ισοδύναμα

$$1 \bmod f, x \bmod f, x^2 \bmod f, \ldots, x^{n-1} \bmod f$$

είναι μια βάση του (διανυσματικού χώρου) E και $[E : F] = n$. Άρα το σώμα E, η αλγεβρική επέκταση του σώματος F, είναι το σύνολο πολυωνύμων βαθμού $< n$

$$E = \{c_0 + c_1\alpha + c_2\alpha^2 + \cdots + c_{n-1}\alpha^{n-1} \mid c_0, c_1, \ldots, c_{n-1} \in F\}$$

Απόδειξη:
Έστω $a(x) \in F[x]$ τυχόν πολυνώμιο. Επειδή $F[x]$ είναι Ευκλείδια περιοχή μπορούμε να διαιρέσουμε το $a(x)$ με το $f(x)$:

$$a(x) = f(x)q(x) + r(x), \quad q(x), r(x) \in F[x], \deg(r(x)) < n.$$

Το $f(x)q(x)$ είναι στο ιδεώδες $\langle f \rangle$ και άρα $a(x) \equiv r(x) \bmod f$. Αυτό δείχνει πως κάθε τάξη υπολοίπων (ισοδυναμίας) του $F[x]/\langle f \rangle$ παρίσταται από ένα πολυώνυμο βαθμού $< n$. Επομένως τα στοιχεία $1, \alpha, \alpha^2, \ldots, \alpha^{n-1}$ καλύπτουν τον διανυσματικό χώρο E.

Θα δείξουμε με την εις άτοπο απαγωγή πως τα στοιχεία $1, \alpha, \alpha^2, \ldots, \alpha^{n-1}$ είναι και γραμμικώς ανεξάρτητα, δηλαδή αποτελούν βάση του χώρου. Ας υποθέσουμε το αντίθετο. Τότε υπάρχει γραμμικός συνδιασμός

$$b_0 + b_1\alpha + b_2\alpha^2 + \cdots + b_{n-1}\alpha^{n-1} = 0$$

στο E με $b_0, b_1, \ldots, b_{n-1} \in F$, και μερικά από αυτα $\neq 0$. Αυτό σημαίνει πως

$$b_0 + b_1\alpha + b_2\alpha^2 + \cdots + b_{n-1}\alpha^{n-1} \equiv 0 \bmod f$$

ή ότι το $f(x)$ διαιρεί το $b_0 + b_1\alpha + b_2\alpha^2 + \cdots + b_{n-1}\alpha^{n-1}$ στο $F[x]$. Αυτό όμως είναι αδύνατο διότι το πολυώνυμο $b_0 + b_1\alpha + b_2\alpha^2 + \cdots + b_{n-1}\alpha^{n-1}$ είναι βαθμού $\leq n - 1$ ενώ το $f(x)$ βαθμού n. Άρα τα στοιχεία $1, \alpha, \alpha^2, \ldots, \alpha^{n-1}$ είναι βάση του διανυσματικού χώρου E και $[E : F] = n$. //

Παράδειγμα:

Ακολουθούμε την κατασκευή του θεωρήματος στην ειδική περίπτωση που $F = \mathbb{R}$ και $f(x) = x^2 + 1$. Τότε το σώμα $E = \mathbb{R}[x]/\langle x^2 + 1\rangle$ είναι μια επέκταση του $\mathbb{R}$ βαθμού 2, μέσα στην οποία βρίσκεται μια ρίζα του $x^2 + 1$. Τα στοιχεία του σώματος επέκτασης του $\mathbb{R}$ είναι της μορφής $a + b\alpha$ με $a,b \in \mathbb{R}$. Η πρόσθεση ορίζεται ως

$$(a + b\alpha) + (c + d\alpha) = (a + c) + (b + d)\alpha.$$

Για τον πολλαπλασιασμό χρησιμοποιούμε το γεγονός ότι $x^2 + 1 = 0$, ή το ότι $\alpha^2 = -1$ στο E. (Αλλιώς, το -1 είναι το υπόλοιπο όταν το x^2 διαιρείται με το $x^2 + 1$ στον $\mathbb{R}[x]$.) Τότε

$$\begin{aligned}(a + b\alpha)\cdot(c + d\alpha) &= ac + (ad + bc)\alpha + bd\alpha^2 \\ &= ac + (ad + bc)\alpha + bd(-1) \\ &= (ac - bd) + (ad + bc)\alpha\end{aligned}$$

Αυτοί είναι και οι τύποι πρόσθεσης και πολλαπλασιασμού στο $\mathbb{C}$. Με άλλα λόγια η απεικόνιση

$$\varphi : \mathbb{R}[x]/\langle x^2 + 1 \rangle \to \mathbb{C}$$
$$a + bx \to a + bi$$

είναι ένας ομοιομορφισμός, και επειδή είναι αμφιμονοσήμαντος είναι ισομορφισμός. Επομένως θα μπορούσαμε να είχαμε ορίσει τον $\mathbb{C}$ με τον ισομορφισμό αυτό, και βάσει του θεωρήματος $\mathbb{C}$ είναι σώμα.

Ορισμός:

Ένα σώμα E που είναι αλγεβρική επέκταση του σώματος F λέγεται ***σώμα διάσπασης*** (splitting field) ενός μη μηδενικού πολυωνύμου $f \in F[x]$ αν το f διασπάται σε γινόμενο γραμμικών παραγόντων στο E, αλλά δεν διασπάται σε κανένα γνήσιο υποσώμα του E. Ένα σώμα F λέγεται ***αλγεβρικά κλειστό*** (algebraically closed) αν και μόνο αν κάθε μη σταθερό πολυώνυμο $f \in F[x]$ έχει μια ρίζα στο F—οπότε το f έχει $\deg(f)$ ρίζες, μετρώντας πολλαπλότητες. (Το $\mathbb{C}$ έχει αυτήν την ιδιότητα.)

Ασκήσεις

1: Έστω ότι $\langle F_n \rangle$ είναι η ακολουθία των αριθμών Fibonacci που ορίζεται από τις αρχικές τιμές $F_0 = 0$, $F_1 = 1$, και την σχέση $F_n = F_{n-1} + F_{n-2}$. Δείξτε ότι κάθε θετικός ακέραιος a μπορεί να γραφτεί σαν

$$a = \sum_{n=2}^{m} a_n F_n \text{ με } a_n \in \{0, 1\}.$$

όπου ο δείκτης n αρχίζει από το 2.

α. Περιγράψτε έναν αλγόριθμο που να βρίσκει την μοναδική αυτή παράσταση του ακεραίου a. (Κάθε μοναδικό σύστημα παράστασης είναι ένα σύστημα αριθμών. Έτσι εδώ έχουμε το σύστημα αριθμών Fibonacci.) Για παράδειγμα ο αριθμός 20 στο σύστημα Fibonacci είναι: $20 = \{0, 1, 0, 1, 0, 1\}_F$ που ισοδυναμεί με $0\,F_2 + 1\,F_3 + 0\,F_4 + 1\,F_5 + 0\,F_6 + 1\,F_7 = 2 + 5 + 13$.

β. Το σύστημα αριθμών Fibonacci μοιάζει πολύ με το δυαδικό σύστημα. Η μόνη διαφορά είναι πως στο πρώτο δεν υπάρχουν ποτέ δύο γειτονικά "1". Εξηγείστε γιατί συμβαίνει αυτό.

2: Σχεδιάστε έναν αλγόριθμο (όμοιο με τον αλγόριθμο πρόσθεσης ακεραίων) για την αφαίρεση δύο ακεραίων πολλαπλής ακρίβειας a και b του ίδιου προσήμου και με $|a| > |b|$. Για τον σκοπό αυτό καθορίστε μια εντολή του επεξεργαστή (ανάλογη με αυτήν που ορίσαμε για την πρόσθεση) που εκτελεί αφαίρεση δύο ακεραίων αριθμών απλής ακριβείας. Επίσης χρησημοποιήστε την σημαία του κρατούμενου για να καταλάβετε αν το αποτέλεσμα είναι αρνητικό ή όχι.

3: Χρησιμοποιήστε την γλώσσα προγραμματισμού C και την βιβλιοθήκη GMP (από την διεύθυνση http://www.swox.com/gmp) για να προγραμματίσετε τον αλγόριθμο πρόσθεσης ακεραίων πολλαπλής ακριβείας (#include <stdio.h>, #include "gmp.h").

4: *α.* Δοθέντος ενός ακεραίου a πως θα βρείτε τα ψηφία του ως προς την βάση β; Υπενθύμιση: Στην πρόσθεση ακεραίων πολλαπλής ακριβείας χρησιμοποιήσαμε

από το Mathematica την συνάρτηση IntegerDigits[] για να βρούμε τα ψηφία του ακεραίου a ως προς την βάση β, και την συνάρτηση FromDigits[] για την αντίστροφη πράξη.

β. Χρησιμοποιήστε την συνάρτηση Fold[] του Mathematica για να βρήτε τον ακέραιο a όταν δίνονται τα ψηφία του ως προς μία βάση β. (Την συνάρτηση αυτή την χρησιμοποιήσαμε στην πρόσθεση πολυωνύμων, για ανάλογο σκοπό.)

5: Στον περιγραφή των αλγορίθμων για τον πολλαπλασιασμό και την διαίρεση δύο πολυωνύμων χρειαζόμαστε έναν άλλο αλγόριθμο που πολλαπλασιάζει ένα πολυώνυμο b με τους ακεραίους a_i ή q_i απλής ή πολλαπλής ακριβείας. Περιγράψτε έναν τέτοιο αλγόριθμο και υπολογίστε το κόστος υπολογισμού του.

6: Στον περιγραφή του αλγορίθμου για τον πολλαπλασιασμό δύο ακεραίων πολλαπλής ακριβείας χρειαζόμαστε έναν άλλο αλγόριθμο για τον πολλαπλασιασμό του ακεραίου πολλαπλής ακριβείας b επί τον ακέραιο απλής ακριβείας a_i. Ο δεύτερος αυτός πολλαπλασιασμός γίνεται με την βοήθεια ειδικής εντολής πολλαπλασιασμού ακεραίων απλής ακριβείας. Περιγράψτε έναν τέτοιο αλγόριθμο και αποδείξτε ότι χρησιμοποιεί $\lambda(a)$ πολλαπλασιασμούς απλής ακριβείας και άλλες τόσες προσθέσεις απλής ακριβείας.

7: Αποδείξτε ότι αν πολλαπλασιάσουμε τους θετικούς ακεραίους $a = \sum_{0 \le i \le n} a_i \beta^i$ και $b = \sum_{0 \le i \le n-1} b_i \beta^i$ με τον ακέραιο $k = \left\lfloor \frac{\beta}{b_{n-1}+1} \right\rfloor$ τότε το πιο σημαντικό ψηφίο του kb, έστω το b_{n-1}^k, ικανοποιεί την ανισότητα $b_{n-1}^k \ge \frac{\beta}{2}$.

8: Ποιές συναρτήσεις αυξάνονται γρηγορότερα:

α. $n^{(\ln n)}$ ή $(\ln n)^n$;

β. $(n!)!$ ή $((n\text{-}1)!)!$ ή $(n-1)!^{n!}$;

9: Πολλαπλασιάστε $(\ln n + \gamma + O(\frac{1}{n}))$ επί $(n + O(\sqrt{n}))$ και εκφράστε το αποτέλεσμα με τον O-συμβολισμό.

10: Στην άσκηση αυτή οι συναρτήσεις FindInstance[] ή Resolve[] του Mathematica (έκδοση ≥ 5.0) μπορούν να φανούν χρήσιμες.

α. Παρίσταται ο δεκαδικός αριθμός 0.1 ακριβώς στον υπολογιστή σας; Αν όχι

ποιός είναι ο πλησιέστερος αριθμός κινητής υποδιαστολής;

β. *Πως παρίστανται οι αριθμοί* $\frac{1}{2}$, $\frac{2}{3}$ *και* $\frac{3}{5}$ *στον υπολογιστή σας;*

11: *Το παρακάτω πρόγραμμα υπολογίζει το* ϵ *του υπολογιστή στον οποίο γράφτηκε το βιβλίο αυτό*

```
eps = 1; eps = 0.5 eps; epsp1 = eps + 1;
While[epsp1 - 1 > 0, eps = 0.5 eps; epsp1 = eps + 1]; eps = eps / 0.5
```

2.22045×10^{-16}

```
eps == $MachineEpsilon
```

True

Α*ν στον βρόγχο αντικαταστήσουμε το τεστ* "epsp1-1>0" *με το τεστ* "epsp1>1" *το αποτέλεσμα διαφέρει κατά ορισμένες δυνάμεις του* 2.

```
eps = 1; eps = 0.5 eps; epsp1 = eps + 1;
While[epsp1 > 1, eps = 0.5 eps; epsp1 = eps + 1]; eps = eps / 0.5
```

2.84217×10^{-14}

Α*ν και η διαφορά αυτή δεν δημιουργεί προβλήματα, είναι ενδεικτικό της αδυναμίας του Mathematica να συγκρίνει στον δεδομένο υπολογιστή δύο αριθμούς που διαφέρουν ελάχιστα μεταξύ τους—με αποτέλεσμα να καταφεύγουμε σε αφαίρεση.* Πώς *συμπεριφέρεται το Mathematica* (ή *οποιοδήποτε άλλο σύστημα διαθέτετε*) *στον υπολογιστή σας;*

Βιβλιογραφία

1. Akritas, A.G.: *Elements of Computer Algebra with Applications.* Wiley Interscience, New York, 1989.

2. Akritas, A.G.: Floating-Point Arithmetic: Precision and Accuracy with *Mathematica*. *Mathematica* World, April 1994. (http://www.inf.uth.gr/~akritas/publications (#42).)

3. Artin, M.: *Algebra*. Prentice-Hall, Englewood Cliffs, NJ, 1991.

4. Dummit, D.S. and R.M. Foote: *Abstract Algebra*. Prentice-Hall, NJ 1991.

5. Essex, C., Davison, M., and Schulzky, C.: Numerical Monsters. SIGSAM Bulletin **34** (4), 16–32, 2000.

6. Flanders, H.: *Scientific Pascal*. Reston, VA, 1984.

7. Forsythe, G.E., Malcolm, M.A., and C.B. Moler: *Computer Methods for Mathematical Computations*. Prentice-Hall, Englewood Cliffs, NJ, 1977.

8. GMP. http://www.swox.com/gmp

9. Graham, R.L., Knuth, D.E., and O. Patashnik: *Concrete Mathematics*. Addison-Wesley, New York, 1981.

10. Knuth, D.E.: *The art of computer programming*. Vol. 2: *Seminumerical Algorithms*. Addison-Wesley, New York, 1989.

11. Mignotte, M.: *Mathematics for Computer Algebra*. Springer-Verlag, New York, 1991.

Κεφάλαιο 2: Ο κλασσικός και επεκταμένος Ευκλείδειος αλγόριθμος

Όπως έχουμε δει, σε πολλές περιπτώσεις, οι ακέραιοι και τα πολυώνυμα με συντελεστές από ένα σώμα συμπεριφέρονται κατά τον ίδιο τρόπο. Συχνά οι αλγόριθμοι είναι όμοιοι και μπορεί κανείς να βρει μια γενίκευση των δύο περιοχών. Τότε αρκεί να σχεδιάσουμε έναν αλγόριθμο για την γενική περίπτωση και να λύσουμε και τα δύο προβλήματα σύμφωνα με το ρητό: "μ' έναν σμπάρο δυο τριγώνια". Στην περίπτωσή μας η Ευκλείδεια περιοχή παίζει τον ρόλο της ομπρέλας που καλύπτει τις δομικές ομοιότητες στον υπολογισμό ενός μέγιστου κοινού διαιρέτη, ή μκδ, (greatest common divisor or gcd) μεταξύ δύο ακεραίων και δύο πολυωνύμων.

Όπως γνωρίζουμε, ο (κλασσικός) Ευκλείδειος αλγόριθμος για την εύρεση του μέγιστου κοινού διαιρέτη, ή μκδ, δύο ακεραίων 612 και 342 περιλαμβάνει τα εξής βήματα:

$$612 = 1\cdot 342 + 270,$$
$$342 = 1\cdot 270 + 72$$
$$270 = 3\cdot 72 + 54,$$
$$72 = 1\cdot 54 + 18,$$
$$54 = 3\cdot 18 + 0,$$

και 18 είναι ο μκδ των 612 και 342. Μια από τις πιο σημαντικές εφαρμογές του είναι στους κλασματικούς αριθμούς όπου πρέπει να απλοποιούμε το $\frac{342}{612}$ σε $\frac{19}{34}$ για να κρατάμε τους αριθμούς όσο γίνεται μικρότερους. Ο αριθμός L των διαιρέσεων ($L = 5$ στην προκειμένη περίπτωση) λέγεται το **Ευκλείδειο μήκος** (Euclidean length) του ζεύγους $\{612, 342\}$.

Ο παραπάνω αλγόριθμος μπορεί να προσαρμοσθεί και να χρησημοποιηθεί για την εύρεση ενός μέγιστου κοινού διαιρέτη δύο πολυωνύμων.

Στα επόμενα θα παρουσιάσουμε τον Ευκλείδειο αλγόριθμο στην γενική του μορφή, στην Ευκλείδεια περιοχή R, καλύπτοντας συγχρόνως τους ακεραίους και τα πολυώνυμα. Έτσι ο δακτύλιος R μπορεί να είναι ή ο δακτύλιος των ακεραίων (με συνάρτηση μεγέθους την απόλυτη τιμή) ή ο δακτύλιος των πολυωνύμων με συντελεστές από το R (με συνάρτηση μεγέθους τον βαθμό του πολυωνύμου).

Στο *Mathematica* η συνάρτηση GCD[n_1, n_2, …], μας δίνει τον μέγιστο κοινό διαιρέτη των ακεραίων n_i ενώ η συνάρτηση PolynomialGCD[$poly_1$, $poly_2$, …] μας δίνει τον μέγιστο κοινό διαιρέτη των πολυωνύμων $poly_i$. Έτσι για το παράδειγμά μας έχουμε:

```
GCD[612, 342]

18
```

2.1 Ο κλασσικός Ευκλείδειος αλγόριθμος

Αρχικά επαναλαμβάνουμε μερικούς ορισμούς και διάφορες γνωστές έννοιες.

Ορισμός:
Μια ακέραια περιοχή R λέγεται **Ευκλείδεια περιοχή** (Euclidean domain) αν υπάρχει μια **Ευκλείδεια συνάρτηση** (Euclidean function) ή **συνάρτηση μεγέθους** (size function) $\sigma : R\backslash\{0\} \to \mathbb{Z}_{\geq 0}$ από τα μη μηδενικά στοιχεία του R στους μη αρνητικούς ακεραίους, έτσι ώστε $\forall a,b \in R$, $b \neq 0$, $\exists q,r \in R$ (όχι κατ' ανάγκη μοναδικά) ώστε $a = qb + r$ και είτε $r = 0$ είτε $\sigma(r) < \sigma(b)$. Το $q = \text{quo}(a, b)$ λέγεται **πηλίκο** (quotient) και το $r = \text{rem}(a, b)$ **υπόλοιπο** (remainder).

Προσέξτε πως αν ορίσουμε $\sigma(0) = -1$, (ή $\sigma(0) =$ με οποιονδήποτε άλλο αρνητικό αριθμό) τότε μπορούμε να πούμε απλά ότι $\sigma(r) < \sigma(b)$ χωρίς να ξεχωρίζουμε την περίπτωση $r = 0$. Επίσης, ανάλογα με την περίσταση η Ευκλείδεια συνάρτηση σ θα παρίσταται και με διαφορετικά γράμματα.

Στο παράρτημα περί δακτυλίων του πρώτου κεφαλαίου έχουμε παρουσιάσει ορισμένα παραδείγματα με διαφορετικές συναρτήσεις μεγέθους. Ακολουθούν μερικά ακόμα παραδείγματα. Βλέπε και τις ασκήσεις.

Παραδείγματα:
1. Ο δακτύλιος των μιγαδικών ακεραίων (Gaussian integers) $R = \mathbb{Z}[i]$ με συνάρτηση μεγέθους (νορμ) $n : R \to \mathbb{Z}$ που ορίζεται με $n(a) = a\overline{a} = |a|^2 = b^2 + c^2$, όπου $\overline{a}$ είναι ο συζυγής μιγαδικός του $a = b + ci$, με $i = \sqrt{-1}$ και $b, c \in R$.

2. Ο δακτύλιος $R = F[x]$, όπου F είναι σώμα με συνάρτηση μεγέθους τον βαθμό του πολυωνύμου $d(a) = \deg(a)$. Ως βαθμός του μηδενικού πολυωνύμου ορίζεται οποιοσδήποτε αρνητικός αριθμός. Όπως έχουμε δεί, όταν $\text{lc}(b)$ του b είναι μια **μονάδα** (unit) στον R, το πηλίκο και το υπόλοιπο υπολογίζονται μονοσήμαντα (αποδείξτε το).

Ορισμός:

Έστω μία Ευκλείδεια περιοχή R και $a,b,c \in R$. Τότε το c είναι ένας ***μέγιστος κοινός διαιρέτης*** ή *μκδ* (greatest common divisor or gcd) *των* a *και* b *αν*:

α. $c \mid a$ *και* $c \mid b$, *και*

β. *αν* $d \mid a$ *και* $d \mid b$, *τότε* $d \mid c$, $\forall d \in R$.

Κατά *τον ίδιο τρόπο το* c *είναι ένα* ***ελάχιστο κοινό πολλαπλάσιο*** (ή *εκπ*) (least common multiple or lcm) *των* a *και* b *αν*:

α. $a \mid c$ *και* $b \mid c$, *και*

β. *αν* $a \mid d$ *και* $b \mid d$, *τότε* $c \mid d$, $\forall d \in R$.

Τέλος, *ένα στοιχείο* $u \in R$ *λέγεται* ***μονάδα*** (unit) *στο* R *αν υπάρχει κάποιο* $v \in R$ *έτσι ώστε* $uv = vu = 1$ (*δηλαδή* $v = u^{-1}$), *ενώ ένα στοιχείο* $a \in R$ *λέγεται* ***εταιρικό*** (associate) *του στοιχείου* $b \in R$, *αν υπάρχει μια μονάδα* u *έτσι ώστε* $a = ub$. Με *τον συμβολισμό* $a \sim b$ *υποδηλώνουμε εταιρικά στοιχεία*.

Παραδείγματα *τόσο για μονάδες όσο και για εταιρικά στοιχεία ενός δακτυλίου* R *υπάρχουν στο παράρτημα περί δακτυλίων του πρώτου κεφαλαίου*

Σαν *παράδειγμα των ορισμών που μόλις αναφέραμε, βλέπουμε πως το* 2 *είναι ένας μκδ των* 4 *και* 6, *ενώ το* 12 *είναι ένα εκπ των* 4 *και* 6 *στον* Ευκλείδειο *δακτύλιο* $\mathbb{Z}$. Προσέξτε *πως ούτε ο μκδ ούτε το εκπ είναι μοναδικά. Όλοι οι μκδ των* $a,b \in R$ *είναι τα εταιρικά μεταξύ τους στοιχεία και το ίδιο ισχύει και για όλα τα εκπ*.

Οι *μονάδες στον δακτύλιο* $\mathbb{Z}$ *είναι* 1 *και* -1, *και επομένως οι μκδ των* 4 *και* 6 *είναι* 2 *και* -2. *Στον* $\mathbb{Z}$, *με τον συμβολισμό* $\gcd(a, b)$ *ορίζεται ο μοναδικός μη αρνητικός μέγιστος κοινός διαιρέτης των* a *και* b. Ομοίως *με το* $\operatorname{lcm}(a, b)$ *ορίζεται το μοναδικό μη αρνητικό ελάχιστο κοινό πολλαπλάσιο των* a *και* b *και είναι γνωστό πως* $\operatorname{lcm}(a, b) = \frac{|ab|}{\gcd(a, b)}$ (*βλέπε και τις ασκήσεις*). Έτσι, *για* $a \in \mathbb{Z}$, $a < 0$, *έχουμε* $\gcd(a, 0) = \gcd(a, a) = |a|$, *ενώ* $\gcd(a, 1) = 1$ $\forall a \in \mathbb{Z}$. Δύο *ακέραιοι αριθμοί* a, b *λέγονται* ***πρώτοι μεταξύ*** *τους* (coprime or relatively prime) *αν* $\gcd(a, b) = 1$ *και συμβολίζονται με* $a \perp b$. Το *Mathematica συμφωνεί πλήρως με αυτά που λέμε*:

```
GCD[-2, 0] == GCD[-2, -2] == 2

True

LCM[4, -6] == LCM[-4, -6] == 12

True
```

Μέγιστοι κοινοί διαιρέτες και ελάχιστα κοινά πολλαπλάσια δεν υπάρχουν σε όλους ανεξαρτήτως τους δακτυλίους. Σαν παράδειγμα αναφέρουμε τους δακτύλιους των αλγεβρικών ακεραίων $\mathcal{O}_{-19}$, $\mathcal{O}_{-43}$, $\mathcal{O}_{-67}$, ή $\mathcal{O}_{-163}$. Για τις Ευκλείδειες περιοχές όμως υπάρχει πάντα ένας μκδ και συνεπώς και ένα εκπ.

Θεώρημα (ιδιότητες του μκδ ακεραίων):
Για $a, b, c \in \mathbb{Z}$, ισχύουν οι εξής ιδιότητες για τον μέγιστο κοινό διαιρέτη:
α. Αν a και b δεν είναι και τα δύο μηδέν, τότε υπάρχουν ακέραιοι x και y έτσι ώστε $\gcd(a, b) = xa + yb$.
β. $\gcd(a, b) = |a| \iff a \mid b$.
γ. $\gcd(a, b) = \gcd(b, a)$ (μεταθετική).
δ. $\gcd(a, \gcd(b, c)) = \gcd(\gcd(a, b), c)$ (προσεταιριστική).
ε. $\gcd(c \cdot a, c \cdot b) = |c| \cdot \gcd(a, b)$ (επιμεριστική).

Για την απόδειξη της πρώτης ιδιότητας γίνεται χρήση της αρχής της καλής διάταξης και της εις άτοπον απαγωγής (βλέπε και την βιβλιογραφία). (Για $0 < a \le b$ οι συντελεστές x, y μπορούν να υπολογισθούν και με την σειρά Farey, F_b.) Η απόδειξη των άλλων ιδιοτήτων αφήνεται σαν άσκηση.

Ο αλγόριθμος που ακολουθεί υπολογίζει μέγιστους κοινούς διαιρέτες σε τυχαίες Ευκλείδειες περιοχές. Ο δακτύλιος R μπορεί να είναι ή ο δακτύλιος των ακεραίων ή ο δακτύλιος των πολυωνύμων με συντελεστές από το R.

Αλγόριθμος: Ο κλασσικός Ευκλείδειος αλγόριθμος (ΕΑ)

Είσοδος: $a, b \in R$, όπου R είναι Ευκλείδεια περιοχή με συνάρτηση μεγέθους σ.
Έξοδος: Ένας μέγιστος κοινός διαιρέτης των a, b.

==========

1. $a_0 \longleftarrow a,\ a_1 \longleftarrow b$.

2. **while** $a_1 \neq 0$ **do** $\{a_0, a_1\} \longleftarrow \{a_1, \text{rem}(a_0, a_1)\}$.

3. **return** a_0.

==========

Ακολουθεί *ο αλγόριθμος αυτός προγραμματισμένος στο Mathematica για την περίπτωση* $R = \mathbb{Z}$. *Προσέξτε πως για τους* ***θετικούς ακεραίους*** a *και* b *η μόνη αλλαγή που χρειάζεται είναι η αντικατάσταση του* rem *από το* Mod[].

```
gcdPosInt[a_ /; a ∈ Integers && a ≥ 0,
  b_ /; b ∈ Integers && b ≥ 0] := Module[
  {a0 = a, a1 = b},
  While[a1 ≠ 0, {a0, a1} = {a1, Mod[a0, a1]}]; a0
 ]
```

Έτσι βρίσκουμε πως ο μκδ των 612 *και* 342 *είναι* 18

```
gcdPosInt[612, 342]
```

```
18
```

και το αποτέλεσμα είναι το ίδιο με αυτό που υπολογίσαμε στην εισαγωγή του κεφαλαίου αυτού.

Αξίζει *να σημειωθεί πως το πρόσημο του* Mod[a, b] *είναι το ίδιο με το πρόσημο του* b. *Έτσι για τον υπολογισμό του μκδ δύο θετικών ακεραίων έχουμε μία ακολουθία θετικών υπολοίπων. Στις ασκήσεις εξετάζουμε την περίπτωση όπου επιτρέπεται να κάνουμε διαίρεση με αρνητικό υπόλοιπο καθώς επίσης και έναν δυαδικό αλγόριθμο για την εύρεση του μκδ δύο ακεραίων.*

Μπορεί *κανείς να δεί τα υπόλοιπα που υπολογίζονται κατά τη διάρκεια υπολογισμού του μκδ χρησημοποιώντας την συνάρτηση* Trace[].

```
Trace[gcdPosInt[612, 342], Mod[a_, b_]] // Flatten
```

```
{Mod[612, 342], Mod[342, 270], Mod[270, 72], Mod[72, 54], Mod[54, 18]}
```

Στην γενική περίπτωση όμως, οι ακέραιοι a και b μπορεί να είναι και αρνητικοί. Έτσι ένας τρόπος για να βρούμε τον μοναδικό μη αρνητικό μέγιστο κοινό διαιρέτη των a και b είναι να πάρουμε την απόλυτο τιμή του αποτελέσματος. (Την απόλυτο τιμή μας δίνει η συνάρτηση Abs[] του *Mathematica*.) Αυτό μας οδηγεί στην ***κανονική μορφή*** (normal form) ***ενός ακεραίου*** a που ορίζεται ως $\text{normal}(a) = |a|$. Προφανώς $a \sim \text{normal}(a)$ στο $R = \mathbb{Z}$.

Το πρόγραμμα για **όλους τους ακεραίους** γενικά είναι:

```
gcdInteger[a_ /; a ∈ Integers, b_ /; b ∈ Integers] := Module[
  {a0 = a, a1 = b},
  While[a1 ≠ 0, {a0, a1} = {a1, Mod[a0, a1]}]; Abs[a0]
 ]

gcdInteger[612, -342]

18
```

Αξίζει να σημειωθεί πως μπορούμε να πάρουμε την απόλυτη τιμή (κανονική μορφή) είτε **μόνο** του τελικού υπολοίπου που είναι και ο μκδ είτε **όλων** των υπολοίπων (που θα εφαρμόσουμε στο τμήμα 2.2). Για την απλούστευση του κλασσικού Ευκλείδειου αλγορίθμου, κανονικοποιούμε και στα επόμενα μόνο το ***τελευταίο*** υπόλοιπο.

Για να επεκτείνουμε την έννοια της κανονικής μορφής σε οποιαδήποτε Ευκλείδεια περιοχή R, υπενθυμίζουμε πως $\forall a \in R,\ a \sim \text{normal}(a) \Leftrightarrow \exists$ μονάδα $u \in R$ έτσι ώστε $a = u \cdot \text{normal}(a)$. Είναι λογικό λοιπόν αυτήν την μονάδα να την αποκαλούμε τον ***κύριο συντελεστή*** (leading coefficient) $\text{lc}(a)$ του a και να ορίζουμε την ***κανονική μορφή*** του a ως $\text{normal}(a) = \frac{a}{\text{lc}(a)}$, με $\text{lc}(a) \neq 0$. Δηλαδή, το a είναι σε κανονική μορφή (ή ***κανονικοποιημένο***) αν $\text{lc}(a) = 1$. Ορίζουμε $\text{lc}(0) = 1$ οπότε $\text{normal}(0) = 0$. Στην περίπτωση των ακεραίων είναι $\text{lc}(a) = \text{sign}(a)$ και $\text{normal}(a) = \frac{a}{\text{lc}(a)} = |a|$. Στο *Mathematica* η κανονική μορφή των ακεραίων, εκτός από την συνάρτηση Abs[], ορίζεται και ως:

```
normal[0] = 0; normal[a_Integer] := a/Sign[a]
```

```
normal[-7] == normal[7] == 7
```

```
True
```

Για *τις* ***κανονικές μορφές*** *σε οποιαδήποτε* Ευκλείδεια περιοχή *R χρειαζόμαστε τις εξής δύο ιδιότητες*:

- *δύο στοιχεία του R έχουν την ίδια κανονική μορφή εάν και μόνον εάν είναι εταιρικά, και*
- *η κανονική μορφή ενός γινομένου είναι το γινόμενο των κανονικών μορφών.*

Ας *δούμε τώρα την περίπτωση που η* Ευκλείδεια περιοχή *είναι ο δακτύλιος πολυωνύμων* $R = \mathbb{Q}[x]$. Επειδή $\mathbb{Q}$ *είναι σώμα, κάθε μη μηδενικός ρητός αριθμός είναι μονάδα στο* $\mathbb{Q}$, *με συνέπεια* $\forall u \in \mathbb{Q}, u \neq 0$ *και* $\forall a \in R$ *έχουμε* $u \cdot a \sim a$ *στο* $R = \mathbb{Q}[x]$. Αν *λοιπόν θέλουμε να υπολογίσουμε* $\gcd(a, b)$, *με* $a, b \in R$ *πως θα διαλέξουμε τον αντιπρόσωπο ανάμεσα σε όλα τα πολλαπλασια του a; Όπως και στην περίπτωση των ακεραίων, μια λογική επιλογή είναι να πάρουμε το πολυώνυμο με μοναδιαίο κύριο συντελεστή* (**monic** *πολυώνυμο*). Έτσι *αν ο κύριος συντελεστής του a είναι* $\mathrm{lc}(a) \in \mathbb{Q}\backslash\{0\}$ *ορίζουμε σαν την* ***κανονική μορφή*** (normal form) ***του πολυωνύμου*** *το πολυώνυμο* $\mathrm{normal}(a) = \frac{a}{\mathrm{lc}(a)}$. Προφανώς $a \sim \mathrm{normal}(a)$ *στο* $R = \mathbb{Q}[x]$.

Ακολουθεί *ο κλασσικός* Ευκλείδειος *αλγόριθμος προγραμματισμένος στο Mathematica για την περίπτωση* $R = \mathbb{Q}[x]$. Προσέξτε *την αντικατάσταση του* rem *από το* PolynomialRemainder[] *και την κανονικοποίηση του αποτελέσματος στο τέλος.*

```
gcdPoly[a_, b_] := Module[
  {a0 = a, a1 = b},
  While[! Developer`ZeroQ[a1], {a0, a1} =
    {a1, PolynomialRemainder[a0, a1, First[Variables[{a, b}]]]}];
  Expand[a0 / Last[CoefficientList[Together[a0],
      First[Variables[{a, b}]]]]]
 ]
```

Έτσι *βρίσκουμε πως ο μκδ των πολυωνύμων* $x^3 - 7x + 7$ *και* $3x^2 - 7$ *είναι* 1

```
gcdPoly[x^3 - 7 x + 7, 3 x^2 - 7]
```

```
1
```

και το αποτέλεσμα είναι το ίδιο με αυτό που υπολογίζουμε με το Mathematica.

```
PolynomialGCD[x^3 - 7 x + 7, 3 x^2 - 7]
```

```
1
```

Όπως *και πριν*, *μπορεί κανείς να δεί τα υπόλοιπα που υπολογίζονται κατά τη διάρκεια υπολογισμού του μκδ χρησημοποιώντας την συνάρτηση* Trace[].

```
Trace[gcdPoly[x^3 - 7 x + 7, 3 x^2 - 7],
  PolynomialRemainder[a_, b_, x]] // Flatten
```

$$\left\{\text{PolynomialRemainder}\left[7 - 7\,x + x^3,\ -7 + 3\,x^2,\ x\right],\ \text{PolynomialRemainder}\left[-7 + 3\,x^2,\ 7 - \frac{14\,x}{3},\ x\right],\ \text{PolynomialRemainder}\left[7 - \frac{14\,x}{3},\ -\frac{1}{4},\ x\right]\right\}$$

Συνοψίζουμε *λέγοντας πως σε μία* Ευκλείδεια *περιοχή* R, $\forall a,b \in R$, *ορίζουμε σαν* $\gcd(a, b)$ *το μοναδικό κανονικοποιημένο εταιρικό στοιχείο όλων των μκδ των* a *και* b (*αν υπάρχει*). Αντίστοιχα *ορίζουμε σαν* $\text{lcm}(a, b)$ *το μοναδικό κανονικοποιημένο εταιρικό στοιχείο όλων των εκπ των* a *και* b. Έτσι $\gcd(a, b) > 0$ *για* $R = \mathbb{Z}$, *και* $\gcd(a, b)$ *είναι ένα πολυώνυμο με μοναδιαίο κύριο συντελεστή* (**monic** *πολυώνυμο*) *για* $R = \mathbb{Q}[x]$, *αν ένα τουλάχιστον από τα* a, b *είναι μη μηδενικό*. Και *στις δύο περιπτώσεις*, $\gcd(0, 0) = 0$.

2.2 Ο *επεκταμένος* Ευκλείδειος *αλγόριθμος*

Σε πολλές εφαρμογές είναι απαραίτητο να εκφράσουμε τον gcd(a, b) σαν γραμμική συνάρτηση των a και b, δηλαδή gcd(a, b) = $sa + tb$ (βλέπε Θεώρημα ιδιοτήτων του μκδ). Οι συντελεστές s, t ονομάζονται **συντελεστές Bézout**. Ένας τρόπος για να βρούμε τους συντελεστές s, t είναι πρώτα να εφαρμόσουμε τον κλασσικό Ευκλείδειο αλγόριθμο και μετά να "πάμε προς τα πίσω". Πηγαίνοντας πίσω στο παράδειγμα για την εύρεση του μκδ των ακεραίων 612 και 342 έχουμε:

$$\begin{aligned} 18 &= 72 - 1\cdot 54 = 72 - 1\cdot(270 - 3\cdot 72) \\ &= 4\cdot 72 - 270 = 4\cdot(342 - 1\cdot 270) - 270 \\ &= 4\cdot 342 - 5\cdot 270 = 4\cdot 342 - 5\cdot(612 - 1\cdot 342) \\ &= 9\cdot 342 - 5\cdot 612, \end{aligned}$$

και $s = -5$ ενώ $t = 9$. Ένας άλλος τρόπος εύρεσης των συντελεστών s, t (με πολλές εφαρμογές) είναι ο **επεκταμένος Ευκλείδειος αλγόριθμος**. Οι συντελεστές Bézout s, t υπολογίζονται τώρα εκφράζοντας ως $s_i a + t_i b$ **κάθε** υπόλοιπο a_i που υπολογίζεται στην εκτέλεση του κλασσικού Ευκλείδειου αλγόριθμου. Αυτό αποτελεί και την κεντρική ιδέα του αλγορίθμου. Δηλαδή για τους ακέραιους a = 612, b = 342 έχουμε τις εξής ακολουθίες:

$$\begin{array}{ll} a_0 = a & a_0 = s_0 a + t_0 b \\ a_1 = b & a_1 = s_1 a + t_1 b \\ a_2 = a_0 - a_1 q_1 & a_2 = s_2 a + t_2 b \\ a_3 = a_1 - a_2 q_2 & a_3 = s_3 a + t_3 b \\ \quad\vdots & \quad\vdots \\ a_i = a_{i-2} - a_{i-1} q_{i-1} & a_i = s_i a + t_i b \\ \quad\vdots & \quad\vdots \\ a_k = a_{k-2} - a_{k-1} q_{k-1} & a_k = s_k a + t_k b \\ 0 = a_{k-1} - a_k q_k & 0 = s_{k+1} a + t_{k+1} b \end{array}$$

Επαναλαμβάνουμε πως ο k λέγεται το Ευκλείδειο μήκος του ζεύγους $\{a, b\}$. Στην αριστερή στήλη είναι οι διαρέσεις, όπως και πριν, μόνο που τώρα λύνουμε ως προς τα υπόλοιπα. Στην δεξιά στήλη **κάθε** υπόλοιπο εκφράζεται σαν $s_i a + t_i b$ και σκοπός μας είναι να βρούμε έναν τρόπο υπολογισμού αυτών των συντελεστών s_i, t_i (η κεντρική ιδέα του αλγορίθμου). Προφανώς, για τους ακεραίους $a = 612$, $b = 342$, οι αρχικές τιμές των συντελεστών είναι $s_0 = 1$, $t_0 = 0$, $s_1 = 0$, $t_1 = 1$. Θέλουμε να βρούμε τις τιμές των υπόλοιπων s_i, t_i για $2 \le i \le k$.

Για να δούμε πως θα υπολογίζουμε τις τιμές των s_i, t_i για $2 \le i \le k$, στην περίπτωση **των θετικών ακεραίων** (που ισούνται με τις κανονικές τους μορφές) κάνουμε το εξής: εξισώνουμε, στο βήμα i, τις τιμές του υπολοίπου a_i που παίρνουμε από την δεξιά και την αριστερή στήλη. Έτσι έχουμε την **βασική σχέση** του επεκταμένου Ευκλείδειου αλγόριθμου

$$a_i = s_i a + t_i b = a_{i-2} - a_{i-1} q_{i-1} = (s_{i-2} a + t_{i-2} b) - (s_{i-1} a + t_{i-1} b) q_{i-1} = (s_{i-2} - s_{i-1} q_{i-1}) a + (t_{i-2} - t_{i-1} q_{i-1}) b.$$

Η σχέση αυτή μας υποδεικνύει ότι: αφού βρούμε το πηλίκο $\mathrm{quo}(a_{i-2}, a_{i-1})$ οι συντελεστές s_i, t_i υπολογίζονται όπως και το υπόλοιπο a_i:

$$\begin{aligned} q_{i-1} &= \mathrm{quo}(a_{i-2}, a_{i-1}) \\ a_i &= a_{i-2} - a_{i-1} q_{i-1} \\ s_i &= s_{i-2} - s_{i-1} q_{i-1} \\ t_i &= t_{i-2} - t_{i-1} q_{i-1} \end{aligned}$$

Προσέξτε πως ο κύριος συντελεστής, των θετικών ακεραίων είναι $\mathrm{lc}(\cdot) = \mathrm{sign}(\cdot) = 1$ και έτσι η κανονική μορφή τους δεν αλλάζει. Όμως στην γενική περίπτωση των ακεραίων και πολυωνύμων η βασική σχέση του επεκταμένου Ευκλείδειου αλγόριθμου **κανονικοποιείται** και αντί για $a_i = s_i a + t_i b = \ldots$ έχουμε $\mathrm{lc}(a_i) a_i = s_i a + t_i b = \ldots$

Συνεπώς, οι παραπάνω τύποι υπολογισμού των a_i, s_i, t_i προσαρμόζονται ανάλογα και μας δίνουν τις κανονικοποιημένες μορφές:

$$q_{i-1} = \text{quo}(a_{i-2}, a_{i-1})$$
$$\alpha_i = \text{lc}(a_{i-2} - a_{i-1} q_{i-1})$$
$$a_i = \frac{(a_{i-2} - a_{i-1} q_{i-1})}{\alpha_i}$$
$$s_i = \frac{(s_{i-2} - s_{i-1} q_{i-1})}{\alpha_i}$$
$$t_i = \frac{(t_{i-2} - t_{i-1} q_{i-1})}{\alpha_i}$$

Δηλαδή, *για να πάρουμε την κανονική μορφή διαιρούμε με τον* $\alpha_i = \text{lc}(a_{i-2} - a_{i-1} q_{i-1})$. Οι *αρχικές τιμές επίσης προσαρμόζονται αντίστοιχα και είναι*: $\alpha_0 = \text{lc}(a)$, $a_0 = \frac{a}{\alpha_0}$, $s_0 = \frac{1}{\alpha_0}$, $t_0 = 0$, $\alpha_1 = \text{lc}(b)$, $a_1 = \frac{b}{\alpha_1}$, $s_1 = 0$, $t_1 = \frac{1}{\alpha_1}$.

Το *ερώτημα τώρα είναι πως μπορούμε να περιγράψουμε την συμπεριφορά του επεκταμένου* Ευκλείδειου *αλγόριθμου με μαθηματικούς τύπους που είναι εύκολοι στον χειρισμό τους*. Μικρός *πειραματισμός καθιστά προφανές πως ο καλλίτερος τρόπος είναι να ορίσουμε τους πίνακες*:

$$\sigma_0 = \begin{pmatrix} s_0 & t_0 \\ s_1 & t_1 \end{pmatrix}, \ \tau_i = \begin{pmatrix} 0 & 1 \\ \alpha_{i+1}^{-1} & -q_i \alpha_{i+1}^{-1} \end{pmatrix} \text{ για } 1 \le i \le k,$$

όπου $\sigma_i = \tau_i \sigma_{i-1}$ ή $\sigma_i = \tau_i \ \tau_{i-1} \cdots \ \tau_1 \sigma_0$ *και* k *είναι το* Ευκλείδειο *μήκος του ζεύγους του οποίου βρίσκουμε τον μκδ*. Δηλαδή *από τον πίνακα* σ_k *παίρνουμε τις τιμές των συντελεστών* Bézout. Επίσης *προσέξτε πως η ορίζουσα του* τ_i *είναι* $\det(\tau_i) = -\frac{1}{\alpha_{i+1}}$, *μία μονάδα στον δακτύλιο*. Αυτό *συνεπάγεται πως υπάρχει ο αντίστροφος πίνακας* τ_i^{-1} *και είναι*

```
Inverse[ ( 0          1
           α_{i+1}^-1  -q_i α_{i+1}^-1 ) ] // MatrixForm
```

$$\begin{pmatrix} q_i & \alpha_{1+i} \\ 1 & 0 \end{pmatrix}$$

Το *ακόλουθο θεώρημα διέπει την συμπεριφορά του επεκταμένου* Ευκλείδειου *αλγόριθμου*.

Θεώρημα 2.2.1 (*συμπεριφορά του επεκταμένου* Ευκλείδειου *αλγόριθμου*):
Έχοντας *ορίσει τις αρχικές τιμές* $\alpha_0 = \text{lc}(a)$, $a_0 = \frac{a}{\alpha_0}$, $s_0 = \frac{1}{\alpha_0}$, $t_0 = 0$, $\alpha_1 = \text{lc}(b)$, $a_1 = \frac{b}{\alpha_1}$, $s_1 = 0$, $t_1 = \frac{1}{\alpha_1}$, *τους πίνακες* $\sigma_i = \tau_i \ \tau_{i-1} \cdots \ \tau_1 \sigma_0$, *και τις τελικές συνθήκες*

$\alpha_{k+1} = 1$, $a_{k+1} = 0$, *τότε για* $0 \le i \le k$, *έχουμε*

α. $\gcd(a, b) = \gcd(a_i, a_{i+1}) = a_k$,

β. $\sigma_i \cdot \begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} a_i \\ a_{i+1} \end{pmatrix}$ (*μόνο τα* a, b **δεν** *είναι κανονικοποιημένα*),

γ. $\sigma_i = \begin{pmatrix} s_i & t_i \\ s_{i+1} & t_{i+1} \end{pmatrix}$,

δ. $a_i = s_i a + t_i b$ (*ισχύει και για* $i = k + 1$),

ε. $s_i t_{i+1} - t_i s_{i+1} = \frac{(-1)^i}{\alpha_0 \cdots \alpha_{i+1}}$,

στ. $\gcd(a_i, t_i) = \gcd(a, t_i)$,

ζ. $a = (-1)^i \alpha_0 \cdots \alpha_{i+1} (t_{i+1} a_i - t_i a_{i+1})$, $b = (-1)^i \alpha_0 \cdots \alpha_{i+1} (s_{i+1} a_i - s_i a_{i+1})$.

Απόδειξη:

α. Αυτή *είναι η γνωστή σχέση και αποδεικνύεται από το γεγονός ότι σε μία* Ευκλείδεια *περιοχή αν* $d \mid a_0$ *και* $d \mid a_1$ *τότε* $d \mid a_2 = a_0 - a_1 q_1$ *κτλ.* (Βλέπε *και την απόδειξη* **α′** *μετά την* **β**.)

β. Αποδεικνύεται *με επαγωγή στο* i (induction on i). Για $i = 0$ *ισχύει* $\sigma_0 \cdot \begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} a_0 \\ a_1 \end{pmatrix}$ *λόγω των αρχικών τιμών.* Υποθέτουμε *λοιπόν πως* $\sigma_i \cdot \begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} a_i \\ a_{i+1} \end{pmatrix}$ *και θα δείξουμε πως* $\sigma_{i+1} \cdot \begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} a_{i+1} \\ a_{i+2} \end{pmatrix}$. Εξ *ορισμού όμως ισχύει* $\sigma_{i+1} = \tau_{i+1} \sigma_i$ *οπότε*

$$\sigma_{i+1} \cdot \begin{pmatrix} a \\ b \end{pmatrix} = \tau_{i+1} \sigma_i \cdot \begin{pmatrix} a \\ b \end{pmatrix} = \tau_{i+1} \cdot \begin{pmatrix} a_i \\ a_{i+1} \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ \alpha_{i+2}^{-1} & -q_{i+1} \alpha_{i+2}^{-1} \end{pmatrix} \begin{pmatrix} a_i \\ a_{i+1} \end{pmatrix} = \begin{pmatrix} a_{i+1} \\ a_{i+2} \end{pmatrix}.$$

α′. Παραθέτουμε *μια λίγο γνωστή απόδειξη της ιδιότητας* **α** *που βασίζεται στην απόδειξη της* **β**. Επαναλαπτική *χρήση της* $\sigma_{i+1} = \tau_{i+1} \sigma_i$ *και η* **β** *μας δίνουν την σχέση*

$$\begin{pmatrix} a_k \\ 0 \end{pmatrix} = \tau_k \cdots \tau_{i+1} \sigma_i \cdot \begin{pmatrix} a \\ b \end{pmatrix} = \tau_k \cdots \tau_{i+1} \cdot \begin{pmatrix} a_i \\ a_{i+1} \end{pmatrix}$$

από την οποία βλέπουμε πως το a_k *είναι γραμμικός συνδυασμός των* a_i *και* a_{i+1} *και άρα κάθε κοινός διαιρέτης των* a_i *και* a_{i+1} *διαιρεί το* a_k. Από *την άλλη μεριά, επειδή υπάρχουν οι αντίστροφοι πίνακες* τ_i^{-1} *έχουμε την σχέση*

$$\begin{pmatrix} a_i \\ a_{i+1} \end{pmatrix} = \tau_{i+1}^{-1} \cdots \tau_k^{-1} \cdot \begin{pmatrix} a_k \\ 0 \end{pmatrix}$$

από την οποία βλέπουμε πως τα a_i και a_{i+1} διαιρούνται από το a_k. Άρα, $a_k = \gcd(a_i, a_{i+1})$ επειδή το a_k είναι κανονικοποιημένο. Αυτό ισχύει και για $i = 0$, οπότε έχουμε $a_k = \gcd(a_0, a_1) = \gcd(a, b)$.

γ. Αποδεικνύεται και αυτό με επαγωγή στο i (induction on i). Για $i = 0$ ισχύει $\sigma_0 = \begin{pmatrix} s_0 & t_0 \\ s_1 & t_1 \end{pmatrix}$ εξ ορισμού. Υποθέτουμε λοιπόν πως $\sigma_i = \begin{pmatrix} s_i & t_i \\ s_{i+1} & t_{i+1} \end{pmatrix}$ και θα δείξουμε πως $\sigma_{i+1} = \begin{pmatrix} s_{i+1} & t_{i+1} \\ s_{i+2} & t_{i+2} \end{pmatrix}$. Εξ ορισμού ισχύει $\sigma_{i+1} = \tau_{i+1}\sigma_i$ οπότε

$$\tau_{i+1}\sigma_i = \tau_{i+1}\cdot\begin{pmatrix} s_i & t_i \\ s_{i+1} & t_{i+1} \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ \alpha_{i+2}^{-1} & -q_{i+1}\alpha_{i+2}^{-1} \end{pmatrix}\begin{pmatrix} s_i & t_i \\ s_{i+1} & t_{i+1} \end{pmatrix} =$$

$$\begin{pmatrix} s_{i+1} & t_{i+1} \\ (s_i - q_{i+1}s_{i+1})\alpha_{i+2}^{-1} & (t_i - q_{i+1}t_{i+1})\alpha_{i+2}^{-1} \end{pmatrix} = \begin{pmatrix} s_{i+1} & t_{i+1} \\ s_{i+2} & t_{i+2} \end{pmatrix}.$$

δ. Έπεται από τις ιδιότητες **β** και **γ** (αντικαθιστώντες το σ_i στην **β** με τον πίνακα από την **γ**).

ε. Έπεται από την **γ** παίρνοντας ορίζουσες. Πράγματι,

$$s_i t_{i+1} - t_i s_{i+1} = \det\begin{pmatrix} s_i & t_i \\ s_{i+1} & t_{i+1} \end{pmatrix} = \det(\sigma_i)$$

$$= \det(\tau_i)\cdots\det(\tau_1)\cdot\det\begin{pmatrix} s_0 & t_0 \\ s_1 & t_1 \end{pmatrix} = \frac{(-1)^i}{\alpha_0\cdots\alpha_{i+1}}.$$

Προσέξτε πως επειδή $\frac{(-1)^i}{\alpha_0\cdots\alpha_{i+1}}$ είναι μονάδα στον δακτύλιο, η **ε** συνεπάγεται πως $\gcd(s_i, t_i) = 1$.

στ. Για να αποδείξουμε $\gcd(a_i, t_i) = \gcd(a, t_i)$ αρκεί να δείξουμε πως κάθε διαιρέτης των a_i, t_i είναι και διαιρέτης των a, t_i, και αντίστροφα. Πράγματι, έστω $\rho \in R$ ώστε $\rho \mid t_i$. Αν επιπλέον $\rho \mid a_i$ τότε θα έχουμε $\rho \mid (a_i - t_i b) = s_i a$ και συνεπώς $\rho \mid a$ επειδή $\gcd(s_i, t_i) = 1$. Αντίστροφα, αν επιπλέον $\rho \mid a$ τότε θα έχουμε $\rho \mid (s_i a + t_i b) = a_i$.

ζ. Τέλος, για να αποδείξουμε πως $a = (-1)^i\alpha_0\cdots\alpha_{i+1}(t_{i+1}a_i - t_i a_{i+1})$, $b = (-1)^i\alpha_0\cdots\alpha_{i+1}(s_{i+1}a_i - s_i a_{i+1})$ πολλαπλασιάζουμε και τα δύο μέρη της **β** με σ_i^{-1} και χρησιμοποιώντας την **γ** και **ε** έχουμε

$$\begin{pmatrix} a \\ b \end{pmatrix} = \sigma_i^{-1}\begin{pmatrix} a_i \\ a_{i+1} \end{pmatrix} = (-1)^i \alpha_0 \cdots \alpha_{i+1}\begin{pmatrix} t_{i+1} & -t_i \\ -s_{i+1} & s_i \end{pmatrix}\begin{pmatrix} a_i \\ a_{i+1} \end{pmatrix}$$

αφού ο πίνακας σ_i^{-1} *είναι*

```
Inverse[σi = ( si    ti   ) ] // MatrixForm
         ( si+1  ti+1 )
```

$$\begin{pmatrix} \frac{t_{1+i}}{-s_{1+i}\,t_i + s_i\,t_{1+i}} & -\frac{t_i}{-s_{1+i}\,t_i + s_i\,t_{1+i}} \\ -\frac{s_{1+i}}{-s_{1+i}\,t_i + s_i\,t_{1+i}} & \frac{s_i}{-s_{1+i}\,t_i + s_i\,t_{1+i}} \end{pmatrix}$$

Ακολουθεί *ο επεκταμένος* Ευκλείδειος *αλγόριθμος στην γενική του μορφή*. Ο *δακτύλιος R είναι* ή $\mathbb{Z}$ ή $\mathbb{Q}[x]$.

Αλγόριθμος: Ο *επεκταμένος* Ευκλείδειος *αλγόριθμος* (EEA)

Είσοδος: $a, b \in R$, όπου *R είναι* Ευκλείδεια *περιοχή με συνάρτηση μεγέθους* σ (*απόλυτος τιμή για τους ακεραίους*, *βαθμός για τα πολυώνυμα*), *και εφοδιασμένο με κανονική μορφή* (normal(a) = $\frac{a}{\mathrm{lc}(a)}$).

Έξοδος: Ένας *μέγιστος κοινός διαιρέτης των* a, b *καθώς επίσης και* s, t *έτσι ώστε* gcd(a, b) = $sa + tb$.

==========

1. $\alpha_0 \longleftarrow \mathrm{lc}(a)$, $a_0 \longleftarrow \frac{a}{\alpha_0}$, $s_0 \longleftarrow \frac{1}{\alpha_0}$, $t_0 \longleftarrow 0$,
$\alpha_1 \longleftarrow \mathrm{lc}(b)$, $a_1 \longleftarrow \frac{b}{\alpha_1}$, $s_1 \longleftarrow 0$, $t_1 \longleftarrow \frac{1}{\alpha_1}$.

2. **while** $a_1 \neq 0$ **do** {

$q \longleftarrow \mathrm{quo}(a_0, a_1)$;

$\{\alpha_0, \alpha_1\} \longleftarrow \{\alpha_1, \mathrm{lc}(a_0 - a_1 q)\}$;

$\{a_0, a_1\} \longleftarrow \{a_1, \frac{(a_0 - a_1 q)}{\alpha_1}\}$;

$\{s_0, s_1\} \longleftarrow \{s_1, \frac{(s_0 - s_1 q)}{\alpha_1}\}$;

$\{t_0, t_1\} \longleftarrow \{t_1, \frac{(t_0 - t_1 q)}{\alpha_1}\}$

}.

3. **return** a_0, s_0, t_0.

==========

Ακολουθούν τα αντίστοιχα προγράμματα για το Mathematica τόσο για ακεραίους όσο και για πολυώνυμα. Η κύρια διαφορά έγκειται στον ορισμό του κύριου συντελεστή. Για τους ακεραίους έχουμε δει πως lc(a) = sign(a) ενώ για τα πολυώνυμα είναι lc(a) = a_n όπου $a = \sum_{i=0}^{n} a_i x^i$. Οι δύο περιπτώσεις συνδυάζονται στην συνάρτηση lc[] που μας δίνει τον κύριο συντελεστή ***ακεραίων* ή *πολυωνύμων***:

```
lc[a_] := Module[{},
  (*  a is zero  *)
  If[a == 0, Return[1]];
  (*  a is Integer  *)
  If[IntegerQ[a], Return[Sign[a]]];
  (*  a is Poly  *)
  If[Length[CoefficientList[a, First[Variables[a]]]] > 0,
   Last[CoefficientList[a, First[Variables[a]]]], 1]
 ]
```

Ο επεκταμένος Ευκλείδειος αλγόριθμος για πολυώνυμα προγραμματισμένος στο Mathematica είναι:

```
EEAPoly[a_, b_] :=
 Module[{alpha0 = lc[a], alpha1 = lc[b], a0, a1, s0, s1, t0, t1, q},
  a0 = Expand[a/alpha0]; s0 = 1/alpha0; t0 = 0;
  a1 = Expand[b/alpha1]; s1 = 0; t1 = 1/alpha1;
  (****  Print["   alpha0= ",alpha0,"  a0= ",Expand[a0],
       "  alpha1= ",alpha1,"  a1= ",Expand[a1],"  s0= ",
       s0,"  s1= ",s1,"  t0= ",t0,"  t1= ",t1];  ****)
  While[! Developer`ZeroQ[a1],
   q = PolynomialQuotient[a0, a1, First[Variables[{a, b}]]];
   {alpha0, alpha1} = {alpha1, lc[a0 - a1 q]};
   {a0, a1} = {a1, Expand[(a0 - a1 q)/alpha1]};
   {s0, s1} = {s1, Expand[(s0 - s1 q)/alpha1]};
   {t0, t1} = {t1, Expand[(t0 - t1 q)/alpha1]};
   (****   Print["  q= ",q,"  a0= ",a0,
        "  alpha1= ",alpha1,"  a1= ",a1,"  s0= ",s0,
        "  s1= ",s1,"  t0= ",t0,"  t1= ",t1];  ****)
  ]; {a0, {s0, t0}}
 ]
```

Έτσι *έχουμε για παράδειγμα* (*η συνάρτηση* lc(·) *πρέπει να έχει ενεργοποιηθεί*)

```
{d, {s, t}} = EEAPoly[x^3 - 7 x + 7, 3 x^2 - 7]
```

$$\left\{1, \left\{-\frac{27}{7} - \frac{18\,x}{7}, -4 + \frac{9\,x}{7} + \frac{6\,x^2}{7}\right\}\right\}$$

το οποίο συμφωνεί με το αποτέλεσμα που παίρνουμε με το Mathematica.

```
<< Algebra`PolynomialExtendedGCD`
PolynomialExtendedGCD[x^3 - 7 x + 7, 3 x^2 - 7]
```

$$\left\{1, \left\{-\frac{27}{7} - \frac{18\,x}{7}, -4 + \frac{9\,x}{7} + \frac{6\,x^2}{7}\right\}\right\}$$

Ε*νεργοποιώντας τις* Print statements *στο πρόγραμμα* EEAPoly[] *βλέπουμε τα ενδιάμεσα αποτελέσματα*. Λόγω έλλειψης *χώρου παραθέτουμε μερικά από αυτά*:

```
{d, {s, t}} = EEAPoly[x^3 - 7 x + 7, 3 x^2 - 7]
```

$$\text{alpha0}= 1 \quad \text{a0}= 7 - 7x + x^3 \quad \text{alpha1}= 3 \quad \text{a1}= -\frac{7}{3} + x^2 \quad \text{s0}= 1 \quad \text{s1}= 0 \quad \text{t0}= 0 \quad \text{t1}= \frac{1}{3}$$

$$\text{q}= x \quad \text{a0}= -\frac{7}{3} + x^2 \quad \text{alpha1}= -\frac{14}{3} \quad \text{a1}= -\frac{3}{2} + x \quad \text{s0}= 0 \quad \text{s1}= -\frac{3}{14} \quad \text{t0}= \frac{1}{3} \quad \text{t1}= \frac{x}{14}$$

$$\text{q}= \frac{3}{2} + x \quad \text{a0}= -\frac{3}{2} + x \quad \text{alpha1}= -\frac{1}{12} \quad \text{a1}= 1 \quad \text{s0}= -\frac{3}{14} \quad \text{s1}= -\frac{27}{7} - \frac{18x}{7} \quad \text{t0}= \frac{x}{14} \quad \text{t1}= -4 + \frac{9x}{7} + \frac{6x^2}{7}$$

$$\text{q}= -\frac{3}{2} + x \quad \text{a0}= 1 \quad \text{alpha1}= 1 \quad \text{a1}= 0 \quad \text{s0}= -\frac{27}{7} - \frac{18x}{7} \quad \text{s1}= -6 + \frac{18x^2}{7} \quad \text{t0}= -4 + \frac{9x}{7} + \frac{6x^2}{7} \quad \text{t1}= -6 + 6x - \frac{6x^3}{7}$$

$$\left\{1, \left\{-\frac{27}{7} - \frac{18x}{7}, -4 + \frac{9x}{7} + \frac{6x^2}{7}\right\}\right\}$$

Στο σημείο αυτό αξίζει να παρατηρήσουμε την συμπεριφορά των βαθμών των πολυωνύμων s και t. Προσέξτε πως από το παράδειγμα και την σχέση $\gcd(a, b) = sa + tb$ βλέπουμε πως ισχύει $\deg(s) < \deg(b)$ και $\deg(t) < \deg(a)$. Θα αποδείξουμε κάτι πιο γενικό.

Θεώρημα 2.2.2 (*βαθμοί των s και t*):
Αν κατά την εκτέλεση του επεκταμένο Ευκλείδειου αλγόριθμου για πολυώνυμα γίνουν L διαιρέσεις και θέσουμε $\deg(a) = n_0$, $\deg(b) = n_1$, τότε ο βαθμός κάθε πηλίκου q_i είναι $\deg(q_i) = n_{i-1} - n_i$, $1 \le i \le L$, και ισχύει

$$\deg(s_i) = \sum_{j=2}^{i-1} \deg(q_j) = n_1 - n_{i-1}, \quad \text{για } 2 \le i \le L,$$
$$\deg(t_i) = \sum_{j=1}^{i-1} \deg(q_j) = n_0 - n_{i-1}, \quad \text{για } 1 \le i \le L.$$

Απόδειξη:
Θα αποδείξουμε με επαγωγή μόνον την πρώτη σχέση διότι όμοια αποδεικνύεται και η δεύτερη. Επιπλέον θα δείξουμε ότι ισχύει

$$\deg(s_{i-1}) < \deg(s_i), \quad \text{για } 2 \le i \le L. \tag{1}$$

Σημειώνουμε πως τα s_{L+1}, t_{L+1} δεν μας χρειάζονται και έτσι δεν τα υπολογίζουμε

Για $i = 2$ ισχύει το θεώρημα διότι $s_2 = \frac{s_0 - q_1 s_1}{\alpha_2} = \frac{1 - q_1 \cdot 0}{\alpha_2} = \alpha_2^{-1}$, και επειδή ο βαθμός του μηδενικού πολυωνύμου είναι οποιοσδήποτε αρνητικός αριθμός, έστω -1, έχουμε $\deg(s_1) = -1 < 0 = \deg(s_2)$. Ας υποθέσουμε τώρα πως το θεώρημα και η ανισότητα (1) ισχύουν για $2 \le i \le k$. Θα δείξουμε πως ισχύουν και για $i = k + 1$. Πράγματι, έχουμε

$$\deg(s_{k-1}) < \deg(s_k) < n_{k-1} - n_k + \deg(s_k) = \deg(q_k s_k)$$

από το οποίο συνεπάγεται αφ' ενός μεν πως

$$\deg(s_{k+1}) = \deg(s_{k-1} - q_k s_k) = \deg(q_k) + \deg(s_k) > \deg(s_k)$$

αφ' ετέρου δε πως

$$\deg(s_{k+1}) = \deg(q_k) + \deg(s_k) = \deg(q_k) + \sum_{j=2}^{k-1} \deg(q_j)$$
$$= \sum_{j=2}^{k} \deg(q_j) .//$$

Για τους ακεραίους το αντίστοιχο πρόγραμμα είναι

```
EEA[a_ /; a ∈ Integers, b_ /; b ∈ Integers] :=
 Module[{alpha0 = lc[a], alpha1 = lc[b], a0, a1, s0, s1, t0, t1, q},
  a0 = a/alpha0; s0 = 1/alpha0; t0 = 0;
  a1 = b/alpha1; s1 = 0; t1 = 1/alpha1;
  (****  Print["   alpha0= ",alpha0,"  a0= ",
       a0,"  alpha1= ",alpha1,"  a1= ",a1,"  s0= ",s0,
       "  s1= ",s1,"  t0= ",t0,"  t1= ",t1];  ****)
  While[a1 ≠ 0,
   q = Quotient[a0, a1];
   {alpha0, alpha1} = {alpha1, lc[a0 - a1 q]};
   {a0, a1} = {a1, Expand[(a0 - a1 q)/alpha1]};
   {s0, s1} = {s1, Expand[(s0 - s1 q)/alpha1]};
   {t0, t1} = {t1, Expand[(t0 - t1 q)/alpha1]};
   (****  Print["  q= ",q,"  alpha1= ",
        alpha1,"  a0= ",a0,"  a1= ",a1,"  s0= ",s0,
        "  s1= ",s1,"  t0= ",t0,"  t1= ",t1]; ****)
  ]; {a0, {s0, t0}}
 ]
```

και έχουμε για παράδειγμα (*η συνάρτηση* lc(·) *πρέπει να έχει ενεργοποιηθεί*)

```
{d, {s, t}} = EEA[612, 342]
```

```
{18, {-5, 9}}
```

το οποίο συμφωνεί με το αποτέλεσμα που παίρνουμε με το Mathematica.

```
ExtendedGCD[612, 342]
```

```
{18, {-5, 9}}
```

Κ*αι εδώ*, *ενεργοποιώντας τις* Print statements *στο πρόγραμμα* EEA[] *βλέπουμε τα ενδιάμεσα αποτελέσματα*. Λόγω έλλειψης χώρου παραθέτουμε μερικά από αυτά:

```
{d, {s, t}} = EEA[612, 342]

   alpha0= 1  a0= 612  alpha1=
 1  a1= 342  s0= 1  s1= 0  t0= 0  t1= 1

  q= 1  alpha1= 1  a0= 342  a1= 270  s0= 0  s1= 1  t0= 1  t1= -1

  q= 1  alpha1= 1  a0= 270  a1= 72  s0= 1  s1= -1  t0= -1  t1= 2

  q= 3  alpha1= 1  a0= 72  a1= 54  s0= -1  s1= 4  t0= 2  t1= -7

  q= 1  alpha1= 1  a0= 54  a1= 18  s0= 4  s1= -5  t0= -7  t1= 9

  q= 3  alpha1= 1  a0= 18  a1= 0  s0= -5  s1= 19  t0= 9  t1= -34

{18, {-5, 9}}
```

2.3 Ανάλυση του Ευκλείδειου αλγόριθμου

Για την ανάλυση του χρόνου υπολογισμού του κλασσικού Ευκλείδειου αλγόριθμου θα εξετάσουμε ξεχωριστά τις δύο περιπτώσεις $R = \mathbb{Z}$ και $R = F[x]$, όπου F σώμα. Στην τελευταία περίπτωση ανήκουν τα πεπερασμένα σώματα και το σώμα $\mathbb{Q}[x]$.

Το αποτέλεσμα που θα βρούμε ισχύει και για τον επεκταμένο Ευκλείδειο αλγόριθμο.

■ 2.3.1 $R = \mathbb{Z}$ και το το θεώρημα του Lamé (1844)

Για την περίπτωση των ακεραίων συνιστούμε το άρθρο του Shallit. Για την ανάλυση του χρόνου υπολογισμού του μκδ δύο ακεραίων θα διατυπώσουμε και θα αποδείξουμε το θεώρημα του Lamé (1844). Η απόδειξη του θεωρήματος αυτού είναι σημαντική γιατί δείχνει πως οι αριθμοί της ακολουθίας Fibonacci έχουν το μέγιστο Ευκλείδειο μήκος. Στην συνέχεια θα βελτιώσουμε την ανάλυση και θα αποκτήσουμε καλλίτερο φράγμα για τον αριθμό των διαιρέσεων που γίνονται κατά την διάρκεια του υπολογισμού του μκδ ακεραίων.

Χωρίς περιορισμό της γενικότητας, θεωρούμε πως $a, b \in \mathbb{Z}$, $a \geq b \geq 0$ (αλλιώς εργαζόμαστε με την κανονική τους μορφή). Από την περιγραφή του κλασσικού Ευκλείδειου αλγορίθμου βλέπουμε πως όλη η "δουλειά" γίνεται στο δεύτερο βήμα που υπάρχει ένας βρόγχος της μορφής:

$$\textbf{while } b \neq 0 \textbf{ do } \{a, b\} \longleftarrow \{b, \mathrm{rem}(a, b)\}.$$

Συγκεκριμένα, στο βήμα αυτό γίνεται ένας "αριθμός" διαιρέσεων ακεραίων πολλαπλής ακριβείας. Όπως έχουμε δει, ο χρόνος υπολογισμού κάθε μιας από αυτές τις διαιρέσεις είναι $O(m(n - m + 1))$, όπου $n = \lambda(a)$, $m = \lambda(b)$ και $n - m + 1 = \lambda_\beta(q) = \lambda(q)$, το μήκος του πηλίκου q ως προς την βάση $\beta = 2^{64}$. Για να βρούμε τον χρόνο υπολογισμού του μκδ των a, b μένει να προσδιορίσουμε τον "αριθμό" των διαιρέσεων που εκτελούνται.

Εδώ είναι που υπεισέρχεται το θεώρημα του Lamé που μας λέει ότι ο "αριθμός" των διαιρέσεων είναι $< 5 \cdot \lambda_{10}(b)$, όπου $\lambda_{10}(b)$ είναι ο αριθμός των ψηφίων του μικρότερου ακέραιου b. Συνεπώς, επειδή $5 \cdot \lambda_{10}(b) = O(\lambda(b)) = O(m)$, μια **πρώτη προσέγγιση** του χρόνου υπολογισμού του μκδ των a, b είναι $O(m^2(n - m + 1)) = O(m^2 n)$.

Μπορούμε όμως να βρούμε και καλλίτερο όριο. Συγκεκριμένα, στο βρόγχο του αλγορίθμου γίνεται ένας "αριθμός" διαιρέσεων ακεραίων πολλαπλής ακριβείας. Αν θέσουμε $a_0 = a$ και $a_1 = b$, από τις διαρέσεις αυτές βρίσκονται τα υπόλοιπα a_2, …, a_L.

$$
\begin{aligned}
a_0 &= a \\
a_1 &= b \\
a_2 &= a_0 - a_1 q_1 \\
a_3 &= a_1 - a_2 q_2 \\
&\vdots \\
a_i &= a_{i-2} - a_{i-1} q_{i-1} \\
&\vdots \\
a_L &= a_{L-2} - a_{L-1} q_{L-1} \\
0 &= a_{L-1} - a_L q_L
\end{aligned}
$$

Θέτουμε $n_0 = n = \lambda(a)$, $n_1 = m = \lambda(b)$, $n_2 = \lambda(a_2)$, …, $n_L = \lambda(a_L)$ και έχουμε $n_0 \geq n_1 \geq n_2 \geq \ldots \geq n_L$. Όπως θα δούμε στο Θεώρημα του Lamé, στην περίπτωση με το μέγιστο L που είναι αυτή της ακολουθίας Fibonacci, γνήσιες ανισότητες παρουσιάζονται μετά από κάθε τέσσερα ή πέντε υπόλοιπα.

Θα υπολογίσουμε τώρα τον χρόνο υπολογισμού κάθε μιας από αυτές τις διαιρέσεις και θα προσθέσουμε τα αποτελέσματα. Ξέρουμε πως το υπόλοιπο a_2 υπολογίζεται σε χρόνο $m(n - m + 1) = n_1(n_0 - n_1 + 1)$, το υπόλοιπο a_3 υπολογίζεται σε χρόνο $n_2(n_1 - n_2 + 1)$, …, και το υπόλοιπο $a_{L+1} = 0$ υπολογίζεται σε χρόνο $n_L(n_{L-1} - n_L + 1)$.

Ο συνολικός χρόνος για την εύρεση του μκδ είναι λοιπόν

$$\sum_{i=1}^{L} n_i\,(n_{i-1} - n_i + 1).$$

Αν τώρα $n_i = n_{i-1} - 1 = \cdots = m - i + 1$, για $2 \le i \le L < 5m$ (δηλαδή το μήκος κάθε υπολοίπου a_i είναι μικρότερο κατά μονάδα από το μήκος του υπολοίπου a_{i-1}, και $L < 5m$ όπως προκύπτει από το θεώρημα Lamé), τότε για $m \ge 1$ (που ισχύει πάντα) και $n_{i-1} - n_i = 1$ το παραπάνω άθροισμα γίνεται

$$\begin{aligned}
&\sum_{i=1}^{L} n_i\,(n_{i-1} - n_i + 1) \\
&\qquad \le \sum_{i=1}^{5m} n_i\,(n_{i-1} - n_i + 1) \\
&\qquad = m(n - m + 1) + 2\sum_{i=2}^{5m} (m - i + 1) \\
&\qquad = mn - m^2 + m + (3m - 15m^2) = mn - 16m^2 + 4m \\
&\qquad < mn.
\end{aligned}$$

Στην "χειρότερη" περίπτωση (ακολουθία Fibonacci) που το μήκος ορισμένων υπολοίπων είναι το ίδιο, θα ισχύει $n_{i-1} - n_i = 0$ και το άθροισμα των χρόνων (εκτός του πρώτου) θα είναι προφανώς μικρότερο από το $2\sum_{i=2}^{5m} (m - i + 1)$. Το ίδιο δε συμβαίνει και αν για κάποιο i, $n_{i-1} - n_i > 1$, διότι στην περίπτωση αυτή $n_i < (m - i + 1)$.

Αξίζει να σημειωθεί πως οι παραπάνω σχέσεις επιβεβαιώνονται και από το *Mathematica*. Για παράδειγμα η τελευταία ανισότητα επιβεβαιώνεται

```
Assuming[m ≥ 1 && n ≥ 1, Refine[m n - 16 m^2 + 4 m < m n]]
```

```
True
```

Λήμμα 2.3.1:
Ένα ακριβές φράγμα στον αριθμό πράξεων για τον υπολογισμό του μκδ των a, b με τον κλασσικό Ευκλείδειο αλγόριθμο είναι $O(mn)$, όπου $n = \lambda(a)$, και $m = \lambda(b)$, τα μήκη των ακεραίων a και b αντίστοιχα.

Ακολουθεί το

Θεώρημα του Lamé (1844):
Ο αριθμός των διαιρέσεων που εκτελούνται για την εύρεση του μκδ δύο ακεραίων,

$a, b \in \mathbb{Z}$, $a \geq b \geq 0$, είναι πάντα μικρότερος από τον αριθμό $5 \cdot \lambda_{10}(b)$, όπου $\lambda_{10}(b)$ είναι ο αριθμός των ψηφίων του μικρότερου ακέραιου b, ως προς την βάση $\beta = 10$.

Απόδειξη:

Ας *θεωρήσουμε την ακολουθία των αριθμών* Fibonacci *που ορίζεται από τις αρχικές τιμές $F_0 = 1$, $F_1 = 2$, και την σχέση $F_n = F_{n-1} + F_{n-2}$.* Η *ακολουθία αυτή είναι διαφορετική από εκείνη που ορίσαμε στις ασκήσεις αλλά μας δίνει τους **μη μηδενικούς**, και **μη επαναλαμβανόμενους** όρους*:

1, 2, 3, 5, 8,💡 13, 21, 34, 55, 89,💡 144, 233, 377, 610, 987,💡 1597, 2584, 4181, 6765,💡 10946 *κτλ*

Το *σήμα* 💡 "*φωτίζει*" *την κύρια ιδέα του θεωρήματος του* Lamé *που αμέσως **αποδεικνύουμε** και που είναι η εξής*: *τα μέλη της ακολουθίας* Fibonacci *που έχουν τον ίδιο αριθμό ψηφίων είναι τουλάχιστον τέσσερα και το πολύ πέντε.*

Πράγματι, *αν παραστήσουμε με t_1 τον πρώτο όρο της ακολουθίας* Fibonacci *με $k + 1$ ψηφία, τότε ισχύει*

$$10^k < t_1 < 2 \cdot 10^k$$

διότι ο όρος t_1 είναι το άθροισμα δύο k-ψήφιων όρων. Ομοίως, *επειδή*

$$\frac{10^k}{2} < t_0 < 10^k$$

και $t_2 = t_0 + t_1$, έχουμε

$$\frac{3}{2} \cdot 10^k < t_2 < 3 \cdot 10^k$$

Π*ροχωρώντας κατά τον ίδιο τρόπο έχουμε*

$$\frac{5}{2} \cdot 10^k < t_3 < 5 \cdot 10^k$$

$$4 \cdot 10^k < t_4 < 8 \cdot 10^k$$

$$\frac{13}{2} \cdot 10^k < t_5 < 13 \cdot 10^k$$

$$\tfrac{21}{2}\cdot 10^k < t_6 < 21\cdot 10^k.$$

Συνεπώς, *οι όροι της ακολουθίας* Fibonacci *με* $k + 1$ *ψηφία είναι τουλάχιστον τέσσερις και το πολύ πέντε.*

Αν *δηλώσουμε με* $F_0, F_1, F_2, \ldots$ *τους όρους της ακολουθίας* Fibonacci, *τότε το πλήθος των όρων μέχρι τον* F_n *θα είναι* ***το πολύ*** $5\cdot\lambda_{10}(F_n)$ - 1. (Τ*ροποποιήστε κατάλληλα το πρόγραμμα υπολογισμού των όρων της ακολουθίας στις ασκήσεις και δοκιμάστε.*) Έτσι *για να βρούμε τον μκδ δύο διαδοχικών όρων* F_n, F_{n+1} *ο αριθμός των διαιρέσεων που θα πρέπει να κάνουμε θα είναι* $< 5\cdot\lambda_{10}(F_n)$—*πράγμα που αποτελεί μερική περίπτωση του θεωρήματος* Lamé.

Έτσι *για παράδειγμα, για να βρουμε τον μκδ των* 13 *και* 8 *όπου* $5\cdot\lambda_{10}(8) = 5\cdot 1$ *κάνουμε τις εξής τέσσερις διαιρέσεις* (*η διαίρεση με πηλίκο την μονάδα δεν μετράει*):

$$\begin{aligned} 13 &= 1\cdot 8 + 5,\\ 8 &= 1\cdot 5 + 3,\\ 5 &= 1\cdot 3 + 2,\\ 3 &= 1\cdot 2 + 1,\\ 2 &= 2\cdot 1 \quad . \end{aligned}$$

Έστω *τώρα ότι θέλουμε να βρούμε τον μκδ των ακεραίων* $a, b \in \mathbb{Z}$, $a \geq b$ *γενικά*. Αν *παραστήσουμε με* $a_{n'+2} = a$, $a_{n'+1} = b$, $a_{n'}, \ldots, a_1, a_0$ *την* ***φθίνουσα*** *ακολουθία των υπολοίπων που προκύπτουν από την εφαρμογή του κλασσικού* Ευκλείδειου *αλγόριθμου, τότε έχουμε* $a_i = q\cdot a_{i-1} + a_{i-2}$ *με* $q \geq 1$. Ας *υποθέσουμε επιπλέον ότι ο ακέραιος* b *βρίσκεται ανάμεσα στους αριθμούς* F_{n+1} *και* F_n. Τότε *θα* ***δείξουμε*** *πως τα υπόλοιπα* $a_{n'}, \ldots, a_1, a_0$ *βρίσκονται ανάμεσα στα διαστήματα που σχηματίζονται από διαδοχικά μέλη της ακολουθίας* $F_{n+1}, F_n, \ldots, F_1, F_0$. Προσέξτε *πως εν γένει* $n \neq n'$.

Ας *πάρουμε πρώτα την περίπτωση* $\boldsymbol{q = 1}$ *για* **όλες** *τις διαιρέσεις και ας υποθέσουμε πως υπάρχουν δύο υπόλοιπα* a_h *και* a_{h-1} *στο ίδιο διάστημα* (F_k, F_{k-1}). Δηλαδή *έχουμε* $F_k > a_h > a_{h-1} > F_{k-1}$. Τότε, *επειδή* $F_k = F_{k-1} + F_{k-2}$ *και επειδή*

F_{k-1} *εμφανίζεται στο άθροισμα μόνο μία φορά, θα έχουμε* $a_h = a_{h-1} + a_{h-2}$ *και* $F_{k-2} > a_{h-2}$. *Δηλαδή* ***δεν*** *θα υπάρχει υπόλοιπο στο διάστημα* (F_{k-1}, F_{k-2}). *Στο ίδιο συμπέρασμα καταλήγουμε και για τις περιπτώσεις* $F_k = a_h > a_{h-1} > F_{k-1}$ *και* $F_k > a_h > a_{h-1} = F_{k-1}$.

Άρα, αν $\boldsymbol{q = 1}$ *για* **όλες** *τις διαιρέσεις του κλασσικού* Ευκλείδειου *αλγόριθμου, τότε τα υπόλοιπα κατανέμονται κατά τέτοιο τρόπο (ανάμεσα στους όρους της* ***φθίνουσας*** *ακολουθίας* Fibonacci) *ώστε ποτέ δεν υπάρχουν περισσότερα από δύο υπόλοιπα σε ένα διάστημα από δύο διαδοχικούς όρους. Επιπλέον, κάθε διάστημα με δύο υπόλοιπα ακολουθείται από ένα διάστημα χωρίς κανένα υπόλοιπο.*

Στην συνέχεια θεωρούμε την περίπτωση $\boldsymbol{q > 1}$ *για μία τουλάχιστον διαίρεση. Έστω λοιπόν ότι έχουμε* $a_i = 2 \cdot a_{i-1} + a_{i-2}$ $(q = 2)$ *και ας υποθέσουμε πως το* a_i *βρίσκεται ανάμεσα στους όρους* F_{j+1} *και* F_j. *Τότε έχουμε*

$$a_i - 2 \cdot a_{i-1} > 0 \quad \textit{και} \quad 2 \cdot F_j - F_{j+1} > 0$$

και προσθέτοντάς τα

$$2(F_j - a_{i-1}) - (F_{j+1} - a_i) > 0.$$

Από την τελευταία ανισότητα συμπεραίνουμε πως $F_j - a_{i-1} > 0$ *ή* $F_j > a_{i-1}$. *Ας δούμε που βρίσκεται το* a_{i-1}. *Αν* $F_{j-1} > a_{i-1}$, *τότε το διάστημα* (F_j, F_{j-1}) *θα είναι* "*άδειο.*" *Αν πάλι* $a_{i-1} > F_{j-1}$, *τότε θα έχουμε* $F_{j-2} > a_{i-2}$ *επειδή ισχύουν*: $F_{j+1} = 2 \cdot F_{j-1} + F_{j-2}$ *(απλό στην απόδειξη),* $a_i = 2 \cdot a_{i-1} + a_{i-2}$ *(εξ υποθέσεως) και* $F_{j+1} > a_i$ *(επίσης εξ υποθέσεως). Συνεπώς, το διάστημα* (F_{j-1}, F_{j-2}) *θα είναι* "*άδειο.*"

Άρα, αν $\boldsymbol{q > 1}$ *για μία τουλάχιστον διαίρεση του κλασσικού* Ευκλείδειου *αλγόριθμου, τότε θα υπάρχει τουλάχιστον ένα διάστημα της ακολουθίας* Fibonacci *που* ***δεω*** *θα περιέχει κανένα υπόλοιπο και αυτό* ***δεν θα εξισορροπηθεί*** *από άλλο διάστημα που θα περιέχει δύο υπόλοιπα.*

Βλέπουμε λοιπόν πως για να είναι $n = n'$, *δηλαδή για να έχει η ακολουθία των υπολοίπων* $a_{n'}, \ldots, a_1, a_0$ *τον ίδιο αριθμό μελών με την ακολουθία* $F_n, \ldots, F_1, F_0$ *πρέπει* $\boldsymbol{q = 1}$ *για* **όλες** *τις διαιρέσεις του κλασσικού* Ευκλείδειου *αλγόριθμου και* a_0

= 1. Δηλαδή όπως ορίζεται η ακολουθία Fibonacci για τους μη μηδενικούς, και μη επαναλαμβανόμενους όρους $F_0 = 1$, $F_1 = 2$, $F_2 = 3$, … έτσι πρέπει και στην ακολουθία των υπολοίπων να είναι $a_0 = 1$.

Προσέξτε όμως πως δεν μπορεί να είναι $a_1 = 2$ διότι τότε οι δύο ακολουθίες θα ήταν οι ίδιες και b θα ήταν ίσο με το F_{n+1}, πράγμα που δεν ισχύει. Συνεπώς, η τιμή του a_1 θα είναι τουλάχιστον 3, και η ακολουθία των υπολοίπων θα έχει τελικά έναν η περισσότερους όρους **λιγώτερους** από την ακολουθία Fibonacci. //

Ενώ ο χρόνος υπολογισμού του μκδ **δεν αλλάζει**, εν τούτοις υπάρχουν και άλλα, καλλίτερα του $5 \cdot \lambda_{10}(b)$, φράγματα του αριθμού των διαιρέσεων που εκτελούνται για την εύρεση του μκδ δύο ακεραίων. Αναφέρουμε το καλλίτερο ενώ ένα άλλο το αφήνουμε για τις ασκήσεις.

Από το θεώρημα του Lamé ξέρουμε πως το μεγαλύτερο Ευκλείδειο μήκος δύο ακεραίων, $a, b \in \mathbb{Z}$, $a \geq b \geq 0$, είναι όταν στην εύρεση του μκδ τους τα πηλίκα είναι όλα 1, δηλαδή όταν οι ακέραιοι αυτοί είναι διαδοχικοί αριθμοί Fibonacci. Επιπλέον, από τις ασκήσεις ξέρουμε ότι αν η ακολουθία Fibonacci ορίζεται από τις αρχικές τιμές $F_0 = 0$, $F_1 = 1$, και την σχέση $F_n = F_{n-1} + F_{n-2}$, τότε $F_n \approx \frac{\phi_+^n}{\sqrt{5}}$, όπου $\phi_+ = \frac{1+\sqrt{5}}{2} \approx 1.618$ είναι η **χρυσή τομή** (golden ratio). Άρα, αν $\{a, b\} = \{F_{n+1}, F_n\}$ το Ευκλείδειο μήκος τους είναι

$$L = n - 1 \approx \log_{\phi} \sqrt{5}\, b - 1,$$

που είναι μία πάρα πολύ καλή προσέγγισή του.

Παράδειγμα:
Έστω $\{a, b\} = \{F_{n+1}, F_n\} = \{21, 13\}$. Το θεώρημα του Lamé μας δίνει το φράγμα $5 \cdot \lambda_{10}(13) = 10$ για τον αριθμό των διαιρέσεων, ενώ ο τύπος με την χρυσή τομή το φράγμα

```
Log[GoldenRatio, √5 13.] - 1 // Round
```

```
6
```

Κλείνουμε με το εξής ενδιαφέρον αποτέλεσμα του Dirichlet, το οποίο δίνουμε και σαν άσκηση. Αν a, και b είναι δύο τυχαία επιλεγμένοι ακέραιοι τότε η πιθανότητα ότι $\gcd(a, b) = 1$ είναι $\frac{6}{\pi^2} \approx 0.60793$.

■ 2.3.2 $R = \mathbb{Q}[x]$ ή $R = F[x]$, όπου F πεπερασμένο σώμα

Θα αναλύσουμε τώρα τον Ευκλείδειο αλγόριθμο για την περίπτωση $a, b \in \mathbb{Q}[x]$ (ή $a, b \in F[x]$, όπου F πεπερασμένο σώμα) και $n = \deg(a) \geq \deg(b) = m \geq 0$. Δηλαδή, οι συντελεστές των πολυωνύμων είναι είτε ρητοί πολλαπλής ακριβείας είτε ακέραιοι απλής ακριβείας (από το πεπερασμένο σώμα).

Όπως και στην περίπτωση των ακεραίων, έτσι και εδώ βλέπουμε πως όλη η "δουλειά" γίνεται στο δεύτερο βήμα του κλασσικού Ευκλείδειου αλγορίθμου που υπάρχει ένας βρόγχος της μορφής:

$$\textbf{while } b \neq 0 \textbf{ do } \{a, b\} \longleftarrow \{b, \mathrm{rem}(a, b)\}.$$

Αν θέσουμε $a_0 = a$ και $a_1 = b$, από τις διαρέσεις αυτές βρίσκονται τα υπόλοιπα a_2, …, a_L τα οποία τώρα είναι πολυώνυμα βαθμών $n_0 = n \geq n_1 = m > n_2 > \ldots > n_L$ αντίστοιχα. Η ακολουθία των υπολοίπων αυτών παίζει σημαντικότατο ρόλο και ονομάζεται **ακολουθία πολυωνυμικών υπολοίπων ή απυ** (polynomial remainder sequence or prs). Προφανώς, ο αριθμός L των διαιρέσεων φράσσεται από $L \leq \deg(b) + 1$, και ο βαθμός κάθε πηλίκου είναι $\deg(q_i) = n_{i-1} - n_i$, $1 \leq i \leq L$.

Όπως έχουμε δει, η διαίρεση με υπόλοιπο του πολυωνύμου a_{i-1} βαθμού n_{i-1} με το πολυώνυμο με μοναδιαίο κύριο συντελεστή a_i βαθμού n_i, εκτελείται με το πολύ $2n_i(n_{i-1} - n_i + 1)$ πράξεις στο R. Για την κανονικοποίηση όλων των πολυωνύμων (δηλαδή για να κάνουμε το a_i πολυώνυμο με μοναδιαίο κύριο συντελεστή) απαιτούνται n_i πράξεις. Άρα, για να βρούμε τον μκδ δύο πολυωνύμων (δηλαδή για τον Ευκλείδειο αλγόριθμο) απαιτούνται

$$\sum_{i=1}^{L} 2\, n_i\, (n_{i-1} - n_i + 1) + \sum_{i=0}^{L} n_i$$
$$= 2\sum_{i=1}^{L} n_i\, (n_{i-1} - n_i) + n_0 + 3\sum_{i=1}^{L} n_i$$

πράξεις στον δακτύλιο R.

Εξετάζουμε πρώτα την περίπτωση που η ακολουθία των πολυωνυμικών υπολοίπων είναι **πλήρης**, δηλαδή $n_i = n_{i-1} - 1 = \cdots = m - i + 1$, για $2 \le i \le L \le m + 1$. Στην περίπτωση αυτή ο βαθμός κάθε υπολοίπου a_i είναι μικρότερος κατά μονάδα από τον βαθμό του υπολοίπου a_{i-1}, οπότε $n_{i-1} - n_i = 1$ και το παραπάνω άθροισμα γίνεται

$$\begin{aligned} &2m(n - m) + n + 3m + 5\textstyle\sum_{i=2}^{m+1} (m - i + 1) \\ &\quad = 2mn - 2m^2 + n + 3m + \tfrac{5}{2}(m^2 - m) = \tfrac{m}{2} + \tfrac{m^2}{2} + 2mn + n \\ &\quad \le \tfrac{m}{2} + \tfrac{mn}{2} + 2mn + n \\ &\quad \le \tfrac{5}{2} mn + \tfrac{3}{2} n. \end{aligned}$$

Η παραπάνω ανισότητα (που όπως και για τους ακεραίους ισχύει και για την περίπτωση **μη πλήρους** απυ, δηλαδή $n_{i-1} - n_i > 1$) επιβεβαιώνεται και από το *Mathematica*.

```
Assuming[m ≥ 1 && n ≥ 1 && n ≥ m, Refine[m/2 + m^2/2 + 2 m n + n ≤ 5/2 m n + 3/2 n]]
```

```
True
```

Λήμμα 2.3.2:
Ένα ακριβές φράγμα στον αριθμό πράξεων για τον υπολογισμό του μκδ των πολυωνύμων a, b με τον κλασσικό Ευκλείδειο αλγόριθμο είναι $O(mn)$, όπου $n = \deg(a)$, και $m = \deg(b)$, $n \ge m$.

Σε αντίθεση με τους ακεραίους διακρίνουμε τις εξής περιπτώσεις:

α. Αν $R = \mathbb{Q}[x]$ (οπότε έχουμε αριθμητική πολλαπλής ακριβείας), και $B = \max\{||a||_\infty, ||b||_\infty\}$, η μέγιστη νορμ, τότε ο χρόνος εύρεσης του μκδ δύο πολυωνύμων γίνεται $O(mn \log^2 B) = O(n^2 \log^2 B)$, όπου $\log^2 B = (\log B)^2$.

β. Αν $R = F[x]$, όπου F *πεπερασμένο σώμα* (οπότε *έχουμε αριθμητική απλής ακριβείας*), *τότε ο χρόνος εύρεσης του μκδ δύο πολυωνύμων είναι αυτός που προαναφέραμε*, *δηλαδή* $O(mn)$.

Ασκήσεις

1: α. *Δείξτε πως η νορμ* $n : \mathbb{Z}[i] \to \mathbb{N}$ *με* $n(\alpha) = \alpha\overline{\alpha}$ *στον δακτύλιο των μιγαδικών ακεραίων* (Gaussian integers) $\mathbb{Z}[i]$ *είναι μία Ευκλείδεια συνάρτηση* (*συνάρτηση μεγέθους*). (Υπόδειξη: *Την λύση την δίνει το ακριβές πηλίκο της διαίρεσης στο* $\mathbb{C}$ *των μιγαδικών ακεραίων* α *και* $\beta \in \mathbb{Z}[i]$.)

β. *Δείξτε ότι οι μονάδες στον δακτύλιο* $\mathbb{Z}[i]$ *είναι τα στοιχεία με νορμ* 1. *Ποια είναι αυτά*;

γ. *Σημειώστε πως η* normal$(a + ib) = |a| + i|b|$, $\forall a,b \in \mathbb{Z}$ *δεν είναι κανονική μορφή των μιγαδικών ακεραίων*, *διότι εταιρικά στοιχεία δεν έχουν την ίδια κανονική μορφή*. *Δείξτε πως η κανονική μορφή* normal$(a) = |a|$ *των ακεραίων*, $\forall a \in \mathbb{Z}$, *δεν επεκτείνεται στους μιγαδικούς ακεραίους διότι δεν πληρείται η πολλαπλασιαστική ιδιότητα των μορφών αυτών*. (Υπόδειξη: *Ποια είναι η κανονική μορφη* normal$((1 + i)^2)$;)

2: α. *Αναπτύξτε έναν Ευκλείδειο αλγόριθμο για την εύρεση του μκδ δύο μιγαδικών ακεραίων* (Gaussian integers). *Για* $a,b \in \mathbb{Z}[i]$ *ως* gcd(a, b) *ορίζεται ο μιγαδικός ακέραιος* $d \in \mathbb{Z}[i]$ *που διαιρεί τους a και b και έχει την μέγιστη νορμ* $n(d)$. *Προσέξτε πως η νορμ του μιγαδικού ακεραίου d δεν αλλάζει αν τον πολλαπλασιάσουμε με* ± 1, *ή* $\pm i$, *και έτσι η κάθε μία από αυτές τις* 4 *περιπτώσεις θεωρείται ως "ο μκδ" των a, b*. *Παρ' όλα όσα είπαμε στην άσκηση* **1γ**, *ακολουθούμε το Mathematica και επιλέγουμε για "κανονική μορφή" του d αυτή με θετικό το πραγματικό και φανταστικό μέρος*.

Υπόδειξη: Η *διαίρεση στον δακτύλιο* $\mathbb{Z}[i]$ *γίνεται βάσει της σχέσης*:

$$\frac{c+i\,d}{a+i\,b} = \frac{(c+i\,d)\,(a-i\,b)}{a^2+b^2}\,.$$

Επίσης, *κάθε μιγαδικός αριθμός μπορεί να γραφτεί σαν άθροισμα ενός μιγαδικού ακεραίου και ενός μιγαδικού αριθμού του οποίου και το πραγματικό και το φανταστικό μέρος είναι μεταξύ* $\frac{-1}{2}$ *και* $\frac{1}{2}$. *Επομένως μπορούμε να διαιρέσουμε*

έναν μιγαδικό ακέραιο a με έναν άλλο b και να βρούμε ένα πηλίκο και ένα υπόλοιπο $\in \mathbb{Z}[i]$, όπου η νορμ του υπολοίπου είναι μικρότερη της νορμ του b.

β. Υπολογίστε τον gcd(12277, 399+20i) (κλασσικό και επεκταμένο) και συγκρίνετε το αποτέλεσμα με αυτό του *Mathematica*.

```
GCD[12277, 399 + 20 i]
```

```
89 + 66 i
```

```
ExtendedGCD[12277, 399 + 20 i]
```

```
{89 + 66 i, {-1 - 2 i, 34 + 60 i}}
```

3: Δείξτε πως αν τουλάχιστον ένα από τα $a, b \in \mathbb{Z}$ είναι διάφορα του μηδενός, τότε ο ακέραιος $\frac{|ab|}{\gcd(a,b)}$ είναι το ελάχιστο κοινό πολλαπλάσιο των a και b, lcm(a, b).

4: (Σειρές Farey) Στην ιδιότητα α του Θεωρήματος ιδιοτήτων του μκδ, και για την ειδική περίπτωση που gcd(a, b) = 1 και $0 < a \le b$, μπορούμε να υπολογίσουμε από την σειρά Farey F_b τους ακέραιους x και y έτσι ώστε gcd(a, b) = $ax + by$.

***α.* Ορισμός:**
Η σειρά Farey τάξης n, που συμβολίζεται με F_n, είναι το σύνολο όλων των κλασμάτων $\frac{a}{b}$ μεταξύ 0 και 1, διατεταγμένων σε αύξουσα τάξη έτσι, με gcd(a, b) = 1 και έτσι ώστε οι παρανομαστές να είναι $b \le n$.

Παράδειγμα:
Η σειρά Farey τάξης 3 είναι F_3: $\frac{0}{1}, \frac{1}{3}, \frac{1}{2}, \frac{2}{3}, \frac{1}{1}$.

***β.* Κατασκευή των σειρών Farey:**
Για την κατασκευή της σειράς F_n αρχίζουμε με $F_1 = \frac{0}{1}, \frac{1}{1}$ και μετά επαναλαμβάνουμε την ακόλουθη πράξη τόσες φορές όσες χρειάζεται (οι παρανομαστές πρέπει να είναι $\le n$):

Εισάγετε το κλάσμα $\frac{a+a'}{b+b'}$ ανάμεσα στα διαδοχικά κλάσματα $\frac{a}{b}$ και $\frac{a'}{b'}$.

Παράδειγμα:

Για τον υπολογισμό της F_3, της σειράς Farey τάξης 3, το πρώτο βήμα μας δίνει ένα καινούργιο κλάσμα ανάμεσα στα $\frac{0}{1}$ και $\frac{1}{1}$:

$$\frac{0}{1}, \frac{1}{2}, \frac{1}{1}$$

ενώ το δεύτερο (και τελευταίο) βήμα μας δίνει δύο επιπλέον κλάσματα:

$$F_3: \ \frac{0}{1}, \frac{1}{3}, \frac{1}{2}, \frac{2}{3}, \frac{1}{1}$$

Σημειώστε πως για να κατασκευάσουμε την σειρά F_n από την σειρά F_{n-1} εισάγουμε το κλάσμα $\frac{a+a'}{n}$ ανάμεσα στα διαδοχικά κλάσματα $\frac{a}{b}$ και $\frac{a'}{b'}$ για τα οποία ισχύει $b + b' = n$, δηλαδή το άθροισμα των παρανομαστών είναι n. Έτσι, για να υπολογίσουμε την σειρα F_4 από την σειρά F_3 εισάγουμε τα κλάσματα $\frac{1}{4}$ και $\frac{3}{4}$ σύμφωνα με τον παραπάνω κανόνα:

$$F_4: \ \frac{0}{1}, \frac{1}{4}, \frac{1}{3}, \frac{1}{2}, \frac{2}{3}, \frac{3}{4}, \frac{1}{1}$$

γ. Αποδείξτε πως αν $\frac{a}{b} < \frac{a'}{b'}$ και όλες οι τιμές είναι μη αρνητικές, τότε

$$\frac{a}{b} < \frac{a+a'}{b+b'} < \frac{a'}{b'}.$$

δ. Αποδείξτε επαγωγικά (by induction) πως αν $\frac{a}{b}$ και $\frac{a'}{b'}$ είναι δύο διαδοχικά κλάσματα (σε οποιοδήποτε στάδιο της κατασκευής) τότε

$$a'b - ab' = 1.$$

(Υπόδειξη: Αρχικά ισχύει η ισότητα αυτή. Όταν ένα νέο κλάσμα εισάγεται, ισχύει η ανισότητα του μέρους γ. Αποδείξτε πως η ισότητα ισχύει σε όλα τα στάδια της κατασκευής.)

ε. Κατασκευάστε την σειρά F_7 και βρείτε δύο λύσεις της εξίσωσης $5s + 7t = 1$.

5: Αποδείξτε πως για $a, b, c \in \mathbb{Z}$, ισχύουν οι εξής ιδιότητες για τον μέγιστο κοινό διαιρέτη:

- $\gcd(a, b) = |a| \iff a \mid b$.

- gcd(a, b) = gcd(b, a) (*μεταθετική*).
- gcd(a, gcd(b, c)) = gcd(gcd(a, b), c) (*προσεταιριστική*).
- gcd($c \cdot a$, $c \cdot b$) = $|c| \cdot$gcd(a, b) (*επιμεριστική*).

6: Ο *κλασσικός* Ευκλείδειος *αλγόριθμος για ακεραίους είναι δυνατόν να επιταχυνθεί αν επιτρέπουμε διαιρέσεις με αρνητικό υπόλοιπο*. Δηλαδή *από τις σχέσεις* $a = b*c + d$ *και* $a = b*(c+1) - e$ *επιλέγουμε εκείνη που δίνει το υπόλοιπο με την μικρότερη απόλυτη τιμή*. Γ*ια παράδειγμα*, *προσέξτε πως ο αριθμός των διαιρέσεων είναι* 3 ό*ταν επιτρέπονται αρνητικά υπόλοιπα*:

```
612 == (342) * 2 + (-72)
342 == (-72) * 5 + (-18)
-72 == (-18) * 4 + 0
```

και 5 ό*ταν όλα τα υπόλοιπα είναι όλα θετικά*:

```
612 == 342 * 1 + 270
342 == 270 * 1 + 72
270 == 72 * 3 + 54
 72 == 54 * 1 + 18
 54 == 18 * 3 + 0
```

Τ*ροποποιήστε το πρόγραμμα υπολογισμού του μκδ δύο θετικών ακεραίων* gcdPos-Int[], *ώστε να επιτρέπονται διαιρέσεις με αρνητικό υπόλοιπο*.

7: Α*ν* $a_i = a_{i+1}\, q_{i+1} + a_{i+2}$, $i = 0, \ldots, k\text{-}1$ *είναι η ακολουθία διαιρέσεων για την εύρεση του μκδ των ακεραίων* a_0, a_1 *αποδείξτε την ανισότητα*

$$a_{i+2} < \tfrac{a_i}{2}, \quad i \geq 1.$$

Με *άλλα λόγια*, *η τιμή των υπολοίπων όχι μόνον μειώνεται*, *αλλά μειώνεται σχετικά γρήγορα*. (Υ*πόδειξη*: Ε*ξετάστε τις δύο πιθανές ανισότητες μεταξύ των* a_{i+1} *και* $\frac{a_i}{2}$.)

8: Χ*ρησιμοποιώντας τον κλασσικό και επεκταμένο* Ευκλείδειο *αλγόριθμο βρείτε τον μκδ των παρακάτω ζευγών ακεραίων*: α. 34, 21. β. 136, 51. γ. 481, 325. δ. 8771, 3206.

9: Δυαδικός, *αναγωγικός* (recursive) *αλγόριθμος για την εύρεση του* $d = \gcd(a, b)$ *δύο θετικών ακεραίων* a, b. Χωρίς *περιορισμό της γενικότητας μπορούμε να θεωρήσουμε μόνον θετικούς ακεραίους διότι όπως ξέρουμε* $\gcd(a, b) = \gcd(|a|, |b|)$.

Αλγόριθμος: Δυαδικός αλγόριθμος για την εύρεση του $d = \gcd(a, b)$.

Είσοδος: $a, b \in \mathbb{Z}_{>0}$.

Έξοδος: Ο *μέγιστος κοινός διαιρέτης* $d = \gcd(a, b) \in \mathbb{Z}_{>0}$.

==========

1. **If** $a = b$ **then return** $d \longleftarrow a$.

2. (* Κ*αι οι δύο ακέραιοι είναι άρτιοι* *)

If $\text{rem}(a, 2) = \text{rem}(b, 2) = 0$ **then return** $d \longleftarrow 2 \cdot \gcd(\frac{a}{2}, \frac{b}{2})$.

3. (* Έ*νας από τους δύο ακεραίους είναι άρτιος* *)

Which

$\text{rem}(a, 2) = 0$**, return** $d \longleftarrow \gcd(\frac{a}{2}, b)$,

$\text{rem}(b, 2) = 0$**, return** $d \longleftarrow \gcd(a, \frac{b}{2})$.

4. (* Οι *δύο ακέραιοι είναι περιττοί και άνισοι* *)

Which

$\text{rem}(a, 2) = \text{rem}(b, 2) = 1$ **and** $a > b$**, return** $d \longleftarrow \gcd(a - b, b)$,

$\text{rem}(a, 2) = \text{rem}(b, 2) = 1$ **and** $a < b$**, return** $d \longleftarrow \gcd(a, b - a)$.

==========

α. Π*ρογραμματίστε τον αλγόριθμο αυτόν* (*λαμβάνοντας υπ' όψη πως για τους ακεραίους* rem() = Mod[]) *και δοκιμάστε τα παραδείγματα της άσκησης* 7. Π*οιος είναι ο χρόνος υπολογισμού του δυαδικού αλγορίθμου*;

β. Τ*ροποποιήστε τον αλγόριθμο ώστε να υπολογίζει* $s,t \in \mathbb{N}$ *έτσι ώστε* $sa + tb = \gcd(a, b)$.

10: Έ*στω ότι* F_n *και* F_{n+1} *είναι διαδοχικοί όροι της ακολουθίας* Fibonacci (*που ορίζεται από τις αρχικές τιμές* $F_0 = 0$, $F_1 = 1$, *και την σχέση* $F_n = F_{n-1} + F_{n-2}$). Δ*είξτε ότι* $\gcd(F_{n+1}, F_n) = 1$. Π*όσες διαιρέσεις απαιτούνται για την εύρεση του μκδ*; Α*κολουθεί ένα αναγωγικό πρόγραμμα υπολογισμού των όρων της ακολουθίας*

Fibonacci, *όπου για επιτάγχυνση των υπολογισμών οι υπολογισθέντες όροι αποθηκεύονται*.

```
F[0] = 0; F[1] = 1;
F[n_] := F[n] = F[n - 1] + F[n - 2]
```

Έτσι *υπολογίζουμε τους πρώτους* 13 *μη μηδενικούς όρους*

```
Table[F[n], {n, 13}]
```

```
{1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233}
```

Π*ροσέξτε πως* $\gcd(F_m, F_n) = F_{\gcd(m,n)}$.

```
GCD[F[15], F[5]] == F[GCD[15, 5]]
```

```
True
```

11: Γ*ια την ακολουθία* Fibonacci *που ορίζεται όπως και στην προηγούμενη άσκηση αποδείξτε τα εξής*:

α. **(**Θεώ*ρημα του* Lucas) Α*ποδείξτε πως* $\gcd(F_m, F_n) = F_{\gcd(m,n)}$. (Υπόδ*ειξη*: Ε*φαρμόστε τον* Ευκλείδειο *αλγόριθμο στους ακέραιους* m *και* n, *και χρησιμοποιήστε την ταυτότητα* $F_{n+k} = F_k F_{n+1} + F_{k-1} F_n$ *από την οποία προκύπτει ότι* $F_{k\,n}$ *είναι πολλαπλάσιο του* F_n.)

β. (Λήμμα *του* Yuri Matiyasevich. Χ*ρησιμοποιήθηκε στην περίφημη απόδειξη πως δεν υπάρχει αλγόριθμος που να αποφασίζει αν μία πολυωνυμική εξίσωση πολλών μεταβλητών με ακέραιους συντελεστές έχει ακέραια λύση*.) Α*ν* $n > 2$, *ο αριθμός* Fibonacci F_m *είναι πολλαπλάσιο του* F_n^2 *εάν και μόνον εάν το* m *είναι πολλαπλάσιο του* nF_n.

γ. Η *ακολουθία* Lucas *ορίζεται από τις αρχικές τιμές* $L_0 = 0$, $L_1 = 1$, *και την σχέση* $L_n = aL_{n-1} + L_{n-2}$, *για σταθερά* $a \in \mathbb{N}$. Ι*σχύει* $\gcd(L_m, L_n) = L_{\gcd(m,n)}$;

12: ***α.*** Α*ν* $a,b \in \mathbb{Z}$ *με* $a \geq b \geq 1$, *αποδείξτε πως* $\text{rem}(a, b) \leq \frac{a-1}{2}$. (Υπόδ*ειξη*: Π*ροσέξτε πως* $\text{rem}(a, b) \leq \min(a - b, b - 1)$ *και καθορίστε την σχέση των* $a - b$ *και* $b - 1$ *στις δύο περιπτώσεις* $b \leq \frac{a+1}{2}$ *και* $b > \frac{a+1}{2}$. Βλέπε *και την βιβλιογραφία*.)

β. Χ*ρησιμοποιώντας* (*επαγωγικά*) *το πρώτο μέρος αποδείξτε πως για να βρούμε*

τον μκδ των $a,b \in \mathbb{Z}_{>0}$ *ο αριθμός των διαιρέσεων που εκτελούνται κατά την διάρκεια του* Ευκλειδείου *αλγορίθμου είναι είναι* $\le \lfloor 2 \log_2 M \rfloor + 1$, *όπου* $M = \max(a, b)$.

13: *Έστω ότι* F_n *και* F_{n+1} *είναι διαδοχικοί όροι της ακολουθίας* Fibonacci *που ορίζεται από τις αρχικές τιμές* $F_0 = 0$, $F_1 = 1$, *και την σχέση* $F_n = F_{n-1} + F_{n-2}$. *Αποδείξτε ότι*

$$F_n = \frac{1}{\sqrt{5}} (\phi_+^n - \phi_-^n) \quad \textit{για } n \in \mathbb{N}$$

όπου $\phi_+ = \frac{1+\sqrt{5}}{2} \approx 1.618$ *είναι η* ***χρυσή τομή*** (golden ratio) *και* $\phi_- = \frac{1-\sqrt{5}}{2} \approx -0.618$. *Συμπεράνετε ότι* F_n *είναι ο πλησιέστερος ακέραιος στον* $\frac{\phi_+^n}{\sqrt{5}}$ *για όλα τα* n. (*Το γράμμα* ϕ *είναι προς τιμήν του* Φειδία.)

14: *Κάνοντας χρήση του αποτελέσματος* $H_\infty^{(2)} = \sum_{k>1} k^{-2} = \frac{\pi^2}{6}$, $H_\infty^{(4)} = \frac{\pi^4}{90}$, $H_\infty^{(6)} = \frac{\pi^6}{945}$, $H_\infty^{(8)} = \frac{\pi^8}{9450}$ (*βλέπε και το άρθρο του* Stark) *υπολογίστε την πιθανότητα δύο, τέσσερις, έξι και οχτώ* (*αντίστοιχα*) *ακέραιοι να είναι πρώτοι μεταξύ τους*.

- **Λύσεις ορισμένων ασκήσεων**

Λύση της άσκησης 2β:

Παρουσιάζουμε *δύο προγράμματα*. Το *πρώτο πρόγραμμα δουλεύει επαναληπτικά*

```
gcdGaussianInteger1[a_, b_] /;
  (Re[a] ∈ Integers && Im[a] ∈ Integers || a ∈ Integers) &&
   (Re[b] ∈ Integers && Im[b] ∈ Integers || b ∈ Integers) := Module[
  {a0 = a, a1 = b},
  While[a1 ≠ 0, quo = Round[N[a0/a1]]; rem = a0 - quo a1;
   {a0, a1} = {a1, rem}]; Complex[Abs[Re[a0]], Abs[Im[a0]]]
 ]
```

και το αποτέλεσμα είναι

```
gcdGaussianInteger1[12277, 399 + 20 i]
```

```
89 + 66 i
```

Το *δεύτερο πρόγραμμα δουλεύει επαγωγικά*

```
gcdGaussianInteger2[a_, b_] := Module[
  {q = a/b},
  gcdGaussianInteger2[b, a - b (Round[Re[q]] + I Round[Im[q]])]
 ]

gcdGaussianInteger2[a_, 0 _] := a
```

και το αποτέλεσμα είναι

```
gcdGaussianInteger2[12277, 399 + 20 i]
```

```
-89 - 66 i
```

Π*ροσέξτε πως αν και τα δύο αποτελέσματα διαφέρουν κατά μονάδα*, *έχουμε*

```
Norm[89 + 66 i] == Norm[-89 - 66 i]
```

```
True
```

Λύση της άσκησης 11*β* (Matiyasevich) **:**

Κ*ατ' αρχάς θα αποδείξουμε ορισμένες χρήσιμες ταυτότητες μελών της ακολουθίας* Fibonacci.

***α*. Τ*αυτότητα του* Cassini (1680):** $F_{n+1}F_{n-1} - F_n^2 = (-1)^n$, $n > 0$. (Υπ*άρχει και σχετικό γεωμετρικό παράδοξο που βασίζεται στην ταυτότητα αυτή και βάσει του οποίου τα* $8\times 8 = 64$ *τετράγωνα της σκακιέρας γίνονται* $5\times 13 = 65$!)

Απ*όδειξη*:

Σ*χηματίζουμε τον πίνακα*

$$M_n = \begin{pmatrix} F_{n-1} & F_n \\ F_n & F_{n+1} \end{pmatrix},\ n \geq 1.$$

(Προσέξτε πως δεν υπάρχει πίνακας M_0). Κατόπιν, προσθέτουμε την δεύτερη σειρά στην πρώτη, αντικαθιστούμε την πρώτη σειρά με το άθροισμα, ανταλλάζουμε τις σειρές και αποκτούμε τον πίνακα

$$M_{n+1} = \begin{pmatrix} F_n & F_{n+1} \\ F_{n+1} & F_{n+2} \end{pmatrix}.$$

Επειδή έχουμε $\det(M_{n+1}) = -\det(M_n)$, δηλαδή οι ορίζουσες διαφέρουν μόνον στο πρόσημο, και $\det(M_1) = \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix} = -1$, έπεται πως

$$\det(M_n) = (-1)^n$$

ή

$$F_{n+1}F_{n-1} - F_n^2 = (-1)^n,\ n > 0.\ //$$

β. $F_{n+k} = F_k F_{n+1} + F_{k-1} F_n$.

Απόδειξη:

Με επαγωγή ως προς το n ή το k. //

Αν στην ταυτότητα ***β*** θέσουμε $k = n$, τότε προκύπτει

$$F_{2n} = F_n F_{n+1} + F_{n-1} F_n$$

από την οποία βλέπουμε πως το F_{2n} είναι πολλαπλάσιο του F_n. Ομοίως,

$$F_{3n} = F_{2n} F_{n+1} + F_{2n-1} F_n$$

και μπορούμε να συμπεράνουμε πως το F_{3n} είναι πολλαπλάσιο του F_n. Επαγωγικά έχουμε ότι το $\boldsymbol{F_{kn}}$ ***είναι πολλαπλάσιο του*** $\boldsymbol{F_n}$:

$$F_{kn} = F_{kn} F_{n+1} + F_{kn-1} F_n.$$

Για να ***αποδείξουμε*** τώρα το λήμμα του Matiyasevich εξετάζουμε την ακολουθία $\mathrm{mod}(F_{kn}, F_n^2)$, $k = 1, 2, 3, \ldots$ για να δούμε πότε είναι $\mathrm{mod}(F_{kn}, F_n^2) = 0$. (Ξέρουμε από τα παραπάνω πως το m πρέπει να έχει την μορφή kn για να είναι $\mathrm{mod}(F_m, F_n) = 0$.) Αρχικά έχουμε $\mathrm{mod}(F_n, F_n^2) = F_n$, που δεν είναι μηδέν. Μετά έχουμε

$$F_{2n} = F_n F_{n+1} + F_{n-1} F_n \equiv 2F_n F_{n+1} \pmod{F_n^2},$$

από την $F_{n+k} = F_k F_{n+1} + F_{k-1} F_n$ και το γεγονός ότι $F_{n+1} \equiv F_{n-1} \pmod{F_n}$. Ομοίως,

$$F_{2n+1} = F_{n+1}^2 + F_n^2 \equiv F_{n+1}^2 \pmod{F_n^2}.$$

Η ισοδυναμία αυτή μας επιτρέπει να υπολογίσουμε (πάντα με την βοήθεια της $F_{n+k} = F_k F_{n+1} + F_{k-1} F_n$)

$$F_{3n} = F_n F_{2n+1} + F_{n-1} F_{2n} \equiv F_n F_{n+1}^2 + F_{n+1}(2F_n F_{n+1}) = 3F_n F_{n+1}^2 \pmod{F_n^2}.$$

και

$$F_{3n+1} = F_{n+1} F_{2n+1} + F_n F_{2n} \equiv F_{n+1}^3 + F_n(2F_n F_{n+1}) \equiv F_{n+1}^3 \pmod{F_n^2}.$$

Γενικά βρίσκουμε με επαγωγή στο k πως

$$F_{kn} \equiv kF_n F_{n+1}^{k-1} \pmod{F_n^2}.$$

και

$$F_{kn+1} \equiv F_{n+1}^k \pmod{F_n^2}.$$

Επειδή δε $\gcd(F_{n+1}, F_n) = 1$, έχουμε

$$F_{kn} \equiv 0 \pmod{F_n^2} \Leftrightarrow kF_n \equiv 0 \pmod{F_n^2} \Leftrightarrow k \equiv 0 \pmod{F_n}$$

και το λήμμα απεδείχθη. //

Λύση της άσκησης 13 με γεννήτριες συναρτήσεων:

Η άσκηση αυτή λύνεται εύκολα με επαγωγή και χρήση της εξίσωσης $y^2 - y - 1 = 0$, εμείς όμως θα την λύσουμε με γεννήτριες συναρτήσεων.

Οι ***γεννήτριες συναρτήσεων*** (generating functions) *είναι ένας απλός τρόπος για να λύνουμε αναγωγικές εξισώσεις* (recurrences) *όπως η* $F_n = F_{n-1} + F_{n-2}$ *που ορίζει την ακολουθία* Fibonacci *με αρχικές τιμές* $F_0 = 0$, $F_1 = 1$.

Ας *θεωρήσουμε την δυναμοσειρά*

$$F(z) = F_0 + F_1 z + F_2 z^2 + \cdots = \sum_{n=0}^{\infty} F_n z^n .$$

Αν *μπορέσουμε να βρούμε έναν απλό τύπο για την* $F(z)$ *τότε μάλλον θα μπορέσουμε να βρούμε έναν απλό τύπο για τους συντελεστές* F_n *της δυναμοσειράς* (*γεννήτριας συναρτήσεων*).

Πολλαπλασιάζοντας *την δυναμοσειρά* $F(z)$ *επί* z *και επί* z^2 *έχουμε*:

$$\begin{aligned} F(z) &= F_0 + F_1 z + F_2 z^2 + F_3 z^3 + F_4 z^4 + \cdots , \\ zF(z) &= F_0 z + F_1 z^2 + F_2 z^3 + F_3 z^4 + \cdots , \\ z^2 F(z) &= F_0 z^2 + F_1 z^3 + F_2 z^4 + \cdots . \end{aligned}$$

Αφαιρούμε *τώρα τις τελευταίες δύο εξισώσεις από την πρώτη και έχουμε*

$$F(z) - zF(z) - z^2 F(z) = z$$

διότι $F_n = F_{n-1} + F_{n-2}$, *και* $F_0 = 0$, $F_1 = 1$. Λύνοντας *ως προς* $F(z)$ *έχουμε τον τύπο*

$$F(z) = \frac{z}{1 - z - z^2} .$$

Όσο *και αν φαίνεται απίστευτο*, *όλες τις πληροφορίες για την ακολουθία* Fibonacci *περιέχονται στην έκφραση* $\frac{z}{1 - z - z^2}$. Το *μόνο που μένει είναι να μετατρέψουμε σε δυναμοσειρά την έκφραση αυτή*. Με *αυτόν τον τρόπο θα έχουμε βρει τον τύπο για τους συντελεστές* F_n.

Για *τον σκοπό αυτό*, *πρώτα αναλύουμε την έκφραση* $\frac{z}{1 - z - z^2}$ *σε* ***μερικά κλάσματα*** *και μετά χρησιμοποιούμε την γνωστή μας σχέση* $\frac{1}{1-\alpha z} = 1 + \alpha z + \alpha^2 z^2 + \alpha^3 z^3 + \cdots$. Αυτό *σημαίνει πως πρέπει πρώτα να βρούμε σταθερές* A, B, α, *και* β *ώστε*

$$\frac{z}{1 - z - z^2} = \frac{A}{1 - \alpha z} + \frac{B}{1 - \beta z} .$$

Η τελευταία εξίσωση γράφεται σαν

$$\frac{z}{1-z-z^2} = \frac{A}{1-\alpha z} + \frac{B}{1-\beta z} = \frac{A-A\beta z+B-B\alpha z}{(1-\alpha z)(1-\beta z)},$$

απ' όπου προκύπτει ότι οι σταθερές που ζητάμε πρέπει να ικανοποιούν το ακόλουθο σύστημα πολυωνυμικών εξισώσεων

$$(A+B) - (A\beta + B\alpha)z = z,$$
$$(1-\alpha z)(1-\beta z) = 1 - z - z^2.$$

Οι σταθερές α, β βρίσκονται από την δεύτερη εξίσωση κάνοντας το ακόλουθο τρικ: Εισάγουμε μια νέα μεταβλητή y και βρίσκουμε την παραγοντοποίηση της εξίσωσης

$$y^2 - yz - z^2 = (y-\alpha z)(y-\beta z).$$

Μετά θέτουμε $y = 1$ και έχουμε τους παράγοντες της $1 - z - z^2$. Οι ρίζες της $y^2 - yz - z^2$ δίνονται από τον τύπο

$$\frac{z \pm \sqrt{z^2+4z}}{2} = \frac{1 \pm \sqrt{5}}{2} z.$$

Άρα

$$y^2 - yz - z^2 = (y - \frac{1+\sqrt{5}}{2} z)(y - \frac{1-\sqrt{5}}{2} z).$$

και έχουμε βρει τις τιμές των α και β. Δηλαδή $\alpha = \phi_+ = \frac{1+\sqrt{5}}{2} \approx 1.61803$ και $\beta = \phi_- = \frac{1-\sqrt{5}}{2} \approx -0.61803$. Με άλλα λόγια είναι ρίζες της εξίσωσης $y^2 - y - 1 = 0$.

Έχοντας βρει τις τιμές των α και β από την πρώτη εξίσωση του συστήματος βρίσκουμε πως $A = -B$ και η εξίσωση αυτή γίνεται

$$-A\phi_- + A\phi_+ = 1.$$

Άρα, $A = \frac{1}{\phi_+ - \phi_-} = \frac{1}{\sqrt{5}}$ και η δυναμοσειρά γράφεται:

$$F(z) = \frac{z}{1-z-z^2} = \frac{A}{1-\alpha z} + \frac{B}{1-\beta z} = \frac{1}{\sqrt{5}} \left(\frac{1}{1-\phi_+ z} - \frac{1}{1-\phi_- z} \right).$$

Αναπτύσσοντας τώρα τα κλάσματα σε δυναμοσειρές και συγκρίνοντας συντελεστές βλέπουμε πως

$$F_n = \frac{1}{\sqrt{5}} (\phi_+^n - \phi_-^n) \quad \text{για } n \in \mathbb{N}. \text{ //}$$

Λύση της άσκησης 14:

Αποδεικνύουμε το αποτέλεσμα για την περίπτωση δύο τυχαία επιλεγμένων ακεραίων a, b. Έστω $p = \text{Prob}\{\gcd(a, b) = 1\}$, την οποία θέλουμε να υπολογίσουμε. Με την βοήθεια της p ας υπολογίσουμε την $\text{Prob}\{\gcd(a, b) = k\}$. Προσέξτε πως $\gcd(a, b) = k$ αν το a είναι πολλαπλάσιο του k και αν το b είναι πολλαπλάσιο του k.

Όμως, το a είναι πολλαπλάσιο του k με πιθανότητα $\frac{1}{k}$ και το b είναι πολλαπλάσιο του k με πιθανότητα $\frac{1}{k}$ και $\gcd(a, b) = 1$ με πιθανότητα p. Με άλλα λόγια $\text{Prob}\{\gcd(a, b) = k\} = \frac{1}{k^2}$. Αθροίζοντας για όλα τα k έχουμε $p\ (\sum_{k>1} k^{-2}) = p\ (\frac{\pi^2}{6}) = 1$ απ' όπου προκύπτει $p = \frac{6}{\pi^2} \approx 0.60793$.

Η συνάρτηση $H_\infty^{(r)} = \zeta(r)$, είναι η συνάρτηση του Riemann (Riemann's zeta function).

Βιβλιογραφία

1. Akritas, A.G.: *Elements of Computer Algebra with Applications.* Wiley Interscience, New York, 1989.

2. Dirichlet, P.G.L.: Über die Bestimmung der mittleren Werthe in der Zahlentheorie. *Abhandlungen der Königlich Preussischen Akademie der Wissenschaften*, 69–83, 1849. Werke, Zweiter Band, ed. L. Kronecker, 1897, 51–66. Reprint by Chelsea Publishing Co., NY, 1968.

3. Graham, R.L., Knuth, D.E., and O. Patashnik: *Concrete Mathematics*. Addison-Wesley, New York, 1981.

4. Lamé, G.: Note sur la limite du nombre des divisions dans la recherche du plus grand commun diviseur entre deux nombres entiers. *Comptes Rendus de l' Académie des Sciences* (Paris) **19**, 867–870, 1844.

5. Shallit, J.: Origins of the Analysis of the Euclidean Algorithm. *Historia Mathematica* **21**, 401–419, 1994.

6. Stark, E.L.: The Series $\sum_{d>1} k^{-s}$, s = 2, 3, 4, …, once more. *Mathematics Magazine* **47**, 197–202, 1974.

Κεφάλαιο 3: Εφαρμογές του Ευκλείδειου αλγόριθμου

Στο κεφάλαιο αυτό παρουσιάζουμε ορισμένες εφαρμογές του Ευκλείδειου αλγόριθμου. Οι εφαρμογές αυτές παίζουν σημαντικό ρόλο όχι μόνο στην Υπολογιστική Άλγεβρα αλλά σε πολλές περιοχές των Μαθηματικών, και γενικά των εφαρμοσμένων επιστημών.

Συγκεκριμένα, θα αναφερθούμε: α. στην αριθμητική υπολοίπων, β. στον υπολογισμό των πολλαπλασιαστικών αντιστρόφων και δυνάμεων modulo έναν ακέραιο, γ. στο ανάπτυγμα ρητών αριθμών σε συνεχή κλάσματα, και δ. στην λύση γραμμικών Διοφαντικών εξισώσεων.

Στο επόμενο κεφάλαιο, και αφού μελετήσουμε τον (Ελληνο)-Κινεζικό αλγόριθμο υπολοίπων θα δούμε την εφαρμογή της αριθμητικής υπολοίπων στην εκτέλεση αριθμητικής ακεραίων πολλαπλής ακριβείας.

3.1 Αριθμητική υπολοίπων

Στο Παράρτημα Γ, και συγκεκριμένα στην συζήτησή μας για δακτυλίους και δακτυλίους πολυωνύμων, ετέθησαν οι θεωρητικές βάσεις για υπόλοιπα mod n και αριθμητική υπολοίπων αντίστοιχα. Ο υπολογισμός υπολοίπων θεωρείται ειδική περίπτωση του Ευκλείδειου αλγόριθμου όπου το Ευκλείδειο μήκος είναι 1, ή όπου, με άλλα λόγια, κάνουμε μόνον μία διαίρεση.

Η βασική έννοια είναι ο δακτύλιος των τάξεων υπολοίπων του R modulo I, όπου το ιδεώδες I μπορεί να είναι είτε στον δακτύλιο των ακεραίων, $R = \mathbb{Z}$, είτε στον δακτύλιο των πολυωνύμων, $R = \mathbb{Q}[x]$. Υπολογιστικά εργαζόμαστε με τους **αντιπροσώπους** των τάξεων υπολοίπων modulo I.

Για $R = \mathbb{Z}$, και $I = 6\mathbb{Z} = \{6r, r \in \mathbb{Z}\}$, οι τάξεις υπολοίπων είναι $R/I = \mathbb{Z}/6\mathbb{Z} = \{0 \bmod 6\mathbb{Z}, 1 \bmod 6\mathbb{Z}, \ldots, 5 \bmod 6\mathbb{Z}\}$. Πιο απλά γράφουμε $\mathbb{Z}/6\mathbb{Z} = \{0, 1, \ldots, 5\} = \mathbb{Z}_6$.

Αντίστοιχα, αν $R = \mathbb{Q}[x]$ και $I = \langle f \rangle$, όπου $f = x^2 - x - 1$, ένα πολυώνυμο με μοναδιαίο κύριο συντελεστή, τότε $3x + 5 \bmod f$, είναι μια τάξη υπολοίπων που αποτελείται από όλα τα πολυώνυμα που έχουν υπόλοιπο $3x + 5$ αν διαιρεθούν με το $f = x^2 - x - 1$.

Ορισμός:

α. Υπενθυμίζουμε πως $a \equiv b \bmod n$, που διαβάζεται "*a και b είναι* ***ισοδύναμα*** ή ***ισοϋπόλοιπα*** (*a* is congruent to *b*, or they are congruent) **modulo** *n*", σημαίνει πως $a - b$ διαιρείται με το n, ή ισοδύναμα πως τα a και b αφήνουν το ίδιο υπόλοιπο ύστερα από διαίρεση με το n.

β. **Αριθμητική υπολοίπων** (modular arithmetic) λέγεται η διαδικασία εκτέλεσης υπολογισμών με υπόλοιπα αριθμητικών εκφρασεων modulo κάποιον μη μηδενικό ακέραιο n.

Οπως έχουμε δει, διαιρέσεις ακεραίων με υπόλοιπο γίνονται εύκολα αν πρώτα κάνουμε κανονικοποίηση του διαιρετέου και διαιρέτη. Βλέπε και το αντίστοιχο θεώρημα. Το ίδιο εύκολα υπολογίζονται τα υπόλοιπα και για πολυώνυμα. Έτσι

περνάμε κατ΄ευθείαν σε εφαρμογές της αριθμητικής υπολοίπων χωρίς να ασχοληθούμε εδώ με το πρόβλημα υπολογισμού των υπολοίπων.

Έστω για παράδειγμα πως `expr` είναι μία έκφραση ακεραίων με τις αριθμητικές πράξεις +, -, *, /, και θέλουμε να υπολογίσουμε το υπόλοιπο της `expr` modulo *n*. Ο υπολογισμός μπορεί να γίνει πολύ αποτελεσματικά αν πρώτα υπολογίσουμε το υπόλοιπο modulo *n* κάθε ακεραίου και, μετά, εκτελώντας μία-μία τις πράξεις στον δακτύλιο των ακεραίων υπολογίζουμε ταυτόχρονα το υπόλοιπο modulo *n* κάθε ενδιάμεσου αποτελέσματος.

Παράδειγμα:
Έστω πως `expr` = 8*13 - 21 και θέλουμε το υπόλοιπο modulo *n* = 5. Σύμφωνα με τα παραπάνω έχουμε

$$\texttt{expr} = 8*13 - 21 \equiv 3*3 + 4 \equiv 4 + 4 \equiv 3 \bmod 5.$$

Προσέξτε πως τα ενδιάμεσα αποτελέσματα είναι πάντα μικρότερα του n^2 (= 25) και πως χρησημοποιήσαμε το λεγόμενο ***μη αρνητικό σύστημα υπολοίπων*** modulo *n* (non negative residue system) που αποτελείται από τους ακεραίους {0, 1, …, *n* - 1}. Η συνάρτηση `Mod[]` του Mathematica κάνει χρήση αυτού του συστήματος.

Ισοδύναμα, θα μπορούσαμε να χρησημοποιήσουμε το ***συμμετρικό σύστημα υπολοίπων*** modulo *n* (symmetric residue system) ή ***το σύστημα υπολοίπων*** modulo *n* ***με την ελάχιστη απόλυτη τιμή*** (least absolute value residue system) που για ***περιττό*** ακέραιο *n* αποτελείται από τους ακεραίους $\{0, \pm 1, \ldots, \pm \frac{(n-1)}{2}\}$. Το αποτέλεσμα θα ήταν

$$\texttt{expr} = 8*13 - 21 \equiv (-2)*(-2) + (-1) \equiv -1 - 1 \equiv -2 \bmod 5.$$

Το συμμετρικό σύστημα υπολοίπων modulo *n* είναι απαραίτητο στην περίπτωση των πολυωνύμων.

Θεώρημα (Ιδιότητες ισοδυναμιών mod *n*):
α. Αν $a \equiv b \bmod m$ και $d \mid m$, τότε $a \equiv b \bmod d$.
β. Αν $a \equiv b \bmod m$ και $a \equiv b \bmod n$, τότε $a \equiv b \bmod \text{lcm}(m, n)$, όπου lcm(*m*, *n*) είναι

το ελάχιστο κοινό πολλαπλάσιο των m, n.

γ. Αν $a \equiv c \bmod m$ και $b \equiv d \bmod m$, τότε $a + b \equiv c + d \bmod m$, $a - b \equiv c - d \bmod m$, και $a \cdot b \equiv c \cdot d \bmod m$.

δ. (Απαλοιφή) Αν $a \cdot b \equiv a \cdot c \bmod m$, τότε $b \equiv c \bmod \frac{m}{\gcd(a, m)}$. Αν $\gcd(a, m) = 1$, τότε από $a \cdot b \equiv a \cdot c \bmod m$, συνεπάγεται $b \equiv c \bmod m$.

Η απόδειξη του θεωρήματος αυτού αφήνεται για άσκηση. Βλέπε και την βιβλιογραφία. Προσέξτε όμως και το ακόλουθο

Παράδειγμα:

Από την ισοδυναμία $3 \cdot 4 \equiv 3 \cdot 6 \bmod 6$ συνεπάγεται $4 \equiv 6 \bmod 2$, ενώ από την ισοδυναμία $3 \cdot 4 \equiv 3 \cdot 9 \bmod 5$ συνεπάγεται $4 \equiv 9 \bmod 5$. Στην πρώτη περίπτωση **αλλάζει** το modulus και από 6 γίνεται 2.

Εφαρμογή 1η:

Σαν πρώτη εφαρμογή της αριθμητικής υπολοίπων ας εξηγήσουμε την γνωστή από το δημοτικό σχολείο μέθοδο ελέγχου: "ένας ακέραιος διαιρείται από το 3 ή το 9 εάν και μόνον εάν το άθροισμα των ψηφίων του διαιρείται από το 3 ή το 9". (Βλέπε και τις ασκήσεις για ανάλογη εξήγηση του ελέγχου διαίρεσης ακεραίου με 11.)

Εξήγηση:

Έστω ο θετικός ακέραιος $a = \sum_{0 \le i \le k} a_i \, 10^i$, ως προς την βάση $\beta = 10$. Προσέξτε πως $10 \equiv 1 \bmod 3$ (ή mod 9) και επομένως $a \equiv \sum_{0 \le i \le k} a_i \, 1^i = \sum_{0 \le i \le k} a_i \bmod 3$ (ή mod 9).

Εφαρμογή 2η:

Επίσης γνωστή από το δημοτικό σχολείο είναι και η μέθοδος ελέγχου των αποτελεσμάτων της πρόσθεσης και του πολλαπλασιασμού δύο ακεραίων—γνωστή στα Αγγλικά σαν μέθοδος "πετάγματος των 9" (method of "casting out 9's").

Έστω ότι $a \otimes b = c$, όπου $\otimes$ σημαίνει πρόσθεση ή πολλαπλασιασμό. Για να δούμε αν το αποτέλεσμα c είναι σωστό, προσθέτουμε επανειλημμένα τα ψηφία κάθε ενός των a, b, και c μέχρις ότου το άθροισμα είναι μονοψήφιο. (Για

παράδειγμα, αν $a = 87$ εκτελούμε επανειλημμένα τις προσθέσεις $8 + 7 = 15$ και $1 + 5 = 6$. Το ίδιο και για τους b, c.) Έστω $\overline{a}$, $\overline{b}$, και $\overline{c}$ τα αντίστοιχα μονοψήφια αποτελέσματα. Για να είναι σωστό το αποτέλεσμα c πρέπει $\overline{c} = \overline{\overline{a} \otimes \overline{b}}$, όπου $\overline{\overline{a} \otimes \overline{b}}$ είναι ο μονοψήφιος ακέραιος που προκύπτει από την πράξη $\overline{a} \otimes \overline{b}$.

Παράδειγμα:
Κάνοντας τις πράξεις με μολύβι και χαρτί, βρίσκουμε το γινόμενο $123*45 = 5535$. Το αποτέλεσμα είναι σωστό γιατί αφού υπολογίσουμε $\overline{123} = 6$, $\overline{45} = 9$, και $\overline{5535} = \overline{18} = 9$, βλέπουμε πως το άθροισμα των ψηφίων του γινομένου $\overline{123}*\overline{45} = 6*9 = 54$ ισούται με 9 ($= \overline{5535}$).

Εξήγηση:
Όλα βασίζονται στο ότι ο ακέραιος, ας πούμε, abc γράφεται και σαν $100a + 10b + c = 99a + 9b + (a + b + c)$. Άρα $abc \equiv (a + b + c) \bmod 9$.

Η βασική διαδικασία του τεστ αυτού είναι η επανειλημμένη πρόσθεση των ψηφίων ενός ακεραίου μέχρις ότου το άθροισμα τους είναι μονοψήφιο. Αυτό που κάνουμε είναι να βρίσκουμε το υπόλοιπο mod 9. Έτσι για τον ακέραιο 87 ο ακέραιος 6 (που προέρχεται από $8 + 7 = 15$ και $1 + 5 = 6$) προκύπτει απ'ευθείας

```
Mod[87, 9]

6
```

Άρα ο έλεγχος συνίσταται στον υπολογισμό των $\overline{a} \equiv a \bmod 9$, $\overline{b} \equiv b \bmod 9$ και $\overline{c} \equiv c \bmod 9$. Το αποτέλεσμα είναι σωστό αν $\overline{c} \equiv \overline{a}$ +/* $\overline{b} \bmod 9$.

Αξίζει να σημειωθεί πως με την παλιά αυτή μέθοδος ελέγχου 10% των τυχαίων σφαλμάτων περνούν απαρατήρητα.

Παράδειγμα:
Η σχολική μέθοδος ελέγχου ***δεν*** μπορεί να "βρει" το λάθος στον πολλαπλασιασμό $123*54 = 5535$.

Η παραπάνω διαδικασία γενικεύεται ως εξής:

Έστω για παράδειγμα ότι θέλουμε να ελέγξουμε αν το πρόγραμμα `Times[]` του *Mathematica* που κάνει πολλαπλασιασμό ακεραίων πολλαπλής ακριβείας δουλεύει σωστά. Ο έλεγχος αυτός πρέπει να γίνει χωρίς να χρησιμοποιηθεί το πρόγραμμα `Times[]`. Δηλαδή, το ερώτημα είναι αν το ακόλουθο γινόμενο είναι σωστό

```
a = Random[Integer, {10^40, 10^41}]; b = Random[Integer, {10^40, 10^41}];
c = a b
```

```
720921763828750206445674647330746747305700629816890617367595900258 9
 707792255807496
```

Η απάντηση δίνεται με ένα ***τεστ υπολοίπων*** (modular test) γνωστό και σαν ***τεστ "δακτυλικών αποτυπωμάτων"*** (fingerprinting test): δηλαδή, παίρνουμε έναν πρώτο αριθμό p απλής ακριβείας και ελέγχουμε αν $a \cdot b \equiv c \bmod p$ ή ισοδύναμα αν $a \cdot b - c$ διαιρείται με το p, ή ισοδύναμα αν τα $a \cdot b$ και c αφήνουν το ίδιο υπόλοιπο ύστερα από διαίρεση με το p.

Ο έλεγχος γίνεται ως εξής: παίρνουμε $\bar{a} = a \bmod p$, $\bar{b} = b \bmod p$, και $\bar{c} = c \bmod p$, και ελέγχουμε αν $\bar{a} \cdot \bar{b} \equiv \bar{c} \bmod p$. Στην περίπτωσή μας το τεστ είναι θετικό.

```
p = Prime[10^9]
```

```
22801763489
```

```
ā = Mod[a, p]; b̄ = Mod[b, p]; c̄ = Mod[c, p];
```

```
Mod[ā b̄, p] == Mod[c̄, p]
```

```
True
```

Κατά πόσο όμως μπορούμε να είμαστε σίγουροι για το αποτέλεσμα; Χωρίς να μπαίνουμε σε λεπτομέρειες, αρκεί να αναφέρουμε πως με διάφορα τρικ (όπως τεστ με διάφορους πρώτους αριθμούς) μπορούμε να αυξήσουμε όσο θέλουμε την πιθανότητα να είμαστε σίγουροι για το αποτέλεσμα. (Εδώ συγκεκριμένα δεν χρειάζεται να ανησυχούμε. Η συνάρτηση `Times[]` δουλεύει σωστά.)

Η *πιο απλή εφαρμογή της αριθμητικής υπολοίπων σε πολυώνυμα με συντελεστές από έναν δακτύλιο R είναι όταν έχουμε για* modulus *το πολυώνυμο πρώτου βαθμού* $x - u$, $u \in R$. (Βλέπε *και τον ομοιομορφισμό διατίμησης*.) *Για κάθε πολυώνυμο* $f(x) \in R[x]$, *το u είναι ρίζα του πολυωνύμου* $f(x) - f(u)$ *και επομένως διαιρείται από το* $x - u$. *Άρα αν θέσουμε* $q(x) = \frac{f(x) - f(u)}{x - u}$, *τότε* $f(x) = q(x)\cdot(x - u) + f(u)$, *όπου* $f(u)$ *είναι το υπόλοιπο*. *Συνεπώς ο βαθμός του* $f(u)$ *είναι μικρότερος του* $1 = \deg(x - u)$, *δηλαδή* $f(u)$ *είναι σταθερά*.

Π*ροσέξτε όμως πως από* $f(x) = q(x)\cdot(x - u) + f(u)$ *προκύπτει* $f(x) \equiv f(u) \bmod (x - u)$ *με συνέπεια οι υπολογισμοί* modulo $(x - u)$ *να είναι ισοδύναμοι με τον υπολογισμό της τιμής* $f(u)$. *Πράγματι, αν* $f(x) = x^3 - 7x + 7$,

```
f[x_] = x^3 - 7 x + 7; f[2] == PolynomialMod[x^3 - 7 x + 7, x - 2]
True
```

Η *τελευταία παρατήρηση μας οδηγεί σε ένα* ***τεστ "δακτυλικών αποτυπωμάτων" για πολυωνυμικές εκφράσεις***.

Έ*στω για παράδειγμα ότι θέλουμε να δούμε αν* $f(x) = (x - 1)(x - 2)(x - 3) - 3$ *ισούται με* $g(x) = x^3 - 6x^2 + 10x - 9$. *Ο έλεγχος θα γίνει με υπολογισμούς* modulo $(x - u)$ *για διάφορες τιμές του u*, *ή με άλλα λόγια θα συγκρίνουμε τα* $f(u)$ *και* $g(u)$ *για διάφορες τιμές του u*.

Α*ρχίζουμε με* $u = 0$. *Στην περίπτωση αυτή* $f(x) \equiv f(0) \bmod (x - 0)$ *με* $f(0) = (-1)(-2)(-3) - 3 = -9$ *και* $g(x) \equiv g(0) \bmod (x - 0)$ *με* $g(0) = -9$ (*ανάλογο με τον υπολογισμό του a* mod 10, *για τον ακέραιο a*). *Άρα* $f(x) \equiv g(x) \bmod (x - 0)$.

Γ*ια την τιμή* $u = 1$ *έχουμε* $f(x) \equiv f(1) \bmod (x - 1)$, *με* $f(1) = -3$ *και* $g(x) \equiv g(1) \bmod (x - 1)$ *με* $g(1) = 1 - 6 + 10 - 9 = -4$ (*ανάλογο με τον υπολογισμό του αθροίσματος των ψηφίων ενός ακεραίου a*). *Άρα* $f(x) \not\equiv g(x) \bmod (x - 1)$.

Τέλος *αναφέρουμε πως πολλές φορές χρειάζεται να κάνουμε πράξεις σε δακτύλιους "διπλών" τάξεων υπολοίπων της μορφής* $\mathbb{Z}_n[x]/\langle f\rangle$, *δηλαδή* modulo *ακεραίους* ***και*** modulo *πολυώνυμα*. *Σε αυτές τις περιπτώσεις οι πράξεις επί των*

συντελεστών γίνονται modulo *n και οι πράξεις επί των πολυωνύμων γίνονται* modulo *f*.

Γ*ια παράδειγμα*, *στον* $\mathbb{Z}[x]$ *έχουμε*

```
(2 x + 3) (x^3 - 7 x + 7) // Expand
```

```
21 - 7 x - 14 x^2 + 3 x^3 + 2 x^4
```

ενώ, *στον* $\mathbb{Z}_3[x]$ *γίνεται*

```
PolynomialMod[(2 x + 3) (x^3 - 7 x + 7), 3]
```

```
2 x + x^2 + 2 x^4
```

Όσ*ον αφορά διαίρεση με υπόλοιπο έχουμε*

```
PolynomialRemainder[2 x^4 + x^2 + 2 x, x^2 - 7, x]
```

```
105 + 2 x
```

που στον $\mathbb{Z}_3[x]$ *γίνεται*

```
PolynomialMod[2 x + 105, 3]
```

```
2 x
```

Το *αποτέλεσμα είναι το ίδιο με αυτό που βρίσκουμε παίρνοντας το διπλό* "*υπόλοιπο*" *του* $(2x + 3)(x^3 - 7x + 7)$ modulo 3 ***και*** $(x^2 - 7)$.

```
PolynomialMod[(2 x + 3) (x^3 - 7 x + 7), {3, x^2 - 7}]
```

```
2 x
```

Δ*ηλαδή*,

$$(2x + 3)(x^3 - 7x + 7) \equiv 2x \bmod (x^2 - 7) \quad \textit{στον } \mathbb{Z}_3[x]$$

ή *με άλλα λόγια*

$$(2\alpha + 3)\,(\alpha^3 - 7\alpha + 7) \equiv 2\alpha \quad \sigma\tau o\nu\ \mathbb{Z}_3[x]/\langle x^2 - 7\rangle$$

όπου $\alpha = x \bmod (x^2 - 7) \in \mathbb{Z}_3[x]/\langle x^2 - 7\rangle$.

3.2 Αντίστροφοι mod *n*

Όπως είπαμε, στην αριθμητική υπολοίπων εργαζόμαστε με τα υπόλοιπα modulo n. Το προσθετικό αντίστροφο για κάθε αριθμό $a \in \mathbb{Z}_n$ είναι ο αριθμός x έτσι ώστε $x + a = n \equiv 0 \bmod n$. Προφανώς $x = n - a$.

Σε αντίθεση, το πολλαπλασιαστικό αντίστροφο ορίζεται ως η λύση της εξίσωσης ισοϋπολοίπων $x \cdot a \equiv 1 \bmod n$. Η εξίσωση αυτή δεν έχει πάντα λύση, διότι απλούστατα κάθε στοιχείο ενός δακτυλίου εν γένει δεν αντιστρέφεται (δεν είναι μονάδα).

Θα παρουσιάσουμε δύο τρόπους λύσης της $x \cdot a \equiv 1 \bmod n$. Ο πρώτος τρόπος σχετίζεται με τον επεκταμένο Ευκλείδειο αλγόριθμο (EEA) ενώ ο δεύτερος με το θεώρημα του Fermat. Για τους χρόνους υπολογισμού των δύο μεθόδων βλέπε τις ασκήσεις και τα Λήμματα 2.3.1 και 2.3.2.

■ 3.2.1 Αντίστροφοι mod *n* με τον Ευκλείδη (EEA)

Θεώρημα 3.2.1:
Έστω R ένας Ευκλείδειος δακτύλιος, $a, n \in R$, και $S = R/nR$ ο δακτύλιος των τάξεων υπολοίπων mod n. Τότε το στοιχείο $a \bmod n \in S$ (που είναι αντιπρόσωπος μιας τάξης υπολοίπων) είναι μονάδα στον S εάν και μόνον εάν $\gcd(a, n) = 1$.

Απόδειξη:
"⇒" Έχουμε

$$\begin{aligned} & a \text{ αντιστρέφεται } \bmod n \\ & \Leftrightarrow \exists x \in R \text{ ώστε } x \cdot a \equiv 1 \bmod n \\ & \Leftrightarrow \exists x, y \in R \text{ ώστε } x \cdot a + y \cdot n = 1 \\ & \Rightarrow \gcd(a, n) = 1. \end{aligned}$$

"⇐" Αντίστροφα, αν $\gcd(a, n) = 1$, τότε ο επεκταμένος Ευκλείδειος αλγόριθμος μας παρέχει τέτοια στοιχεία $x, y \in R$.

Παραδείγματα:

1. $R = \mathbb{Z}$, $n = 8$, *και* $a = 6$. Επειδή gcd(8, 6) = 2, *βάσει του* Θεωρήματος 3.2.1 *δεν υπάρχει το* 6^{-1}, *ο αντίστροφος του* 6. Αντίθετα *επειδή* gcd(8, 7) = 1,

```
ExtendedGCD[a = 7, n = 8]
```

```
{1, {-1, 1}}
```

υπάρχει το 7^{-1}, *το αντίστροφο του* 7. Από *τον επεκταμένο* Ευκλείδειο *αλγόριθμο βλέπουμε πως* (-1)·7 + 1·8 = 1 *και συνεπώς* (-1)·7 ≡ 7·7 ≡ 1 mod 8. Άρα, *το* 7 (≡ -1 mod 8) *είναι το αντίστροφο του* 7 mod 8.

2. $R = \mathbb{Q}[x], f = x^3 - 7x + 7$ *και* $a = x^2 - 7$.

```
<< Algebra`PolynomialExtendedGCD`
PolynomialExtendedGCD[a = x^2 - 7, f = x^3 - 7 x + 7]
```

$$\left\{1, \left\{-\frac{x}{7}, \frac{1}{7}\right\}\right\}$$

Από *τον επεκταμένο* Ευκλείδειο *αλγόριθμο βλέπουμε πως* $\frac{-x}{7}(x^2 - 7) + \frac{1}{7}(x^3 - 7x + 7) = 1$ *και συνεπώς το* $\frac{-x}{7}$ *είναι το αντίστροφο του* $x^2 - 7$ modulo $x^3 - 7x + 7$.

Από *το* Θεώρημα 3.2.1 *συνεπάγεται πως αν ο δακτύλιος των τάξεων υπολοίπων είναι* $S = \mathbb{Z}/p\mathbb{Z} = \mathbb{Z}_p$, *όπου* $p \in \mathbb{Z}_{>0}$ *πρώτος αριθμός ή* $S = F[x]/\langle f \rangle$, *όπου* F *είναι σώμα και* $f \in F[x]$ *ένα μη παραγοντοποιήσιμο πολυώνυμο, τότε κάθε στοιχείο του* $S\backslash\{0\}$ *είναι μονάδα, και το* S *είναι σώμα* (*βλέπε και την βιβλιογραφία*).

Αντί *για* $\mathbb{Z}_p$ *θα χρησιμοποιούμε εναλλακτικά και τον συμβολισμό* F_p, *για το πεπερασμένο σώμα με* p *στοιχεία*. Επιπλέον, *αν* $f \in \mathbb{Z}_p[x] = \mathbb{F}_p[x]$ *είναι ένα μη παραγοντοποιήσιμο πολυώνυμο βαθμού* n, *τότε* $\mathbb{F}_p[x]/\langle f \rangle$ *είναι ένα πεπερασμένο σώμα με* $q = p^n$ *στοιχεία και θα συμβολίζεται με* $\mathbb{F}_q$.

Το *παρακάτω θεώρημα μας λέει πως υπολογίζοντας* modulo *ένα μη παραγοντοποιήσιμο πολυώνυμο* f *είναι ισοδύναμο με το να* "*προσθέτουμε στο σώμα* F *μια ρίζα του* f."

Θεώρημα 3.2.2:

Έστω F ένα σώμα και $f \in F[x]$ ένα μη παραγοντοποιήσιμο πολυώνυμο με μοναδιαίο κύριο συντελεστή. Τότε το $E = F[x]/\langle f \rangle$ είναι μία αλγεβρική επέκταση του σώματος F (ίσως και σώμα διάσπασης του f), και $f(\alpha) = 0$ για $\alpha = x \bmod f \in E$.

Απόδειξη:

Βάσει του Θεωρήματος 3.2.1, το E είναι σώμα και $f(\alpha) = f(x \bmod f) = (f(x) \bmod f) = 0$.

Παράδειγμα:

Ας θεωρήσουμε τον δακτύλιο $\mathbb{F}_2[x] = \mathbb{Z}_2[x]$ και το μη παραγοντοποιήσιμο πολυώνυμο με μοναδιαίο κύριο συντελεστή $f = x^3 + x + 1$. Το f δεν έχει ρίζες στο $\mathbb{Z}_2$ όπως βλέπουμε

```
NSolve[x^3 + x + 1 == 0]
```

```
{{x → -0.682328}, {x → 0.341164 - 1.16154 i}, {x → 0.341164 + 1.16154 i}}
```

και άρα $\mathbb{F}_8 = \mathbb{F}_2[x]/\langle f \rangle$ είναι σώμα. Τα 8 στοιχεία του σώματος αυτού είναι της μορφής

$$\{c_0 + c_1\alpha + c_2\alpha^2 \mid c_0, c_1, c_2 \in \mathbb{Z}_2\}$$

όπου $\alpha = x \bmod f$. Δηλαδή είναι τα εξής:

$$\{0, 1, \alpha, \alpha^2, \alpha + \alpha^2, 1 + \alpha, 1 + \alpha^2, 1 + \alpha + \alpha^2\}$$

Έστω ότι τώρα θέλουμε να βρούμε το αντίστροφο του x modulo $f = x^3 + x + 1$. Από τον επεκταμένο Ευκλείδειο αλγόριθμο

```
PolynomialMod[
 PolynomialExtendedGCD[a = x, f = x^3 + x + 1], 2]
```

```
{1, {1 + x^2, 1}}
```

βλέπουμε πως $x\cdot(1 + x^2) + 1\cdot(x^3 + x + 1) = 1$ στο $\mathbb{F}_2[x]$ και συνεπώς το $(1 + x^2)$ είναι το αντίστροφο του x modulo $x^3 + x + 1$. Ισοδύναμα, στο σώμα $\mathbb{F}_8 = \mathbb{F}_2[x]/\langle f = x^3 + x + 1 \rangle$ έχουμε $\alpha^{-1} = 1 + \alpha^2$.

Προτυπο

Υπενθυμίζουμε πως αν n είναι ένας θετικός ακέραιος τότε το σύνολο $\mathbb{Z}_n^\times$ συμβολίζει τα στοιχεία του $\mathbb{Z}_n$ που έχουν πολλαπλασιαστικό αντίστροφο. Το σύνολο αυτό είναι η ***ομάδα των μονάδων*** του δακτυλίου $\mathbb{Z}_n$, και το Θεώρημα 3.2.1 μας λέει πως $\mathbb{Z}_n^\times = \{a \bmod n \mid \gcd(a, n) = 1\}$. Η ***συνάρτηση φ του*** **Euler** (Euler's totient function φ), φ: $\mathbb{Z}_{>0} \to \mathbb{Z}_{>0}$ μετράει τον πληθικό αριθμό της ομάδας $\mathbb{Z}_n^\times$, δηλαδή $\forall n$ μας δίνει τον αριθμό των θετικών ακεραίων $\leq n$ που είναι πρώτοι προς το n. Στο Mathematica υπάρχει η συνάρτηση

```
EulerPhi[7]
```

```
6
```

Εξ ορισμού $\varphi(1) = 1$. Αν το $n = p$ είναι πρώτος αριθμός, όλα τα μη μηδενικά στοιχεία του $\mathbb{Z}_p = \mathbb{F}_p$ είναι μονάδες (αντιστρέψιμα) και επομένως $\varphi(p) = p - 1$.

Γενικότερα, αν το n είναι δύναμη πρώτου αριθμού, $n = p^e$, από το Θεώρημα 3.2.1 συνεπάγεται πως ένα στοιχείο $k \in \mathbb{Z}_{p^e}$ είναι αντιστρέψιμο modulo p^e εάν και μόνον εάν ο p δεν διαιρεί τον k.

Όμως, στον $\mathbb{Z}_{p^e}$ ο p διαιρεί τα στοιχεία ℓp, όπου $0 \leq \ell < p^{e-1}$. Συνεπώς,

$$\varphi(p^e) = p^e - p^{e-1} = p^{e-1}(p - 1).$$

Πράγματι, έχουμε για παράδειγμα

```
EulerPhi[2^3] == 2^2 (2 - 1)
```

```
True
```

Κλείνοντας, αξίζει να σημειώσουμε πως στο $\mathbb{Z}_p$, όπου p πρώτος αριθμός, ισχύει ο "***κανόνας των αρχαρίων***"

$$(a + b)^p \equiv a^p + b^p \bmod p,$$

διότι στο ανάπτυγμα

$$(a + b)^p = \sum_{i=0}^{p} \binom{p}{i} a^{p-i}\, b^i$$

εκτός από τον πρώτο και τον τελευταίο, οι διωνυμικοί συντελεστές $\binom{p}{i} = \frac{p!}{i!\,(p-i)!}$ διαιρούνται όλοι από το p. Επαγωγικά αποδεικνύεται πως για $\forall i \in \mathbb{N}$ ισχύει

$$(a + b)^{p^i} \equiv a^{p^i} + b^{p^i} \bmod p.$$

■ 3.2.2 Αντίστροφοι mod *n* με το Fermat

Για να υπολογίσουμε τον αντίστροφο του $a \bmod n$, ή αλλιώς για να λύσουμε την εξίσωση $x \cdot a \equiv 1 \bmod n$, με την βοήθεια του "μικρού" θεωρήματος του Fermat, χρησιμοποιούμε την σχέση $a^{p-1} \equiv 1 \bmod p$, που αποδεικνύουμε παρακάτω. Η σχέση αυτή γράφεται και $a^{p-2} \cdot a \equiv 1 \bmod p$, απ' όπου προκύπτει πως $a^{-1} \equiv a^{p-2} \bmod p$. (Βλέπε όμως και Θεώρημα 3.2.3 παρακάτω.) Στο επόμενο τμήμα θα δούμε πως υψώνουμε σε δυνάμεις.

Θεώρημα 3.2.3 (Το "μικρό" θεώρημα του Fermat του 1640):
Αν p είναι πρώτος αριθμός και $a \in \mathbb{Z}$, τότε $a^p \equiv a \bmod p$ και αν ο a δεν διαιρείται από τον p, τότε $a^{p-1} \equiv 1 \bmod p$.

Απόδειξη:
Αποδεικνύουμε την $a^{p-1} \equiv 1 \bmod p$, από την οποία άμεσα προκύπτει η $a^p \equiv a \bmod p$.

Ας θεώρήσουμε τους αριθμούς $a, 2\cdot a, \ldots, p\cdot a$. Αν τους διαιρέσουμε με το p, τα υπόλοιπα που προκύπτουν θα είναι όλα διαφορετικά μεταξύ τους. (Διότι αν υποθέσουμε πως οι αριθμοί $i\cdot a$ και $j\cdot a$ αφήνουν το ίδιο υπόλοιπο τότε $(i - j)\cdot a$ θα ήταν πολλαπλάσιο του p, αφού τα υπόλοιπα θα έχουν φύγει. Συνεπώς, αφού εξ υποθέσεως το p δεν διαιρεί τον a και είναι πρώτος, έπεται πως διαιρεί τον $(i - j)$—άτοπο, διότι οι i, j είναι μικρότεροι του p.)

Άρα, τα υπόλοιπα που προκύπτουν θα είναι οι αριθμοί $0, 1, 2, \ldots, p - 1$, σε κάποια σειρά. Προφανώς, το 0 προκύπτει από την διαίρεση του $p\cdot a$ με το p και αν το εξαιρέσουμε έχουμε:

$$a^{p-1}(p - 1)! \equiv (p - 1)! \bmod p$$

ή αλλοιώς

$$a^{p-1}(p - 1)! - (p - 1)! \equiv 1 \bmod p$$

απ΄όπου έπεται

$$(a^{p-1} - 1)(p - 1)! \equiv 1 \bmod p.$$

Επειδή το p δεν διαιρεί τον $(p - 1)!$ και είναι πρώτος, έπεται πως διαιρεί τον $(a^{p-1} - 1)$ και άρα $a^{p-1} \equiv 1 \bmod p$. //

Θεώρημα 3.2.4 (Γενίκευση του Euler):
Αν $\gcd(a, n) = 1$, τότε $a^{\varphi(n)} \equiv 1 \bmod n$ και επομένως $a^{-1} \equiv a^{\varphi(n)-1} \bmod n$.

Απόδειξη:
Ανάλογη με την τελευταία. //

Από τα παραπάνω συμπεραίνουμε πως για να υπολογίσουμε τον αντίστροφο του $a \bmod n$, με την βοήθεια του "μικρού" θεωρήματος του Fermat ή της γενίκευσής του, χρειάζεται να υπολογίσουμε $a^{p-2} \bmod p$ ή $a^{\varphi(n)-1} \bmod n$, δηλαδή να υψώσουμε σε δυνάμεις modulo κάποιον αριθμό. Το θέμα αυτό εξετάζουμε στο επόμενο τμήμα.

> **Σημείωση:**
> Θα πρέπει να προσέξουμε πως το Θεώρημα 3.2.3 του Fermat ισχύει μόνο όταν ο p είναι πρώτος αριθμός και δεν μπορεί να χρησιμοποιηθεί σαν τεστ για την εύρεση πρώτων αριθμών, όπως έκαναν παλαιότερα οι Κινέζοι μαθηματικοί.
>
> Συγκεκριμένα, οι Κινέζοι ισχυριζόντουσαν πως ο αριθμός n είναι πρώτος εάν και μόνον εάν ισχύει $a^n \equiv a \bmod n$ ή $a^{n-1} \equiv 1 \bmod n$. Το τεστ παρέμεινε αδιαφιλονίκητο για πολλούς αιώνες—ακόμα και ο Leibniz το πίστευε—έως ότου το 1819 ο Γάλλος Pierre Frédéric Sarrus έδωσε αντιπαράδειγμα.
>
> Ο Sarrus υπέδειξε πως ο αριθμός $2^{341} - 2$ είναι πολλαπλάσιο του 341,

```
Mod[2^341 - 2, 341]

0
```

αν και ο 341 *δεν είναι πρώτος διότι* 341 = 11·31.

```
FactorInteger[341]
```

```
{{11, 1}, {31, 1}}
```

Αυτό *φαίνεται επίσης και από το γεγονός ότι* $2^{10} \equiv 1 \bmod 341$, *οπότε* $2^{340} \equiv (2^{10})^{34} \equiv 1^{34} \equiv 1 \bmod 341$ *και συνεπώς* $341 \mid (2^{341} - 2)$. Ο 341 *είναι ο μικρότερος αριθμός ως προς την βάση* 2.

Μετά *την ανακάλυψη του* Sarrus *βρέθηκαν και άλλοι αριθμοί*, όπως *ο* 91 = 7·13 (*ο μικρότερος αριθμός*) *ως προς την βάση* 3. *Έτσι έχουμε για παράδειγμα*

```
Mod[3^91 - 3, 91]
```

```
0
```

Οι *σύνθετοι* (*μη πρώτοι*) *αριθμοί που* "*πληρούν*" *το* Θεώρημα *του* Fermat *λέγονται* ***ψευτοπρώτοι***. *Για κάθε βάση υπάρχουν άπειροι ψευτοπρώτοι*.

Υπάχρχει *και μία ειδική κατηγορία αριθμών που είναι ψευτοπρώτοι ως προς κάθε βάση*. Αυτοί *λέγονται αριθμοί* Carmichael, *προς τιμή του μαθηματικού που τους ανακάλυψε το* 1909. Τέτοιοι *αριθμοί είναι για παράδειγμα ο* 561 = 3·11·17 *και ο* 1729 = 7·13·19, όπως *φαίνεται από τα παρακάτω*. (Η *συνάρτηση* `Prime[i]` *του Mathematica μας δίνει τον i-στό πρώτο αριθμό*.)

```
Table[Mod[Prime[i]^561 - Prime[i], 561], {i, 20}]
```

```
{0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}
```

```
Table[Mod[Prime[i]^1729 - Prime[i], 1729], {i, 20}]
```

```
{0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}
```

3.3 Ύψωση σε δύναμη με την δυαδική μέθοδο

Υπάρχουν δύο τρόποι για να υψώσουμε σε δύναμη κάποιο στοιχείο ενός δακτυλίου R. Έστω για παράδειγμα a^k, $a \in R$, $k \in \mathbb{Z}_{>0}$: Ο πρώτος τρόπος, αποκαλούμενος και μέθοδος "της ωμής δύναμης" ("brute force approach"), είναι να πολλαπλασιάσουμε επί το a επανειλημμένα k - 1 φορές. Ο δεύτερος τρόπος, αποκαλούμενος και "**δυαδική μέθοδος**" ή μέθοδος "**ύψωσε στο τετράγωνο και πολλαπλασίασε**" ("binary method" or "square and multiply" method) είναι πιο αποτελεσματικός και ήταν γνωστός στην Ινδία 2000 χρόνια πριν.

Η δυαδική μέθοδος για τον υπολογισμό του a^k, $a \in R$, $k \in \mathbb{Z}_{>0}$, δουλεύει ως εξής: Γράφουμε τον k στην δυαδική του μορφή, δηλαδή $k = \sum_{0 \le i \le n} b_i 2^i$. Μετά αντικαθιστούμε κάθε "1" με το ζεύγος των γραμμάτων "SM" και κάθε "0" με το γράμμα "S". Διαγράφουμε το "SM" που είναι στο αριστερό μέρος. Η ακολουθία των γραμμάτων που μένουν μας δίνει τον κανόνα υπολογισμού του a^k, αν ερμηνεύσουμε το "S" σαν "ύψωσε στο τετράγωνο" και το "M" σαν "πολλαπλασίασε επί το a". Εννοείται πως οι πράξεις γίνονται στον δακτύλιο R.

Παράδειγμα:
Έστω $R = \mathbb{Z}/11\mathbb{Z} = \mathbb{Z}_{11}$. Για να υπολογίσουμε το 4^{-1} με το Θεώρημα του Fermat πρέπει να υπολογίσουμε το 4^9 στον δακτύλιο $\mathbb{Z}_{11}$. Η δυαδική παράσταση του 9 είναι {1,0,0,1}, από την οποία προκύπτει αρχικά η ακολουθία SM, S, S, SM και τελικά η S, S, SM. Άρα, το 4^9 υπολογίζεται **στον $\mathbb{Z}_{11}$** υψώνοντας στο τετράγωνο 3 φορές και πολλαπλασιάζοντας επί 4, δηλαδή $4^9 = (((4^2)^2)^2 4 \in \mathbb{Z}_{11}$. Δηλαδή έχουμε $4^2 \equiv 5 \bmod 11$, $5^2 \equiv 3 \bmod 11$, $3^2 \equiv 9 \bmod 11$, και $9 \cdot 4 \equiv 3 \equiv 4^{-1} \bmod 11$.

Προφανώς, η μέθοδος αυτή είναι πολύ πιο γρήγορη από το να υπολογίσουμε πρώτα το $4^9 = 262144$ και ύστερα να βρούμε το υπόλοιπο mod 11.

Αλγόριθμος: Δυαδική μέθοδος ύψωσης σε δύναμη

Είσοδος: Το ζεύγος $\{a, k\}$. $a \in R$, όπου R είναι ένας δακτύλιος με μονάδα και $k \in \mathbb{Z}_{>0}$.

Έξοδος: $a^k \in R$.

==========

1. $k = \sum_{0 \le i \le n} b_i 2^i = 2^n + b_{n-1} \cdot 2^{n-1} + \cdots + b_1 \cdot 2 + b_0$;

$c \leftarrow a$.

2. **for** $i = n - 1, n - 2, \ldots, 0$ **do**

if $b_i = 1$ **then** $c \leftarrow c^2 \cdot a$ **else** $c \leftarrow c^2$.

3. **return** c.

==========

Αξίζει να σημειωθεί πως ο αλγόριθμος δουλεύει από τα αριστερά προς τα δεξιά όσον αφορά τα δυαδικά ψηφία b_i, τα οποία πρέπει να αποθηκευτούν. Στις ασκήσεις ζητείται να τροποποιήσετε τον αλγόριθμο ώστε να δουλεύει από τα δεξιά προς τα αριστερά όσον αφορά τα δυαδικά ψηφία, τα οποία να υπολογίζονται ένα-ένα χωρίς να αποθηκεύονται. (Βλέπε και την βιβλιογραφία.)

Ακολουθεί το πρόγραμμα `binaryExp[]`, που εφαρμόζει τον αλγόριθμο στο Mathematica. Στις ασκήσεις ζητείται να προγραμματίσετε την μέθοδο με την ακολουθία των γραμμάτων. Αξίζει να προσέξετε την συνάρτηση `Fold[]`.

```
binaryExp[a_ /; a ∈ Integers, k_ /; k ∈ Integers && k > 0] :=
 Module[{bits},
  bits = Drop[IntegerDigits[k, 2], 1
   ];
  Fold[If[#2 == 1, #1^2 a, #1^2] &, a, bits]
 ]
```

Έτσι για παράδειγμα έχουμε

```
binaryExp[1234567, 89] == 1234567^89

True
```

Ας δούμε τώρα τον χρόνο υπολογισμού του αλγορίθμου. Εκτελούνται $\lfloor \log_2 k \rfloor$ τετραγωνισμοί και $w(k) - 1 \le \lfloor \log_2 k \rfloor$ πολλαπλασιασμοί στο R, όπου $w(k)$ είναι ο αριθμός των "1" στην δυαδική παράσταση του k. Συνεπώς γίνονται συνολικά το πολύ $2 \log_2 k$ πολλαπλασιασμοί.

Ο αριθμός $w(k)$ είναι γνωστός και σαν το ***βάρος* Hamming (**Hamming weight). Στο *Mathematica* το βάρος Hamming ενός ακεραίου υπολογίζεται από την συνάρτηση

```
DigitCount[k = 9, 2, 1]
```

```
2
```

Η δυαδική μέθοδος ύψωσης σε δύναμη αποτελεί και την βάση μιας μεθόδου πολλαπλασιασμού ακεραίων, γνωστής σαν ***μέθοδος των*** "**Ρώσων *χωρικών***" ("Russian peasant" method). Βασίζεται στο γεγονός ότι αν b είναι άρτιος τότε $a \cdot b = (2a) \cdot (\frac{b}{2})$ ενώ αν b είναι περιττός τότε $a \cdot b = (2a) \cdot \frac{(b-1)}{2} + a$. Την εξηγούμε με ένα παράδειγμα.

Έστω ότι θέλουμε να πολλαπλασιάσουμε τους ακεραίους 38 και 19. Θα κάνουμε διπλασιασμούς ακεραίων, διαιρέσεις δια 2 και πρόσθεση.

Στην πρώτη σειρά της πρώτης στήλης του πίνακα που ακολουθεί είναι ο 38 και κάθε επόμενη σειρά είναι το διπλάσιο της προηγούμενης.

Στην πρώτη σειρά της δεύτερης στήλης του πίνακα γράφουμε το 19. Για κάθε σειρά της στήλης αυτής ισχύει το εξής: αν ο αριθμός της σειράς είναι άρτιος, απλά διαιρούμε δια 2 και γράφουμε το πηλίκο στην επόμενη σειρά. Αν ο αριθμός της σειράς είναι περιττός, αφαιρούμε μονάδα πριν τον διαιρέσουμε δια 2 και μεταφέρουμε τον αντίστοιχο (για την σειρα) αριθμό της πρώτης στήλης στην τρίτη.

Στο τέλος προσθέτουμε τους αριθμούς της τρίτης στήλης.

a	*b*	*sum*
38	19	38

76	9	76
152	4	-
304	2	-
608	1	608
το γινόμενο:		**722**

Ακολουθεί *η εφαρμογή της μεθόδου αυτής στο Mathematica*. Προσέξτε πως *αντικαταστήσαμε την δεύτερη στήλη με αντιστροφή της σειράς των δυαδικών ψηφίων του b και την τρίτη στήλη με εσωτερικό γινόμενο*.

```
binaryMultiplication[a_ /; a ∈ Integers, b_ /; b ∈ Integers] :=
 Module[{bits},
  bits = IntegerDigits[b, 2] // Reverse;
  bits.Table[2^(n-1) a, {n, 1, Length[bits]}]
 ]
```

Έτσι *έχουμε*

```
binaryMultiplication[38, 19] == 38 19
```

```
True
```

3.4 Ευκλείδειος αλγόριθμος και συνεχή κλάσματα

Τα συνεχή κλάσματα παίζουν σημαντικό ρόλο σε διάφορες περιοχές των Μαθηματικών. Σε ένα από τα επόμενα κεφάλαια θα δούμε πως τα συνεχή κλάσματα αποτελούν την βάση της πιο γρήγορης και αποτελεσματικής μεθόδου για την απομόνωση των πραγματικών ριζών πολυωνύμων με ακέραιους συντελεστές. Σε αυτό το κεφάλαιο εισάγουμε τα συνεχή κλάσματα με την βοήθεια του Ευκλείδειου αλγόριθμου.

Έστω λοιπόν R μία Ευκλείδεια περιοχή, $a_0, a_1 \in R$ και $q_i, a_i \in R$, $1 \le i \le L$ τα πηλίκα και υπόλοιπα για τα a_0, a_1 στον κλασσικό Ευκλείδειο αλγόριθμο. Τότε, απαλείφοντας διαδοχικά τα υπόλοιπα έχουμε

$$\frac{a_0}{a_1} = \frac{q_1 a_1 + a_2}{a_1} = q_1 + \frac{a_2}{a_1} = q_1 + \frac{1}{\frac{a_1}{a_2}} = q_1 + \frac{1}{q_2 + \frac{a_3}{a_2}} =$$

$$= q_1 + \frac{1}{q_2 + \frac{1}{\frac{a_2}{a_3}}} = \ldots = q_1 + \cfrac{1}{q_2 + \cfrac{1}{q_3 + \cfrac{1}{\ddots + \frac{1}{q_L}}}}.$$

Αυτό είναι το ανάπτυγμα του $\frac{a_0}{a_1}$ σε ***συνεχή κλάσματα*** (continued fractions expansion) και ***συμβολίζεται*** με $\frac{a_0}{a_1} = \{q_1, \ldots, q_L\}$. Οι αριθμοί $q_1, \ldots, q_L$ λέγονται ***μερικά πηλίκα*** (partial quotients) και είναι όλοι τους θετικοί. Στην συνέχεια το ανάπτυγμα ενός αριθμού σε συνεχή κλάσματα θα συμβολίζεται ως $\{c_1, \ldots, c_L\}$. Αν χρησιμοποιήσουμε τους πρώτους k αριθμούς $\{c_1, \ldots, c_k\}$, $k < L$, για να προσεγγίσουμε το $\frac{a_0}{a_1}$, τότε έχουμε την k-στή ***συγκλίνουσα*** (kth convergent) του $\frac{a_0}{a_1}$.

Αξίζει να σημειωθεί πως επειδή οι "αριθμητές" παραπάνω είναι όλοι τους 1 το ανάπτυγμα είναι μοναδικό. (Γενικά θα μπορούσαν να είναι αριθμητές οποιαδήποτε στοιχεία του R.)

Παράδειγμα:

Ας θεωρήσουμε το κλάσμα $\frac{8}{5}$. Ο Ευκλείδειος αλγόριθμος για $a_0 = 8$ και $a_1 = 5$ μπορεί να γραφτεί σαν

$$c_1 = \lfloor \frac{a_0}{a_1} \rfloor = \lfloor \frac{8}{5} \rfloor = 1, \quad \frac{a_2}{a_1} = \frac{a_0}{a_1} - c_1 = \frac{3}{5},$$
$$c_2 = \lfloor \frac{a_1}{a_2} \rfloor = \lfloor \frac{5}{3} \rfloor = 1, \quad \frac{a_3}{a_2} = \frac{a_1}{a_2} - c_2 = \frac{2}{3},$$
$$c_3 = \lfloor \frac{a_2}{a_3} \rfloor = \lfloor \frac{3}{2} \rfloor = 1, \quad \frac{a_4}{a_3} = \frac{a_2}{a_3} - c_3 = \frac{1}{2},$$
$$c_4 = \lfloor \frac{a_3}{a_4} \rfloor = \lfloor \frac{2}{1} \rfloor = 2, \quad \frac{a_5}{a_4} = \frac{a_3}{a_4} - c_4 = 0,$$

και το ανάπτυγμα του $\frac{8}{5}$ *σε συνεχές κλάσμα είναι*

$$\frac{8}{5} = \{1, 1, 1, 2\} = 1 + \frac{1}{1+\frac{1}{1+\frac{1}{2}}}.$$

Με *το Mathematica βρίσκουμε το ίδιο αποτέλεσμα*:

```
ContinuedFraction[8/5]
```

```
{1, 1, 1, 2}
```

Σημειώνουμε *πως ο* Ευκλείδειος *αλγόριθμος μπορεί να χρησιμοποιηθεί μόνο για το ανάπτυγμα ρητών αριθμών σε συνεχή κλάσματα*. Παρ' όλα *αυτά, αν η* Ευκλείδεια *περιοχή* $R = \mathbb{Z}$, *τότε και κάθε στοιχείο α των πραγματικών αριθμών* $\mathbb{R}$ *έχει ανάπτυγμα σε συνεχή κλάσματα, με την έννοια ότι τα αρχικά τμήματα συγκλίνουν στο α ως προς την απόλυτο τιμή*.

Επίσης, *φορτώνοντας το πακέτο*

```
<< NumberTheory`ContinuedFractions`
```

μπορούμε να έχουμε την πραγματική μορφή του συνεχούς κλάσματος.

```
ContinuedFractionForm[ContinuedFraction[8/5]]
```

$$1 + \frac{1}{1 + \frac{1}{1+\frac{1}{2}}}$$

από την οποία πάλι βρίσκουμε το αρχικό κλάσμα

```
Normal[%]
```

$$\frac{8}{5}$$

Για την εύρεση του αναπτύγματος σε συνεχή κλάσματα ενός πραγματικού αριθμού α εργαζόμαστε ως εξής: Θέτουμε $\alpha_1 = \alpha$, $c_1 = \lfloor \alpha_1 \rfloor$, $\beta_2 = \alpha_1 - c_1$, $\alpha_2 = \frac{1}{\beta_2}$, και γενικά $c_i = \lfloor \alpha_i \rfloor$, $\beta_{i+1} = \alpha_i - c_i$, $\alpha_{i+1} = \frac{1}{\beta_{i+1}}$. Προσέξτε πως για όλα τα i έχουμε $0 \le \beta_i \le 1$, και το ανάπτυγμα σταματάει όταν $\beta_i = 0$, που συμβαίνει εάν και μόνον εάν $\alpha \in \mathbb{Q}$.

Παράδειγμα:
Ας υπολογίσουμε τα πρώτα τέσσερα μερικά πηλίκα του αναπτύγματος του π σε συνεχή κλάσματα. Εδώ δεν μπορεί να εφαρμοσθεί ο Ευκλείδειος αλγόριθμος αλλά θεωρούμε γνωστή μία προσέγγιση του π. Έστω λοιπόν

```
N[π, 6]
```

```
3.14159
```

οπότε

$\alpha_1 = 3.14159$, $c_1 = 3$, $\beta_2 = 0.14159$,
$\alpha_2 = \frac{1}{0.14159}$, $c_2 = 7$, $\beta_3 = 0.06265$,
$\alpha_3 = \frac{1}{0.06265}$, $c_3 = 15$, $\beta_4 = 0.9617$,
$\alpha_4 = \frac{1}{0.9617}$, $c_4 = 1$, $\beta_5 = 0.03983$,
κτλ

Με το Mathematica υπολογίζουμε τα 13 πρώτα μερικά πηλίκα του π

```
ContinuedFraction[π, 13]
```

```
{3, 7, 15, 1, 292, 1, 1, 1, 2, 1, 3, 1, 14}
```

και ορισμένων άλλων αριθμών:

```
ContinuedFraction[√3, 13]
```

```
{1, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2}
```

(Προσέξτε εδώ τα επαναλαμβανόμενα ψηφία. Το ανάπτυγμα αυτό γράφεται και σαν $\{1, 1, 2, \overline{1, 2}\}$.)

```
ContinuedFraction[(1 + √5)/2, 13]
```

```
{1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1}
```

(Το ανάπτυγμα αυτό γράφεται και σαν $\{1, 1, \overline{1}\}$.)

```
ContinuedFraction[E, 13]
```

```
{2, 1, 2, 1, 1, 4, 1, 1, 6, 1, 1, 8, 1}
```

Το ανάπτυγμα άρρητων αριθμών, $\alpha \in \mathbb{R}$, σε συνεχή κλάσματα είναι ένας πολύ καλός τρόπος προσέγγισης του α με ρητούς αριθμούς που έχουν "μικρούς" παρανομαστές. (Με το θέμα αυτό ασχολείται η θεωρία Διοφαντικών προσεγγίσεων.)

Όσον αφορά τις συγκλίνουσες μπορούμε να πούμε πως αν

$$p_{-1} = 0,\ p_0 = 1,\quad p_i = c_i\, p_{i-1} + p_{i-2},\quad i \geq 1$$
$$q_{-1} = 1,\ q_0 = 0,\quad q_i = c_i\, q_{i-1} + q_{i-2},\quad i \geq 1$$

τότε η k-στή συγκλίνουσα του $\frac{a_0}{a_1}$ είναι $r_k = \frac{p_k}{q_k}$, $\gcd(p_k, q_k) = 1$. Ισχύει το εξής θεώρημα η απόδειξη του οποίου βρίσκεται στην βιβλιογραφία:

Θεώρημα 3.4.1:
Αν θεωρήσουμε το (πεπερασμένο ή άπειρο) ανάπτυγμα σε συνεχή κλάσματα $\alpha = \{c_1, \ldots, c_L, \ldots\}$ και συμβολίσουμε με $r_k = \frac{p_k}{q_k}$ την k-στή συγκλίνουσα στο α, τότε ισχύουν:

α. $p_k q_{k-1} - p_{k-1} q_k = (-1)^k,\qquad k \geq 0,$

β. $r_k - r_{k-1} = \frac{(-1)^k}{q_k\, q_{k-1}}$, $\qquad k \geq 2$,

γ. $r_k - r_{k-2} = \frac{(-1)^{k-1}\, c_k}{q_k\, q_{k-2}}$, $\qquad k \geq 3$,

δ. *Για τις περιττές τιμές του k οι συγκλίνουσες αποτελούν μία μονοτονικά αύξουσα ακολουθία με όριο το α, ενώ για τις άρτιες τιμές του k οι συγκλίνουσες αποτελούν μονοτονικά φθίνουσα ακολουθία με όριο το α. Επιπλέον, κάθε* r_{2k-1} *είναι μικρότερο από κάθε* r_{2k} *και κάθε συγκλίνουσα* r_k, $k \geq 3$, *βρίσκεται μεταξύ των δύο προηγουμένων σιγκλινουσών.*

Προσοχή:

Το *παραπάνω θεώρημα ισχύει μόνο για την περίπτωσή μας που το ανάπτυγμα σε συνεχή κλάσματα αρχίζει με την αρίθμηση* c_1. Αν *η αρίθμηση αρχίζει με* c_0 (όπως *σε άλλους συγγραφείς*) *τότε τα πράγματα αλλάζουν ελαφρώς* (*βλέπε την βιβλιογραφία*).

Παράδειγμα:

Ας *προσέξουμε το μέρος* ***δ*** *του* Θεωρήματος 3.4.1. Όπως *είδαμε, το ανάπτυγμα του* $\frac{8}{5}$ *σε συνεχές κλάσμα είναι*:

$$\frac{8}{5} = \{1, 1, 1, 2\} = 1 + \frac{1}{1 + \frac{1}{1 + \frac{1}{2}}}.$$

Οι *συγκλίνουσες είναι*: $r_1 = \frac{p_1}{q_1} = \frac{c_1\, p_0 + p_{-1}}{c_1\, q_0 + q_{-1}} = 1$, $r_2 = \frac{p_2}{q_2} = \frac{c_2\, p_1 + p_0}{c_2\, q_1 + q_0} = 2$, $r_3 = \frac{p_3}{q_3} = \frac{c_3\, p_2 + p_1}{c_3\, q_2 + q_1} = \frac{3}{2}$, $r_4 = \frac{c_4\, p_3 + p_2}{c_4\, q_3 + q_2} = \frac{8}{5}$. *Προσέξτε πως η τιμή* $\frac{8}{5}$ *προσεγγίζεται εναλλάξ μια φορά από αριστερά και μία από τα δεξιά. Για τις περιττές τιμές του k, οι συγκλίνουσες* $r_1 = 1$, $r_3 = \frac{3}{2}$ *αποτελούν αύξουσα ακολουθία ενώ για τις άρτιες τιμές του k, οι συγκλίνουσες* $r_2 = 2$, $r_4 = \frac{8}{5}$ *αποτελούν φθίνουσα ακολουθία.*

Τα *ίδια αποτελέσματα βρίσκουμε και με το Mathematica.*

```
Convergents[8/5]
```

$\{1, 2, \frac{3}{2}, \frac{8}{5}\}$

Αξίζει να σημειωθούν τα εξής:

Α. Η προσέγγιση ικανοποιεί την σχέση

$$|\alpha - \frac{p_k}{q_k}| \leq \frac{1}{q_k^2}.$$

Παράδειγμα:

Σε κάθε ανάπτυγμα του π σε συνεχή κλάσματα, έστω με 5 όρους

```
ContinuedFraction[π, 5]
```

```
{3, 7, 15, 1, 292}
```

αντιστοιχεί ο ρητός αριθμός (η 5-τη συγκλίνουσα) $r_5 = \frac{p_5}{q_5} = \frac{103993}{33102}$. Αυτό φαίνεται είτε από την συνάρτηση

```
Convergents[ContinuedFraction[π, 5]] // Last
```

$$\frac{103993}{33102}$$

είτε από την συνάρτηση

```
FromContinuedFraction[{3, 7, 15, 1, 292}]
```

$$\frac{103993}{33102}$$

Η δεκαδική μορφή της συγκλίνουσας είναι 3.141592653012

```
N[103993/33102, 13]
```

```
3.141592653012
```

και συγκρίνοντάς την με το π βλέπουμε πως έχουμε υπολογίσει τα πρώτα 10 ψηφία (του π).

```
N[π, 20]
```

```
3.1415926535897932385
```

Προσέξτε πως η ποιότητα της προσέγγισης είναι $|\pi - \frac{p_5}{q_5}| \le \frac{1}{q_5^2}$,

```
Abs[N[π, 20] - 103993/33102] < 1/33102^2
```

```
True
```

όπου $\texttt{N}[\pi, 20] - \frac{103993}{33102} = 0.0000000005778906344$ όπως φαίνεται στον παρακάτω πίνακα. Ο αριθμός των μηδενικών μας δείχνει πόσα ψηφία ταυτίζονται.

```
Table[{rn = FromContinuedFraction[ContinuedFraction[π, i]],
   N[rn, 11], N[π, 20] - rn}, {i, 1, 5}] // MatrixForm
```

$$\begin{pmatrix} 3 & 3.0000000000 & 0.1415926535897932385 \\ \frac{22}{7} & 3.1428571429 & -0.0012644892673496187 \\ \frac{333}{106} & 3.1415094340 & 0.0000832196275290875 \\ \frac{355}{113} & 3.1415929204 & -2.667641890624\times 10^{-7} \\ \frac{103993}{33102} & 3.1415926530 & 5.778906344\times 10^{-10} \end{pmatrix}$$

Στην ***πρώτη σειρά***: **(α)** η πρώτη στήλη δείχνει την 1-τη συγκλίνουσα, r_1, του π που είναι 3, **(β)** η δεύτερη στήλη δείχνει την δεκαδική τιμή (τα 11 πρώτα ψηφία) της συγκλίνουσα r_1 που είναι 3.0000000000 και **(γ)** η τρίτη στήλη δείχνει την ποιότητα της προσέγγισης που είναι 0.1415926535897932385, δηλαδή βρήκαμε μόνον το πρώτο ψηφίο του π. Στην ***δεύτερη σειρά***: **(α)** η πρώτη στήλη δείχνει την 2-τη συγκλίνουσα, r_2, του π που είναι $\frac{22}{7}$, …

Β. Ένα πολύ ενδιαφέρον αποτέλεσμα των Gauß-Kuzmin μας πληροφορεί πως στο ανάπτυγμα σε συνεχή κλάσματα σχεδόν όλων των αριθμών, η πιθανότητα το k-στό μερικό πηλίκο c_k να είναι ίσο με έναν θετικό ακέραιο j είναι

$$\text{Prob}[c_k = j] = \log_2 \frac{(j+1)^2}{j(j+2)}.$$

Αυτό σημαίνει πως για όλους σχεδόν τους αριθμούς η πιθανότητα να είναι $c_k = 1$ είναι περίπου 0.41.

Γ. Ενδιαφέρουσα είναι και η ιστορία υπολογισμού του π. Ο Αρχιμήδης (278-212 πΧ) έδωσε μία γεωμετρική μέθοδο υπολογισμού του π χρησημοποιώντας κανονικά πολύγωνα (με 96 πλευρές) περιγραμμένα και εγγεγραμμένα σε κύκλο. Απέδειξε πως $3\frac{10}{11} < \frac{25344\ (\text{περίμετρος πολυγώνου})}{8069\ (\text{διάμετρος κύκλου})} < \pi < \frac{29376}{9347} < 3\frac{1}{7}$, δηλαδή υπελόγισε τα πρώτα 2 δεκαδικά ψηφία του π (βλέπε και την βιβλιογραφία).

$$\mathbf{N}\left[\frac{\mathbf{29376}}{\mathbf{9347}}\right] - \mathbf{N}\left[\frac{\mathbf{25344}}{\mathbf{8069}}\right]$$

```
0.00191692
```

Μέχρι τον Νεύτωνα η γεωμετρική μέθοδος του Αρχιμήδη χρησιμοποιούταν για την προσέγγιση του π. Αυξάνοντας τον αριθμό των πλευρών των κανονικών πολυγώνων αυξάνεται και η προσέγγιση του π.

Ο Γάλλος μαθηματικός Francois Viète (1540-1603) υπελόγισε το π με ακρίβεια εννέα δεκαδικών ψηφίων με την βοήθεια κανονικού πολυγώνου με 393216 πλευρές. Και στις αρχές του 17ου αιώνα, ο Ολλανδός Ludolph van Ceulen υπελόγισε το π με ακρίβεια 35 δεκαδικών ψηφίων με την βοήθεια κανονικών πολυγώνων με 2^{62} πλευρές!

Ο Νεύτωνας (περίπου το 1670) χρησιμοποιώντας τεχνικές απειροστικού λογισμού και μόλις εννέα όρους ενός αναπτύγματος δυναμοσειράς, με ένα από τα ωραιότερα θεωρήματα των Μαθηματικών, υπελόγισε το π με ακρίβεια εφτά δεκαδικών ψηφίων!

Δ. Εύλογα γεννιέται η ερώτηση σε τι μας χρησιμεύει ο υπολογισμός όλο και περισσότερων ψηφίων του π. Η απάντηση είναι πως ο υπολογισμός πολλών ψηφίων του π επιτυγχάνεται με την βοήθεια γρήγορων αλγορίθμων για μεγάλης ακρίβειας ακεραίους και αριθμούς κινητής υποδιαστολής και είναι ένα καλό τεστ για το computer hardware.

Το *τεστ αυτό δεν συγχωρεί κανένα κατασκευαστικό λάθος και εκτελείται συνήθως σε υπερυπολογιστές πριν φύγουν από το εργαστάσιο*. Με *ένα μεγάλης κλίμακας τεστ από την θεωρία των αριθμών* (twin primes), *ανακαλύφτηκε το* 1996 *το κατασκευαστικό λάθος που είχαν πάνω από ένα εκατομμύριο* Pentium CPU's *και η εταιρία* Intel *αναγκάστηκε να τα αντικαταστήσει με δικά της έξοδα*.

Ακολουθούν *δύο τρόποι παράστασης συνεχών κλασμάτων*: *ο πρώτος χρησιμοποιεί τις λεγόμενες* ***γραμμικές κλασματικές συναρτήσεις*** (linear fractional functions, or Möbius substitutions) *και ο δεύτερος τα* ***συνεχόμενα πολυώνυμα*** (continuants).

■ 3.4.1 Παράσταση συνεχών κλασμάτων με πίνακες γραμμικών κλασματικών συναρτήσεων

Έστω γ ένας θετικός πραγματικός αριθμός και $\{c_1, c_2, c_3, \ldots\}$ το ανάπτυγμά του σε συνεχή κλάσματα. Ας θεωρήσουμε τους (θετικούς πραγματικούς) αριθμούς $\gamma_1 = \{c_2, c_3, \ldots\}$ και $\gamma_2 = \{c_3, c_4, \ldots\}$.

Οι αριθμοί γ_1 και γ_2 συνδέονται με τον γ με τις εξής σχέσεις:

$$\gamma = c_1 + \frac{1}{\gamma_1} = \frac{c_1\,\gamma_1 + 1}{\gamma_1} = \varphi(\gamma_1)$$

ενώ

$$\gamma_1 = c_2 + \frac{1}{\gamma_2} = \frac{c_2\,\gamma_2 + 1}{\gamma_2} = \chi(\gamma_2).$$

Έτσι έχουμε

$$\gamma = (\varphi\circ\chi)(\gamma_2) = \frac{(c_1\,c_2+1)\,\gamma_2+c_1}{c_2\,\gamma_2+1}.$$

Οι συναρτήσεις φ και χ που είναι κλάσματα δύο γραμμικών πολυωνύμων ονομάζονται **γραμμικές κλασματικές συναρτήσεις** (linear fractional functions, or Möbius substitutions), και η δράση τους αντιπροσωπεύεται από έναν πίνακα. Έτσι,

$$\varphi(x) = \frac{a\,x+b}{c\,x+d} \iff \begin{pmatrix} a & b \\ c & d \end{pmatrix}.$$

Η σύνθεση των συναρτήσεων φ και χ αντιπροσωπεύεται από το γινόμενο των πινάκων

$$\begin{pmatrix} c_1 & 1 \\ 1 & 0 \end{pmatrix}\begin{pmatrix} c_2 & 1 \\ 1 & 0 \end{pmatrix} = \begin{pmatrix} c_1\,c_2 + 1 & c_1 \\ c_2 & 1 \end{pmatrix}$$

που συμφωνεί με το αποτέλεσμα $\gamma = (\varphi\circ\chi)(\gamma_2)$ που υπολογίσαμε παραπάνω. Τονίζουμε πως η ορίζουσα των πινάκων αυτών είναι ± 1, και συνεπώς υπάρχει και ο αντίστροφος πίνακας.

Από τα παραπάνω συμπεραίνουμε πως στο ανάπτυγμα του γ σε συνεχή κλάσματα μπορούμε να αντιστοιχίσουμε το γινόμενο των πινάκων

$$\begin{pmatrix} c_1 & 1 \\ 1 & 0 \end{pmatrix}\begin{pmatrix} c_2 & 1 \\ 1 & 0 \end{pmatrix}\begin{pmatrix} c_3 & 1 \\ 1 & 0 \end{pmatrix}\cdots$$

και αν $r_{k-1} = \frac{p_{k-1}}{q_{k-1}}$ *και* $r_k = \frac{p_k}{q_k}$ *είναι δύο διαδοχικές συγκλίνουσες τότε έχουμε* (*με την βοήθεια των αναγωγικών τύπων υπολογισμού των συγκλινουσών*):

$$\begin{pmatrix} p_k & p_{k-1} \\ q_k & q_{k-1} \end{pmatrix}\begin{pmatrix} c_{k+1} & 1 \\ 1 & 0 \end{pmatrix} = \begin{pmatrix} c_{k+1}\,p_k + p_{k-1} & p_k \\ c_{k+1}\,q_k + q_{k-1} & q_k \end{pmatrix} = \begin{pmatrix} p_{k+1} & p_k \\ q_{k+1} & q_k \end{pmatrix}.$$

Έτσι *έχουμε τον εξής συμβολισμό*:

$$\gamma = \{c_1, c_2, \ldots\} = \begin{pmatrix} c_1 & 1 \\ 1 & 0 \end{pmatrix}\begin{pmatrix} c_2 & 1 \\ 1 & 0 \end{pmatrix}\cdots.$$

Αν *χρησιμοποιήσουμε τους k πρώτους όρους θα έχουμε*

$$\begin{pmatrix} p_k & p_{k-1} \\ q_k & q_{k-1} \end{pmatrix} = \begin{pmatrix} c_1 & 1 \\ 1 & 0 \end{pmatrix}\begin{pmatrix} c_2 & 1 \\ 1 & 0 \end{pmatrix}\cdots\begin{pmatrix} c_k & 1 \\ 1 & 0 \end{pmatrix},$$

όπου

$$\frac{p_k}{q_k} = \{ c_1, c_2, \ldots, c_k \}.$$

■ 3.4.2 Παράσταση συνεχών κλασμάτων με συνεχόμενα πολυώνυμα (continuants)

Τα *πολυώνυμα αυτά μελετήθηκαν διεξοδικά από τον* Euler. Το ***συνεχόμενο πολυώνυμο*** $K_n(x_1, x_2, \ldots, x_n)$ *έχει n παραμέτρους και ορίζεται αναγωγικά ως εξής*:

$$K_0() = 1,$$
$$K_1(x_1) = x_1,$$
$$K_n(x_1, \ldots, x_n) = x_n \cdot K_{n-1}(x_1, \ldots, x_{n-1}) + K_{n-2}(x_1, \ldots, x_{n-2}).$$

Στο *Mathematica τα πολυώνυμα ορίζονται αναγωγικά ως εξής* (*ο δείκτης n δεν χρειάζεται όταν συνεπάγεται*):

```
K[] := 1;
K[x_] := K[x] = x ;
K[a___, x_, y_] := K[a, x, y] = y K[a, x] + K[a]
```

(Στις ασκησεις ζητείται να προγραμματισθεί με έναν άλλο τρόπο, που δεν στηρίζεται σε pattern matching.) Για παράδειγμα, μετά το $K_1(x_1)$ έχουμε:

```
K[x1, x2]
```

$1 + x_1 x_2$

```
K[x1, x2, x3] // Expand
```

$x_1 + x_3 + x_1 x_2 x_3$

```
K[x1, x2, x3, x4] // Expand
```

$1 + x_1 x_2 + x_1 x_4 + x_3 x_4 + x_1 x_2 x_3 x_4$

Με επαγωγή εύκολα βλέπουμε πως $K_n(1, 1, \ldots, 1) = F_{n+1}$, ο $(n + 1)$-στός αριθμός Fibonacci. Αυτό επιβεβαιώνεται και από το *Mathematica*. Πράγματι, αν ενεργοποιήσουμε τον κώδικα

```
F[0] = 0; F[1] = 1;
F[n_] := F[n] = F[n – 1] + F[n – 2]
```

για τον υπολογισμό των αριθμών της ακολουθίας Fibonacci έχουμε:

```
n = 18;
K[Apply[Sequence, Table[1, {i, n}]]] == F[n + 1]
```

```
True
```

όπου η εντολή `Apply[Sequence,Table[1,{i, n}]]` μας δημιουργεί την ακολουθία των 1 μέσα στην συνάρτηση `K[]`.

Ο Euler παρατήρησε πως το $K_n(x_1, \ldots, x_n)$ μπορεί να βρεθεί αν αρχίσουμε με το γινόμενο $x_1 \cdot x_2 \cdots x_n$ και μετά διαγράψουμε συνεχόμενα ζευγάρια $x_k \cdot x_{k+1}$ με όλους τους δυνατούς τρόπους. Με τον τρόπο αυτό μπορούμε να κατασκευάσουμε την ακολουθία των κώδικων Morse των n παραμέτρων. Μια τελίτσα συνεισφέρει μία μονάδα στο μήκος και αντιστοιχεί σε μεταβλητή που περιέχεται στον όρο ενώ

μια παύλα συνεισφέρει δύο μονάδες στο μήκος και αντιστοιχεί σε ένα ζεύγος που εξαιρείται.

Για παράδειγμα, αν $n = 4$, τότε έχουμε την εξής ακολουθία κώδικων Morse:

. . . . . . – . – . – . . – –

Έτσι, για παράδειγμα, . . – αντιστοιχεί στον όρο $x_1 \cdot x_2$. Ένας κώδικας Morse μήκους n με k παύλες έχει $n - 2k$ τελίτσες και $n - k$ σύμβολα συνολικά. Αυτές οι τελίτσες και παύλες μπορούν να τακτοποιηθούν με $\binom{n-k}{k}$ τρόπους. Αν λοιπόν αντικαταστήσουμε κάθε τελίτσα με το γράμμα z και κάθε παύλα με τον αριθμό 1 προκύπτει:

$$K_n(z, z, \ldots, z) = \sum_{k=0}^{n} \binom{n-k}{k} z^{n-2k}.$$

Ξέρουμε επίσης πως ο αριθμός των όρων ενός συνεχόμενου πολυωνύμου είναι ένας αριθμός Fibonacci. Έτσι έχουμε

$$F_{n+1} = \sum_{k=0}^{n} \binom{n-k}{k} z^{n-2k}.$$

Εξ αιτίας της σχέσης με τον κώδικα Morse τα συνεχόμενα πολυώνυμα έχουν την ακόλουθη συμμετρία

$$K(x_1, x_2, \ldots, x_n) = K(x_n, \ldots, x_2, x_1),$$

η οποία στις ασκήσεις αποδεικνύεται επαγωγικά. Η συμμετρία αυτή έχει σαν αποτέλεσμα να ισχύει και ο συμμετρικός ορισμός

$$K_n(x_1, \ldots, x_n) = x_1 \cdot K_{n-1}(x_2, \ldots, x_n) + K_{n-2}(x_3, \ldots, x_n).$$

Τα συνεχόμενα πολυωνύμα σχετίζονται και με τον Ευκλείδειο αλγόριθμο. Έστω για παράδειγμα ότι ο υπολογισμός $\gcd(a, b)$ τελειώνει σε τρία βήματα. Δηλαδή έστω ότι έχουμε:

$$\begin{aligned}
\gcd(a, b) &= \gcd(a_0, a_1) && a_0 = a, \quad a_1 = b,\\
&= \gcd(a_1, a_2) && a_2 = \mathrm{rem}(a_0, a_1),\\
&= \gcd(a_2, a_3) && a_3 = \mathrm{rem}(a_1, a_2),\\
&= \gcd(a_3, 0) = a_3. && 0 = \mathrm{rem}(a_2, a_3).
\end{aligned}$$

Τότε, έχουμε:

$$\begin{aligned}
a_3 &= a_3 && = a_3 \cdot K()\\
a_2 &= q_3 a_3 && = a_3 \cdot K(q_3)\\
a_1 &= q_2 a_2 + a_3 && = a_3 \cdot K(q_2, q_3)\\
a_0 &= q_1 a_1 + a_2 && = a_3 \cdot K(q_1, q_2, q_3)
\end{aligned}$$

Γενικά, αν ο Ευκλείδειος αλγόριθμος βρίσκει τον μκδ d των a και b σε k βήματα, έχοντας υπολογίσει την ακολουθία των πηλίκων $q_1, q_2, \ldots, q_k$, τότε $a = d \cdot K(q_1, q_2, \ldots, q_k)$ και $b = d \cdot K(q_2, \ldots, q_k)$.

Τα συνεχόμενα πολυώνυμα σχετίζονται επίσης και με τα συνεχή κλάσματα, από όπου πήραν και το όνομά τους. Έτσι έχουμε για παράδειγμα

$$a_1 + \cfrac{1}{a_2 + \cfrac{1}{a_3 + \cfrac{1}{\ddots + \cfrac{1}{a_k}}}} = \frac{K(a_1, a_2, \ldots, a_k)}{K(a_2, \ldots, a_k)}.$$

Δηλαδή, έχουμε τις σχέσεις

$$\begin{aligned}
p_k &= K(a_1,\ a_2,\ \ldots,\ a_k),\\
q_k &= K(a_2,\ a_2,\ \ldots,\ a_k),
\end{aligned}$$

όπου $r_k = \frac{p_k}{q_k}$ η k-στή συγκλίνουσα. Άρα

$$\frac{K(a_1, a_2, \ldots, a_k)}{K(a_2, \ldots, a_k)} = \{a_1,\ a_2,\ \ldots,\ a_k\}.$$

Πράγματι, με το Mathematica έχουμε για παράδειγμα

```
FromContinuedFraction[{1, 2, 3, 4}] == K[1, 2, 3, 4]/K[2, 3, 4]
```

```
True
```

3.5 Γραμμικές Διοφαντικές εξισώσεις

Η λύση γραμμικών Διοφαντικών εξισώσεων είναι ακόμα μία εφαρμογή του επεκταμένου Ευκλείδειου αλγόριθμου. Έστω $a,b,d \in \mathbb{Z}$ και θέλουμε να βρούμε ακέραιες λύσεις $s,t \in \mathbb{Z}$ της εξίσωσης

$$sa + tb = d. \tag{1}$$

Το σύνολο όλων των πραγματικών λύσεων της εξίσωσης αυτής είναι μία ευθεία στο επίπεδο $\mathbb{R}^2$, που μπορεί να γραφτεί σαν το άθροισμα $u + U$, μιας συγκεκριμένης λύσης $u = \{s_0, t_0\} \in \mathbb{R}^2$ και του συνόλου U της ομογενούς εξίσωσης

$$sa + tb = 0. \tag{2}$$

Η ιδιότητα αυτή γενικεύεται και για τις ακέραιες λύσεις, όπως φαίνεται στο ακόλουθο θεώρημα.

Θεώρημα 3.5.1:
Έστω R μία Ευκλείδεια περιοχή, $a,b,d \in R$ όπου a, b δεν είναι και τα δύο μηδέν (οπότε $\gcd(a, b) \neq 0$).

α. Η εξίσωση (1) έχει λύση εάν και μόνον εάν $\gcd(a, b)$ διαιρεί το d.

β. Αν $\{s_0, t_0\} \in R^2$ είναι μία λύση της εξίσωσης (1), τότε το σύνολο όλων των λύσεων είναι $\{s_0, t_0\} + U$, όπου

$$U = R\cdot\left(\frac{b}{\gcd(a, b)}, \frac{-a}{\gcd(a, b)}\right) \subseteq R^2.$$

είναι το σύνολο των λύσεων της ομογενούς εξίσωσης (2).

γ. Αν $R = F[x]$, όπου F είναι ένα σώμα, η εξίσωση (1) έχει λύση και $\deg(a) + \deg(b) - \deg(\gcd(a, b)) > \deg(d)$, τότε υπάρχει μία μοναδική λύση $\{s, t\} \in R^2$ της (1) ώστε $\deg(s) < \deg(b) - \deg(\gcd(a, b))$ και $\deg(t) < \deg(a) - \deg(\gcd(a, b))$.

Απόδειξη:

α. Αν $\{s, t\} \in R^2$ είναι λύση της (1), τότε gcd(a, b) διαιρεί $sa + tb$ και επομένως d. Αντίστροφα, έστω ότι gcd(a, b) διαιρεί d και ισχύει $d = k\cdot\gcd(a, b)$. Τότε με τον επεκταμένο Ευκλείδειο αλγόριθμο βρίσκουμε s_g, t_g ώστε $s_g a + t_g b = \gcd(a, b)$, οπότε μία λύση της (1) είναι $\{s, t\} = \{s_g\cdot k, t_g\cdot k\}$

β. Επειδή τα στοιχεία $\frac{a}{\gcd(a,b)}$ και $\frac{b}{\gcd(a,b)}$ είναι πρώτα μεταξύ τους έχουμε από την (2):

$$s\frac{a}{\gcd(a,b)} = -t\frac{b}{\gcd(a,b)} \Leftrightarrow \exists k \in R \mid s = k\frac{b}{\gcd(a,b)} \text{ και } t = k\frac{-a}{\gcd(a,b)} \Leftrightarrow \{s, t\} \in U.$$

(Τα πρόσημα των s, t θα μπορούσαν να είναι και αντίθετα. Η παρουσίασή μας ακολουθεί το Mathematica

```
Reduce[3 s + 5 t == 0, {s, t}, Integers]

C[1] ∈ Integers && s == 5 C[1] && t == -3 C[1]
```

όπως βλέπουμε στο παράδειγμα.)

Η απόδειξη του (**β**) τελειώνει με την παρατήρηση ότι από την (1) έχουμε

$$(s - s_0)a + (t - t_0)b = 0 \Leftrightarrow \{s - s_0, t - t_0\} \in U.$$

γ. Έστω $\{s_0, t_0\} \in R^2$ μία λύση της (1). Για να αποδείξουμε την **ύπαρξη** λύσης που πληρεί τις συνθήκες διαιρούμε κατ' αρχάς το t_0 με το $\frac{a}{\gcd(a,b)}$ για να λάβουμε $q,t \in R$ ώστε $t_0 = q\frac{a}{\gcd(a,b)} + t$. Προφανώς $\deg(t) < \deg(a) - \deg(\gcd(a, b))$, όπως το θέλουμε, και $t = t_0 - q\frac{a}{\gcd(a,b)}$. Θέτουμε τώρα $s = s_0 + q\frac{b}{\gcd(a,b)}$ και βλέπουμε (από το (**β**)) πως $\{s, t\} = \{s_0, t_0\} + q\{\frac{b}{\gcd(a,b)}, \frac{-a}{\gcd(a,b)}\}$ είναι λύση της (1).

Μένει να δείξουμε πως $\deg(s) < \deg(b) - \deg(\gcd(a, b))$. Όμως έχουμε, εξ υποθέσεως, $\deg(d) < \deg(a) + \deg(b) - \deg(\gcd(a, b))$, και εξ ορισμού $\deg(t) < \deg(a) - \deg(\gcd(a, b))$. Από αυτά συνεπάγεται πρώτα $\deg(tb) < \deg(a) + \deg(b) - \deg(\gcd(a, b))$ και έπειτα $\deg(s) + \deg(a) = \deg(sa) = \deg(d - tb) < \deg(a) + \deg(b) - \deg(\gcd(a, b))$, απ' όπου προκύπτει η ζητούμενη ανισότητα.

Η *αποδείξη της* ***μοναδικότητας*** *της λύσης συνεπάγεται από το γεγονός ότι* (*όπως ξέρουμε από την* (***β***)) *μια άλλη λύση θα ήταν της μορφής* $\{s_{\text{other}} = s + \frac{k\,b}{\gcd(a,b)}, t_{\text{other}} = t - \frac{k\,b}{\gcd(a,b)}\}$. *Όμως* $\deg(s_{\text{other}} = s + \frac{k\,b}{\gcd(a,b)}) = \deg(\frac{k\,b}{\gcd(a,b)}) \geq \deg(b) - \deg(\gcd(a, b))$ *και επομένως* $\deg(s_{\text{other}})$ *δεν πληρεί την συνθήκη*.

Παραδείγματα:

Η *εύρεση μιάς λύσης της γραμμικής Διοφαντικής εξίσώσης* $3s + 5t = 1$ *με το Mathematica επιτυγχάνεται με την συνάρτηση* `ExtendedGCD[]`

```
ExtendedGCD[3, 5]
```

```
{1, {2, -1}}
```

και είναι {2, -1}. Η *εύρεση της γενικής λύσης της ίδιας εξίσωσης* (*ή της αντίστοιχης ομογενούς εξίσωσης*) *επιτυγχάνεται με την συνάρτηση* `Reduce[]`

```
Reduce[3 s + 5 t == 1, {s, t}, Integers]
```

```
C[1] ∈ Integers && s == 2 + 5 C[1] && t == -1 - 3 C[1]
```

Στις *ασκήσεις ζητείται να γραφτεί για την περίπτωση αυτή* (*και χωρίς την χρήση της* `Reduce[]`) *ένα μικρό πρόγραμμα στο Mathematica*.

Όσον *αφορά τα πολυώνυμα έχουμε την μοναδική λύση*

```
<< Algebra`PolynomialExtendedGCD`;
{d, {s, t}} = PolynomialExtendedGCD[a = x^3 - 7 x + 7, b = 3 x^2 - 7]
```

$$\left\{1, \left\{-\frac{27}{7} - \frac{18\,x}{7}, -4 + \frac{9\,x}{7} + \frac{6\,x^2}{7}\right\}\right\}$$

```
s a + t b // Expand
```

```
1
```

που πληρεί τις συνθήκες

```
Exponent[s, x] < Exponent[b, x] – Exponent[d, x]
```

```
True
```

```
Exponent[t, x] < Exponent[a, x] – Exponent[d, x]
```

```
True
```

Ασκήσεις

1: Κάνοντας χρήση των ορισμών αποδείξτε πως:

α. Αν $a \equiv b \bmod m$ και $d \mid m$, τότε $a \equiv b \bmod d$.

β. Αν $a \equiv b \bmod m$ και $a \equiv b \bmod n$, τότε $a \equiv b \bmod \operatorname{lcm}(m, n)$, όπου $\operatorname{lcm}(m, n)$ είναι το ελάχιστο κοινό πολλαπλάσιο των m, n.

γ. Αν $a \equiv c \bmod m$ και $b \equiv d \bmod m$, τότε $a + b \equiv c + d \bmod m$, $a - b \equiv c - d \bmod m$, και $a \cdot b \equiv c \cdot d \bmod m$.

δ. (Απαλοιφή) Αν $a \cdot b \equiv a \cdot c \bmod m$, τότε $b \equiv c \bmod \frac{m}{\gcd(a, m)}$. Αν $\gcd(a, m) = 1$, τότε από $a \cdot b \equiv a \cdot c \bmod m$, συνεπάγεται $b \equiv c \bmod m$.

2: Αποδείξτε πως ο ακέραιος $a = \sum_{0 \le i \le k} a_i 10^i$ διαιρείται από το 11 εάν και μόνον εάν το εναλλασσόμενο άθροισμα των ψηφίων του $a_0 - a_1 + a_2 - a_3 \pm \cdots (-1)^k a_k$ διαιρείται από το 11. Υπόδειξη: $10^k \equiv (-1)^{k+1} \bmod 11$.

3: Αν F ένα σώμα και $f \in F[x]$ ένα μη παραγοντοποιήσιμο πολυώνυμο βαθμού n και με μοναδιαίο κύριο συντελεστή, αποδείξτε πως οι αριθμητικές πράξεις (πρόσθεσης, πολλαπλασιασμού, διαίρεσης με αντιστρέψιμο στοιχείο) στο σώμα $E = F[x]/\langle f \rangle$ εκτελούνται με $O(n^2)$ πράξεις στο F. (Υπόδειξη: Λήμματα 2.3.1 και 2.3.2.)

4: Αποδείξτε πως οι αριθμητικές πράξεις στο $\mathbb{Z}_n$, εκτελούνται με $O(\lambda(n)^2)$ πράξεις σε λέξεις του υπολογιστή, όπου $\lambda(n)$ είναι το μήκος του n ως προς την βάση β.

5: Αποδείξτε το "μικρό" θεώρημα του Fermat με την χρήση του "κανόνα των αρχαρίων".

6: Αποδείξτε πως ο αντίστροφος ενός αριθμού mod p υπολογίζεται με το θεώρημα του Fermat με $O(\lambda(p)^3)$ πράξεις σε λέξεις του υπολογιστή. (Η μέθοδος αυτή προτιμάται για υπολογισμούς με το χέρι.) Σε αντίθεση, με τον επεκταμένο Ευκλείδειο αλγόριθμο απαιτούνται $O(\lambda(p)^2)$ πράξεις σε λέξεις του υπολογιστή, όπου $n = \lambda(a)$, και $m = \lambda(b)$. (Υπόδειξη: Άσκηση 4 και Τμήμα 3.3.)

7: Τροποποιήστε τον αλγόριθμο για την δυαδική μέθοδο ύψωσης σε δύναμη ώστε να δουλεύει από δεξιά προς τα αριστερά και να μην χρειάζεται να αποθηκεύονται τα δυαδικά ψηφία του εκθέτη.

8: Προγραμματίστε στο *Mathematica* την δυαδική μέθοδο ύψωσης σε δύναμη με τη ακολουθία των γραμμάτων "S" και "M", δηλαδή όπως ακριβώς περιγράφεται στο παράδειγμα του Τμήματος 3.3.

9: Με την βοήθεια της σχέσης

$$\begin{pmatrix} p_k & p_{k-1} \\ q_k & q_{k-1} \end{pmatrix} = \begin{pmatrix} c_1 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} c_2 & 1 \\ 1 & 0 \end{pmatrix} \cdots \begin{pmatrix} c_k & 1 \\ 1 & 0 \end{pmatrix},$$

αποδείξτε ότι $p_k q_{k-1} - p_{k-1} q_k = (-1)^k$, για $k \geq 0$ (σχέση (α) του Θεωρήματος 3.4.1). (Υπόδειξη: Υπολογίστε τις ορίζουσες.)

10: Αποδείξτε με επαγωγή την σχέση συμμετρίας για τα συνεχόμενα πολυώνυμα

$$K(x_1, x_2, \ldots, x_n) = K(x_n, \ldots, x_2, x_1),$$

11: $\boldsymbol{\alpha}$**.** Αποδείξτε την σχέση

$$K(x_1, \ldots, x_n) = x_1 \cdot K(x_2, \ldots, x_n) + K(x_3, \ldots, x_n).$$

χρησιμοποιώντας την σχέση

$$\frac{K(x_1, x_2, \ldots, x_n)}{K(x_2, \ldots, x_n)} = \{x_1, x_2, \ldots, x_n\} = x_1 + \frac{1}{\{x_2, \ldots, x_n\}}.$$

$\boldsymbol{\beta}$**.** Προγραμματίστε τον αναγωγικό ορισμό του συνεχόμενου πολυώνυμου χωρίς pattern matching.

12: Γράψτε ένα πρόγραμμα στο *Mathematica* που να λύνει γραμμικές Διοφαντικές εξισώσεις.

■ **Λύσεις ορισμένων ασκήσεων**

Λύση της άσκησης 8:

```
binaryExp2[a_ /; a ∈ Integers, k_ /; k ∈ Integers && k > 0] :=
 Module[{code, code2},
  code = ToCharacterCode[
    ToString[IntegerDigits[k, 2] /. {1 → SM, 0 → S}]];
  code2 = Drop[Select[code, 77 ≤ # ≤ 83 &], 2];
  Fold[If[#2 == 83, #1^2, #1 a] &, a, code2]
 ]
```

Λύση της άσκησης 10:

Είναι προφανές ότι ισχύει για $n = 0, 1$. Για την επαγωγή έχουμε

$$\begin{aligned} K(x_1, \ldots, x_n) &= x_n \cdot K(x_1, \ldots, x_{n-1}) + K(x_1, \ldots, x_{n-2}) \\ &= x_n \cdot K(x_{n-1}, \ldots, x_1) + K(x_{n-2}, \ldots, x_2) \\ &= K(x_n, \ldots, x_1). \end{aligned}$$

Λύση της άσκησης 11:

α. Εδώ έχουμε:

$$\begin{aligned} \frac{K(x_1, x_2, \ldots, x_k)}{K(x_2, \ldots, x_k)} &= \{x_1,\ x_2,\ \ldots,\ x_k\} = x_1 + \frac{1}{\{x_2, \ldots, x_k\}} \\ &= x_1 + \frac{1}{\frac{K(x_2, \ldots, x_k)}{K(x_3, \ldots, x_k)}} \\ &= \frac{x_1 \cdot K(x_2, \ldots, x_k) + K(x_3, \ldots, x_k)}{K(x_2, \ldots, x_k)} \end{aligned}$$

Εξισώνοντας τους αριθμητές βρίσκουμε την σχέση.

β.

```
Clear[K];
K[{}] := 1;
K[{q_}] := q;
K[q_List] := Last[q] K[Drop[q, -1]] + K[Drop[q, -2]] /; Length[q] > 1
```

```
K[{x1, x2, x3, x4}] // Expand
```

```
1 + x1 x2 + x1 x4 + x3 x4 + x1 x2 x3 x4
```

Λύση της άσκησης 12:

Το ακόλουθο πρόγραμμα μας επιστρέφει μία μερική λύση μαζί με το ζεύγος $\{\frac{b}{d}, -\frac{a}{d}\}$ που μας δίνει την γενική λύση όταν προστεθεί στην μερική.

```
linearDiophantine[a_ x_ + b_ y_ == c_] := Module[
  {d, s, t},
  {d, {s, t}} = ExtendedGCD[a, b];
  If[Mod[c, d] == 0, {{x → c s / d, y → c t / d}, {b / d, -a / d}}, {}]
 ]
```

```
Clear[s, t]; sol = linearDiophantine[3 s + 5 t == 1]
```

```
{{s → 2, t → -1}, {5, -3}}
```

Αν τώρα θέσουμε

```
expr = 3 s + 5 t
```

```
3 s + 5 t
```

βλέπουμε πως για διάφορα k έχουμε διαφορετικές λύσεις

```
Table[expr /. {s → k First[Last[sol]] + Last[First[First[sol]]],
   t → k Last[Last[sol]] + Last[Last[First[sol]]]}, {k, 3, 11}]
```

```
{1, 1, 1, 1, 1, 1, 1, 1, 1}
```

Βιβλιογραφία

1. Akritas, A.G.: *Elements of Computer Algebra with Applications.* Wiley Interscience, New York, 1989.

Dunham, W.: *Journey Through Genius, The Great Theorems of Mathematics*. Wiley Science Editions, New York, 1990.

2. Graham, R.L., Knuth, D.E., and O. Patashnik: *Concrete Mathematics*. Addison-Wesley, New York, 1981.

Κεφάλαιο 4: Παρεμβολή

Μία βασική ιδέα της Υπολογιστικής Άλγεβρας είναι η χρήση διαφόρων μορφών παραστάσεων των υπό εξέταση αντικειμένων για την επιτάχυνση των υπολογισμών.

Έτσι, για παράδειγμα, αν για την παράσταση των πολυωνύμων χρησιμοποιήσουμε τους συντελεστές ο πολλαπλασιασμός τους γίνεται σε χρόνο $O(n^2)$, ενώ αν χρησιμοποιήσουμε τις τιμές τους σε αρκετα σημεία ο πολλαπλασιασμός τους γίνεται (όπως θα δούμε) σε χρόνο $O(n \log n)$.

Υπάρχουν βασικά δύο τρόποι παράστασης των υπό εξέταση αντικειμένων.

Ο ***πρώτος τρόπος*** βασίζεται στην επιλογή μιας βάσης, οπότε το αντικείμενο παρίσταται σαν ανάπτυγμα σε δυνάμεις αυτής της βάσης.

Παραδείγματα αμέσου ενδιαφέροντος είναι: (***α***) η παράσταση των πολυωνύμων με τους συντελεστές τους, όπου η βάση είναι η μεταβλητή x, και (***β***) η δεκαδική ή δυαδική ή β-δική παράσταση των ακεραίων όπου η βάση είναι το 10 ή 2 ή β αντίστοιχα.

Ο ***δεύτερος τρόπος*** βασίζεται στην επιλογή μιας n-διάστατης βάσης, με συνέπεια το αντικείμενο να παρίσταται και αυτό σαν ένα n-διάστατο διάνυσμα.

Παραδείγματα αμέσου ενδιαφέροντος είναι: (***α***) η παράσταση των πολυωνύμων με τις τιμές τους σε διάφορα σημεία $u_0, \ldots, u_{n-1}$, όπου η βάση είναι $x - u_0, \ldots, x - u_{n-1}$ και (***β***) η παράσταση ακεραίων με τα υπόλοιπα modulo διάφορους πρώτους αριθμούς $p_0, \ldots, p_{n-1}$, οι οποίοι αποτελούν και την βάση.

Οταν έχουμε να επιλύσουμε ένα πρόβλημα πρέπει να επιλέξουμε την παράσταση στην οποία λύνεται ευκολότερα. Σημαντικό είναι επίσης να μπορούμε να μετατρέπουμε το πρόβλημα από την μία παράσταση στην άλλη.

Εδώ αξίζει να τονίσουμε την ομοιότητα μεταξύ του υπολογισμού της τιμής ενός πολυωνύμου στο σημείο *u* και του υπολοίπου ενός ακεραίου modulo έναν πρώτο αριθμό *p*.

Όπως όμως έχουμε δεί, ο υπολογισμός της τιμής ενός πολυωνύμου στο σημείο *u* είναι το ίδιο με τον υπολογισμό του υπολοίπου του πολυωνύμου modulo *x - u*. Κατά συνέπεια, ο υπολογισμός του υπολοίπου ενός ακεραίου modulo τον πρώτο αριθμό *p* μπορεί να θεωρηθεί ως "ο υπολογισμός της τιμής του ακεραίου στο σημείο *p*."

Η ***αντίστροφη*** πράξη υπολογισμού των συντελεστών του πολυωνύμου από τις τιμές του σε διάφορα σημεία λέγεται ***παρεμβολή*** (interpolation).

Για τους ακεραίους αυτό επιτυγχάνεται με τον **Κινεζικό αλγόριθμο υπολοίπων** ο οποίος, με βάση τα παραπάνω, αντιστοιχεί με την "παρεμβολή του ακεραίου από τις τιμές του σε διάφορους πρώτους."

Στην συνέχεια θα εξετάσουμε αυτά τα θέματα αλλαγής παράστασης πολυωνύμων και ακεραίων και θα δούμε πως ο Κινεζικός αλγόριθμος υπολοίπων για τα πολυώνυμα είναι η πιο γενική μορφή παρεμβολής.

4.1 Εκτίμηση και μετάθεση πολυωνύμων με την μέθοδο των Ruffini-Horner

Εστω το πολυώνυμο $f(x) = \sum_{i=0}^{n-1} f_i x^i \in F[x]$, όπου F είναι ένα σώμα. Εκτίμηση (evaluation) του πολυωνύμου $f(x)$ σε ένα σημείο $u \in F$ είναι ο υπολογισμός της τιμής $f(u)$. Η ***εκτίμηση*** του πολυωνύμου $f(x)$ στα διαφορετικά σημεία $u_0, \ldots, u_{n-1}$, και η (αντίστροφη πράξη) ***παρεμβολή*** του πολυωνύμου από τις τιμές $f(u_0), \ldots, f(u_{n-1})$, συνιστούν την πιο σημαντική αλλαγή παράστασης και είναι το θέμα αυτού του κεφαλαίου.

Στενά συνδεδεμένη με την εκτίμηση είναι και η ***μετάθεση*** (translation) — των ριζών — του πολυωνύμου $f(x)$, δηλαδή ο υπολογισμός των συντελεστών του πολυωνύμου $f(x + u)$. Η μετάθεση θα μας χρειαστεί στην συζήτηση περί απομόνωσης ριζών πολυωνύμων.

Στο εδάφιο αυτό θα μελετήσουμε τους αλγόριθμους των Ruffini-Horner για την εκτίμηση και μετάθεση πολυωνύμων.

Για την ιστορία αναφέρουμε πως από τους περισσότερους συγγραφείς, οι μέθοδοι αυτοί ονομάζονται απλά Horner. Όμως, από την βιβλιογραφία βλέπουμε πως ο Ruffini (1804) προηγήθηκε του Horner κατά 15 χρόνια. Εκτός των άλλων, στην εργασία του 1804 ο Ruffini απέδειξε πως δεν μπορούμε με "τύπο" να υπολογίσουμε τις ρίζες πολυωνύμου βαθμού 5. Το αποτέλεσμα αυτό γενικεύθηκε από τον Abel για πολυώνυμα βαθμού ≥ 5.

Αρχίζουμε λοιπόν με την εκτίμηση στο σημείο $u \in F$ του πολυωνύμου $f(x) \in F[x]$, όπου F είναι ένα σώμα και

$$f(x) = f_0 + f_1 x + \cdots + f_{n-1} x^{n-1}.$$

Θέλουμε δηλαδή να υπολογίσουμε την τιμή $f(u)$. Η μέθοδος των Ruffini-Horner συνίσταται στον (αποτελεσματικό) υπολογισμό του $f(u)$ με την βοήθεια του τύπου

$$f(u) = f_0 + u(f_1 + \cdots + u(f_{n-2} + u f_{n-1})\ldots).$$

Αν *ο βαθμός του πολυωνύμου είναι n* - 1 = 5, *ο τύπος αυτός προγραμματίζεται απλούστατα στο Mathematica με την συνάρτηση* `Fold[]`

```
Fold[(#1 u + #2) &, 0, Reverse[{f0, f1, f2, f3, f4, f5}]]
```

```
f0 + u (f1 + u (f2 + u (f3 + u (f4 + u f5))))
```

Έτσι *το πρόγραμμα για την* ***εκτίμηση*** *ενός πολυωνύμου στο σημείο u με την μέθοδο των* Ruffini-Horner *είναι*

```
RuffiniHornerEvaluation[f_, u_] :=
 Fold[(#1 u + #2) &, 0, Reverse[CoefficientList[f, Variables[f]]]]
```

Αν *για παράδειγμα* $f(x) = x^3 - 7x + 7$, *τότε την τιμή* $f(3)$ *μπορούμε να την υπολογίσουμε και με την καινούργια μας συνάρτηση*

```
RuffiniHornerEvaluation[x^3 - 7 x + 7, 3] == x^3 - 7 x + 7 /. x → 3
```

```
True
```

Προφανώς *για τον υπολογισμό της τιμής* $f(u)$ *χρειάζονται n πολλαπλασιασμοί και προσθέσεις*. Αν *δε υποθέσουμε πως* $f(x) \in \mathbb{Z}[x]$, *και λάβουμε υπ' όψη το μέγεθος των συντελεστών του* $f(x)$ *και του u, τότε στην χειρότερη περίπτωση απαιτούνται* $O(n^2 \lambda(u)\lambda(\|f\|_\infty)$ *αριθμητικές πράξεις* (*βλέπε και την βιβλιογραφία*).

Λόγω *της σημασίας της μεθόδου αυτής, αξίζει να δούμε μερικές ακόμη λεπτομέρειες*. Συγκεκριμένα, *ας πάρουμε το γενικό πολυώνυμο 3ου βαθμού* $f_0 + f_1 x + f_2 x^2 + f_3 x^3$, *και ας χρησιμοποιήσουμε την συνάρτηση* `FoldList[]` *για να δούμε τα* ***ενδιάμεσα*** *αποτελέσματα για την εκτίμηση του πολυωνύμου στο σημείο u*.

```
Drop[FoldList[(#1 u + #2) &, 0, {f3, f2, f1, f0}], 1]
```

```
{f3, f2 + u f3, f1 + u (f2 + u f3), f0 + u (f1 + u (f2 + u f3))}
```

Αν *τώρα γράψουμε σε μία σειρά τους συντελεστές* $\{f_3, f_2, f_1, f_0\}$ *του πολυωνύμου* (*συμπεριλαμβανομένων και των μηδενικών συντελεστών*) *και σε μία*

δεύτερη σειρά (από κάτω) γράψουμε τα ενδιάμεσα αποτελέσματα μαζί με την τιμή $f(u)$, τότε προκύπτει η εξής μέθοδος για τον υπολογισμό με το χέρι:

$$\{ f_3, \quad f_2, \quad f_1, \quad f_0 \}$$
$$\{a = f_3, \quad b = f_2 + ua, \quad c = f_1 + ub, \quad f(u) = f_0 + uc \}$$

Ο πρώτος όρος, a, της δεύτερης σειράς είναι ο κύριος συντελεστής του πολυωνύμου και απλά αντιγράφεται από την πρώτη σειρά. Ο δεύτερος όρος, b, υπολογίζεται πολλαπλασιάζοντας τον a επί u και προσθέτοντας τον f_2. Ο τρίτος όρος, c, υπολογίζεται πολλαπλασιάζοντας τον b επί u και προσθέτοντας τον f_1. Και τέλος, ο τελευταίος όρος, $f(u)$, υπολογίζεται πολλαπλασιάζοντας τον c επί u και προσθέτοντας τον f_0. Ακολουθεί η ερμηνεία των a, b, c ως συντελεστών ενός πολυωνύμου.

Με γνωστή την ισοδυναμία $f(x) \equiv f(u) \bmod (x - u)$ — δηλαδή η εκτίμηση του $f(x)$ σε ένα σημείο u είναι το ίδιο με τον υπολογισμό του υπολοίπου modulo $(x - u)$ — το παραπάνω σχήμα (με τα ενδιάμεσα αποτελέσματα) είναι γνωστό και σαν **συνθετική διαίρεση** (synthetic division algorithm). Δηλαδή, αν διαιρέσουμε το $f_3 x^3 + f_2 x^2 + f_1 x + f_0$ με το $(x - u)$, τότε το πηλίκο είναι το πολυώνυμο $ax^2 + bx + c$ και το υπόλοιπο $f(u)$. (Οι συντελεστές a, b, c υπολογίστηκαν μαζί με το υπόλοιπο.)

Έτσι για παράδειγμα για την εκτίμηση του πολυωνύμου $x^3 - 7x + 7$ στο σημείο $u = 3$ έχουμε:

$$\{f_3 = 1, \quad f_2 = 0, \quad f_1 = -7, \quad f_0 = 7 \}$$
$$\{ a = 1, \quad b = 3, \quad c = 2, \quad f(u) = 13 \}$$

ή απλά

$$\{1, \quad 0, \quad -7, \quad 7\}$$
$$\{1, \quad 3, \quad 2, \quad 13\}$$

όπου το πηλίκο είναι $x^2 + 3x + 2$ και το υπόλοιπο 13. Στο Mathematica η συνθετική διαίρεση προγραμματίζεται ως εξής (**προσοχή:** τα πολυώνυμα με συμβολικούς συντελεστές **πρέπει** να έχουν μεταβλητή x):

```
syntheticDivision[f_, u_] := Module[
  {newCoefs},
  newCoefs = FoldList[(#1 u + #2 ) &, 0, Reverse[CoefficientList[
      f, If[Length[Variables[f]] > 1, x, Variables[f]]]]];
  {Drop[Drop[newCoefs, 1], -1], Last[newCoefs]}
 ]
```

και για το παράδειγμά μας έχουμε:

```
f = x^3 - 7 x + 7; {quo, rem} = syntheticDivision[f, 3]
```

```
{{1, 3, 2}, 13}
```

Π*ροφανώς το πηλίκο παρίσταται από την λίστα των συντελεστών του*. Εύ*κολα όμως το μετατρέπουμε στην συνήθη μορφή πολυωνύμου ως εξής*:

```
n = Length[quo]; var = First[Variables[f]];
powers = Reverse[Table[var^(i-1), {i, n}]];
quo.powers
```

```
2 + 3 x + x^2
```

Εν *γένει*, *αν έχουμε μία λίστα συντελεστών ενός πολυωνύμου* (*με τον κύριο συντελεστή* ***πρώτο*** *στοιχείο της λίστας*) *και θέλουμε να της δώσουμε την συνήθη μορφή πολυωνύμου χρησιμοποιούμε το ακόλουθο πρόγραμμα*:

```
polyForm[coefList_, var_] := Module[{n, powers},
  n = Length[coefList]; powers = Reverse[Table[var^(i-1), {i, n}]];
  coefList.powers]
```

Ας *δούμε τώρα πως με την χρήση* ***συνθετικών διαιρέσεων*** *υπολογίζονται οι συντελεστές του πολυωνύμου* $f(x + u)$, *δηλαδή πως γίνεται η μετάθεση πολυωνύμων*. Η *πράξη αυτή είναι τόσο σημαντική ώστε πρέπει να μπορούμε να την κάνουμε με το χέρι*.

Γ*ια μία μερική απόδειξη για πολυώνυμα βαθμού τρία χρησιμοποιούμε το Mathematica*, *και αφήνουμε την γενίκευσή της για άσκηση*.

Έστω λοιπόν το γενικό πολυώνυμο τρίτου βαθμού $f(x) = f_0 + f_1 x + f_2 x^2 + f_3 x^3$

```
f[x_] = Sum[f_i x^i, {i, 0, 3}]
```

$$f_0 + x\, f_1 + x^2\, f_2 + x^3\, f_3$$

και έστω ότι θέλουμε να υπολογίσουμε τους συντελεστές του $f(x + u)$. Με το Mathematica εύκολα βλέπουμε πως οι ζητούμενοι συντελεστές είναι

```
f[x_ + u_] = Collect[Expand[Sum[f_i (x + u)^i, {i, 0, 3}]], x]
```

$$f_0 + u\, f_1 + u^2\, f_2 + u^3\, f_3 + x^3\, f_3 + x^2\, (f_2 + 3\, u\, f_3) + x\, (f_1 + 2\, u\, f_2 + 3\, u^2\, f_3)$$

αλλά το ερώτημα είναι πως προέκυψαν. Ας τους δούμε έναν-έναν.

Προσέξτε πως ο συντελεστής του x^0 στο πολυώνυμο $f(x + u)$ είναι

```
Coefficient[f[x + u], x, 0]
```

$$f_0 + u\, f_1 + u^2\, f_2 + u^3\, f_3$$

ο οποίος ισούται με το υπόλοιπο της συνθετικής διαίρεσης του $f(x)$ με το $(x - u)$

```
{quo, rem} = syntheticDivision[f[x], u];
Expand[rem]
```

$$f_0 + u\, f_1 + u^2\, f_2 + u^3\, f_3$$

Επειδή το (πρώτο) πηλίκο quo είναι σε μορφή λίστας συντελεστών

```
Expand[quo]
```

$$\{f_3,\ f_2 + u\, f_3,\ f_1 + u\, f_2 + u^2\, f_3\}$$

το μετετρέπουμε σε συνήθες πολυώνυμο

```
f[x_] = polyForm[Expand[quo], x]
```

$$f_1 + u\, f_2 + u^2\, f_3 + x^2\, f_3 + x\, (f_2 + u\, f_3)$$

Στην συνέχεια προσέξτε πως ο συντελεστής του x^1 στο πολυώνυμο $f(x + u)$ είναι:

```
Coefficient[f[x + u], x, 1]
```

$$f_1 + 2\, u\, f_2 + 3\, u^2\, f_3$$

ο οποίος ισούται με το υπόλοιπο της συνθετικής διαίρεσης του πρώτου πηλίκου — νέου $f(x)$ — με το $(x - u)$

```
{quo, rem} = syntheticDivision[f[x], u];
Expand[rem]
```

$$f_1 + 2\, u\, f_2 + 3\, u^2\, f_3$$

Επειδή πάλι το δεύτερο πηλίκο quo είναι σε μορφή λίστας συντελεστών

```
Expand[quo]
```

$$\{f_3,\ f_2 + 2\, u\, f_3\}$$

το μετετρέπουμε σε συνήθες πολυώνυμο

```
f[x_] = polyForm[Expand[quo], x]
```

$$f_2 + 2\, u\, f_3 + x\, f_3$$

Τέλος βλέπουμε πως ο συντελεστής του x^2 στο πολυώνυμο $f(x + u)$ είναι:

```
Coefficient[f[x + u], x, 2]
```

$$f_2 + 3\, u\, f_3$$

ο οποίος ισούται με το υπόλοιπο της συνθετικής διαίρεσης του δεύτερου πηλίκου — νέου $f(x)$ — με το $(x - u)$

```
{quo, rem} = syntheticDivision[f[x], u];
Expand[rem]
```

```
f2 + 3 u f3
```

Το τρίτο πηλίκο quo είναι σε μορφή λίστας συντελεστών

```
Expand[quo]
```

```
{f3}
```

αλλά αυτήν την φορά σταματάμε την διαδικασία, επειδή ισούται με τον κύριο συντελεστή του αρχικού πολυωνύμου $f(x)$ — ο οποίος είναι και ο κύριος συντελεστής του $f(x + u)$.

```
Coefficient[f[x + u], x, 3]
```

```
f3
```

Βλέπουμε λοιπόν πως η μετάθεση ενός πολυωνύμου επιτυγχάνεται με επανειλημμένη εφαρμογή του αλγορίθμου της συνθετικής διαίρεσης και στο Mathematica το πρόγραμμα είναι ως εξής (προσέξτε πως για εξοικονομία χρόνου δεν χρησιμοποιούμε την συνάρτηση `polyForm[]` που ορίσαμε παραπάνω):

```
polyTranslation[f_, u_] := Module[{n, powersPoly, quo, rem, temp = f,
   translatedCoefficients = {}, var = First[Variables[f]]},
  {quo, rem} = syntheticDivision[temp, u];
  n = Length[quo];
  powersPoly = Reverse[Table[var^(i-1), {i, n + 1}]];
  powers = Drop[powersPoly, 1];
  PrependTo[translatedCoefficients, Expand[rem]];
  While[n ≥ 2,
   temp = quo.powers; {quo, rem} = syntheticDivision[temp, u];
   PrependTo[translatedCoefficients, Expand[rem]];
   n = n – 1; powers = Drop[powers, 1]];
  PrependTo[translatedCoefficients, First[quo]];
  translatedCoefficients.powersPoly
 ]
```

Έτσι έχουμε για παράδειγμα

```
polyTranslation[f[x], u]
```

```
f0 + u f1 + u^2 f2 + u^3 f3 + x^3 f3 + x^2 (f2 + 3 u f3) + x (f1 + 2 u f2 + 3 u^2 f3)
```

ή *για* $f(x) = x^3 - 7x + 7$ *και* $u = 1$,

```
f[x_] = x^3 - 7 x + 7; polyTranslation[f[x], 1]
```

```
1 - 4 x + 3 x^2 + x^3
```

Για *το τελευταίο αυτό αριθμητικό παράδειγμα οι υπολογισμοί με το χέρι συγκεντρώνονται στους παρακάτω πίνακες*

Ruffini (1804)

$$\begin{array}{cccc} \{1, & 0, & -7, & 7\} \\ & \{1, & 1, & -6, \quad \mathbf{1}\} \\ & & \{1, & 2, \quad -\mathbf{4}\} \\ & & & \{1, \quad \mathbf{3}\} \\ & & & \{\mathbf{1}\} \end{array}$$

Horner (1819)

$$\begin{array}{cccc} \{1, & 0, & -7, & 7\} \\ \{1, & 1, & -6, & \mathbf{1}\} \\ \{1, & 2, & -\mathbf{4}\} & \\ & \{1, & \mathbf{3}\} & \\ & \{\mathbf{1}\} & & \end{array}$$

και οι συντελεστές του πολυωνύμου $f(x + 1)$ *διαβάζονται από κάτω προς τα πάνω* (ή *δεξιά-επάνω*). Αξίζει *να σημειωθεί πως στην περίπτωση που* $u = 1$ *δεν γίνονται καθόλου πολλαπλασιασμοί*!

Επειδή *η συνθετική διαίρεση εφαρμόζεται επανειλημμένα σε πολυώνυμα συνεχώς μειωμένου βαθμού, για τον υπολογισμό των συντελεστών του πολυωνύμου* $f(x + u)$, *χρειάζονται* $\sum_{i=1}^{n} i = \frac{n(n+1)}{2} = \mathrm{O}(n^2)$ *πολλαπλασιασμοί και προσθέσεις*. Αν *δε υποθέσουμε πως* $f(x) \in \mathbb{Z}[x]$, *και λάβουμε υπ' όψη το μέγεθος των συντελεστών του* $f(x)$ *και του* u, *τότε στην χειρότερη περίπτωση απαιτούνται* $O(n^3 \lambda(u)^2 \lambda(\|f\|_\infty)$ *αριθμητικές πράξεις* (*βλέπε και την βιβλιογραφία*).

4.2 Πίνακες Vandermonde

Οι πίνακες Vandermonde έχουν δύο ιδιότητες που τους κάνουν ιδιαίτερα χρήσιμους στους αλγόριθμους εκτίμησης και παρεμβολής: (**α**) εύκολα διαπιστώνεται αν η ορίζουσά τους δεν είναι μηδέν (δηλαδή αν ο πίνακας είναι non-singular), και (**β**) συστήματα γραμμικών εξισώσεων οι συντελεστές των οποίων σχηματίζουν πίνακα Vandermonde λύνονται εύκολα με αριθμητική απείρου ακριβείας.

Οι πίνακες Vandermonde μας δίνουν μια σφαιρική εικόνα της εκτίμησης και παρεμβολής. Για να δούμε την εικόνα αυτή έστω τα πολυώνυμα βαθμού **μικρότερου** του n, $f(x) = \sum_{i=0}^{n-1} f_i x^i \in F[x]$, F σώμα, και τα σημεία $u_0, u_1, \ldots, u_{n-1} \in F$. Τότε η απεικόνιση εκτίμησης $\varepsilon : F^n \to F^n$ ορίζεται ως:

$$\varepsilon(f_0, \ldots, f_{n-1}) = (\sum_{i=0}^{n-1} f_i u_0^i, \ldots, \sum_{i=0}^{n-1} f_i u_{n-1}^i)$$

και παρίσταται από τον **πίνακα Vandermonde**

$$V_n = \mathrm{Vdm}(u_0, \ldots, u_{n-1}) = \begin{pmatrix} 1 & u_0 & u_0^2 & \ldots & u_0^{n-1} \\ 1 & u_1 & u_1^2 & \ldots & u_1^{n-1} \\ 1 & u_2 & u_2^2 & \ldots & u_2^{n-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & u_{n-1} & u_{n-1}^2 & \ldots & u_{n-1}^{n-1} \end{pmatrix} \in F^{n \times n}.$$

Το ακόλουθο πρόγραμμα μας δίνει τον πίνακα Vandermonde τάξης $n \times n$

```
Vandermonde[u_List] := Module[
  {i, j, mat = {}, r, n = Length[u]},
  For[j = 0, j < n, j++, r = {};
   r = For[i = 0, i < n, i++, AppendTo[r, u[[j + 1]]^i]]
     AppendTo[mat, r]]; mat
 ]
```

Ακολουθούν ένα συμβολικό και ένα αριθμητικό παράδειγμα.

```
Vandermonde[{u₀, u₁, u₂}] // MatrixForm
```

$$\begin{pmatrix} 1 & u_0 & u_0^2 \\ 1 & u_1 & u_1^2 \\ 1 & u_2 & u_2^2 \end{pmatrix}$$

Στο συμβολικό παράδειγμα σημειώστε πως αν έχουν αποδοθεί τιμές στις μεταβλητές u_i αυτές θα πρέπει να καθαρίσθούν εκτελώντας την εντολή `For[i=0, i<n, i++, uᵢ=.]`

```
Vandermonde[{1, 2, 3}] // MatrixForm
```

$$\begin{pmatrix} 1 & 1 & 1 \\ 1 & 2 & 4 \\ 1 & 3 & 9 \end{pmatrix}$$

Η *ορίζουσα ενός πίνακα* Vandermonde *υπολογίζεται πολύ εύκολα ως εξής*: *Για κάθε* $0 \le i \le n - 2$, *πολλαπλασιάζουμε την* i*-στή στήλη του πίνακα με* u_0 *και αφαιρούμε το αποτέλεσμα από την* $(i + 1)$*-στή στήλη*. Οπότε, *η ορίζουσα του πίνακα*, $\mathrm{Det}(V_n)$, *ισούται με την ορίζουσα*:

$$\mathrm{Det}\begin{pmatrix} 1 & 0 & 0 & \dots & 0 \\ 1 & u_1 - u_0 & u_1^2 - u_0\,u_1 & \dots & u_1^{n-1} - u_0\,u_1^{n-2} \\ 1 & u_2 - u_0 & u_2^2 - u_0\,u_2 & \dots & u_2^{n-1} - u_0\,u_2^{n-2} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & u_{n-1} - u_0 & u_{n-1}^2 - u_0\,u_{n-1} & \dots & u_{n-1}^{n-1} - u_0\,u_{n-1}^{n-2} \end{pmatrix}$$

πράγμα που σημαίνει πως αν αναπτύξουμε κατά μήκος της πρώτης σειράς — και βγάλουμε τον κοινό παράγοντα $(u_i - u_0)$ *από κάθε σειρα — έχουμε*:

$$\begin{aligned} \mathrm{Det}(V_n) &= \mathrm{Det}(V_{n-1}) \prod_{i=1}^{n-1} (u_i - u_0) \\ &= \mathrm{Det}(V_{n-2}) \prod_{i=2}^{n-1} (u_i - u_1) \prod_{i=1}^{n-1} (u_i - u_0) \\ &= \dots \\ &= \prod_{i>j=0}^{n-1} (u_i - u_j). \end{aligned}$$

Προφανώς, *αν* $u_i = u_j$ *για κάποιο* $i \ne j$, *τότε οι σειρές* i *και* j *είναι ίσες και η ορίζουσα είναι μηδέν* (*ο πίνακας είναι* singular). Αντίθετα, *αν* $u_i \ne u_j$ *για όλα τα* i,

j τότε ο πίνακας (*και επομένως η απεικόνιση ε*) *αντιστρέφεται και* V_n^{-1} *είναι ο πίνακας της απεικόνισης παρεμβολής*.

Γ*ια* $n = 3$ *η ορίζουσα του πίνακα* Vandermonde, *όπως μας το επιβεβαιώνει και το Mathematica*, *είναι* $(u_1 - u_0)(u_2 - u_0)(u_2 - u_1)$.

```
Det[Vandermonde[{u₀, u₁, u₂}]] // Factor
```

```
-(u₀ - u₁) (u₀ - u₂) (u₁ - u₂)
```

Βλέπουμε *λοιπόν καθαρά πως η ορίζουσα ενός πίνακα* Vandermonde *είναι μη μηδενική εάν και μόνον εάν* $u_i \neq u_j$ *για όλα τα* i, j.

Ο *αντίστροφος ενός πίνακα* Vandermonde V_n *μπορεί να υπολογισθεί με το ακόλουθο "τρικ" με* $O(n^2)$ *πράξεις στο σώμα F*: Πολλαπλασιάζουμε *τον πίνακα* V_n *με έναν γενικό πίνακα A διαστάσεων* $n \times n$.

$$\begin{pmatrix} 1 & u_0 & \dots & u_0^{n-1} \\ 1 & u_1 & \dots & u_1^{n-1} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & u_{n-1} & \dots & u_{n-1}^{n-1} \end{pmatrix} \begin{pmatrix} a_{01} & a_{02} & \dots & a_{0,n-1} \\ a_{11} & a_{12} & \dots & a_{1,n-1} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n-1,1} & a_{n-1,2} & \dots & a_{n-1,n-1} \end{pmatrix}$$

Τ*α στοιχεία* a_{ij} *πίνακα A επιλέγονται έτσι ώστε το παραπάνω γινόμενο να είναι ο ταυτοτικός πίνακας*. Γ*ια να δούμε πως θα επιλέξουμε τα στοιχεία* a_{ij} *του πίνακα A*, *παρατηρούμε πως τα στοιχεία της πρώτης γραμμής του γινομένου είναι εκτιμήσεις πολυωνύμων βαθμού* n - 1 *της ακόλουθης μορφής* (*το γράμμα L είναι για τον* Lagrange):

$$a_{0\,j} + u_0 a_{1\,j} + u_0^2 a_{2\,j} + \cdots + u_0^{n-1} a_{n-1,j} = L_j(u_0), \quad 0 \le j \le n - 1.$$

Ε*πομένως*, *το γινόμενο των δύο παραπάνω πινάκων θα είναι της μορφής*

$$\begin{pmatrix} L_0(u_0) & L_1(u_0) & \dots & L_{n-1}(u_0) \\ L_0(u_1) & L_1(u_1) & \dots & L_{n-1}(u_1) \\ \vdots & \vdots & \ddots & \vdots \\ L_0(u_{n-1}) & L_1(u_{n-1}) & \dots & L_{n-1}(u_{n-1}) \end{pmatrix},$$

και θέλουμε να ισούται με τον ταυτοτικό πίνακα — για να είναι ο πίνακας A ο αντίστροφος του V_n. Πρέπει λοιπόν να είναι $L_j(u_i) = 0$ για $i \neq j$ και $L_i(u_i) = 1$. Για να πετύχουμε το ζητούμενο, θέτουμε

$$L_j(x) = \prod_{i \neq j,\, 0 \leq i \leq n-1} \frac{x - u_i}{u_j - u_i},$$

και εξασφαλίζουμε δύο πράγματα: (α) από τον παρανομαστή ότι τα διαγώνια στοιχεία είναι 1 και (β) από τον αριθμητή ότι τα μη διαγώνια στοιχεία είναι 0.

Έχοντας προσδιορίσει τα **βαθμού** n - 1 πολυώνυμα $L_j(x)$, $0 \leq j \leq n$ - 1, βρίσκουμε και τα στοιχεία a_{ij}, $0 \leq i \leq n$ - 1, (της **στήλης** j) του πίνακα A από τους συντελεστές των αντίστοιχων δυνάμεων του x. Δηλαδή έχουμε

$$A = V_n^{-1} = \{\text{στήλη}_0, \text{στήλη}_1, \ldots, \text{στήλη}_{n-1}\}$$

όπου για $0 \leq j \leq n$ - 1,

$$\begin{aligned} \text{στήλη}_j &= \text{συντελεστές του} \prod_{i \neq j,\, 0 \leq i \leq n-1} \frac{x - u_i}{u_j - u_i} \\ &= \text{coef}(L_j(x)). \end{aligned}$$

Τα πολυώνυμα $L_j(x)$, $0 \leq j \leq n$ - 1, προσδιορίζονται με την βοήθεια του γενικού πολυωνύμου

$$L(x) = \prod_{i=0}^{n-1} x - u_i$$

βαθμού n. Το ανάπτυγμα του πολυωνύμου αυτού υπολογίζεται με $O(n^2)$ αριθμητικές πράξεις στο σώμα F και ο συντελεστής του όρου x^n είναι η μονάδα.

Έχοντας ορίσει το γενικό πολυώνυμο $L(x)$ κάθε ένα από τα πολυώνυμα $L_j(x)$, $0 \leq j \leq n$ - 1, υπολογίζονται με $O(n)$ αριθμητικές πράξεις ως εξής: Ο **αριθμητής** του $L_j(x)$ είναι το πηλίκο $\frac{L(x)}{x - u_j}$ που υπολογίζεται με $O(n)$ πράξεις στο σώμα F — επειδή ο διαιρέτης είναι πρώτου βαθμού. Ο δε **παρανομαστής** του $L_j(x)$ είναι η τιμή του αριθμητή $\frac{L(x)}{x - u_j}$ στο σημείο $x = u_j$, πράξη που επίσης υπολογίζεται με $O(n)$ πράξεις στο σώμα F.

Παράδειγμα:

Για να υπολογίσουμε τον 3×3 πίνακα A (τον αντίστροφο του πίνακα Vandermonde V_3 που ορίσαμε παραπάνω) πρέπει να ορίσουμε τα βαθμού 2 πολυώνυμα

$$L_j(x) = \prod_{i\neq j,\, 0\le i\le 2} \frac{x-u_i}{u_j-u_i}, \quad 0 \le j \le 2.$$

Για τον σκοπό αυτό ορίζουμε πρώτα το γενικό πολυώνυμο $L(x) = \prod_{i=0}^{2} x-u_i$, βαθμού 3

```
n = 3; L[x_] := Product[(x - u_i), {i, 0, n-1}]; L[x]
```

$$(x-u_0)\,(x-u_1)\,(x-u_2)$$

Όπως αναφέραμε, από το $L(x)$ υπολογίζουμε κάθε ένα από τα (βαθμού 2) πολυώνυμα $L_j(x)$, $0 \le j \le 2$, ως εξής: Ο αριθμητής του πολυωνύμου $L_j(x)$ είναι $\frac{L(x)}{x-u_j}$ ενώ ο παρανομαστής του είναι η τιμή του αριθμητή στο σημείο $x = u_j$.

Για το πολυώνυμο $L_0(x)$ έχουμε:

```
num = L[x]/(x - u0); denom = num /. x → u0; L0 = num/denom
```

$$\frac{(x-u_1)\,(x-u_2)}{(u_0-u_1)\,(u_0-u_2)}$$

και οι **συντελεστές** του (που αποτελούν και την **πρώτη στήλη** του αντίστροφου πίνακα A) είναι

```
row0 = CoefficientList[L0, x] // Simplify
```

$$\left\{\frac{u_1\,u_2}{(u_0-u_1)\,(u_0-u_2)},\ -\frac{u_1+u_2}{(u_0-u_1)\,(u_0-u_2)},\ \frac{1}{(u_0-u_1)\,(u_0-u_2)}\right\}$$

Για το πολυώνυμο $L_1(x)$ και την **δεύτερη στήλη** του A έχουμε:

```
num = L[x]/(x - u1); denom = num /. x → u1; L1 = num/denom;
row1 = CoefficientList[L1, x] // Simplify
```

$$\left\{\frac{u_0\,u_2}{(-u_0+u_1)\,(u_1-u_2)},\ \frac{u_0+u_2}{(u_0-u_1)\,(u_1-u_2)},\ \frac{1}{(-u_0+u_1)\,(u_1-u_2)}\right\}$$

Τέλος, *για το πολυώνυμο* $L_2(x)$ *και την* ***τρίτη στήλη*** *του* A *έχουμε*:

```
num = L[x]/(x - u2); denom = num /. x → u2; L2 = num/denom;
row2 = CoefficientList[L2, x] // Simplify
```

$$\left\{\frac{u_0\,u_1}{(u_0-u_2)\,(u_1-u_2)},\ -\frac{u_0+u_1}{(u_0-u_2)\,(u_1-u_2)},\ \frac{1}{(u_0-u_2)\,(u_1-u_2)}\right\}$$

Ετσι *έχουμε τον πίνακα* A

```
Transpose[{row0, row1, row2}] // MatrixForm
```

$$\begin{pmatrix} \frac{u_1\,u_2}{(u_0-u_1)\,(u_0-u_2)} & \frac{u_0\,u_2}{(-u_0+u_1)\,(u_1-u_2)} & \frac{u_0\,u_1}{(u_0-u_2)\,(u_1-u_2)} \\ -\frac{u_1+u_2}{(u_0-u_1)\,(u_0-u_2)} & \frac{u_0+u_2}{(u_0-u_1)\,(u_1-u_2)} & -\frac{u_0+u_1}{(u_0-u_2)\,(u_1-u_2)} \\ \frac{1}{(u_0-u_1)\,(u_0-u_2)} & \frac{1}{(-u_0+u_1)\,(u_1-u_2)} & \frac{1}{(u_0-u_2)\,(u_1-u_2)} \end{pmatrix}$$

και το αποτέλεσμα είναι το ίδιο με αυτό που υπολογίζουμε με το Mathematica

```
Inverse[Vandermonde[{u0, u1, u2}]] // Simplify // MatrixForm
```

$$\begin{pmatrix} \frac{u_1\,u_2}{(u_0-u_1)\,(u_0-u_2)} & -\frac{u_0\,u_2}{(u_0-u_1)\,(u_1-u_2)} & \frac{u_0\,u_1}{(u_0-u_2)\,(u_1-u_2)} \\ -\frac{u_1+u_2}{(u_0-u_1)\,(u_0-u_2)} & \frac{u_0+u_2}{(u_0-u_1)\,(u_1-u_2)} & -\frac{u_0+u_1}{(u_0-u_2)\,(u_1-u_2)} \\ \frac{1}{(u_0-u_1)\,(u_0-u_2)} & -\frac{1}{(u_0-u_1)\,(u_1-u_2)} & \frac{1}{(u_0-u_2)\,(u_1-u_2)} \end{pmatrix}$$

Το *ακόλουθο πρόγραμμα στο Mathematica υπολογίζει τον αντίστροφο ενός πίνακα* Vandermonde *διαστάσεων* $n \times n$ *σε χρόνο* $O(n^2)$.

```
inverseVandermonde[u_List] :=
 Module[{denom, i, mat, n = Length[u], num, L, poly, r, x},
  L[x_] := Product[(x - u[[i + 1]]), {i, 0, n - 1}]; mat = {}; For[i = 0, i < n, i++,
   num = L[x]/(x - u[[i + 1]]); denom = num /. x → u[[i + 1]]; poly = num/denom;
  r = CoefficientList[poly, x] // Expand // Simplify; AppendTo[mat, r]];
  Transpose[mat]]
```

Πράγματι έχουμε:

```
inverseVandermonde[{u0, u1, u2}] // MatrixForm
```

$$\begin{pmatrix} \frac{u_1 u_2}{(u_0-u_1)(u_0-u_2)} & \frac{u_0 u_2}{(-u_0+u_1)(u_1-u_2)} & \frac{u_0 u_1}{(u_0-u_2)(u_1-u_2)} \\ -\frac{u_1+u_2}{(u_0-u_1)(u_0-u_2)} & \frac{u_0+u_2}{(u_0-u_1)(u_1-u_2)} & -\frac{u_0+u_1}{(u_0-u_2)(u_1-u_2)} \\ \frac{1}{(u_0-u_1)(u_0-u_2)} & \frac{1}{(-u_0+u_1)(u_1-u_2)} & \frac{1}{(u_0-u_2)(u_1-u_2)} \end{pmatrix}$$

Και τελειώνουμε με ένα αριθμητικό παράδειγμα,

```
inverseVandermonde[{1, 2, 3}] // MatrixForm
```

$$\begin{pmatrix} 3 & -3 & 1 \\ -\frac{5}{2} & 4 & -\frac{3}{2} \\ \frac{1}{2} & -1 & \frac{1}{2} \end{pmatrix}$$

όπου το αποτέλεσμα συμφωνεί με αυτό που υπολογίζει το Mathematica.

```
Inverse[Vandermonde[{1, 2, 3}]] // MatrixForm
```

$$\begin{pmatrix} 3 & -3 & 1 \\ -\frac{5}{2} & 4 & -\frac{3}{2} \\ \frac{1}{2} & -1 & \frac{1}{2} \end{pmatrix}$$

4.3 Παρεμβολή Lagrange

Στην απλούστερη μορφή του το πρόβλημα της πολυωνυμικής παρεμβολής (polynomial interpolation) μας ζητά να βρούμε ένα πολυώνυμο μιας μεταβλητής από τις τιμές του σε ωρισμένα σημεία. Στην ενότητα αυτή θα δούμε πως στο πρόβλημα αυτό υπάρχει πάντα μία μοναδική λύση και θα μάθουμε να την υπολογίζουμε.

Έστω λοιπόν οτι θέλουμε να υπολογίσουμε το πολυώνυμο $f(x) \in F[x]$, F σώμα, βαθμού $< n$

$$f(x) = f_0 + f_1 x + f_2 x^2 + \cdots + f_{n-1} x^{n-1}$$

για το οποίο ξέρουμε τις τιμές του $w_i \in F$ στα σημεία $u_i \in F$, δηλαδή $f(u_i) = w_i$, $0 \le i \le n - 1$. Από τις σχέσεις $f(u_i) = w_i$ προκύπτει το εξής σύστημα γραμμικών εξισώσεων με αγνώστους τους n συντελεστές f_i:

$$\begin{array}{ccccccc} f_0 + f_1 u_0 & + & f_2 u_0^2 & + \cdots + & f_{n-1} u_0^{n-1} & = & w_0 \\ f_0 + f_1 u_1 & + & f_2 u_1^2 & + \cdots + & f_{n-1} u_1^{n-1} & = & w_1 \\ \vdots & & \vdots & & \vdots & & \\ f_0 + f_1 u_{n-1} & + & f_2 u_{n-1}^2 & + \cdots + & f_{n-1} u_{n-1}^{n-1} & = & w_{n-1} \end{array}$$

Στο σύστημα αυτό διακρίνουμε τον πίνακα Vandermonde $V_n = \mathrm{Vdm}(u_0, u_1, \ldots, u_{n-1})$ και το γράφουμε:

$$V_n . \{f_0, f_1, \ldots, f_{n-1}\} = \{w_0, w_1, \ldots, w_{n-1}\}$$

Από την τελευταία σχέση βλέπουμε πως οι συντελεστές του πολυωνύμου δίνονται από την σχέση

$$\{f_0, f_1, \ldots, f_{n-1}\} = V_n^{-1} . \{w_0, w_1, \ldots, w_{n-1}\}$$

η οποία υπολογίζεται με $O(n^2)$ πράξεις στο σώμα F — πράξεις απαιτούμενες για τον υπολογισμό του πίνακα V_n^{-1}.

Όπως όμως έχουμε δει στην προηγούμενη ενότητα,

$$V_n^{-1} = \{\text{στήλη}_0, \text{στήλη}_1, \ldots, \text{στήλη}_{n-1}\}$$
$$= \{\text{coef}(L_0(x)), \text{coef}(L_1(x)), \ldots, \text{coef}(L_{n-1}(x))\}$$

οπότε οι συντελεστές του πολυωνύμου είναι:

$\{f_0, f_1, \ldots, f_{n-1}\}$

$$= \{\text{στήλη}_0, \text{στήλη}_1, \ldots, \text{στήλη}_{n-1}\}.\{w_0, w_1, \ldots, w_{n-1}\}$$

$$= \text{στήλη}_0 w_0 + \text{στήλη}_1 w_1 + \cdots + \text{στήλη}_{n-1} w_{n-1}$$

$$= \text{coef}(L_0(x))w_0 + \text{coef}(L_1(x))w_1 + \cdots + \text{coef}(L_{n-1}(x))w_{n-1}$$

Αν δε συμβολίσουμε με $\text{coef}(L_j(x),\ x^i)$ τον συντελεστή του x^i στο πολυώνυμο

$$L_j(x) = \prod_{i \neq j,\, 0 \le i \le n-1} \frac{x - u_i}{u_j - u_i},$$

τότε έχουμε για $0 \le j \le n - 1$

$$f_j = \text{coef}(L_0(x),\ x^j)w_0 + \text{coef}(L_1(x),\ x^j)w_1 + \cdots + \text{coef}(L_{n-1}(x),\ x^j)w_{n-1}$$

Αντικαθιστώντες τους παραπάνω τύπους των συντελεστών στο πολυώνυμο

$$f(x) = f_0 + f_1 x + f_2 x^2 + \cdots + f_{n-1} x^{n-1}$$

και χρησημοποιώντας ολόκληρα τα πολυώνυμα $L_j(x)$ — αντί για τους συντελεστές δυνάμεων του x — προκύπτει

$$f(x) = \sum_{j=0}^{n-1} L_j(x)\, w_j = L_0(x)w_0 + L_1(x)w_1 + \cdots + L_{n-1}(x)w_{n-1}$$

ή επειδή

$$f(x) = \sum_{j=0}^{n-1} \left(\prod_{i \neq j,\, i=0}^{n-1} \frac{x - u_i}{u_j - u_i} \right) w_j .$$

Αυτός *είναι ο* ***τύπος παρεμβολής*** *του* Lagrange (Lagrange interpolation formula) *με την βοήθεια του οποίου υπολογίζουμε το πολυώνυμο* $f(x)$ *— με την βοήθεια των πολυώνυμων* $L_j(x)$ *και των τιμών* $w_i = f(u_i) \in F$ *στα σημεία* $u_i \in F$, $0 \le i \le n - 1$.

Το *ακόλουθο πρόγραμμα στο Mathematica υλοποιεί τον τύπο παρεμβολής του* Lagrange *κατά τον απλούστερο τρόπο*. Όπως *αναφέραμε το πολυώνυμο υπολογίζεται με* $O(n^2)$ *πράξεις στο σώμα* F.

```
LagrangeInterpolation[w_List, u_List, x_] /;
  Length[w] == Length[u] := Module[
  {i, n = Length[u]},
  (inverseVandermonde[u].w).Table[x^i, {i, 0, n - 1}]
 ]
```

Έτσι *για παράδειγμα*, *αν έχουμε τα σημεία* $\{u_0, u_1, u_2\}$ *και τις τιμές* $\{w_0, w_1, w_2\}$ *του πολυωνύμου σε αυτά*, *βλέπουμε πως το πολυώνυμο είναι βαθμού* 2 (*για να εκτλεσθεί η συνάρτηση* `LagrangeInterpolation[]` *πρέπει να έχουμε ενεργοποιήσει την συνάρτηση* `inverseVandermonde[]` *από την προηγούμενη ενότητα*)

```
f[x_] = LagrangeInterpolation[{w0, w1, w2}, {u0, u1, u2}, x];
Exponent[f[x], x]
```

```
2
```

και, *πράγματι*, *οι τιμές του στα σημεία* u_i, $0 \le i \le 3$, *είναι* w_i.

```
Table[f[ui], {i, 0, 2}] // Simplify
```

```
{w0, w1, w2}
```

Παράδειγμα 1 (*συμβολικό*):

Έστω *ότι θέλουμε να υπολογίσουμε τους συντελεστές* $\{f_0, f_1, f_2\}$ *του πολυωνύμου* $f(x)$ *έτσι ώστε* $f(u_i) = w_i$, $0 \le i \le 2$, $u_0, u_1, u_2 \in F$. *Θα δώσουμε λύση με δύο διαφορετικούς τρόπους*.

Χρησημοποιώντας την συνάρτηση `LagrangeInterpolation[]` που ορίσαμε έχουμε

```
f[x_] = LagrangeInterpolation[{w0, w1, w2}, {u0, u1, u2}, x];
```

και οι συντελεστές του f(x) είναι

```
CoefficientList[f[x], x] // Together
```

$$\left\{\frac{u_1^2 u_2 w_0 - u_1 u_2^2 w_0 - u_0^2 u_2 w_1 + u_0 u_2^2 w_1 + u_0^2 u_1 w_2 - u_0 u_1^2 w_2}{(u_0 - u_1)(u_0 - u_2)(u_1 - u_2)},\ \frac{-u_1^2 w_0 + u_2^2 w_0 + u_0^2 w_1 - u_2^2 w_1 - u_0^2 w_2 + u_1^2 w_2}{(u_0 - u_1)(u_0 - u_2)(u_1 - u_2)},\ \frac{u_1 w_0 - u_2 w_0 - u_0 w_1 + u_2 w_1 + u_0 w_2 - u_1 w_2}{(u_0 - u_1)(u_0 - u_2)(u_1 - u_2)}\right\}$$

Οι ίδιοι συντελεστές προκύπτουν χρησημοποιώντας την συνάρτηση `Solve[]` του *Mathematica* για να λύσουμε το σύστημα $V_n.\{f_0, f_1, f_2\} = \{w_0, w_1, w_2\}$ (για να εκτλεσθεί η συνάρτηση `Solve[]` πρέπει να έχουμε ενεργοποιήσει την συνάρτηση `Vandermonde[]` από την προηγούμενη ενότητα)

```
Solve[Vandermonde[{u0, u1, u2}].{f0, f1, f2} == {w0, w1, w2},
  {f0, f1, f2}] // Simplify
```

$$\left\{\left\{f_0 \to \frac{u_0 u_2 (-u_0 + u_2) w_1 + u_1^2 (u_2 w_0 - u_0 w_2) + u_1 (-u_2^2 w_0 + u_0^2 w_2)}{(u_0 - u_1)(u_0 - u_2)(u_1 - u_2)},\ f_1 \to \frac{u_2^2 (w_0 - w_1) + u_0^2 (w_1 - w_2) + u_1^2 (-w_0 + w_2)}{(u_0 - u_1)(u_0 - u_2)(u_1 - u_2)},\ f_2 \to \frac{u_2 (-w_0 + w_1) + u_1 (w_0 - w_2) + u_0 (-w_1 + w_2)}{(u_0 - u_1)(u_0 - u_2)(u_1 - u_2)}\right\}\right\}$$

Εξετάζοντας προσεκτικά τους συντελεστές του πολυώνυμου $f(x)$ ως προς w_i, όπου $w_i = f(u_i),\ 0 \le i \le 2$, βλέπουμε πως ισχύουν τα όσα είπαμε παραπάνω:

```
Coefficient[f[x], w0] // Simplify
```

$$\frac{(x - u_1)(x - u_2)}{(u_0 - u_1)(u_0 - u_2)}$$

```
Coefficient[f[x], w1] // Simplify
```

$$-\frac{(x-u_0)\,(x-u_2)}{(u_0-u_1)\,(u_1-u_2)}$$

```
Coefficient[f[x], w2] // Simplify
```

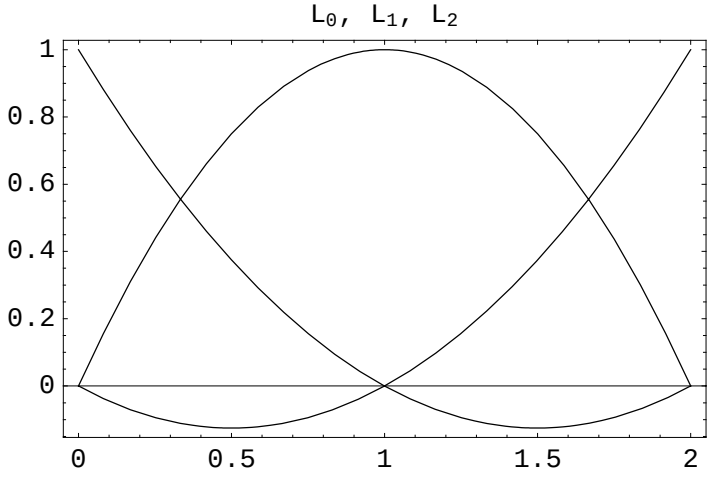
$$\frac{(x-u_0)\,(x-u_1)}{(u_0-u_2)\,(u_1-u_2)}$$

Δηλαδή *ισχύει ο τύπος παρεμβολής του* Lagrange

$$f(x)=\sum_{j=0}^{2}\left(\prod_{i\neq j,\,i=0}^{2}\frac{x-u_i}{u_j-u_i}\right)w_j.$$

Ας *δούμε τώρα τις γραφικές παραστάσεις των πολυωνύμων* $L_j(x)$, *για τα σημεία* $u_i = i$.

```
n = 3; L[x_] = ∏_{i=0}^{n-1} (x - u_i); L[x];
For[i = 0, i < 3, i++, u_i = i];
For[i = 0, i < 3, i++, num = L[x]/(x - u_i);
 denom = num /. x → u_i; L_i = num/denom]; Clear[i];
Plot[{L0, L1, L2}, {x, 0, 2}, Frame → True, PlotLabel → "L0, L1, L2"];
```



Αμέσως *μετά την χρήση τους καθαρίζουμε τα σημεία* u_i, *και τα πολυώνυμα* L_i, $0 \le i \le 2$.

```
For[i = 0, i < 3, i++, u_i =.]; For[i = 0, i < 3, i++, L_i =.]; Clear[i];
```

Παράδειγμα 2 (αριθμητικό):

Ας *δούμε τώρα μία εφαρμογή του τύπου παρεμβολής του* Lagrange *σε ένα αριθμητικό παράδειγμα*. *Έστω λοιπόν ότι θέλουμε να βρούμε το πολυώνυμο* $f(x)$ *έτσι ώστε* $f(u_i) = w_i$, $0 \le i \le 3$, *όπου* $u_0 = 1$, $u_1 = 2$, $u_2 = 3$ *και* $u_3 = 4$, *ενώ* $w_0 = 5$, $w_1 = -3$, $w_2 = 6$ *και* $w_3 = -4$. Η *απάντηση δίνεται άμεσα*

```
f[x_] = LagrangeInterpolation[{5, -3, 6, -4}, {1, 2, 3, 4}, x]
```

$$66 - \frac{199\,x}{2} + \frac{89\,x^2}{2} - 6\,x^3$$

και ισχύουν οι συνθήκες του προβλήματος

```
Table[f[x], {x, 1, 4}]
```

```
{5, -3, 6, -4}
```

■ 4.3.1 Εφαρμογές στην διατήρηση κοινού μυστικού

Έστω *ότι θέλουμε να δώσουμε σε* n *άτομα ένα κοινό μυστικό* (*π.χ. τον κωδικό αριθμό ενός τραπεζικού λογαριασμού*) *έτσι ώστε* **όλοι** *μαζί να μπορούν να το βρουν*, *αλλά κανένα γνήσιο υποσύνολο των* n *ατόμων να μην είναι σε θέση να το βρει*.

Για *να επιτύχουμε τον σκοπό μας ταυτίζουμε τα πιθανά μυστικά με στοιχεία του πεπερασμένου σώματος* $\mathbb{F}_p = \mathbb{Z}/\langle p\rangle$ *για κάποιο κατάλληλο* p. Αν *λοιπόν ο κωδικός του τραπεζικού λογαριασμού* (PIN) *είναι τετραψήφιος διαλέγουμε έναν πρώτο αριθμό* $p > 10000$, *έστω* 10099.

Κατόπιν *διαλέγουμε* $2n - 1$ *τυχαίους αριθμούς* $f_1, f_2, \ldots, f_{n-1}, u_0, u_1, \ldots, u_{n-1}$ *όπου τα* u_i *είναι όλα διάφορα του μηδενός και μεταξύ τους*, (*δηλαδή* $u_i \neq u_j$ *για* $i \neq j$), *θέτουμε* f_0 = *κοινό μυστικό*, *ορίζουμε το πολυώνυμο*

$$f(x) = f_0 + f_1 x + f_2 x^2 + \cdots + f_{n-1}\,x^{n-1} \in \mathbb{F}_p[x]$$

και δίνουμε στο άτομο i την τιμή $w_i = f(u_i) \in \mathbb{F}_p$, $0 \le i \le n - 1$.

Αν τώρα βρεθούν **όλοι** μαζί θα μπορέσουν να υπολογίσουν το πολυώνυμο $f(x)$ από τον τύπο παρεμβολής του Lagrange και να βρουν τον σταθερό όρο f_0. Αν όμως βρεθούν **λιγότεροι** από $n - 1$ τότε δεν θα έχουν καμία ένδειξη για τον σταθερό όρο και όλοι οι αριθμοί του $\mathbb{F}_p$ είναι το ίδιο πιθανοί.

Παράδειγμα:
Έστω ότι ο κωδικός του τραπεζικού λογαριασμού (PIN) είναι 5665 (το κοινό μυστικό), και θέλουμε να τον μοιράσουμε σε 5 άτομα. Ορίζουμε τους 9 τυχαίους αριθμούς στο διάστημα από 0 μέχρι 10000

```
Table[u_i = Random[Integer, {0, 10000}], {i, 0, 4}]
```

```
{9559, 3575, 3213, 5462, 5968}
```

```
Table[f_i = Random[Integer, {0, 10000}], {i, 1, 4}]
```

```
{6538, 4746, 200, 1193}
```

και τον δέκατο

```
f_0 = 5665;
```

Κατόπιν ορίζουμε το πολυώνυμο 4ου βαθμού από τους συντελεστές του f_i, $0 \le i \le 4$.

```
f[x_] = Table[f_i, {i, 0, 4}].Table[x^i, {i, 0, 4}]
```

```
5665 + 6538 x + 4746 x^2 + 200 x^3 + 1193 x^4
```

και υπολογίζουμε τις τιμές $w_i = f(u_i)$ στο σώμα $\mathbb{F}_{10099}$, τις οποίες μοιράζουμε στα 5 άτομα

```
Table[w_i = Mod[f[u_i], 10099], {i, 0, 4}]
```

```
{7770, 4249, 85, 8717, 5702}
```

Π*ροσέξτε πως μόνον όταν βρεθούν και τα* 5 *άτομα και χρησιμοποιήσουν τις τιμές* w_i, $0 \le i \le 4$, *μπορούν να βρούν τον κωδικό* (5665)

```
PolynomialMod[LagrangeInterpolation[
  {w0, w1, w2, w3, w4}, {u0, u1, u2, u3, u4}, x], 10099]
```

$$5665 + 6538\,x + 4746\,x^2 + 200\,x^3 + 1193\,x^4$$

Α*ν βρεθούν* 4 ή *λιγότερα άτομα θα αποτύχουν*

```
PolynomialMod[
 LagrangeInterpolation[{w1, w2, w3, w4}, {u1, u2, u3, u4}, x], 10099]
```

$$7131 + 6315\,x + 8461\,x^2 + 1226\,x^3$$

4.4 Κινεζικός αλγόριθμος υπολοίπων

Όπως αναφέραμε και στην εισαγωγή του Κεφαλαίου αυτού, ο **Κινεζικός αλγόριθμος υπολοίπων** (Chinese remainder algorithm, CRA) είναι για τους ακεραίους ότι είναι η παρεμβολή Lagrange για τα πολυώνυμα. Δηλαδή ο Κινεζικός αλγόριθμος υπολοίπων παρεμβάλει έναν ακέραιο από τα υπόλοιπα (τις "τιμές" w_i) που προκύπτουν αν διαιρεθεί ο ακέραιος με διάφορους αριθμούς m_i (τα "σημεία" u_i) που είναι πρώτοι μεταξύ τους, δηλαδή $\gcd(m_i, m_j) = 1$ για $i \neq j$. Η τελευταία συνθήκη ικανοποιείται άμεσα αν οι αριθμοί m_i είναι πρώτοι.

Στο *Mathematica* η συνάρτηση `ChineseRemainder[]` ισχύει μόνο για ακεραίους και βρίσκεται στο πακέτο

```
<< NumberTheory`NumberTheoryFunctions`
```

το οποίο πρέπει πρώτα να ενεργοποιήσουμε. Αν τώρα για παράδειγμα ξέρουμε πως ένας ακέραιος δίνει υπόλοιπα $w_0 = 1$, $w_1 = 2$ και $w_2 = 5$ όταν διαιρεθεί με τους πρώτους $m_0 = 2$, $m_1 = 5$ και $m_2 = 7$ αντίστοιχα, τότε ο **μοναδικός** ακέραιος στο διάστημα από 0 μέχρι $70 = 2 \cdot 5 \cdot 7$ που ικανοποιεί τις συνθήκες αυτές είναι ο 47

```
ChineseRemainder[{1, 2, 5}, {2, 5, 7}]
47
```

Πράγματι βλέπουμε πως

```
Mod[47, {2, 5, 7}]
{1, 2, 5}
```

Ας δούμε λεπτομερώς πως λύνεται το παράδειγμα αυτό που γράφεται και σαν

$$x \equiv 1 \bmod 2,$$
$$x \equiv 2 \bmod 5,$$
$$x \equiv 5 \bmod 7.$$

Η *πρώτη ισοδυναμία ικανοποιείται από τις τιμές* $x = 1 + 2i$, $i \in \mathbb{Z}$. Γι*α να υπολογίσουμε το* i *αντικαθιστούμε την τιμή του* x *στην δεύτερη ισοδυναμία που γίνεται* $1 + 2i \equiv 2 \bmod 5$ ή $2i \equiv (2 - 1) \bmod 5 \equiv 1 \bmod 5$. Ε*φαρμόζουμε τώρα τον επεκταμένο* Ευκλείδειο *αλγόριθμο* (ΕΕΑ) *στους ακεραίους* 2 *και* 5 *και βρίσκουμε τον πολλαπλασιαστικό αντίστροφο του* 2 mod 5, *που είναι* 3. Ε*πομένως έχουμε* $3 \cdot 2i \equiv 3 \bmod 5$ ή $i \equiv 3 \bmod 5$ ή $i = 3 + 5j$, $j \in \mathbb{Z}$.

Άρα η λύση x *των δύο πρώτων ισοδυναμιών είναι* $x = 1 + 2i = 1 + 2(3 + 5j) = 7 + 2 \cdot 5j$, ή $x \equiv 7 \bmod 2 \cdot 5$ *και μένει να λύσουμε το ακόλουθο σύστημα* 2 *ισοδυναμιών*:

$$x \equiv 7 \bmod 2 \cdot 5,$$
$$x \equiv 5 \bmod 7 \ .$$

Όπ*ως και προηγούμενα*, *η πρώτη ισοδυναμία ικανοποιείται από τις τιμές* $x = 7 + 2 \cdot 5j$ *και για να βρούμε την τιμή* j *αντικαθιστούμε την τιμή του* x *στην δεύτερη ισοδυναμία που γίνεται* $7 + 2 \cdot 5j \equiv 5 \bmod 7$ ή $2 \cdot 5j \equiv (5 - 7) \bmod 7$ ή $2 \cdot 5j \equiv -2 \bmod 7 \equiv 5 \bmod 7$. Ο *πολλαπλασιαστικός αντίστροφος του* 10 mod 7 *είναι ο ίδιος με τον αντίστροφο του* 3 mod 7, *δηλαδή* 5. Ε*πομένως* $j \equiv 5 \cdot 5 \bmod 7 \equiv 4 \bmod 7$ *και* $j = 4 + 7k$, $k \in \mathbb{Z}$.

Άρα η λύση x *και των τριών ισοδυναμιών είναι* $x = 1 + 2i = 1 + 2(3 + 5j) = 7 + 2 \cdot 5j = 7 + 2 \cdot 5(4 + 7k)$, ή $x \equiv 47 \bmod 2 \cdot 5 \cdot 7$.

Αξ*ίζει να τονισθεί πως η λύση στο παραπάνω παράδειγμα βρέθηκε πρώτα* mod 2, *μετά* mod $2 \cdot 5$ *και τέλος* mod $2 \cdot 5 \cdot 7$. Ο *τρόπος αυτός της* "*σταδιακής*" *προσέγγισης της λύσης — είναι πάρα πολύ σημαντικός και — θα χρησιμοποιηθεί στην επόμενη ενότητα*.

Ακ*ολουθεί ένας πολύ ωραίος προγραμματισμός στο Mathematica του* Κ*ινεζικού αλγόριθμου υπολοίπων — μόνο για ακεραίους*. Βλ*έπε και την βιβλιογραφία*. Π*ροσέξτε πως αντί για τον* ΕΕΑ *χρησιμοποιούμε την συνάρτηση* `PowerMod[]` *και εκμεταλλευόμαστε στο έπακρο την ικανότητα του Mathematica να κάνει πράξεις σε λίστες* (*η ιδιότητα Listable διαφόρων συναρτήσεων*).

```
ourIntCRA[w_List, m_List] /; Length[w] == Length[m] :=
 Module[{mProd = Apply[Times, m], mm},
  mm = mProd / m;
  Mod[Apply[Plus, w mm PowerMod[mm, -1, m]], mProd]
 ]

ourIntCRA[{1, 2, 5}, {2, 5, 7}]

47
```

Το *πρόγραμμα αυτό θέλει προσεκτική μελέτη για να γίνει κατανοητή η δύναμη και ομορφιά του Mathematica*. Στις *ασκήσεις μπορεί κανείς να δει πως χρησιμοποιώντας τον* EEA *χάνουμε την απλότητα του προγράμματος*!

Στ*ην συνέχεια εξετάζουμε το* Κ*ινεζικό θεώρημα υπολοίπων στην γενική του μορφή για δακτυλίους, βλέπε και την βιβλιογραφία, και αναπτύσσουμε τον γενικό αλγόριθμο*.

Π*ρώτα ορίζουμε το* ***γινόμενο*** *ενός συνόλου* (*μεταθετικών*) *δακτυλίων* (direct product of rings) *ως το γινόμενο μεταθετικών* (*αβελιανών*) *ομάδων που γίνεται δακτύλιος ορίζοντας τον πολλαπλασιασμό ως προς τις συνιστώσες*.

Σ*υγκεκριμένα*, *αν* R_1 *και* R_2 *είναι δύο μεταθετικοί δακτύλιοι*, *τότε το γινόμενό τους είναι* $R_1 \times R_2 = \{(r_1, r_2) \mid r_1 \in R_1, r_2 \in R_2\}$, *δηλαδή το σύνολο των διατεταγμένων ζευγαριών* (r_1, r_2) *όπου πρόσθεση και πολλαπλασιασμός εκτελούνται ως προς τις συνιστώσες*:

$$(r_1, r_2) + (s_1, s_2) = (r_1 + s_1, r_2 + s_2)$$

και

$$(r_1, r_2)(s_1, s_2) = (r_1 s_1, r_2 s_2).$$

Σ*την συνέχεια της συζήτησή μας υποθέτουμε πως* R *είναι μία* Ευκλείδεια *περιοχή και πως έχουμε επιλέξει τα στοιχεία* $m_0, m_1, \ldots, m_{n-1} \in R$, *έτσι ώστε* $\gcd(m_i, m_j) = 1$ *για* $i \neq j$, $0 \le i, j \le n - 1$. Επίσης *ορίζουμε το στοιχείο* $m = m_0 m_1 \cdots m_{n-1} = \operatorname{lcm}(m_0, m_1, \ldots, m_{n-1}) \in R$, *το ελάχιστο κοινό πολλαπλάσιο των* $m_0, m_1, \ldots, m_{n-1}$.

Για κάθε έναν από τα στοιχεία m_i, $0 \le i \le n$ - 1, ορίζουμε το κύριο ιδεώδες $\langle m_i \rangle$ στο R, οπότε έχουμε τους κανονικούς ομοιομορφισμούς δακτυλίων

$$\varphi_i : R \to R/\langle m_i \rangle$$

όπου για $r \in R$ έχουμε

$$\varphi_i(r) \to r \bmod m_i .$$

Συνδιάζοντας **όλους** τους κανονικούς ομοιομορφισμούς φ_i ορίζουμε τον ομοιομορφισμό δακτυλίων

$$\varphi : R \to R/\langle m_0 \rangle \times \cdots \times R/\langle m_{n-1} \rangle$$

όπου για $r \in R$ έχουμε

$$\varphi(r) \to \{r \bmod m_0, \ldots, r \bmod m_{n-1}\}.$$

Θεώρημα 4.4.1 (Κινεζικό Θεώρημα Υπολοίπων):
Έστω $m_0, m_1, \ldots, m_{n-1} \in R$, R Ευκλείδεια περιοχή, έτσι ώστε $\gcd(m_i, m_j) = 1$ για $i \ne j$, $0 \le i, j \le n$ - 1. Τότε ο ομοιομορφισμός $\varphi : R \to R/\langle m_0 \rangle \times \cdots \times R/\langle m_{n-1} \rangle$ είναι μία απεικόνιση μονοσήμαντος και επί (surjection), με πυρήνα $\langle m \rangle$, όπου $m = m_0 m_1 \cdots m_{n-1}$, και ισχύει τόσο ο ισομορφισμός δακτυλίων

$$R/\langle m \rangle \cong R/\langle m_0 \rangle \times \cdots \times R/\langle m_{n-1} \rangle$$

όσο και ο ισομορφισμός ομάδων των πολλαπλασιαστικών ομάδων

$$(R/\langle m \rangle)^\times \cong (R/\langle m_0 \rangle)^\times \times \cdots \times (R/\langle m_{n-1} \rangle)^\times$$

Απόδειξη:
Δίνουμε μία κατασκευαστική απόδειξη που θα αποτελέσει την βάση του αντίστοιχου αλγορίθμου.

Έστω ότι $r \in R$. Τότε

$r \in \ker(\varphi)$

$\Leftrightarrow \varphi(r) = \{r \bmod m_0, \ldots, r \bmod m_{n-1}\} = \{0, \ldots, 0\}$

$\Leftrightarrow m_i \mid r,\ 0 \le i \le n - 1$

$\Leftrightarrow \operatorname{lcm}(m_0, m_1, \ldots, m_{n-1}) \mid r$

$\Leftrightarrow m \mid r$

και επομένως $\ker(\varphi) = \langle m \rangle$.

Για *να αποδείξουμε πως η απεικόνηση είναι μονοσήμαντος και επί αρκεί να δείξουμε πως για κάθε μοναδιαίο διάνυσμα* $e_i = \{0, \ldots, 0, 1, 0, \ldots, 0\} \in R/\langle m_0 \rangle \times \cdots \times R/\langle m_{n-1} \rangle$ *υπάρχει ένα στοιχείο* $L_i \in R$, — *αντίστοιχο των πολυωνύμων* $L_i(x)$ *της μεθόδου παρεμβολής* Lagrange — *έτσι ώστε* $\varphi(L_i) = e_i$, $0 \le i \le n - 1$. *Εξηγήσεις στις ασκήσεις*.

Το L_i *βρίσκεται αν εφαρμόσουμε τον επεκταμένο* Ευκλείδειο *αλγόριθμο* (ΕΕΑ) *στα στοιχεία* m_i *και* $\frac{m}{m_i}$ (*όπου* $\frac{m}{m_i} = m_0 \cdots m_{i-1} m_{i+1} \cdots m_{n-1}$), *βρούμε* $s_i, t_i \in R$ *έτσι ώστε* $s_i \frac{m}{m_i} + t_i m_i = 1 = \gcd(\frac{m}{m_i}, m_i)$, *και θέσουμε* $L_i = s_i \frac{m}{m_i}$.

Έτσι *έχουμε αφ' ενός μεν* $L_i \equiv 0 \bmod m_j$, $0 \le j \ne i \le n - 1$, *αφ' ετέρου δε* $L_i \equiv 1 \bmod m_i$ *διότι*

$$L_i = s_i \frac{m}{m_i} \equiv s_i \frac{m}{m_i} + t_i m_i = 1 \bmod m_i.$$

Άρα, $\varphi(L_i) = e_i$ *όπως το θέλαμε*.

Σημείωση: *Δεν πρέπει να ξεχνάμε πως για κάθε* i, $0 \le i \le n - 1$, *με τον* ΕΕΑ *βρίσκουμε το* s_i *που είναι το* ***πολλαπλασιαστικό αντίστροφο*** *του* $\frac{m}{m_i} \bmod m_i$.

Όσον *αφορά τον ισομορφισμό δακτυλίων*

$$R/\langle m \rangle \cong R/\langle m_0 \rangle \times \cdots \times R/\langle m_{n-1} \rangle$$

αυτός ισχύει από το θεώρημα ομοιομορφισμού δακτυλίων και τα παραπάνω. Ο *δε ισομορφισμός ομάδων των πολλαπλασιαστικών ομάδων*

$$(R/\langle m\rangle)^\times \cong (R/\langle m_0\rangle)^\times \times \cdots \times (R/\langle m_{n-1}\rangle)^\times$$

ισχύει διότι αν $r \in R$, *τότε έχουμε*

το r αντιστρέφεται mod m

$\Leftrightarrow \gcd(r, m) = 1$

$\Leftrightarrow \gcd(r, m_i) = 1$, *για* $0 \le i \le n - 1$

$\Leftrightarrow$ *το r αντιστρέφεται* mod m_i, *για* $0 \le i \le n - 1$.//

Αλγόριθμος: Κινεζικός αλγόριθμος υπολοίπων (ΚΑΥ ή CRA)

Είσοδος: $w_0, w_1, \ldots, w_{n-1} \in R$, όπου R Ευκλείδεια περιοχή, και $m_0, m_1, \ldots, m_{n-1} \in R$, έτσι ώστε $\gcd(m_i, m_j) = 1$ για $i \ne j$, $0 \le i, j \le n - 1$.
Έξοδος: $r = \sum_{i=0}^{n-1} L_i w_i \in R$, έτσι ώστε $r \equiv w_i \mod m_i$ για $0 \le i \le n - 1$. (Όπως αναφέραμε στην απόδειξη $L_i \equiv 0 \mod m_j$, $0 \le j \ne i \le n - 1$, και $L_i \equiv 1 \mod m_i$.)
==========

1. $m \leftarrow m_0 m_1 \cdots m_{n-1}$; $r \leftarrow 0$.

2. for $0 \le i < n$ **do** {

εφαρμόζουμε τον ΕΕΑ (ενότητα 2.2) στα $\frac{m}{m_i}, m_i$

και υπολογίζουμε $s_i, t_i \in R$, έτσι ώστε $s_i \frac{m}{m_i} + t_i m_i = 1$ (s_i είναι το πολ/κό αντίστροφο του $\frac{m}{m_i} \mod m_i$);

$L_i \leftarrow s_i \frac{m}{m_i}$;

$r \leftarrow r + L_i w_i$

}.

3. return rem(r, m).
==========

Πράγματι, ο αλγόριθμος κάνει αυτό που θέλουμε διότι για κάθε i, $0 \le i \le n - 1$, έχουμε

$$\sum_{i=0}^{n-1} L_i w_i = \sum_{i=0}^{n-1} s_i \frac{m}{m_i} w_i \equiv s_i \frac{m}{m_i} w_i \mod m_i$$

$$\equiv w_i \mod m_i$$

επειδή $s_i \frac{m}{m_i} \equiv 1 \bmod m_i$.

Για την περίπτωση $\boldsymbol{R} = \mathbb{Z}$ έχουμε το παρακάτω πρόγραμμα στο Mathematica

```
ourIntegerCRA[w_List, m_List] /; Length[w] == Length[m] :=
 Module[{a, b, g, L, mProd = Apply[Times, m], n = Length[w], r = 0},
  For[i = 0, i < n, i++, b = m[[i + 1]]; a = mProd/b; {g, {s, t}} =
    ExtendedGCD[a, b]; L = s a; r += L w[[i + 1]]]; Mod[r, mProd]
 ]
```

Έτσι βρίσκουμε τον ακέραιο 47

```
ourIntegerCRA[{1, 2, 5}, {2, 5, 7}]
```

```
47
```

για τον οποίο

```
Mod[47, {2, 5, 7}]
```

```
{1, 2, 5}
```

και το αποτέλεσμα είναι το ίδιο με αυτό που υπολογίσαμε στην αρχή της ενότητας αυτής.

Αν τώρα πάρουμε $\boldsymbol{R} = \boldsymbol{F}[\boldsymbol{x}]$, F σώμα, και $m_i = x - u_i$, για $0 \le i \le n - 1$, όπου $u_0, u_1, \ldots, u_{n-1} \in F$ είναι ανά δύο διάφορα, τότε για $f \in F[x]$ θα έχουμε

$$f \equiv f(u_i) \bmod (x - u_i), \quad 0 \le i \le n - 1$$

και επομένως ο ομοιομορφισμός δακτυλίων θα είναι

$$\varphi : F[x] \to F[x]/\langle x - u_0 \rangle \times \cdots \times F[x]/\langle x - u_{n-1} \rangle$$

όπου για $f \in F[x]$ έχουμε

$$\varphi(f) \to \{f(u_0), \ldots, f(u_{n-1})\}.$$

Ξέρουμε όμως πως ο ομοιομορφισμός αυτός ειναι ο ομοιομορφισμός διατίμησης ή εκτίμησης (evaluation homomorphism) στα σημεία $u_0, u_1, \ldots, u_{n-1} \in F$. Επιπλέον, τα στοιχεία $L_i \in R = F[x]$, βαθμού $< n$, είναι τώρα ακριβώς τα πολυώνυμα $L_i(x) = \prod_{i \neq j,\, 0 \le j \le n-1} \frac{x - u_j}{u_i - u_j}$, που ορίσαμε στην μέθοδο παρεμβολής Lagrange.

Αν λοιπόν $w_0, w_1, \ldots, w_{n-1} \in F$, το πολυώνυμο $f = \sum_{i=0}^{n-1} L_i w_i \in F[x]$ δεν είναι άλλο από το πολυώνυμο παρεμβολής Lagrange που ικανοποιεί

$$f(u_i) = w_i, \quad 0 \le i \le n - 1.$$

Το ακόλουθο πρόγραμμα στο Mathematica ισχύει για $R = F[x]$. Προσέξτε πως τώρα χρησιμοποιούμε `PolynomialExtendedGCD[]` και `PolynomialMod[]` αντί `ExtendedGCD[]` και `Mod[]` αντίστοιχα. Βλέπε και τις ασκήσεις.

```
<< Algebra`PolynomialExtendedGCD`;
ourPolyCRA[w_List, m_List] /; Length[w] == Length[m] :=
 Module[{a, b, g, L, mProd = Apply[Times, m], n = Length[w], f = 0},
  For[i = 0, i < n, i++, b = m[[i + 1]]; a = mProd/b;
   {g, {s, t}} = PolynomialExtendedGCD[a, b]; L = s a; f += L w[[i + 1]]];
  PolynomialMod[f, mProd]
 ]
```

Έτσι βρίσκουμε το πολυώνυμο

```
f[x_] = ourPolyCRA[{5, -3, 6, -4}, {x - 1, x - 2, x - 3, x - 4}]
```

$$66 - \frac{199\,x}{2} + \frac{89\,x^2}{2} - 6\,x^3$$

για το οποίο ισχύει

```
Map[f[#] &, {1, 2, 3, 4}]
```

```
{5, -3, 6, -4}
```

Το αποτέλεσμα αυτό είναι το ίδιο με εκείνο που υπολογίζουμε με παρεμβολή Lagrange (για να εκτλεσθεί η συνάρτηση `LagrangeInterpolation[]` πρέπει να την έχουμε ενεργοποιήσει στην προηγούμενη ενότητα).

```
fL[x_] = LagrangeInterpolation[{5, -3, 6, -4}, {1, 2, 3, 4}, x]
```

$$66 - \frac{199\,x}{2} + \frac{89\,x^2}{2} - 6\,x^3$$

```
Map[fL[#] &, {1, 2, 3, 4}]
```

```
{5, -3, 6, -4}
```

Επισημαίνουμε πως στα δύο τελευταία προγράμματα εφαρμόσαμε πιστά την περιγραφή του Κινεζικού αλγόριθμου υπολοίπων και έτσι χρησιμοποιήσαμε την εντολή `For[]`. Στις ασκήσεις ζητείται να προγραμματίσετε τον αλγόριθμο ***χωρίς*** την `For[]`, εκμεταλλευόμενοι την ιδιότητα του Mathematica να δρα σε λίστες.

Συμπέρασμα:
Όταν χρησιμοποιούμε τον Κινεζικό αλγόριθμο υπολοίπων με τα m_i να είναι n διαφορετικά ***γραμμικά*** πολυώνυμα, $m_i = x - u_i$, $0 \le i \le n - 1$, είναι το ***ίδιο*** σαν να κάνουμε παρεμβολή πολυωνύμου στα n διαφορετικά σημεία u_i, $0 \le i \le n - 1$.

Επιπλέον ο Κινεζικό αλγόριθμο υπολοίπων μας λέει πως το πολυώνυμο παρεμβολής είναι μοναδικό modulo $\prod_{i=0}^{n-1}(x - u_i)$ και άρα υπάρχει ένα μόνο πολυώνυμο $f = \sum_{i=0}^{n-1} L_i\, w_i \in F[x]$ βαθμού $< n$ που λύνει το πρόβλημα της παρεμβολής.

Άρα πρέπει να θεωρούμε τον Κινεζικό αλγόριθμο υπολοίπων σαν ***γενίκευση*** της παρεμβολής. Η γενίκευση αυτή φαίνεται όταν τα πολυώνυμα m_i ***δεν είναι γραμμικά*** και οι τιμές w_i, $0 \le i \le n - 1$, είναι και αυτές πολυώνυμα. Ένα παράδειγμα που μας οδηγεί στο επόμενο θέμα είναι και το ακόλουθο

```
fH[x_] = ourPolyCRA[{x, 1, 8}, {x^2, (x - 1)^3, x - 4}]
```

$$x + \frac{383\,x^2}{108} - \frac{239\,x^3}{36} + \frac{131\,x^4}{36} - \frac{59\,x^5}{108}$$

για το οποίο ισχύει

```
Map[PolynomialMod[fH[x], #] &, {x^2, (x - 1)^3, x - 4}]
```

```
{x, 1, 8}
```

Αυτό *αντιστοιχεί στην παρεμβολή* Hermite, *που είναι και το θέμα της ακόλουθης υποενότητας*.

■ 4.4.1 Π*αρεμβολή* Hermite

Η *παρεμβολή* Hermite *είναι μία γενίκευση της παρεμβολής πολυώνυμων όπου σε κάθε σημείο δίνεται όχι μόνο η τιμή της συνάρτησης αλλά και οι τιμές μερικών από τις πρώτες παραγώγους της*. Ισοδύναμα *λέμε ότι σε κάθε σημείο δίνεται ένα μερικό ανάπτυγμα της συνάρτησης κατά* Taylor.

Όταν *ο δακτύλιος* R *είναι* $\mathbb{Z}[x]$ ή $\mathbb{Q}[x]$ ή $\mathbb{R}[x]$ ή *τέλος* $\mathbb{C}[x]$, *το ανάπτυγμα κατά* Taylor *του* f *γύρω από το* u *(που ανήκει στο* $\mathbb{Z}$ ή *στο* $\mathbb{Q}$ ή *στο* $\mathbb{R}$ ή *τέλος στο* $\mathbb{C}$) *είναι της γνωστής μας μορφής*

$$f(x) = \frac{f^{(n)}(u)}{n!}(x - u)^n + \cdots + f^{(1)}(u)(x - u) + f(u),$$

όπου $f^{(i)}$ *είναι η* i*-στή παράγωγος*. Τονίζουμε *πως για* $R = \mathbb{Z}[x]$ *και* $u \in \mathbb{N}$, *οι συντελεστές* $\frac{f^{(k)}(u)}{k!}$ *είναι όλοι τους ακέραιοι (βλέπε και τις ασκήσεις)*. Για $k \le n$, *από το παραπάνω ανάπτυγμα συνεπάγεται πως*

$$f(x) \equiv f_{k-1}\cdot(x - u)^{k-1} + \cdots + f_1\cdot(x - u) + f_0 \mod (x - u)^k$$

και λέμε πως έχουμε ***το ανάπτυγμα κατά*** **Taylor** ***του*** f ***γύρω από το*** u ***τάξης*** k. Όταν R *είναι ένας μεταθετικός δακτύλιος*, $u \in R$ *και* $f \in R[x]$, $\deg(f) \le n$, *η ονομασία αυτή είναι σε πλήρη συμφωνία με τον* ***γενικό ορισμό*** *του αναπτύγματος κατά* Taylor *του* f *γύρω από το* u, *που είναι*

$$f(x) = f_n\cdot(x - u)^n + \cdots + f_1\cdot(x - u) + f_0.$$

Έστω *τώρα ότι* F *είναι σώμα*, $u_0, u_1, \ldots, u_{n-1} \in F$, *και διάφορα μεταξύ τους*, $e_0, e_1, \ldots, e_{n-1} \in \mathbb{N}$, *και* $w_0, w_1, \ldots, w_{n-1} \in F[x]$, *έτσι ώστε* $\deg(w_i) < e_i$ *για όλα τα* i.

Το ***πρόβλημα της παρεμβολής του*** **Hermite** *είναι να υπολογίσουμε* $f \in F[x]$, $\deg(f) < s = e_0 + e_1 + \cdots + e_{n-1}$ *έτσι ώστε για όλα τα* i, *το* w_i *είναι το ανάπτυγμα κατά* Taylor *του* f *γύρω από το* u_i *τάξης* e_i. *Ισοδύναμα, πρέπει να υπολογίσουμε* $f \in F[x]$, *με τον Κινεζικό αλγόριθμο υπολοίπων (για πολυώνυμα) έτσι ώστε*

$$f(x) \equiv w_i \mod (x - u_i)^{e_i},\ 0 \le i \le n - 1.$$

Παράδειγμα:

Έστω *ότι θέλουμε να βρούμε ένα πολυώνυμο* $f \in \mathbb{Q}[x]$ *βαθμού* < 6 *έτσι ώστε*: *στο σημείο* $u_0 = 0$, *έχουμε* $f(0) = 0$ *και* $f^{(1)}(0) = 1$, *στο σημείο* $u_1 = 1$, *έχουμε* $f(1) = 1$, $f^{(1)}(1) = 0$ *και* $f^{(2)}(1) = 0$, *και στο σημείο* $u_2 = 4$, *έχουμε* $f(4) = 8$. *Για να βρούμε το* $f \in \mathbb{Q}[x]$ *παρατηρούμε τα εξής*:

α. *Στο σημείο* $u_0 = 0$, *το μερικό ανάπτυγμα κατά* Taylor *του* f *γύρω από το* u_0 *είναι τάξης* $e_0 = 2$, *επειδή δίδονται* $f(0) = 0$ *και* $f^{(1)}(0) = 1$. *Δηλαδή, έχουμε*

$$w_0 = f(0) + f^{(1)}(0) \cdot x = x$$

Στην ορολογία του Κινεζικού αλγόριθμου υπολοίπων έχουμε την ισοδυναμία

$$f(x) \equiv x \mod x^2.$$

β. *Στο σημείο* $u_1 = 1$, *το μερικό ανάπτυγμα κατά* Taylor *του* f *γύρω από το* u_1 *είναι τάξης* $e_1 = 3$, *επειδή δίδονται* $f(1) = 1$, $f^{(1)}(1) = 0$ *και* $f^{(2)}(1) = 0$. *Δηλαδή, έχουμε*

$$w_1 = f(1) + f^{(1)}(1) \cdot (x - 1) + f^{(2)}(1) \cdot (x - 1)^2 = 1$$

Στην ορολογία του Κινεζικού αλγόριθμου υπολοίπων έχουμε την ισοδυναμία

$$f(x) \equiv 1 \mod (x - 1)^3.$$

γ. Τέλος *στο σημείο* $u_2 = 4$, *το μερικό ανάπτυγμα κατά* Taylor *του* f *γύρω από το* u_2 *είναι τάξης* $e_2 = 1$, *επειδή δίδεται* $f(4) = 8$. *Δηλαδή, έχουμε*

$$w_2 = f(4) = 8$$

Στην ορολογία του Κινεζικού αλγόριθμου υπολοίπων έχουμε την ισοδυναμία

$$f(x) \equiv 8 \mod (x - 4).$$

Με άλλα λόγια, *θέλουμε να λύσουμε το πρόβλημα παρεμβολής του* Hermite *με τα εξής δεδομένα*: $u_0 = 0, u_1 = 1, u_2 = 4, e_0 = 2, e_1 = 3, e_2 = 1, w_0 = x, w_1 = 1, w_2 = 8$.

Με *τον* Κινεζικό *αλγόριθμο υπολοίπων* (*για πολυώνυμα*) *βρίσκουμε αμέσως το πολυώνυμο* $f \in \mathbb{Q}[x]$, 5*ου βαθμού*

```
f[x_] = ourPolyCRA[{x, 1, 8}, {x^2, (x - 1)^3, x - 4}]
```

$$x + \frac{383\,x^2}{108} - \frac{239\,x^3}{36} + \frac{131\,x^4}{36} - \frac{59\,x^5}{108}$$

για το οποίο ισχύουν οι αρχικές συνθήκες

```
{f[0] == 0, f'[0] == 1}
```

```
{True, True}
```

```
{f[1] == 1, f'[1] == 0, f''[1] == 0}
```

```
{True, True, True}
```

```
f[4] == 8
```

```
True
```

ή *ισοδύναμα*

```
Map[PolynomialMod[f[x], #] &, {x^2, (x - 1)^3, x - 4}]
```

```
{x, 1, 8}
```

Το *πρόβλημα παρεμβολής του* Hermite *χρειάζεται τόσο χρόνο για να λυθεί όσο απαιτεί ο* Κινεζικός *αλγόριθμος υπολοίπων*. Η *ανάλυση του τελευταίου ακολουθεί*.

■ 4.4.2 Ανάλυση του Κινεζικού αλγόριθμου υπολοίπων

Όσον αφορά τον χρόνο υπολογισμού του Κινεζικού αλγόριθμου υπολοίπων (ΚΑΥ) διακρίνουμε δύο περιπτώσεις:

α. $R = F[x]$

Στην περίπτωση αυτή υπολογίζουμε πρώτα το πολυώνυμο $m = m_0 m_1 \cdots m_{n-1}$ εκτελώντας τους διαδοχικούς πολλαπλασιασμούς $m_0 m_1$, $(m_0 m_1) m_2$, …, $(m_0 m_1 \cdots m_{n-2}) m_{n-2}$. Ξέρουμε όμως πως το γινόμενο δύο πολυωνύμων a, b, βαθμού n_a και n_b αντίστοιχα, υπολογίζεται με $2(n_a + 1)(n_b + 1)$ πράξεις. Έτσι, τα διαδοχικά γινόμενα υπολογίζονται με

$$
\begin{aligned}
&2\textstyle\sum_{i=0}^{n-1} (d_0 + \cdots + d_{i-1} + 1)\,(d_i + 1) \\
&\qquad = 2\textstyle\sum_{0 \le j < i \le n-1} d_j (d_i + 1) + 2\sum_{i=1}^{n-1} (d_i + 1) \\
&\qquad < \textstyle\sum_{i,j=0}^{n-1} d_j (d_i + 1) = 2(\sum_{j=0}^{n-1} d_j)(\sum_{i=0}^{n-1} (d_i + 1)) \\
&\qquad = 2\, n_m (n_m + n) \\
&\qquad = O(n_m^2)
\end{aligned}
$$

πράξεις στο σώμα F.

Κατόπιν, για $0 \le i \le n - 1$, εκτελούμε διαιρέσεις της μορφής $\frac{m}{m_i}$, εφαρμόζουμε τον επεκταμένο Ευκλείδειο αλγόριθμο στα πολυώνυμα $\frac{m}{m_i}$ και m_i, και εκτελούμε πολλαπλασιασμούς της μορφής $s_i(\frac{m}{m_i})$ και $w_i(s_i \frac{m}{m_i})$.

Όσον αφορά τη διαίρεση δύο πολυωνύμων a, b, βαθμού n_a και n_b αντίστοιχα, ξέρουμε πως υπολογίζεται με $(2n_b + 1)(n_a - n_b + 1)$ πράξεις. Επομένως, όλες μαζί οι διαιρέσεις υπολογίζονται με

$$
\begin{aligned}
&\textstyle\sum_{i=0}^{n-1} (2\, d_i + 1)\,(n_m - d_i + 1) \\
&\qquad \le 2 n_m \textstyle\sum_{i=0}^{n-1} (d_i + 1) \\
&\qquad = 2\, n_m (n_m + n) \\
&\qquad = O(n_m^2)
\end{aligned}
$$

πράξεις στο στο σώμα F.

Όσον αφορά την εφαρμογή του επεκταμένου Ευκλείδειου αλγόριθμου στα πολυώνυμα a, b, βαθμού n_a και n_b αντίστοιχα, ξέρουμε πως υπολογίζεται με $n_a n_b$ πράξεις και επιστρέφει s και t ώστε $sa + tb = \gcd(a, b)$, όπου $\deg(s) < \deg(b)$.

Επομένως, κάθε εκτέλεση του επεκταμένου Ευκλείδειου αλγόριθμου στα πολυώνυμα $\frac{m}{m_i}$ και m_i εκτελείται με $O((n_m - d_i)d_i)$ πράξεις, και συνολικά όλες μαζί με $O(n_m^2)$ πράξεις (όπως και οι διαιρέσεις παραπάνω). Επιπλέον, επειδή $\deg(s_i) < \deg(m_i) = d_i$ κάθε πολλαπλασιασμός των πολυωνύμων s_i και $\frac{m}{m_i}$ εκτελείται με $O(d_i^2)$ πράξεις και συνολικά όλοι μαζί με

$$\begin{aligned}\sum_{i=0}^{n-1} d_i^2 &\le n_m \sum_{i=0}^{n-1} d_i \\ &= O(n_m^2)\end{aligned}$$

πράξεις στο σώμα F. Όσον αφορά τον πολλαπλασιασμό των πολυωνύμων w_i και $s_i \frac{m}{m_i}$ — επειδή $\deg(w_i) < \deg(m_i) = d_i$, και $\deg(s_i \frac{m}{m_i}) < \deg(m) = n_m$ — κάθε ένας εκτελείται με $O(d_i n_m)$ πράξεις και συνολικά όλοι μαζί με $O(n_m^2)$ πράξεις στο σώμα F.

Στο τέλος έχουμε να υπολογίσουμε το υπόλοιπο της διαίρεσης του πολυωνύμου $\sum_{i=0}^{n-1} s_i \frac{m}{m_i} w_i$ με το m. Επειδή όμως $\deg(\sum_{i=0}^{n-1} s_i \frac{m}{m_i} w_i) < 2n_m$ και $\deg(m) = n_m$ η διαίρεση αυτή γίνεται με $(n_m + 1)(2n_m - n_m + 1)$ ή $O(n_m^2)$ πράξεις στο σώμα F.

Λήμμα 4.4.1:
Έστω $m_0, m_1, \ldots, m_{n-1} \in R = F[x]$, όπου F σώμα, $\deg(m_i) = d_i \ge 1$, για κάθε i, $0 \le i \le n - 1$, $m = m_0 m_1 \cdots m_{n-1}$, και $n_m = \deg(m) = \sum_{i=0}^{n-1} d_i$. Έστω επίσης $w_0, w_1, \ldots, w_{n-1} \in R = F[x]$, με $\deg(w_i) < \deg(m_i) = d_i$. Τότε ο Κινεζικός αλγόριθμος υπολοίπων για πολυώνυμα υπολογίζει το μοναδικό $f \in F[x]$, $\deg(f) < n_m$,

$$f \equiv w_i \mod m_i \text{ για } 0 \le i \le n - 1$$

με $O(n_m^2)$ πράξεις στο σώμα F.

***β.* $R = \mathbb{Z}$**

Για τους ακεραίους έχουμε το ακόλουθο Λήμμα την απόδειξη του οποίου αφήνουμε για άσκηση.

Λήμμα 4.4.2:
Έστω $m_0, m_1, \ldots, m_{n-1} \in \mathbb{N}$, $m = m_0 m_1 \cdots m_{n-1}$ και $n_m = \lambda(m)$ το μήκος του ακεραίου m. Έστω επίσης $w_0, w_1, \ldots, w_{n-1} \in \mathbb{Z}$, όπου $0 \le |w_i| < m_i$ για κάθε i, $0 \le i \le n - 1$. Τότε ο Κινεζικός αλγόριθμος υπολοίπων για ακεραίους υπολογίζει τον μοναδικό ακέραιο $f \in \mathbb{Z}$, $0 \le f < m$,

$$f \equiv w_i \mod m_i \text{ για } 0 \le i \le n - 1$$

με $O(n_m^2)$ πράξεις σε λέξεις του υπολογιστή.

4.5 Παρεμβολή Newton

Όπως είδαμε στην προηγούμενη ενότητα, αν πάρουμε $R = F[x]$, F σώμα, και χρησιμοποιήσουμε τον Κινεζικό αλγόριθμο υπολοίπων με τα **γραμμικά** πολυώνυμα $m_i = x - u_i$, για $0 \le i \le n - 1$, όπου $u_0, u_1, \ldots, u_{n-1} \in F$ είναι ανά δύο διάφορα, τότε θα βρούμε $f \in F[x]$ έτσι ώστε

$$f \equiv f(u_i) \bmod (x - u_i), \quad 0 \le i \le n - 1.$$

Ο αλγόριθμος υπολογίζει το πολυώνυμο παρεμβολής f με την βοήθεια των στοιχείων $L_i \in R = F[x]$, βαθμού $< n$, τα οποία είναι ακριβώς τα πολυώνυμα $L_i(x) = \prod_{i \ne j,\, 0 \le j \le n-1} \frac{x - u_j}{u_i - u_j}$, που ορίσαμε στην μέθοδο παρεμβολής Lagrange.

Στην ενότητα αυτή θα παρουσιάσουμε μία διαφορετική μορφή του αλγόριθμου αυτού, όπου το πολυώνυμο παρεμβολής f θα προσεγγίζεται σταδιακά: πρώτα $\bmod x - u_0$, μετά $\bmod (x - u_0)\cdot(x - u_1)$, …, και τέλος $\bmod (x - u_0)\cdots(x - u_{n-1})$ (βλέπε και το αντίστοιχο παράδειγμα για τους ακεραίους στην προηγούμενη ενότητα). Όπως θα δούμε, το αποτέλεσμα είναι ο τύπος παρεμβολής του Νεύτωνα.

Ακολουθεί ο Κινεζικός αλγόριθμος υπολοίπων όπου η λύση προσεγγίζεται σταδιακά. Ο γενικός αλγόριθμος που παρουσιάζουμε βασίζεται στο παράδειγμα της προηγούμενης ενότητας και αποτελείται από δύο αλγορίθμους: τον CRA2, για την περίπτωση 2 ισοδυναμιών, και τον CRAn, για την περίπτωση n ισοδυναμιών. Ο CRAn χρησιμποποιεί τον CRA2 και λύνει τις ισοδυναμίες κατά ζεύγη. Μας ενδιαφέρει όμως μόνο η περίπτωση των πολυωνύμων (βλέπε και την βιβλιογραφία).

Αλγόριθμος: Κινεζικός αλγόριθμος υπολοίπων για πολυώνυμα — δύο ισοδυναμίες (ΚΑΥ2 ή CRA2)

Είσοδος: w_0, w_1 που ανήκουν στην Ευκλείδεια περιοχή $R = F[x]$, F σώμα, και $m_0, m_1 \in R$, έτσι ώστε $\gcd(m_0, m_1) = 1$ και $\deg(m_0) \le n - 2$, για n που ορίζεται στον CRAn, και $\deg(m_1) = 1$.
Έξοδος: $f \in R$, έτσι ώστε $f \equiv w_i \bmod m_i$ για $i = 0, 1$.

==========

1. $f \leftarrow \text{rem}(w_0, m_0)$.

2. *εφαρμόζουμε τον* EEA (*ενότητα* 2.2) *στα* m_0, m_1 *και υπολογίζουμε* $s, t \in R$, *έτσι ώστε* $sm_0 + tm_1 = 1$ (*s είναι το πολλαπλασιαστικό αντίστροφο του* $m_0 \bmod m_1$);

$q \leftarrow \text{rem}(s(w_1 - f), m_1)$

3. return $f \leftarrow f + qm_0$.

==========

Αλγόριθμος: Κινεζικός αλγόριθμος υπολοίπων για πολυώνυμα — n ισοδυναμίες (ΚΑΥ*n* ή CRA*n*)

Είσοδος: $w_0, w_1, \ldots, w_{n-1}$ *που ανήκουν στην* Ευκλείδεια περιοχή $R = F[x]$, F *σώμα, και* $m_0, m_1, \ldots, m_{n-1} \in R$, *έτσι ώστε* $\gcd(m_i, m_j) = 1$ *για* $i \neq j$, $0 \leq i, j \leq n - 1$ *και* $\mathbf{deg(m_i) = 1}$ *για* $0 \leq i \leq n - 1$.

Έξοδος: $f \in R$, *έτσι ώστε* $\deg(f) < n$ *και* $f \equiv w_i \bmod m_i$ *για* $0 \leq i \leq n - 1$.

==========

1. $m \leftarrow 1$; $W_0 \leftarrow w_0$.

2. for $0 \leq i < n - 1$ **do** {

$m \leftarrow m \cdot m_i$;

εφαρμόζουμε τον CRA2 *στα ζεύγη* $\{W_i, w_{i+1}\}$ *και* $\{m_0 m_1 \cdots m_i, m_{i+1}\}$, *και υπολογίζουμε* W_{i+1}, *την προσέγγιση* $\bmod m_0 m_1 \cdots m_i m_{i+1}$.

}.

3. return $f \leftarrow W_{n-1}$.

==========

Τα *παρακάτω προγράμματα στο Mathematica ισχύουν για* $\boldsymbol{R = F[x]}$.

```
(**  2 congruencies  **)
<< Algebra`PolynomialExtendedGCD`;
ourPolyCRA2[w_List, m_List] /; Length[w] == Length[m] := Module[
  {g, q, f, s, t},
  f = PolynomialMod[w[[1]], m[[1]]];
  {g, {s, t}} = PolynomialExtendedGCD[m[[1]], m[[2]]];
  q = PolynomialMod[s (w[[2]] - f), m[[2]]];
  f = f + q m[[1]]
 ]

(**  n congruencies  **)
ourPolyCRAn[w_List, m_List] /; Length[w] == Length[m] :=
 Module[{mProd = 1, V, W = w[[1]]},
  (** Print["η προσέγγιση είναι  ",
      PolynomialMod[W,137],"  mod ",mProd  m[[1]]]; **)
  For[i = 0, i < Length[w] - 1, i++,
   mProd = mProd m[[i + 1]];
   V = ourPolyCRA2[{W, w[[i + 2]]}, {mProd, m[[i + 2]]}];
   W = V; (** Print["η προσέγγιση είναι  ",
      PolynomialMod[W,137],"  mod ",mProd  m[[i+2]]] **)
  ]; W
 ]
```

Έστω τώρα $R = \mathbb{Z}_n[x]/\langle f \rangle$, όπου έχουμε αριθμητική με διπλό υπόλοιπο, modulo f και modulo n, και έστω $n = 137$. Έστω επί πλέον ότι παίρνουμε

$$w_0 = 8,\ w_1 = 35,\ w_2 = 50,$$
$$w_3 = 37,\ w_4 = 127,\ w_5 = 99,\ w_6 = 39$$

και

$$m_0 = x + 5,\ m_1 = x + 15,\ m_2 = x + 25,$$
$$m_3 = x + 35,\ m_4 = x + 45,\ m_5 = x + 55,\ m_6 = x + 65,$$

και θέλουμε να βρούμε το πολυώνυμο $f \in R$, βαθμού ≤ 6, ώστε να ισχύει $f \equiv w_i \mod m_i$ για $0 \le i \le 6$.

Το αποτέλεσμα είναι $f(x) = x^6 + 1$, και αν ενεργοποιήσουμε τις Print statements στο πρόγραμμα βλέπουμε ότι αυτό προσεγγίζεται σταδιακά από τα εξής πολυώνυμα:

```
PolynomialMod[ourPolyCRAn[{8, 35, 50, 37, 127, 99, 39},
  {x + 5, x + 15, x + 25, x + 35, x + 45, x + 55, x + 65}], 137]
```

η προσέγγιση είναι $8 \bmod 5 + x$

η προσέγγιση είναι $63 + 11\,x \bmod (5 + x)(15 + x)$

η προσέγγιση είναι $127 + 92\,x + 52\,x^2 \bmod (5 + x)(15 + x)(25 + x)$

η προσέγγιση είναι $132 + 57\,x + 85\,x^2 + 19\,x^3 \bmod (5 + x)(15 + x)(25 + x)(35 + x)$

η προσέγγιση είναι $17 + 115\,x + 34\,x^2 + 77\,x^3 + 11\,x^4 \bmod (5 + x)(15 + x)(25 + x)(35 + x)(45 + x)$

η προσέγγιση είναι $120 + 39\,x + 48\,x^2 + 112\,x^3 + 116\,x^4 + 94\,x^5 \bmod (5 + x)(15 + x)(25 + x)(35 + x)(45 + x)(55 + x)$

η προσέγγιση είναι $1 + x^6 \bmod (5 + x)(15 + x)(25 + x)(35 + x)(45 + x)(55 + x)(65 + x)$

$1 + x^6$

Προσέξτε την σταδιακή προσέγγιση πρώτα $\bmod (x + 5)$, έπειτα $\bmod (x + 5)(x + 15)$, … και τέλος $\bmod (x + 5)(x + 15)(x + 25)(x + 35)(x + 45)(x + 55)(x + 65)$. Επίσης προσέξτε πόσο διαφέρει το τελικό πολυώνυμο $x^6 + 1$ από τα ενδιάμεσα!

Το φαινόμενο αυτό της σταδιακής προσέγγισης θα το συναντήσουμε αμέσως παρακάτω στην παρεμβολή του Νεύτωνα και αργότερα στο κεφάλαιο για την παραγοντοποίηση των πολυωνύμων.

Έστω τώρα η Ευκλείδεια περιοχή $R = F[x]$, F σώμα, $f \in R$, $\deg(f) < n$ και έστω ότι $f(u_i) = w_i$, για $0 \le i < n$ — δηλαδή και εδώ είναι $m_i = x - u_i$. Με την **παρεμβολή του Νεύτωνα** υπολογίζουμε σταθερές $\lambda_i \in F$ έτσι ώστε

$$f(x) = \lambda_0 + \lambda_1(x - u_0) + \lambda_2(x - u_0)(x - u_1) + \cdots + \lambda_n(x - u_0)\cdots(x - u_{n-1})$$

Αν τώρα συμβολίσουμε με $f^{(k)}(x)$ το πολυώνυμο που προσεγγίζει το f με μόνο τις πρώτες k από τις σταθερές λ_i, τότε έχουμε

$$f(x) = f^{(k)}(x) + \lambda_k(x - u_0)\cdots(x - u_{k-1}) + \cdots + \lambda_n(x - u_0)\cdots(x - u_{n-1}).$$

Στην ορολογία του Κινεζικού αλγόριθμου υπολοίπων λέμε πως το $f^{(k)}(x)$ προσεγγίζει το $f(x)$ modulo $(x - u_0)\cdots(x - u_{k-1})$.

Οι σταθερές λ_i, $0 \le i < n$, μπορούν εύκολα να υπολογισθούν. Πράγματι, για την σταθερά λ_0 έχουμε

$$f(u_0) = w_0 = \lambda_0,$$

ενώ η λ_1 είναι η λύση της εξίσωσης

$$f(u_1) = w_1 = w_0 + \lambda_1(u_1 - u_0).$$

Γενικά, έχοντας υπολογίσει τις σταθερές $\lambda_0, \lambda_1, \ldots, \lambda_{k-1}$ η επόμενη σταθερά λ_k είναι η λύση της εξίσωσης

$$f(u_k) = w_k = \lambda_0 + \lambda_1(u_k - u_0) + \cdots + \lambda_k(u_k - u_0)\cdots(u_k - u_{k-1}).$$

Ισοδύναμα, μπορούμε να εξετάσουμε το θέμα και από την σκοπιά των προσεγγιστικών πολυωνύμων.

Πράγματι, με μία σταθερά — ή modulo $(x - u_0)$ — το πολυώνυμο παρεμβολής f προσεγγίζεται από το

$$f^{(1)}(x) = w_0,$$

ενώ με δύο σταθερές — ή modulo $(x - u_0)(x - u_1)$ — το πολυώνυμο f προσεγγίζεται από το

$$f^{(2)}(x) = w_0 + \frac{w_1 - w_0}{u_1 - u_0}(x - u_0).$$

Προσέξτε πως ισχύει $w_1 = f^{(1)}(u_0) + \lambda_1(u_1 - u_0)$ και γενικά

$$w_k = f^{(k)}(u_{k-1}) + \lambda_k(u_k - u_0)\cdots(u_k - u_{k-1}).$$

Λύνοντας ως προς λ_k έχουμε τον εξής επαγωγικό τύπο για τα προσεγγιστικά πολυώνυμα

$$f^{(k+1)}(x) = f^{(k)}(x) + (w_k - f^{(k)}(u_{k-1}))\frac{(x-u_0)\cdots(x-u_{k-1})}{(u_k-u_0)\cdots(u_k-u_{k-1})}.$$

Με άλλα λόγια, το πολυώνυμο παρεμβολής f προσεγγίζεται από το $f^{(k+1)}(x)$ modulo $(x - u_0)\cdots(x - u_k)$, και τελικά $f(x) = f^{(n-1)}(x)$.

Στο ακόλουθο πρόγραμμα στο Mathematica ορίζουμε την συνάρτηση NewtonInterpolation[], η οποία μας δίνει το πολυώνυμο παρεμβολής προσεγγίζοντάς το σταδιακά, όπως ακριβώς και ο Κινεζικός αλγόριθμος υπολοίπων που ορίσαμε στην ενότητα αυτή. Το πρόγραμμα είναι μία εφαρμογή του επαγωγικού τύπου για τα προσεγγιστικά πολυώνυμα $f^{(k)}(x)$.

```
NewtonInterpolation[w_List, m_List] /;
  Length[w] == Length[m] := Module[
  {approxf, i, k},
  approxf[0, x_] = w[[1]];
  approxf[k_, x_] := approxf[k, x] = approxf[k - 1, x] + (w[[k]] -
        approxf[k - 1, -First[m[[k]]]]) (∏_{i=1}^{k-1} (x + First[m[[i]]])) /
      (∏_{i=1}^{k-1} (-First[m[[k]]] + First[m[[i]]]));
  approxf[Length[m], x]
 ]
```

Έτσι, με σταδιακές προσεγγίσεις βρίσκουμε το ίδιο πολυώνυμο που βρίκαμε παραπάνω με τον Κινεζικό αλγόριθμο υπολοίπων

```
w = {8, 35, 50, 37, 127, 99, 39};
m = {x + 5, x + 15, x + 25, x + 35, x + 45, x + 55, x + 65};

PolynomialMod[NewtonInterpolation[w, m], 137]
```

$1 + x^6$

Όμως *τα πολυώνυμα που υπολογίζονται επαγωγικά και τα οποία προσεγγίζουν διαδοχικά το* $f(x)$ modulo $(x - u_0)\cdots(x - u_k)$, $k = 0, 1, \ldots, n$ *είναι τα ίδια προσεγγιστικά πολυώνυμα που υπολογίσαμε προηγουμένως με τον* Κινεζικό *αλγόριθμο υπολοίπων*. Πράγματι, *αν ενεργοποιήσουμε τον παρακάτω κώδικα* (*που είναι κάνει* όλη *την δουλειά στην συνάρτηση* NewtonInterpolation[])

```
approxf[0, x_] = w[[1]]; approxf[k_, x_] := approxf[k, x] =
  approxf[k - 1, x] + (w[[k]] - approxf[k - 1, -First[m[[k]]]])
    (∏_{i=1}^{k-1} (x + First[m[[i]]])) / (∏_{i=1}^{k-1} (-First[m[[k]]] + First[m[[i]]]))
```

για το ίδιο παράδειγμα έχουμε:

```
Table[
  Print["η προσέγγιση είναι  ", PolynomialMod[approxf[i, x], 137],
   "   mod  ", ∏_{j=1}^{i} (x + First[m[[j]]]), {i, 1, 7}];
```

η προσέγγιση είναι $8 \mod 5 + x$

η προσέγγιση είναι $63 + 11\,x \mod (5 + x)(15 + x)$

η προσέγγιση είναι $127 + 92\,x + 52\,x^2 \mod (5 + x)(15 + x)(25 + x)$

η προσέγγιση είναι $132 + 57\,x + 85\,x^2 + 19\,x^3 \mod (5 + x)(15 + x)(25 + x)(35 + x)$

η προσέγγιση είναι $17 + 115\,x + 34\,x^2 + 77\,x^3 + 11\,x^4 \mod (5 + x)(15 + x)(25 + x)(35 + x)(45 + x)$

η προσέγγιση είναι $120 + 39\,x + 48\,x^2 + 112\,x^3 + 116\,x^4 + 94\,x^5 \mod (5 + x)(15 + x)(25 + x)(35 + x)(45 + x)(55 + x)$

η προσέγγιση είναι $1 + x^6 \mod (5 + x)(15 + x)(25 + x)(35 + x)(45 + x)(55 + x)(65 + x)$

Έτσι βλέπουμε πως η παρεμβολή του Νεύτωνα είναι ισοδύναμη με τον Κινεζικό αλγόριθμο υπολοίπων — όταν αυτός υπολογίζει το πολυώνυμο παρεμβολής με ακολουθία προσεγγίσεων.

4.6 Μετασχηματισμός Fourier

Οι μέθοδοι παρεμβολής που συζητήσαμε μέχρι τώρα δεν θέτουν περιορισμούς στα σημεία εκτίμησης (evaluation points) που χρησιμοποιούν και απαιτούν $O(n^2)$ πράξεις για τον υπολογισμό του πολυωνύμου παρεμβολής βαθμού $\le n - 1$. Στην ενότητα αυτή θα παρουσιάσουμε μία μέθοδο παρεμβολής που σαν σημεία εκτίμησης χρησιμοποιεί τις ρίζες του πολυωνύμου $x^n - 1$ και απαιτεί μόνο $O(n \log n)$ πράξεις.

Έστω λοιπόν R δακτύλιος, $\omega \in R$ και $n \in \mathbb{N}$. Αν $\omega^n = 1$ τότε το ω είναι μία ***n*-στή ρίζα της μονάδας** ή ***ρίζα της μονάδας τάξης n*** (n-th root of unity). Προφανώς κάθε μία ρίζα του $x^n - 1$ είναι μία n-στή ρίζα της μονάδος.

Αν ω είναι μία n-στή ρίζα της μονάδος και επιπλέον $\omega^{n/p} - 1$ ***δεν*** είναι μηδενοδιαιρέτης (δηλαδή $\omega^{n/p} \neq 1$) για κανέναν πρώτο διαιρέτη p του n, τότε λέμε πως το ω είναι μία ***αρχέγονη n-στή ρίζα της μονάδας*** ή ***αρχέγονη ρίζα της μονάδας, τάξης n*** (primitive n-th root of unity).

Το σύνολο των ριζών της μονάδας, τάξης n, αποτελεί μία κυκλική πολλαπλασιαστική ομάδα. Έτσι, ισοδύναμα, ονομάζουμε αρχέγονες n-στές ρίζες της μονάδος όσες από τις n ρίζες του $x^n - 1$ έχουν τάξη n — είναι δηλαδή ***γεννήτριες*** της κυκλικής ομάδας (βλέπε και την βιβλιογραφία).

Κατά συνέπεια αν ω είναι μία αρχέγονη ρίζα της μονάδας, τάξης n, οι άλλες αρχέγονες ρίζες είναι τα στοιχεία ω^ℓ, όπου $1 \le \ell < n$, και $\gcd(\ell, n) = 1$, διότι αυτές είναι οι γεννήτριες της κυκλικής ομάδας τάξης n. Άρα υπάρχουν $\varphi(n)$ αρχέγονες ρίζες της μονάδας, τάξης n, όπου φ είναι η συνάρτηση φ του Euler.

Παράδειγμα:
Στον δακτύλιο $\boldsymbol{R} = \mathbb{C}$, το πολυώνυμο $x^8 - 1$ έχει τις εξής 8 ρίζες

```
sol = Solve[x^8 - 1 == 0]
```

```
{{x → -1}, {x → -ⅈ}, {x → ⅈ}, {x → 1},
 {x → -(-1)^(1/4)}, {x → (-1)^(1/4)}, {x → -(-1)^(3/4)}, {x → (-1)^(3/4)}}
```

οι οποίες γράφονται και σαν

```
solutions = x /. sol;
Table[ComplexExpand[solutions[[k + 1]]], {k, 0, 7}]
```

$$\left\{-1,\ -\text{i},\ \text{i},\ 1,\ -\frac{1+\text{i}}{\sqrt{2}},\ \frac{1+\text{i}}{\sqrt{2}},\ \frac{1-\text{i}}{\sqrt{2}},\ -\frac{1-\text{i}}{\sqrt{2}}\right\}$$

Τις *ρίζες αυτές τις παριστάνουμε και ως* $\omega_k = e^{2\pi i k/8} = \cos(\frac{2\pi k}{n}) + i\,\sin(\frac{2\pi k}{n})$, $k = 0, 1, \ldots, 7$. *Στο μιγαδικό επίπεδο τα σημεία αυτά παρίστανται από* 8 *ισοαπέχοντα σημεία στον μοναδιαίο κύκλο, αρχίζοντας από το σημείο* {1, 0}, *που αντιστοιχεί στο* $k = 0$. (Ο *κύκλος διατρέχεται με φορά αντίθετη των δεικτών του ωρολογίου.*) Έτσι *έχουμε*

```
Table[ω_k = ℯ^(2πⅈk/8), {k, 0, 7}]
```

$$\left\{1,\ e^{\frac{\text{i}\pi}{4}},\ \text{i},\ e^{\frac{3\text{i}\pi}{4}},\ -1,\ e^{-\frac{3\text{i}\pi}{4}},\ -\text{i},\ e^{-\frac{\text{i}\pi}{4}}\right\}$$

ή *ισοδύναμα*

```
Table[ComplexExpand[ω_k = ℯ^(2πⅈk/8)], {k, 0, 7}]
```

$$\left\{1,\ \frac{1+\text{i}}{\sqrt{2}},\ \text{i},\ -\frac{1-\text{i}}{\sqrt{2}},\ -1,\ -\frac{1+\text{i}}{\sqrt{2}},\ -\text{i},\ \frac{1-\text{i}}{\sqrt{2}}\right\}$$

Το $\omega_1 = e^{2\pi i/8}$ *είναι μία αρχέγονη* 8*-η ρίζα της μονάδος διότι εκτός του ότι είναι ρίζα του* $x^8 - 1$, *η έκφραση* $\omega_1^{n/p} - 1$ ***δεν*** *είναι μηδενοδιαιρέτης για* $n = 8$ *και* $p = 2$

```
n = 8; p = 2; ω_1^(n/p) - 1
```

```
-2
```

Ισοδύναμα, *η ρίζα* ω_1 *είναι γεννήτρια της κυκλικής ομάδας*.

```
Table[ComplexExpand[ω1^k], {k, 0, 7}]
```

$$\left\{1,\ \frac{1+i}{\sqrt{2}},\ i,\ -\frac{1-i}{\sqrt{2}},\ -1,\ -\frac{1+i}{\sqrt{2}},\ -i,\ \frac{1-i}{\sqrt{2}}\right\}$$

Αντίθετα, *η ρίζα* ω_2 *δεν είναι αρχέγονη ρίζα της μονάδος τάξης* 8 *διότι οι δυνάμεις της δεν παράγουν τα* 8 *διαφορετικά στοιχεία της ομάδας*.

```
Table[ComplexExpand[ω2^k], {k, 0, 7}]
```

```
{1, i, -1, -i, 1, i, -1, -i}
```

Ισοδύναμα, *η έκφραση* $\omega_2^{n/p}$ - 1 ***είναι*** *μηδενοδιαιρέτης για* $n = 8$ *και* $p = 2$

```
n = 8; p = 2; ω2^(n/p) - 1
```

```
0
```

επειδή το 0 *είναι μηδενοδιαιρέτης κάθε δακτυλίου*.

Στην *περίπτωσή μας υπάρχουν συνολικά* $\varphi(8) = 4$

```
EulerPhi[8]
```

```
4
```

αρχέγονες ρίζες της μονάδας, τάξης 8, *και αν εξαιρέσουμε την* ω_1 *που έχουμε ήδη βρει οι υπόλοιπες* 3 *είναι οι εξής*:

```
{ω1^3, ω1^5, ω1^7} // ComplexExpand
```

$$\left\{-\frac{1-i}{\sqrt{2}},\ -\frac{1+i}{\sqrt{2}},\ \frac{1-i}{\sqrt{2}}\right\}$$

διότι gcd(3, 8) = gcd(5, 8) = gcd(7, 8) = 1. Πράγματι, *έχουμε για παράδειγμα*

```
Table[ω1^(3 k), {k, 0, 7}] // ComplexExpand
```

$$\left\{1, -\frac{1-i}{\sqrt{2}}, -i, \frac{1+i}{\sqrt{2}}, -1, \frac{1-i}{\sqrt{2}}, i, -\frac{1+i}{\sqrt{2}}\right\}$$

Αξίζει να σημειωθεί πως ο δακτύλιος $R = \mathbb{R}^{\times}\backslash\{0\}$, έχει αρχέγονες ρίζες **μόνον** τάξης 2, και πως, γενικά, δεν έχουν όλοι οι δακτύλιοι αρχέγονες ρίζες κάθε τάξης. Ισχύει το εξής:

Θεώρημα 4.6.1:
Έστω δακτύλιος R και $\ell, n \in \mathbb{N}$ έτσι ώστε $1 \le \ell < n$. Αν $\omega \in R$ είναι μία αρχέγονη ρίζα της μονάδας, τάξης n, τότε ισχύει:

$$\sum_{k=0}^{n-1} \omega^{\ell k} = 0.$$

Απόδειξη:
Επειδή ω είναι αρχέγονη ρίζα της μονάδας, τάξης n, έπεται πως είναι γεννήτρια όλων των άλλων ριζών της μονάδας. Συνεπώς, η έκφραση $\omega^{\ell} - 1$ δεν είναι μηδενοδιαιρέτης στο R για $1 \le \ell < n$.

Από την σχέση

$$\omega^{\ell n} = (\omega^n)^{\ell} = 1^{\ell} = 1$$

προκύπτει

$$\omega^{\ell n} - 1 = (\omega^{\ell} - 1)\sum_{k=0}^{n-1} (\omega^{\ell})^k = 0$$

και επειδή $\omega^{\ell} - 1$ δεν είναι μηδενοδιαιρέτης στο R έχουμε κατ' ανάγκη $\sum_{k=0}^{n-1} \omega^{\ell k} = 0$.//

Έστω δακτύλιος R, $n \in \mathbb{N}$ έτσι ώστε $1 \le n$ και έστω ότι $\omega \in R$ είναι μία αρχέγονη ρίζα της μονάδας, τάξης n. Στα επόμενα το πολυώνυμο $f = \sum_{i=0}^{n-1} f_i x^i \in R[x]$, βαθμού $< n$ θα παρίσταται από την λίστα των συντελεστών του $\{f_0, \ldots, f_{n-1}\} \in R^n$.

Ορισμός (DFT): Η γραμμική απεικόνιση (ισομορφισμός) $V_{\omega}: R^n \to R^n$, που **υπολογίζει τις τιμές ενός πολυωνύμου στις δυνάμεις του ω**, δηλαδή $V_{\omega}: \{f_0, \ldots,$

$f_{n-1}\} \to \frac{1}{\sqrt{n}}\{f(1), \ldots, f(\omega^{n-1})\}$, *ονομάζεται ο* ***διακριτός μετασχηματισμός Fourier*** (*Discrete Fourier Transform* or DFT).

Εξήγηση του όρου $\frac{1}{\sqrt{n}}$ δίνεται παρακάτω. Οι τιμές του πολυωνύμου στις δυνάμεις του ω, $\{f(1), \ldots, f(\omega^{n-1})\}$, συμβολίζονται και ως $\{\tilde{f}_0, \ldots, \tilde{f}_{n-1}\}$. Προσέξτε πως ο πίνακας V_ω που αντιστοιχεί στον μετασχηματισμό του Fourier είναι ο γνωστός μας πίνακας Vandermonde — χωρίς να δηλώνεται η τάξη του n. Δηλαδή

$$\{\tilde{f}_0, \ldots, \tilde{f}_{n-1}\}^{\mathrm{T}} = V_\omega \,.\, \{f_0, \ldots, f_{n-1}\}^{\mathrm{T}}$$

ή αναλυτικότερα

$$\begin{pmatrix} \tilde{f}_0 \\ \tilde{f}_1 \\ \tilde{f}_2 \\ \vdots \\ \tilde{f}_{n-1} \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega & \omega^2 & \cdots & \omega^{n-1} \\ 1 & \omega^2 & \omega^4 & \cdots & \omega^{2(n-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega^{n-1} & \omega^{2(n-1)} & \cdots & \omega^{(n-1)^2} \end{pmatrix} \begin{pmatrix} f_0 \\ f_1 \\ f_2 \\ \vdots \\ f_{n-1} \end{pmatrix}$$

Επειδή ω είναι μία αρχέγονη ρίζα της μονάδας, τάξης n, τα στοιχεία $1, \omega, \ldots, \omega^{n-1}$ είναι διαφορετικά μεταξύ τους και αν R είναι σώμα, ο πίνακας Vandermonde αντιστρέφεται και $(V_\omega)^{-1}$ είναι ο πίνακας της παρεμβολής για τα n σημεία. (Για την συνέχεια ενεργοποιήστε τους κώδικες για τους πίνακες Vandermonde στην αντίστοιχη ενότητα.)

Ορισμός (iDFT): Η γραμμική απεικόνιση $(V_\omega)^{-1}: R^n \to R^n$, που ***υπολογίζει την ακολουθία*** $\{f_0, \ldots, f_{n-1}\}$ ***από την ακολουθία*** $\{\tilde{f}_0, \ldots, \tilde{f}_{n-1}\}$, δηλαδή $(V_\omega)^{-1}$: $\{\tilde{f}_0, \ldots, \tilde{f}_{n-1}\} \to \frac{1}{\sqrt{n}}\{f_0, \ldots, f_{n-1}\}$, ονομάζεται ο ***αντίστροφος διακριτός μετασχηματισμός Fourier*** (*inverse Discrete Fourier Transform* or iDFT). Δηλαδή έχουμε

$$\{f_0, \ldots, f_{n-1}\}^{\mathrm{T}} = (V_\omega)^{-1} \,.\, \{\tilde{f}_0, \ldots, \tilde{f}_{n-1}\}^{\mathrm{T}}$$

Παράδειγμα:

Αν $\omega = i$ είναι η αρχέγονη τέταρτη ρίζα της μονάδας έχουμε τον πίνακα V_ω:

```
ω = i; V[ω_] := Vandermonde[{ω^0, ω^1, ω^2, ω^3}];
V[ω] // MatrixForm
```

$$\begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & i & -1 & -i \\ 1 & -1 & 1 & -1 \\ 1 & -i & -1 & i \end{pmatrix}$$

Με *την μέθοδο που περιγράψαμε στην ενότητα για τους πίνακες* Vandermonde *ο αντίστροφος του πίνακα* V_ω, *δηλαδή* $(V_\omega)^{-1}$, *είναι*

```
VInv[ω_] := inverseVandermonde[{ω^0, ω^1, ω^2, ω^3}];
VInv[ω] // MatrixForm
```

$$\begin{pmatrix} \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \\ \frac{1}{4} & -\frac{i}{4} & -\frac{1}{4} & \frac{i}{4} \\ \frac{1}{4} & -\frac{1}{4} & \frac{1}{4} & -\frac{1}{4} \\ \frac{1}{4} & \frac{i}{4} & -\frac{1}{4} & -\frac{i}{4} \end{pmatrix}$$

Στην *περίπτωση όμως που* ω *είναι μία αρχέγονη ρίζα της μονάδας, τάξης* n, *όπως στο παράδειγμά μας, ο αντίστροφος υπολογίζεται άμεσα από τον τύπο* $(V_\omega)^{-1} = \frac{1}{n} V_{\omega^{-1}}$ *οπότε*

$$\{f_0, \ldots, f_{n-1}\}^{\mathrm{T}} = \frac{1}{n} V_{\omega^{-1}} . \{\tilde{f}_0, \ldots, \tilde{f}_{n-1}\}^{\mathrm{T}}.$$

Αυτό *συνεπάγεται από το ακόλουθο*

Θεώρημα 4.6.2:

Έστω *ο μεταθετικός δακτύλιος* R, $n \in \mathbb{N}$ *έτσι ώστε* $n \geq 1$, *και* $\omega \in R$ *μία αρχέγονη ρίζα της μονάδας, τάξης* n. Τότε *ισχύει*

$$V_\omega . V_{\omega^{-1}} = nI,$$

όπου I *είναι ο ταυτοτικός πίνακας διαστάσεων* $n \times n$.

Απόδειξη:

Επειδή ω *είναι μία αρχέγονη ρίζα της μονάδας, τάξης* n, *ισχύει* $\omega^n = 1$ *και* $\omega^\ell \neq 1$

για $1 \le \ell \le n$ - 1. *Συνεπώς, αν για κάποιο* $1 \le \ell \le n$ - 1 *έχουμε* $(\omega^{-1})^{\ell} = (\omega^{\ell})^{-1} = 1 = 1^{-1}$, *τότε θα πρέπει να έχουμε αναγκαστκά και* $\omega^{\ell} = 1$, *πράγμα αδύνατο.*

Δηλαδή, αν το $\omega \in R$ *είναι μία αρχέγονη ρίζα της μονάδας, τάξης n, το ίδιο ισχύει και για το* ω^{-1} *και ο πίνακας* $V_{\omega^{-1}}$ *είναι μη μηδενικός.*

Γ*ια* $0 \le i,j \le n$ - 1, *το ij-στό στοιχείο του* $V_{\omega}.V_{\omega^{-1}}$ *είναι*

$$\sum_{k=0}^{n-1} \omega^{i\,k}\,\omega^{-\,k\,j} = \sum_{k=0}^{n-1} \omega^{k(i-j)} = \begin{cases} 0 & \gamma\iota\alpha\ i \neq j \\ n & \gamma\iota\alpha\ i = j \end{cases}$$

διότι αν $i = j$ *το στοιχείο* $(V_{\omega}.V_{\omega^{-1}})_{ij}$ *είναι* $\sum_{k=0}^{n-1} 1 = n$, *ενώ αν* $i \neq j$ *το στοιχείο* $(V_{\omega}.V_{\omega^{-1}})_{ij}$ *είναι* $\sum_{k=0}^{n-1} \omega^{k(i-j)} = 0$ *λόγω του* Θεωρήματος 4.6.1.//

Χ*ρειάζεται όμως προσοχή διότι αν υπολογίσουμε τον αντίστροφο μετασχηματισμό* Fourier *ενός διανύσματος κατ' αυτόν τον τρόπο θα βρούμε το αρχικό διάνυσμα πολλαπλασιασμένο επί n.* Το *πρόβλήμα αυτό λύνεται είτε διαιρώντας το αποτέλεσμα δια n είτε ενσωματώνοντας τον παράγοντα* $\frac{1}{\sqrt{n}}$ *στους μετασχηματισμούς — πρακτική που εφαρμόζεται στο Mathematica και τηρούμε.*

Π*αράδειγμα (συνέχεια του προηγούμενου*):
Ο *αντίστροφος του πίνακα* V_{ω} *είναι*

```
1/4 V[ω^-1] // MatrixForm
```

$$\begin{pmatrix} \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \\ \frac{1}{4} & -\frac{i}{4} & -\frac{1}{4} & \frac{i}{4} \\ \frac{1}{4} & -\frac{1}{4} & \frac{1}{4} & -\frac{1}{4} \\ \frac{1}{4} & \frac{i}{4} & -\frac{1}{4} & -\frac{i}{4} \end{pmatrix}$$

και ισχύει η ταυτότητα

```
V[ω].(1/4 V[ω^-1]) == IdentityMatrix[4]
```

```
True
```

Στην συνέχεια παρουσιάζουμε ένα πολύ σημαντικό αλγόριθμο — τον γρήγορο μετασχηματισμό Fourier ή FFT — που υπολογίζει τον διακριτό μετασχηματισμό Fourier πολύ αποτελεσματικά.

Ο γρήγορος μετασχηματισμός Fourier είναι ένας από τους πιο πολυχρησημοποιούμενους αλγόριθμους. Αναπτύχθηκε από τους Cooley και Tukey το 1965, αλλά η αρχική ανακάλυψη (1805) οφείλεται στον Γκάους (Gauss) — δηλαδή πριν από την ανακάλυψη του Fourier το 1807. (Όμως η εργασία του Γκάους δημοσιεύθηκε μετά τον θάνατό του.) Ενδιαφέρον έχει και το ιστορικό άρθρο στην βιβλιογραφία.

Είναι τόσο σημαντικός ο αλγόριθμος αυτός που τον παρουσιάζουμε με δύο διαφορετικούς τρόπους: πρώτα με παραγοντοποίηση πινάκων και μετά με συνέλιξη πολυωνύμων.

■ 4.6.1 Ο γρήγορος μετασχηματισμός Fourier (FFT) — με παραγοντοποίηση πινάκων

Στην υποενότητα αυτή θα θεωρήσουμε το πρόβλημα όπου πολλαπλασιάζουμε τον $n \times n$ πίνακα V_ω με μία οποιαδήποτε ακολουθία n στοιχείων, $n \in \mathbb{N}$. Μία αποτελεσματική λύση στο πρόβλημα αυτό συνεπάγεται έναν γρήγορο αλγόριθμο για τον υπολογισμό τόσο του μετασχηματισμού Fourier όσο και του αντιστρόφου του.

Ο κλασσικός αλγόριθμος πολλαπλασιασμού του $n \times n$ πίνακα V_ω με ένα διάνυσμα n στοιχείων απαιτεί $O(n^2)$ πράξεις. Τόσες πράξεις όμως απαιτούνται και από τις προηγούμενες μεθόδους παρεμβολής. Μπορούμε όμως να παραγοντοποιήσουμε τον πίνακα V_ω σε ένα γινόμενο από $\log n$ πίνακες, όπου κάθε ένας έχει μόνο δύο μη μηδενικά στοιχεία σε κάθε σειρά. Ο πολλαπλασιασμός του διανύσματος με έναν από αυτούς τους πίνακες απαιτεί $O(n)$ πράξεις και με όλους αυτούς τους πίνακες απαιτεί $O(n \log n)$ πράξεις.

Για να εξηγήσουμε την παραγοντοποίηση, ***συμβολίζουμε τώρα*** τον $n \times n$ πίνακα V_ω με δείκτη την διάσταση n, σαν V_n. Στο Mathematica ορίζουμε την συνάρτηση

```
V[n_] := Vandermonde[Table[ω^k, {k, 0, n - 1}]];
```

και για $\omega = -1$, *την αρχέγονη ρίζα της μονάδας, τάξης* 2, *έχουμε*

```
ω = -1; V[2] // MatrixForm
```

$$\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

ενώ για $\omega = i$, *την αρχέγονη ρίζα της μονάδας, τάξης* 4, *έχουμε*

```
ω = i; V[4] // MatrixForm
```

$$\begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & i & -1 & -i \\ 1 & -1 & 1 & -1 \\ 1 & -i & -1 & i \end{pmatrix}$$

Ας *συμβολίσουμε με* P_4 *τον* (permutation) *πίνακα που αντιμεταθέτει τις μεσαίες στήλες του* V_4. Ο *πίνακας* P_4 *είναι*

```
P4 = {{1, 0, 0, 0}, {0, 0, 1, 0}, {0, 1, 0, 0}, {0, 0, 0, 1}};
```

και έστω το γινόμενό του με τον παραπάνω πίνακα, $V_4.P_4$

```
ω = i; V[4].P4 // MatrixForm
```

$$\begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & i & -i \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -i & i \end{pmatrix}$$

Προσέξτε *πως στον πίνακα* $V_4.P_4$, *μετρώντας από το* 0, *οι δύο πρώτες στήλες είναι οι* ***άρτιες*** *στήλες του* V_4, *ενώ οι τελευταίες δύο στήλες είναι οι* ***περιττές*** *στήλες του* V_4.

Παρατηρούμε *πως ισχύει η εξής παραγοντοποίηση*:

$$V_4.P_4 = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & i & -i \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -i & i \end{pmatrix} = \begin{pmatrix} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} & \begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix}.\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \\ \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} & \begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix}.\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \end{pmatrix}$$

$$= \begin{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} & \begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix} \\ \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} & -\begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix} \end{pmatrix}.\begin{pmatrix} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} & \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix} \\ \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix} & \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \end{pmatrix}.$$

Αν δε συμβολίσουμε με I_n τον ταυτοτικό πίνακα τάξης n, και με W_n τον διαγώνιο πίνακα τάξης n, με διαγώνια στοιχεία $1, \omega, \omega^2, \ldots, \omega^{n-1}$ τότε η παραγοντοποίηση του $V_4.P_4$ παίρνει την μορφή (προσέξτε πως στον πίνακα W_n το ω είναι μία αρχέγονη ρίζα της μονάδας, **τάξης 2n**)

$$V_4.P_4 = \begin{pmatrix} I_2 & W_2 \\ I_2 & -W_2 \end{pmatrix}.\begin{pmatrix} V_2 & 0 \\ 0 & V_2 \end{pmatrix} = \begin{pmatrix} V_2 & W_2 V_2 \\ V_2 & -W_2 V_2 \end{pmatrix}.$$

Ο χωρισμός ενός πίνακα σε υποπίνακες και η επανασύστασή του, επιτυγχάνεται με την βοήθεια των παρακάτω προγραμμάτων:

```
blockMatrix[matrix_?MatrixQ, blocksize_: {2, 2}] :=
 Partition[matrix, blocksize] /;
  Apply[And, Map[IntegerQ, (Dimensions[matrix] / blocksize)]]
unBlockTensor[blocks_] :=
 Partition[Flatten[Transpose[blocks, {1, 3, 2}]],
  {Apply[Times, Dimensions[blocks][[{2, 4}]]]}]
```

Τονίζουμε πως στους πίνακες $\mathbf{V_n}$ και $\mathbf{W_{n/2}}$ η αρχέγονη ρίζα της μονάδας είναι **τάξης n**.

Έτσι, αφού ορίσουμε τους πίνακες I_n και W_n με τις συναρτήσεις

```
Id[n_] := IdentityMatrix[n]
W[n_] := DiagonalMatrix[Table[ω^k, {k, 0, n - 1}]]
```

επιβεβαιώνουμε την ταυτότητα $V_4.P_4 = \begin{pmatrix} I_2 & W_2 \\ I_2 & -W_2 \end{pmatrix}.\begin{pmatrix} V_2 & 0 \\ 0 & V_2 \end{pmatrix}$

```
ω =.; (V[4].P4 /. {ω → e^(π i/2)} // N) ==
  unBlockTensor[( Id[2]   W[2] /. {ω → e^(π i/2)} )].
                  Id[2]  -W[2] /. {ω → e^(π i/2)}
   unBlockTensor[( V[2] /. {ω → -1}  Id[2] /. {1 → 0} )] // N
                   Id[2] /. {1 → 0}  V[2] /. {ω → -1}
```

```
True
```

Στην γενική περίπτωση έχουμε το εξής θεώρημα το οποίο βασικά είναι ο αλγόριθμος FFT.

Θεώρημα 4.6.3:

Έστω δακτύλιος R, $n = 2m$, όπου $m \in \mathbb{N}$, και έστω P_n ο (permutation) πίνακας που αντιμεταθέτει τις στήλες του V_n έτσι ώστε οι πρώτες m στήλες του $V_n.P_n$ είναι οι άρτιες στήλες του V_n και οι τελευταίες m στήλες είναι οι περιττές στήλες του V_n. Τότε

$$V_n.P_n = \begin{pmatrix} I_m & W_m \\ I_m & -W_m \end{pmatrix}.\begin{pmatrix} V_m & 0 \\ 0 & V_m \end{pmatrix} = \begin{pmatrix} V_m & W_m V_m \\ V_m & -W_m V_m \end{pmatrix}$$

Απόδειξη:

Έστω $(V_n.P_n)_{\mathrm{ij}}$ το στοιχείο ij του πίνακα $V_n.P_n$, και έστω ω_n μια αρχέγονη ρίζα της μονάδας, τάξης n. Θα δείξουμε πως ισχύει η ισότητα αυτή των πινάκων αποδεικνύοντας την ισότητα των στοιχείων για κάθε ένα τεταρτημόριο χωριστά. Στο ***πρώτο τεταρτημόριο*** του $V_n.P_n$, με τις **άρτιες** στήλες του V_n, έχουμε

$$(V_n.P_n)_{\mathrm{ij}} = \omega_n^{i(2j)} = \omega_m^{\mathrm{ij}} = (V_m)_{\mathrm{ij}}$$

επειδή από $\omega_n^n = \omega_n^{2m} = 1$ έπεται $\omega_n^2 = \omega_m$. Στο ***δεύτερο τεταρτημόριο*** του $V_n.P_n$, με τις **άρτιες** στήλες του V_n, έχουμε

$$(V_n.P_n)_{i+m,j} = \omega_n^{(i+m)(2j)} = \omega_m^{(i+m)j} = \omega_m^{\mathrm{ij}} = (V_m)_{\mathrm{ij}}$$

επειδή $\omega_m^m = 1$. Στο ***τρίτο τεταρτημόριο*** του $V_n.P_n$, με τις **περιττές** στήλες του V_n, έχουμε

$$(V_n.P_n)_{i,j+m} = \omega_n^{i(2j+1)} = \omega_n^i \omega_m^{\mathrm{ij}} = (W_m V_m)_{\mathrm{ij}}$$

επειδή το διαγώνιο στοιχείο της i-στής σειράς του πίνακα W_m είναι ω_n^i. Τέλος, στο **τέταρτο τεταρτημόριο** του $V_n.P_n$, με τις **περιττές** στήλες του V_n, έχουμε

$$(V_n.P_n)_{i+m,j+m} = \omega_n^{(i+m)(2j+1)} = \omega_n^{i+m}\omega_n^{i(2j)} = -\omega_n^i\omega_m^{ij} = (-W_m V_m)_{ij}$$

επειδή από $\omega_n^n = \omega_n^{2m} = 1$ έπεται $\omega_n^m = -1$ .//

Ας δούμε τώρα το **κόστος υπολογισμού** του μετασχηματισμού Fourier ενός τυχαίου διανύσματος με n στοιχεία. Αν χρησιμοποιήσουμε τον πίνακα V_n τότε, όπως ξέρουμε, απαιτούνται n^2 πολλαπλασιασμοί. Αν όμως χρησιμοποιήσουμε την ανάλυσή του $V_n.P_n$ στους πίνακες $\begin{pmatrix} I_m & W_m \\ I_m & -W_m \end{pmatrix}.\begin{pmatrix} V_m & 0 \\ 0 & V_m \end{pmatrix}$ τότε απαιτούνται $\frac{n^2}{2}$ + 2n πολλαπλασιασμοί, σχεδόν οι μισοί, διότι

▪ ο πίνακας $\begin{pmatrix} I_m & W_m \\ I_m & -W_m \end{pmatrix}$, που λέγεται "**πεταλούδα**" (butterfly), έχει σε κάθε σειρά δύο μη μηδενικά στοιχεία και επομένως το γινόμενο με ένα διάνυσμα απαιτεί 2n πολλαπλασιασμούς.

▪ ο πίνακας $\begin{pmatrix} V_m & 0 \\ 0 & V_m \end{pmatrix}$ έχει σε κάθε σειρά $\frac{n}{2}$ μη μηδενικά στοιχεία και επομένως το γινόμενο με ένα διάνυσμα απαιτεί $\frac{n^2}{2}$ πολλαπλασιασμούς.

Η διαδικασία μπορεί να επαναληφθεί εισαγάγοντας έναν νέο πίνακα $P'_{n/2}$ που έχει δύο αντίγραφα του πίνακα $P_{n/2}$ και δρα στα δύο αντίγραφα του πίνακα V_m — στον $\begin{pmatrix} V_m & 0 \\ 0 & V_m \end{pmatrix}$ — αντιμεταθέτοντας τις στήλες τους. Αυτό που προκύπτει είναι η εξής παραγοντοποίηση του πίνακα $V_n.P_n.P'_{n/2}$:

$$V_n.P_n.P'_{n/2} =$$
$$\begin{pmatrix} I_{\frac{n}{2}} & W_{\frac{n}{2}} \\ I_{\frac{n}{2}} & -W_{\frac{n}{2}} \end{pmatrix}.\begin{pmatrix} I_{\frac{n}{4}} & W_{\frac{n}{4}} & 0 & 0 \\ I_{\frac{n}{4}} & -W_{\frac{n}{4}} & 0 & 0 \\ 0 & 0 & I_{\frac{n}{4}} & W_{\frac{n}{4}} \\ 0 & 0 & I_{\frac{n}{4}} & -W_{\frac{n}{4}} \end{pmatrix}.\begin{pmatrix} V_{\frac{n}{4}} & 0 & 0 & 0 \\ 0 & V_{\frac{n}{4}} & 0 & 0 \\ 0 & 0 & V_{\frac{n}{4}} & 0 \\ 0 & 0 & 0 & V_{\frac{n}{4}} \end{pmatrix}$$

που ελαττώνει τον αριθμό των πολλαπλασιασμών (σχεδόν) ακόμα κατά ήμισυ. Προσέξτε επίσης πως οι δύο πρώτοι πίνακες έχουν μόνο δύο μη μηδενικά στοιχεία σε κάθε σειρά. Επαναλαμβάνοντας την διαδικασία αυτή log n φορές προκύπτει

ένας (τελικός) πίνακας που είναι εντελώς διαγώνιος. Αυτή είναι η βασική ιδέα του FFT.

Στον αλγόριθμο του Θεωρήματος 4.6.3 μπορούμε να αντιμεταθέσουμε τις στήλες μία φορά από την αρχή. Το αποτέλεσμα είναι ένα ενδιαφέρον σχέδιο. Ας δούμε για παράδειγμα την εφαρμογή του τύπου του Θεωρήματος 4.6.3 **δύο φορές** σε έναν 8×8 πίνακα. Αναφερόμαστε μόνον στις στήλες του πίνακα V_8 που τις παριστάνουμε ως Σ_i, δηλαδή έχουμε $V_8 = (\Sigma_0, \Sigma_1, \Sigma_2, \Sigma_3, \Sigma_4, \Sigma_5, \Sigma_6, \Sigma_7)$.

Την πρώτη φορά οι στήλες του V_8 αντιμεταθέτονται και έχουμε

$$V_8.P_8 = (\Sigma_0, \Sigma_2, \Sigma_4, \Sigma_6, \Sigma_1, \Sigma_3, \Sigma_5, \Sigma_7).$$

Κατόπιν έχουμε τον πίνακα P_4 που δρα ξεχωριστά στις πρώτες 4 στήλες και στις τελευταίες 4 στήλες:

$$(\Sigma_0, \Sigma_2, \Sigma_4, \Sigma_6).P_4 = (\Sigma_0, \Sigma_4, \Sigma_2, \Sigma_6)$$
$$(\Sigma_1, \Sigma_3, \Sigma_5, \Sigma_7).P_4 = (\Sigma_1, \Sigma_5, \Sigma_3, \Sigma_7)$$

έτσι ώστε το τελικό αποτέλεσμα είναι

$$V_8.P_8.P_4' = (\Sigma_0, \Sigma_4, \Sigma_2, \Sigma_6, \Sigma_1, \Sigma_5, \Sigma_3, \Sigma_7).$$

Αυτό είναι μια διάταξη που προκύπτει από "***αναστροφή των δυαδικών ψηφίων***" (bit reversal) των δεικτών των στηλών.

Αναλυτικά, οι δείκτες των στηλών $\Sigma_0, \Sigma_2, \Sigma_4, \Sigma_6, \Sigma_1, \Sigma_3, \Sigma_5, \Sigma_7$ είναι

```
n = 3; Table[i, {i, 0, 2^n - 1}]
```

```
{0, 1, 2, 3, 4, 5, 6, 7}
```

ή σε δυαδική μορφή

```
indeces = Map[(dig = IntegerDigits[#, 2]; digL = Length[dig];
    If[digL < n, Flatten[PrependTo[dig, Table[0, {i, n - digL}]]],
     IntegerDigits[#, 2]]) &, Table[i, {i, 0, 2^n - 1}]]
```

```
{{0, 0, 0}, {0, 0, 1}, {0, 1, 0}, {0, 1, 1},
 {1, 0, 0}, {1, 0, 1}, {1, 1, 0}, {1, 1, 1}}
```

Αντιστρέφοντας τα δυαδικά ψηφία κάθε δείκτη

```
indecesReversed = Map[Reverse, indeces]
```

```
{{0, 0, 0}, {1, 0, 0}, {0, 1, 0}, {1, 1, 0},
 {0, 0, 1}, {1, 0, 1}, {0, 1, 1}, {1, 1, 1}}
```

και επαναφέροντας τους δείκτες στην βάση 10

```
Map[FromDigits[#, 2] &, indecesReversed]
```

```
{0, 4, 2, 6, 1, 5, 3, 7}
```

παίρνουμε την διάταξη των στηλών $\Sigma_0, \Sigma_4, \Sigma_2, \Sigma_6, \Sigma_1, \Sigma_5, \Sigma_3, \Sigma_7$.

Η "αναστροφή των δυαδικών ψηφίων" είναι το πρώτο που γίνεται στα στοιχεία ενός διανύσματος όταν θέλουμε να υπολογίσουμε τον μετασχηματισμό Fourier του. Το παράδειγμα που ακολουθεί εξηγεί τις λεπτομέρειες.

Παράδειγμα:
Χρησιμοποιώντας το Θεώρημα 4.6.3 θα βρούμε τον μετασχηματισμό Fourier του διανύσματος {0, 1, 2, 3, 4, 5, 6, 7} με 8 στοιχεία. Τονίζουμε πως ο αριθμός των στοιχείων του διανύσματος πρέπει να είναι δύναμη του 2.

Ο άμεσος τρόπος είναι να πολλαπλασιάσουμε το διάνυσμα επί τον πίνακα Vandermonde τάξης 8. Πράγματι έχουμε

```
ω = e^(2 π i/8);
fTran = V[8].{0, 1, 2, 3, 4, 5, 6, 7} // N
```

```
{28., -4. - 9.65685 i, -4. - 4. i, -4. - 1.65685 i,
 -4., -4. + 1.65685 i, -4. + 4. i, -4. + 9.65685 i}
```

Τονίζουμε πως ο αντίστροφος μετασχηματισμός με την βοήθεια του Θεωρήματος 4.6.2 μας δίνει το αρχικό διάνυσμα πολλαπλασιασμένο επί 8

```
ω = (e^(2 π i/8))^-1;
V[8].fTran // Chop // Rationalize
```

```
{0, 8, 16, 24, 32, 40, 48, 56}
```

και πως, επιπλέον, έχουμε

$$\frac{\texttt{fTran}}{\sqrt{8}} == \texttt{Fourier[\{0, 1, 2, 3, 4, 5, 6, 7\}]}$$

```
True
```

Όμως, ο παραπάνω τρόπος υπολογισμού του μετασχηματισμού Fourier απαιτεί $O(n^2)$ πολλαπλασιασμούς. Για να τους ελαττώσουμε στο $O(n \log n)$ πρέπει να παραγοντοποιήσουμε τον πίνακα $V[8]$ με βάση το Θεώρημα 4.6.3. Δηλαδή πρέπει να υπολογίσουμε το ανάπτυγμα

$$V_8.P_8.P_4' = \begin{pmatrix} I_4 & W_4 \\ I_4 & -W_4 \end{pmatrix}.\begin{pmatrix} \begin{pmatrix} I_2 & W_2 \\ I_2 & -W_2 \end{pmatrix} & 0 \\ 0 & \begin{pmatrix} I_2 & W_2 \\ I_2 & -W_2 \end{pmatrix} \end{pmatrix}.\begin{pmatrix} V_2 & 0 & 0 & 0 \\ 0 & V_2 & 0 & 0 \\ 0 & 0 & V_2 & 0 \\ 0 & 0 & 0 & V_2 \end{pmatrix}$$

Για τον σκοπό αυτό πρέπει:

(**Α**) να υπενθυμίσουμε τον ορισμό των πινάκων I_n, και W_n, και να ορίσουμε αφ' ενός μεν τους μηδενικούς πίνακες z_n αφ' ετέρου δε τους πίνακες "πεταλούδα" $B_n = \begin{pmatrix} I_n & W_n \\ I_n & -W_n \end{pmatrix}$.

```
ω =.;
Id[n_] := IdentityMatrix[n];
W[n_] := DiagonalMatrix[Table[ω^k, {k, 0, n - 1}]]
z[n_] := IdentityMatrix[n] /. {1 → 0}
B[n_] := unBlockTensor[( Id[n]   W[n]
                         Id[n]  -W[n] )]
```

και (**B**) να ορίσουμε τους πίνακες: ***β*1:** P_8 (ο P_4 έχει ήδη ορισθεί)

$$\mathbf{P_8} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix};$$

η δράση και χρήση του οποίου φαίνεται από το παράδειγμα

```
{0, 1, 2, 3, 4, 5, 6, 7}.P8
```

```
{0, 2, 4, 6, 1, 3, 5, 7}
```

και ***β*2:** P_4' ο οποίος είναι ένας 8×8 πίνακας που περιέχει δύο αντίγραφα του P_4 για να δρα στα αντίγραφα των πινάκων V_4.

$$\mathbf{PP_4} = \mathbf{unBlockTensor}\left[\begin{pmatrix} \mathbf{P_4} & \mathbf{z[4]} \\ \mathbf{z[4]} & \mathbf{P_4} \end{pmatrix}\right];$$

Με τους ορισμούς αυτούς, κάθε ένας από τους 3 πίνακας του αναπτύγματος του $V_8.P_8.P_4'$ έχει δύο μη μηδενικά στοιχεία σε κάθε σειρά. Έτσι τώρα σχηματίζουμε τον πρώτο πίνακα με τα 4 αντίγραφα του V_2 (***προσέξτε*** την ***αρχέγονη ρίζα*** της μονάδας — είναι τάξης 2!)

$$\omega = -1;\ \mathbf{mat1} = \mathbf{unBlockTensor}\left[\begin{pmatrix} \mathbf{V[2]} & \mathbf{z[2]} & \mathbf{z[2]} & \mathbf{z[2]} \\ \mathbf{z[2]} & \mathbf{V[2]} & \mathbf{z[2]} & \mathbf{z[2]} \\ \mathbf{z[2]} & \mathbf{z[2]} & \mathbf{V[2]} & \mathbf{z[2]} \\ \mathbf{z[2]} & \mathbf{z[2]} & \mathbf{z[2]} & \mathbf{V[2]} \end{pmatrix}\right];$$

κάνουμε "αναστροφή των δυαδικών ψηφίων" στο αρχικό διάνυσμα

```
bitReversed = P8.PP4.{0, 1, 2, 3, 4, 5, 6, 7}
```

```
{0, 4, 2, 6, 1, 5, 3, 7}
```

και υπολογίζουμε το πρώτο ενδιάμεσο αποτέλεσμα που είναι

```
firstResult = mat1.bitReversed
```

```
{4, -4, 8, -4, 6, -4, 10, -4}
```

Κατόπιν *σχηματίζουμε τον δεύτερο πίνακα με τα* 2 *αντίγραφα του* V_4 *τα οποία έχουν αντικατασταθεί από τους πίνακες "πεταλούδα"* $\begin{pmatrix} I_2 & W_2 \\ I_2 & -W_2 \end{pmatrix}$ (***προσέξτε*** *την* ***αρχέγονη ρίζα*** *της μονάδας — είναι τάξης* 4!)

```
ω = e^(π i/2);
mat2 = unBlockTensor[(B[2]  z[4]
                      z[4]  B[2])];
```

και υπολογίζουμε το δεύτερο ενδιάμεσο αποτέλεσμα που είναι

```
secondResult = mat2.firstResult
```

```
{12, -4 - 4 i, -4, -4 + 4 i, 16, -4 - 4 i, -4, -4 + 4 i}
```

Τέλος, *σχηματίζουμε τον τρίτο πίνακα "πεταλούδα"* B_4 (*με* ***αρχέγονη ρίζα*** *της μονάδας, τάξης* 8!)

```
ω = e^(2 π i/8); mat3 = B[4];
```

και υπολογίζουμε το τελικό αποτέλεσμα που είναι

```
finalResult = mat3.secondResult // N
```

```
{28., -4. - 9.65685 i, -4. - 4. i, -4. - 1.65685 i,
 -4., -4. + 1.65685 i, -4. + 4. i, -4. + 9.65685 i}
```

Πολλαπλασιάζοντάς *το με* $\frac{1}{\sqrt{8}}$ *παίρνουμε το ίδιο αποτέλεσμα με αυτό που δίνει το Mathematica*

```
1/√8 finalResult == Fourier[{0, 1, 2, 3, 4, 5, 6, 7}]
```

```
True
```

Όπως έχουμε δει, αν $\omega = e^{2\pi i/8}$ είναι μία αρχέγονη ρίζα της μονάδας, τάξης 8, τότε $\omega^2 = e^{\pi i/2}$ είναι μία αρχέγονη ρίζα της μονάδας, τάξης 4, και $(\omega^2)^2 = \omega^4 = e^{\pi i}$ είναι μία αρχέγονη ρίζα της μονάδας, τάξης 2.

Ένας άλλος τρόπος για να κάνουμε το παράδειγμα βασίζεται στον πολλαπλασιασμό διανυσμάτων με πίνακες "πεταλούδα". Ο τρόπος αυτός ταιριάζει με τις hardware εφαρμογές, και είναι ο εξής:

Αρχίζουμε κάνοντας "αναστροφή των δυαδικών ψηφίων" και το διάνυσμα γίνεται {0, 4, 2, 6, 1, 5, 3, 7}.

```
vector = {0, 4, 2, 6, 1, 5, 3, 7};
```

Κατόπιν, για $r = 1$, πολλαπλασιαζουμε, ανά δύο, τα στοιχεία του διανύσματος επί τον πίνακα $B_1 = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$. Δηλαδή έχουμε:

```
vectorPairs1 = Partition[vector, 2]
```

```
{{0, 4}, {2, 6}, {1, 5}, {3, 7}}
```

οπότε το πρώτο αποτέλεσμα είναι

```
output1 = Map[Dot[B[1], #] &, vectorPairs1]
```

```
{{4, -4}, {8, -4}, {6, -4}, {10, -4}}
```

Για $r = 2$, πολλαπλασιαζουμε, ανά τέσσερα, τα στοιχεία του πρώτου αποτελέσματος επί τον πίνακα $B_2 = \begin{pmatrix} I_2 & W_2 \\ I_2 & -W_2 \end{pmatrix}$. Δηλαδή έχουμε:

```
vectorPairs2 = Map[Flatten, Partition[output1, 2]]
```

```
{{4, -4, 8, -4}, {6, -4, 10, -4}}
```

οπότε το δεύτερο αποτέλεσμα είναι

```
ω = e^(π i/2); output2 = Map[Dot[B[2], #] &, vectorPairs2] // Flatten // N
```

```
{12., -4. - 4. i, -4., -4. + 4. i, 16., -4. - 4. i, -4., -4. + 4. i}
```

Γ*ια* $r = 3$, *πολλαπλασιαζουμε τα στοιχεία του δεύτερου αποτελέσματος επί τον πίνακα* $B_4 = \begin{pmatrix} I_4 & W_4 \\ I_4 & -W_4 \end{pmatrix}$ *και έχουμε το τελικό αποτέλεσμα*

```
ω = e^(2 π i/8);
output3 = 1/Sqrt[8] B[4].output2 // N // Chop
```

```
{9.89949, -1.41421 - 3.41421 i, -1.41421 - 1.41421 i,
 -1.41421 - 0.585786 i, -1.41421, -1.41421 + 0.585786 i,
 -1.41421 + 1.41421 i, -1.41421 + 3.41421 i}
```

που είναι το ίδιο με αυτό που δίνει το Mathematica

```
output3 == Fourier[{0, 1, 2, 3, 4, 5, 6, 7}]
```

```
True
```

Β*ασιζόμενοι στον στον πολλαπλασιασμό διανυσμάτων με πίνακες* "*πεταλούδα*" *ο αλγόριθμος είναι ως εξής*:

Αλγόριθμος: Γρήγορος μετασχηματισμός Fourier (FFT) — με πίνακες "πεταλούδα"

Είσοδος: Το *διάνυσμα* $\{f_0, \ldots, f_{n-1}\} \in R^n$, $n = 2^k \in \mathbb{N}$, *το οποίο θεωρείται και η λίστα των συντελεστών του πολυωνύμου* $f = \sum_{i=0}^{n-1} f_i x^i \in R[x]$, *βαθμού* $< n$. Υπάρχει $\omega \in R$, *μία ρίζα της μονάδας, τάξης* n.
Έξοδος: $V_\omega(f) = \frac{1}{\sqrt{n}} \{f(1), \ldots, f(\omega^{n-1})\} \in R^n$.

==========

1. Κάνουμε *αναδιάταξη των στοιχείων του διανύσματος*. Το *πρώτο αποτέλεσμα*

προκύπτει πολλαπλασιάζοντας επί τον πίνακα "πεταλούδα" B[1] *τα στοιχεία του νέου διανύσματος ανά ζεύγη.*

2. for $2 \le i < \log_2 n$ **do** {*χωρίζουμε το αποτελέσματα σε ομάδες των* 2^i *στοιχείων, ορίζουμε την ρίζα της μονάδας τάξης* 2^i, *τον αντίστοιχο πίνακα "πεταλούδα"* $B[2^{i-1}]$ *και πολλαπλασιάζουμε με αυτόν κάθε ομάδα των* 2^i *στοιχείων για να σχηματίσουμε το νέο αποτέλεσμα.*}.

3. Τέλος, *αν* $\log_2 n > 1$, *ορίζουμε την ρίζα της μονάδας τάξης* $2^{\log_2 n}$, *και πολλαπλασιάζουμε το αποτέλεσμα επί τον αντίστοιχο πίνακα "πεταλούδα"* $\frac{1}{\sqrt{n}} B[2^{\log_2 n - 1}]$ *για να σχηματίσουμε το τελικό αποτέλεσμα. Σε περίπτωση που* $\log_2 n = 1$, *πολλαπλασιάζουμε το αποτέλεσμα επί τον ταυτοτικό πίνακα* $\frac{1}{\sqrt{2}} \mathrm{Id}[2]$ *για να σχηματίσουμε το τελικό αποτέλεσμα.*

==========

Για *την περίπτωση* $\boldsymbol{R} = \mathbb{C}$ *έχουμε το παρακάτω πρόγραμμα στο Mathematica. Προσέξτε πως για να λειτουργήσει το πρόγραμμα αυτό εκτός από τον ορισμό των πινάκων "πεταλούδα" χρειάζεται και ένα πρόγραμμα για την αναστροφή των δυαδικών ψηφίων* (*υπάρχει στις ασκήσεις*).

```
fourierMatrixFactorization[list_] /;
  Log[2, Length[list]] ∈ Integers :=
 Module[
  {i, n = Length[list], output, resuffList, vectorTupples},
  resuffList = resuffleList[list];
  vectorTupples = Partition[resuffList, 2];
  output = Map[Dot[B[1], #] &, vectorTupples] // Flatten // N;
  For[i = 2, i < Log[2, n], i++,
   vectorTupples = Map[Flatten, Partition[output, 2^i]];
   ω = ℯ^(2 π i /2^i);
   output = Flatten[Map[Dot[B[2^(i-1)], #] &, vectorTupples]] // N;
  ]; If[Log[2, n] > 1, ω = ℯ^(2 π i /n);
   1/√n B[2^(Log[2,n]-1)].output // N // Chop, 1/√n Id[2].output]
 ]
```

Έτσι για το διάνυσμα $\{0, 1, \ldots, 15\}$ βρίσκουμε

```
fourierMatrixFactorization[Table[i, {i, 0, 2^3 - 1}]]
```

```
{9.89949, -1.41421 - 3.41421 i, -1.41421 - 1.41421 i,
 -1.41421 - 0.585786 i, -1.41421, -1.41421 + 0.585786 i,
 -1.41421 + 1.41421 i, -1.41421 + 3.41421 i}
```

που είναι το ίδιο με το αποτέλεσμα που βρίσκουμε χρησιμοποιώντας την συνάρτηση `Fourier[]` του Mathematica

```
fourierMatrixFactorization[Table[i, {i, 0, 2^3 - 1}]] ==
 Fourier[Table[i, {i, 0, 2^3 - 1}]]
```

```
True
```

■ 4.6.2 Ο γρήγορος μετασχηματισμός Fourier (FFT) — με "συνέλιξη" πολυωνύμων

Όπως και πριν, R είναι δακτύλιος, $n \in \mathbb{N}_{\geq 1}$, και το πολυώνυμο $f = \sum_{i=0}^{n-1} f_i x^i \in R[x]$, βαθμού $< n$ παρίσταται από την λίστα των συντελεστών του $\{f_0, \ldots, f_{n-1}\} \in R^n$.

Η ***συνέλιξη των δύο πολυωνύμων***, (convolution) $f = \sum_{i=0}^{n-1} f_i x^i$ και $g = \sum_{j=0}^{n-1} g_j x^j$ στο $R[x]$ είναι το πολυώνυμο

$$h = f *_n g = \sum_{k=0}^{n-1} h_k x^k \in R[x]$$

όπου

$$h_k = \sum_{i+j \equiv k \bmod n} f_i g_j = \sum_{i=0}^{n-1} f_i g_{k-i} \quad \text{για } 0 \leq k < n,$$

και η αριθμητική στους δείκτες γίνεται modulo n. Αν θεωρήσουμε τους συντελεστές των πολυωνύμων σαν διανύσματα στο R^n τότε κάνουμε λόγο για την ***κυκλική συνέλιξη*** των διανυσμάτων f και g.

Ισχύει το εξής:

Θεώρημα 4.6.4:

Έστω R δακτύλιος, $n \in \mathbb{N}$ έτσι ώστε $n \geq 1$, και έστω $\omega \in R$ μία αρχέγονη ρίζα της μονάδας, τάξης n. Για τα πολυώνυμα $f, g \in R[x]$ βαθμού $\leq n - 1$, ισχύει

$$V_\omega(f *_n g) = V_\omega(f) \bullet V_\omega(g)$$

όπου $\bullet$ δηλώνει τον "στοιχείο προς στοιχείο" πολλαπλασιασμό διανυσμάτων.

Απόδειξη:

Ξέρουμε πως $f *_n g = fg + q(x^n - 1)$ για κάποιο $q \in R[x]$. Οπότε έχουμε

$$(f *_n g)(\omega^i) = f(\omega^i)g(\omega^i) + q(\omega^i)(\omega^{i\,n} - 1) = f(\omega^i)g(\omega^i)$$

για $0 \leq i \leq n - 1$.//

Παράδειγμα:

Έστω $n = 4$, $f = x^3 - 7x + 7$ και $g = 3x^3 + 2x^2 + x + 1$. Τότε η κυκλική συνέλιξη των πολυωνύμων f και g (ή των διανυσμάτων $\{7, -7, 0, 1\}$ και $\{1, 1, 2, 3\}$) είναι το πολυώνυμο

```
n = 4; fCoef = {7, -7, 0, 1}; gCoef = {1, 1, 2, 3};
powersX = Table[x^i, {i, 0, n - 1}];
powersX.ListConvolve[fCoef, gCoef, {1, 1}]
```

```
-13 + 2 x + 10 x^2 + 8 x^3
```

η το διάνυσμα

```
Table[Sum[fCoef[[i + 1]] gCoef[[Mod[k - i, n] + 1]], {i, 0, n - 1}], {k, 0, 3}]
```

```
{-13, 2, 10, 8}
```

*που προκύπτει ισοδύναμα **είτε** χρησιμοποιώντας την συνάρτηση* `ListConvolve[]` *του Mathematica*

```
ListConvolve[fCoef, gCoef, {1, 1}]
```

```
{-13, 2, 10, 8}
```

είτε *παίρνοντας, διαδοχικά, το εσωτερικό γινόμενο της πρώτης σειράς του παρακάτω πίνακα με κάθε μία από τις επόμενες.* Προσέξτε πως στον πίνακα έχουμε ***αντιστρέψει*** *την σειρά του διανύσματος g και* ***προσθέσει*** *στο τέλος τα* 3 *πρώτα στοιχεία του g*

$$\begin{pmatrix} 3 & 2 & 1 & 1 & 3 & 2 & 1 \\ & & & 7 & -7 & 0 & 1 \\ & & 7 & -7 & 0 & 1 & \\ & 7 & -7 & 0 & 1 & & \\ 7 & -7 & 0 & 1 & & & \end{pmatrix},$$

είτε *χρησιμοποιώντας την σχέση* $f *_n g \equiv fg \bmod x^n - 1$.

```
PolynomialMod[(fCoef.powersX) (gCoef.powersX), x^n - 1]
```

```
-13 + 2 x + 10 x^2 + 8 x^3
```

Τ*ονίζουμε την* ***ισοδυναμία*** *ανάμεσα στην συνέλιξη και τον πολλαπλασιασμό στον δακτύλιο* $R[x]/\langle x^n - 1\rangle$. *Στην συνέχεια θα εκμεταλλευτούμε την σχέση αυτή για να αναπτύξουμε τον γρήγορο μετασχηματισμό* Fourier.

1η Σημαντική Παρατήρηση: επιτάχυνση της συνέλιξης.

Ο *τελευταίος υπολογισμός της συνέλιξης μας οδηγεί στην εξής* ***γενίκευση***: Α*ν* $f \in R[x]$ *είναι βαθμού n, ειδικά για διαιρέτες της μορφής* $x^{n/2} - u$, *το* ***υπόλοιπο*** $f \bmod x^{n/2} - u$ *υπολογίζεται,* ***χωρίς διαίρεση****, αν στους "χαμηλούς όρους" του f προσθέσουμε τους "ψηλούς όρους" του f διαιρεμένους με* $x^{n/2}$ *και πολλαπλασιασμένους επί u.*

Έτσι *αν πάρουμε ένα πολυώνυμο βαθμού* $6 < n = 8$ *ως προς την μεταβλητή x και το θεωρήσουμε ως βαθμού* 7 (*με μηδενικό κύριο συντελεστή*)

```
n = 8; f = 7 + 7 x^2 + 8 x^3 - 20 x^4 + 2 x^5 + 3 x^6; var = First[Variables[f]];
temp = CoefficientList[f, var];
AppendTo[temp, 0]
```

```
{7, 0, 7, 8, -20, 2, 3, 0}
```

τότε το υπόλοιπο της διαίρεσής του με το $x^4 - 1$ *υπολογίζεται,* ***χωρίς διαίρεση****, αν προσθέτουμε στους "χαμηλούς όρους" του f*

```
lowerPart = Table[var^i, {i, 0, Floor[(n - 1)/2]}].Take[temp, Floor[n/2]]
```

```
7 + 7 x^2 + 8 x^3
```

τους "ψηλούς όρους" του f διαιρεμένους με x^4 *και πολλαπλασιασμένους επί* $u = 1$

```
higherPart =
 Table[var^(n/2 + i), {i, 0, Floor[(n - 1)/2]}].Take[temp, -Floor[n/2]] / x^(n/2) //
  Expand
```

```
-20 + 2 x + 3 x^2
```

```
lowerPart + higherPart
```

```
-13 + 2 x + 10 x^2 + 8 x^3
```

Πράγματι, *εκτελώντας την διαίρεση έχουμε το ίδιο αποτέλεσμα*

```
PolynomialMod[f, x^4 - 1]
```

```
-13 + 2 x + 10 x^2 + 8 x^3
```

Σημειώνουμε πως η διαίρεση με x^4 *που έγινε παραπάνω δεν έχει κόστος διότι απλώς αλλάζουμε τους εκθέτες.*

Επιπλέον, αν και δεν μας ενδιαφέρει άμεσα, αναφέρουμε πως και το πηλίκο υπολογίζεται, ***επίσης χωρίς διαίρεση****, και είναι οι "ψηλοί όροι" του f διαιρεμένοι με* $x^{n/2}$

```
higherPart == PolynomialQuotient[f, x^4 - 1, x]
True
```

2η Σημαντική Παρατήρηση: *αναγωγική εκτίμηση πολυωνύμου.*

Αν R δακτύλιος, $n \in \mathbb{N}_{\geq 1}$, και $\omega \in R$ μία αρχέγονη ρίζα της μονάδας, τάξης n, τότε οι τιμές του πολυωνύμου $f \in R[x]$, ***βαθμού*** $\leq n - 1$, στις δυνάμεις $1, \omega, \omega^2, \ldots, \omega^{n-1}$ ***ισούνται*** με τις τιμές των υπολοίπων $r_0 = f \bmod x^{n/2} - 1$ και $r_1 = f \bmod x^{n/2} + 1$ στις αντίστοιχες δυνάμεις.

Πράγματι, αν υποθέσουμε πως για κάποια $q_0, q_1, r_0, r_1 \in R[x]$, βαθμού $\leq \frac{n}{2} - 1$, έχουμε

$$f = q_0 (x^{n/2} - 1) + r_0 = q_1 (x^{n/2} + 1) + r_1$$

τότε ισχύουν

$$\begin{aligned} f(\omega^{2i}) &= q_0(\omega^{2i})(\omega^{ni} - 1) + r_0(\omega^{2i}) = r_0(\omega^{2i}) \\ f(\omega^{2i+1}) &= q_1(\omega^{2i+1})(\omega^{ni}\,\omega^{n/2} + 1) + r_1(\omega^{2i+1}) = r_1(\omega^{2i+1}) \end{aligned}$$

Η πρώτη διότι $\omega^{ni} = 1$, και η δεύτερη διότι $\omega^{n/2} = -1$, δεδομένου ότι

$$(\omega^n - 1) = (\omega^{n/2} - 1)(\omega^{n/2} + 1) = 0$$

και $\omega^{n/2} \neq 1$. Επιπλέον, αν θέσουμε $r_1^*(x) = r_1(\omega x)$ προκύπτει $r_1(\omega^{2i+1}) = r_1^*(\omega^{2i})$.

Συνεπώς, οι άρτιες δυνάμεις του ω θα υπολογίζονται με το r_0 ενώ οι περιττές δυνάμεις του ω θα υπολογίζονται με το r_1. Και στις δύο περιπτώσεις, ***υπολογίζουμε τις τιμές των υπολοίπων*** r_0 και r_1^* στις **ίδιες** δυνάμεις $1, \omega^2, \ldots, \omega^{2n-2}$ όπου ω^2 είναι μία αρχέγονη ρίζα της μονάδας, τάξης $\frac{n}{2}$.

Ας εφαρμόσουμε τώρα τα παραπάνω στον γρήγορο μετασχηματισμό Fourier.

Έστω λοιπόν άρτιος $n \in \mathbb{N}_{\geq 1}$, R δακτύλιος, $\omega \in R$ μία αρχέγονη ρίζα της μονάδας, τάξης n και $f \in R[x]$ βαθμού $\leq n - 1$. Αυτό που θέλουμε είναι να υπολογίσουμε την τιμή του f στα σημεία $1, \omega, \omega^2, \ldots, \omega^{n-1}$ εκτελώντας $O(n \log n)$ πράξεις στον

δακτύλιο R — αντί $O(n^2)$. Για τον σκοπό μας χρησιμοποιούμε τις παραπάνω 2 παρατηρήσεις, δηλαδή:

- από την 2η Σημαντική Παρατήρηση (αναγωγική εκτίμηση πολυωνύμων) ξέρουμε πως για να υπολογίσουμε τις τιμές του f στις δυνάμεις $1, \omega, \omega^2, \ldots, \omega^{n-1}$ αρκεί να υπολογίσουμε τις τιμές των υπολοίπων r_0 και r_1^* στις δυνάμεις $1, \omega^2, \ldots, \omega^{2n-2}$ όπου ω^2 είναι μία αρχέγονη ρίζα της μονάδας, τάξης $\frac{n}{2}$.

- από την 1η Σημαντική Παρατήρηση (επιτάχυνση της συνέλιξης) ξέρουμε πως μπορούμε να υπολογίσουμε τα πολυώνυμα $f \bmod x^{n/2} \mp 1$ ***χωρίς διαίρεση!*** Οι συντελεστές του r_0 υπολογίζονται εκτελώντας $\frac{n}{2}$ προσθέσεις ενώ οι συντελεστές του r_1^* υπολογίζονται εκτελώντας $\frac{n}{2}$ πολλαπλασιασμούς με δυνάμεις του ω. Η εφαρμογή αυτής της παρατήρησης μας επιταχύνει τον μετασχηματισμό!

Αναγωγική εφαρμογή των υπολοίπων αυτών $\log n$ φορές (όπου ο συνολικός αριθμός των πολλαπλασιασμών που απαιτείται για τον υπολογισμών όλων των r_k^* είναι $\sum_{k=1}^{\log(n)} \frac{n}{2^k} = n - \frac{n}{2^{\log(n)}}$) μας δίνει τον γρήγορο μετασχηματισμό Fourier με χρόνο $O(n \log n)$.

Παράδειγμα:
Θα επαναλάβουμε το παράδειγμα που κάναμε στην προηγούμενη υποενότητα με την παραγοντοποίηση των πινάκων:

Το πολυώνυμό μας βαθμού $n = 8$ είναι

```
n = 8; powersX = Table[x^i, {i, 0, n - 1}];
fCoef = Table[i, {i, 0, n - 1}]; f = fCoef.powersX
```

$$x + 2x^2 + 3x^3 + 4x^4 + 5x^5 + 6x^6 + 7x^7$$

και για $\omega = e^{2\pi i/8}$, την αρχέγονη ρίζα της μονάδας, τάξης 8, θέλουμε να υπολογίσουμε τις τιμές $f(1), f(\omega), \ldots, f(\omega^7)$, δηλαδή τον μετασχηματισμό Fourier.

Θέτοντας

```
ω = e^(2 π i/n); powersW = Table[ω^i, {i, 0, n - 1}]
```

$$\left\{1, e^{\frac{i\pi}{4}}, i, e^{\frac{3i\pi}{4}}, -1, e^{-\frac{3i\pi}{4}}, -i, e^{-\frac{i\pi}{4}}\right\}$$

έχουμε τον μετασχηματισμό Fourier

```
(f /. x → powersW // N) / Sqrt[n] ==
Fourier[fCoef] // Chop
```

```
True
```

Σύμφωνα όμως *με την* 2*η σημαντική παρατήρηση, στο πρώτο βήμα του* ***γρήγορου*** *μετασχηματισμού* Fourier *υπολογίζουμε τα υπόλοιπα* mod $x^{n/2} \mp 1$, r_0 *και* r_1, *αντίστοιχα, για τα οποία ισχύουν* $f(\omega^{2i}) = r_0(\omega^{2i})$ *και* $f(\omega^{2i+1}) = r_1(\omega^{2i+1})$, *δηλαδή με το* r_0 *υπολογίζουμε τις άρτιες δυνάμεις του* ω, *ενώ με το* r_1 *υπολογίζουμε τις περιττές δυνάμεις του* ω.

Σύμφωνα *με την* 1*η σημαντική παρατήρηση, στην περίπτωσή μας που ο βαθμός του διαιρετέου είναι* $\deg(f) \le n - 1$ *και ο διαιρέτης είναι της μορφής* $x^{n/2} - u$ *έχουμε την ακόλουθη συνάρτηση για τον υπολογισμό των υπολοίπων* r_0 *και* r_1 ***χωρίς διαίρεση*** (*προσέξτε πως* $g = x^{n/2} - u$ *και* $n = \deg(f) > \deg(g)$)

```
remainderWithoutDivision[f_, g_, n_] /;
  n > Exponent[f, x] > Exponent[g, x] &&
   Exponent[g, x] == n / 2 && Length[Apply[List, g]] == 2 :=
 Module[{higherPart, k, lowerPart, temp, var = First[Variables[f]]},
  temp = CoefficientList[f, var];
  k = Length[temp];
  If[k ≠ n, AppendTo[temp, Table[0, {n - k}]]; temp = Flatten[temp]];
  lowerPart =
   Table[var^i, {i, 0, Floor[(n - 1)/2]}].Take[temp, Floor[n/2]];
higherPart = -First[CoefficientList[g, var]]
    Table[var^(n/2 + i), {i, 0, Floor[(n - 1)/2]}].
       Take[temp, -Floor[n/2]] / x^(n/2) // Expand;
  lowerPart + higherPart
 ]
```

Υπολογίζουμε λοιπόν "γρήγορα" τα υπόλοιπα r_0

```
r0 = remainderWithoutDivision[f, x^(n/2) - 1, n]
```

$4 + 6\,x + 8\,x^2 + 10\,x^3$

και r_1

```
r1 = remainderWithoutDivision[f, x^(n/2) + 1, n]
```

$-4 - 4\,x - 4\,x^2 - 4\,x^3$

για να τα εκτιμήσουμε στις άρτιες και περιττές δυνάμεις του ω αντίστοιχα.

```
evenPowersW = Table[powersW[[i + 1]], {i, 0, n - 1, 2}]
```

$\{1, i, -1, -i\}$

```
oddPowersW = Table[powersW[[i + 1]], {i, 1, n, 2}]
```

$\left\{e^{\frac{i\pi}{4}}, e^{\frac{3i\pi}{4}}, e^{-\frac{3i\pi}{4}}, e^{-\frac{i\pi}{4}}\right\}$

Επειδή όμως ο βαθμός των r_0 και r_1 είναι > 1, για να υπολογίσουμε τις τιμές των υπολοίπων r_0 και r_1 στις άρτιες και περιττές δυνάμεις του ω αρκεί, αναγωγικά, να υπολογίσουμε τις τιμές των νέων υπολοίπων $r_{0,0}$, $r_{0,1}$, $r_{1,0}$ και $r_{1,1}$ στις αντίστοιχες δυνάμεις του ω.

Από το r_0 υπολογίζουμε τα δύο νέα υπόλοιπα

```
r00 = remainderWithoutDivision[r0, x^(n/4) - 1, n / 2]
```

$12 + 16\,x$

```
r01 = remainderWithoutDivision[r0, x^(n/4) + 1, n / 2]
```

$-4 - 4\,x$

και το ίδιο κάνουμε και για το r_1 *το οποίο όμως έχουμε αντικαταστήσει με το* r_1^* *όπως αναφέραμε παραπάνω* (Προσέξτε πως τώρα το ω ισούται με το πρώτο στοιχείο της λίστας *oddPowersW*, των περιττών δυνάμεων του ω)

```
r1S = r1 /. x → ω x
```

$$-4 - 4\, e^{\frac{i\pi}{4}}\, x - 4\, i\, x^2 - 4\, e^{\frac{3 i \pi}{4}}\, x^3$$

```
r10 = remainderWithoutDivision[r1S, x^2 - 1, 4]
```

$$(-4 - 4\, i) - 4\, e^{\frac{i\pi}{4}}\, x - 4\, e^{\frac{3 i \pi}{4}}\, x$$

```
r11 = remainderWithoutDivision[r1S, x^2 + 1, 4]
```

$$(-4 + 4\, i) - 4\, e^{\frac{i\pi}{4}}\, x + 4\, e^{\frac{3 i \pi}{4}}\, x$$

Στην *συνέχεια καθορίζουμε σε ποιές από τις δυνάμεις* $1, \omega^2, \ldots, \omega^{2n-2}$ (όπου $\omega^2 = e^{\pi i/2}$ *είναι μία αρχέγονη ρίζα της μονάδας, τάξης* $\frac{n}{2} = 4$) *θα εκτιμηθεί το κάθε ένα από τα υπόλοιπα* $r_{0,0}$, $r_{0,1}$, $r_{1,0}$ *και* $r_{1,1}$.

```
n = Length[evenPowersW];
evenPowersW00 = Table[evenPowersW[[i + 1]], {i, 0, n - 1, 2}]
evenPowersW01 = Table[evenPowersW[[i + 1]], {i, 1, n, 2}]
oddPowersW10 = Table[evenPowersW[[i + 1]], {i, 0, n - 1, 2}]
oddPowersW11 = Table[evenPowersW[[i + 1]], {i, 1, n, 2}]
```

```
{1, -1}
```

```
{i, -i}
```

```
{1, -1}
```

```
{i, -i}
```

Και *τέλος επειδή τα υπόλοιπα* $r_{0,0}$, $r_{0,1}$, $r_{1,0}$ *και* $r_{1,1}$ *είναι βαθμού* 1 *υπολογίζουμε τις τιμές τους στις αντίστοιχες δυνάμεις του* ω

```
even20 = r00 /. x -> evenPowersW00 // N
```

```
{28., -4.}
```

```
odd20 = r01 /. x → evenPowersW01 // N
```

```
{-4. - 4. i, -4. + 4. i}
```

```
even21 = r10 /. x → oddPowersW10 // N
```

```
{-4. - 9.65685 i, -4. + 1.65685 i}
```

```
odd21 = r11 /. x → oddPowersW11 // N
```

```
{-4. - 1.65685 i, -4. + 9.65685 i}
```

και βρίσκουμε τον μετασχηματισμό Fourier

$$\frac{1}{\sqrt{8}}$$ `(Thread[{even20, even21, odd20, odd21}] // Flatten) ==`

```
  Fourier[{0, 1, 2, 3, 4, 5, 6, 7}] // Chop
```

```
True
```

Στις *ασκήσεις ζητείται να προγραμματισθεί ο γρήγορος μετασχηματισμός* Fourier (FFT) *με υπόλοιπα χωρίς διαιρέσεις*.

Το *πρόβλημα με τον γρήγορο μετασχηματισμό* Fourier *όπως τον αναπτύξαμε παραπάνω είναι ότι το μήκος του διανύσματος που μετασχηματίζουμε πρέπει να είναι δύναμη του* 2. *Στην βιβλιογραφία υπάρχει μεγάλος αριθμός επεκτάσεων και τροποποιήσεων της βασικής μορφής του γρήγορου μετασχηματισμού* Fourier (FFT) *που παρουσιάσαμε. Ανάμεσα σε αυτές τις τροποποιήσεις υπάρχουν και εκείνες που επιτρέπουν το μήκος του διανύσματος να είναι γινόμενο μοκρών πρώτων αριθμών*.

Παράρτημα Δ: Διάφορες εφαρμογές του γρήγορου μετασχηματισμού Fourier

Λόγω της σπουδαιότητας του γρήγορου μετασχηματισμού Fourier (FFT) στην ενότητα αυτή θα περιγράψουμε ορισμένες εφαρμογές. Οι πιο σημαντικές είναι η προσέγγιση συναρτήσεων με μιγαδικές τριγωνομετρικές συναρτήσεις, γνωστή και σαν **γρήγορη προσέγγιση Fourier** ή fast Fourier fit (FFF), και η συμπίεση εικόνων με τον **διακριτό μετασχηματισμό συνημιτόνων** ή discrete cosine transform (DCT), μία παραλλαγή του γρήγορου μετασχηματισμού Fourier .

■ Δ1: Γρήγορος πολλαπλασιασμός πολυωνύμων και ακεραίων με FFT

Αυτό είναι ένα πολυσυζητημένο θέμα. Παρουσιάζουμε πρώτα τον γρήγορο πολλαπλασιασμό πολυωνύμων στον οποίο στηρίζεται ο γρήγορος πολλαπλασιασμός ακεραίων.

Έχουμε αναφέρει πως αν R είναι μία ακέραια περιοχή, τότε ένα πολυώνυμο $f = \sum_{i=0}^{n-1} f_i x^i \in R[x]$, βαθμού μικρότερου του $n \in \mathbb{N}$, μπορεί να παρασταθεί **είτε** από την λίστα των συντελεστών του $\{f_0, \ldots, f_{n-1}\}$, **είτε** από την "λίστα τιμών" του σε n διαφορετικά σημεία $u_0, \ldots, u_{n-1} \in R$. Στην περίπτωσή μας, για $0 \le i < n$ έχουμε $u_i = \omega^i$, όπου $\omega \in R$ είναι μία αρχέγονη ρίζα της μονάδας, τάξης n.

Ο λόγος που θεωρούμε την παράσταση πολυωνύμου με την "λίστα τιμών" του είναι ότι ο πολλαπλασιασμός σε αυτήν την μορφή ειναι γρήγορος και εύκολος. Συγκεκριμένα, αν οι "λίστες τιμών" $\{f(u_0), \ldots, f(u_{n-1})\}$ και $\{g(u_0), \ldots, g(u_{n-1})\}$ — στα n διαφορετικά σημεία $u_0, \ldots, u_{n-1}$ — παριστάνουν τα πολυώνυμα f και g, με $\deg(f) + \deg(g) < n$, τότε η "λίστα τιμών" του γινομένου $f \cdot g$ στα σημεία αυτά είναι $\{f(u_0)\cdot g(u_0), \ldots, f(u_{n-1})\cdot g(u_{n-1})\}$.

Επομένως, όταν τα πολυώνυμα παρίστανται με τις "λίστες τιμών" τους το κόστος του πολλαπλασιασμού τους είναι γραμμικό ως προς τον βαθμό τους. Σε αντίθεση, όταν τα πολυώνυμα παρίστανται με τις λίστες των συντελεστών τους δεν υπάρχει μέθοδος πολλαπλασιασμού τους που το κόστος της να είναι γραμμικό ως προς τον βαθμό τους.

Συνεπώς, *ένας γρήγορος τρόπος υπολογισμού των τιμών πολυωνύμων* (*σε n διαφορετικά σημεία*) *και παρεμβολής μας οδηγεί στον ακόλουθο γρήγορο αλγόριθμο πολλαπλασιασμού πολυωνύμων*.

Αλγόριθμος: Γρήγορος πολλαπλασιασμός πολυωνύμων (με FFT)

Είσοδος: Τα *πολυώνυμα* $f,g \in R[x]$, *βαθμού* $\deg(f)$ *και* $\deg(g)$ *αντίστοιχα*. *R είναι δακτύλιος στον οποίο υπάρχουν αρχέγονες ρίζες της μονάδας, τάξης n, για κάθε* $n \in \mathbb{N}$.

Έξοδος: Το *πολυώνυμο* $h = f \cdot g \in R[x]$.

==========

1. Επιλέγουμε $n = 2^k$, *έτσι ώστε* $n > \deg(f) + \deg(g)$ *και ω μία αρχέγονη ρίζα της μονάδας, τάξης n*. (Η *ρίζα ω χρειάζεται στον* FFT.)

2. Θεωρούμε *τα πολυώνυμα f, g βαθμού n* - 1, "*γεμίζοντας*" *τις λίστες των συντελεστών τους με μηδενικά αν χρειάζεται, και εφαρμόζουμε σε αυτές τον* FFT.

(*α*) $F = \text{FFT}(\{f_0,\ f_1,\ \ldots,\ f_{\deg(f)}, 0, \ldots, 0\})$;

(*β*) $G = \text{FFT}(\{g_0,\ g_1,\ \ldots,\ g_{\deg(g)}, 0, \ldots, 0\})$;

3. Τέλος, *πολλαπλασιάζουμε τα F, G σημείο προς σημείο και κάνοντας παρεμβολή στο αποτέλεσμα — με τον αντίστροφο μετασχηματισμό* iFFT — *βρίσκουμε το γινόμενο*

(*γ*) $H = \text{iFFT}(FG)$**.**

==========

Στο *Mathematica οι μετασχηματισμοί* FFT *και* iFFT *γίνονται με τις συναρτήσεις* `Fourier[]` *και* `InverseFourier[]` *αντίστοιχα*. Ακολουθεί *η συνάρτηση* `polyFFTMultiply[]`, *που υπολογίζει το γινόμενο δύο πολυωνύμων με συντελεστές στον δακτύλιο* $\mathbb{C}$.

```
polyFFTMultiply[f_, g_] := Module[
  {x = First[Variables[f]], n0 = Exponent[f, x],
   m0 = Exponent[g, x], flist = CoefficientList[f, x],
   glist = CoefficientList[g, x], productValues, k = 1, n},
  While[2^k ≤ m0 + n0, ++k ]; n = 2^k;
  flist = AppendTo[flist, Table[0, {i, 1, n - (n0 + 1)}]] // Flatten;
  glist = AppendTo[glist, Table[0, {i, 1, n - (m0 + 1)}]] // Flatten;
  productValues = Fourier[flist] Fourier[glist] // Chop;
  (√n InverseFourier[productValues] // Chop // Rationalize).
   Table[x^i, {i, 0, n - 1}]
 ]
```

Παράδειγμα: Έστω ότι μας δίνονται τα δύο πολυώνυμα $f(x) = x^3 - 7x + 7$ και $g(x) = 3x^2 - 7$, βαθμών $\deg(f) = 3$ και $\deg(g) = 2$ αντίστοιχα.

```
Clear[f, g];
f[x_] = x^3 - 7 x + 7; g[x_] = 3 x^2 - 7;
```

Με τον κλασσικό τρόπο, το γινόμενό τους είναι

```
f[x] g[x] // Expand
```

```
-49 + 49 x + 21 x^2 - 28 x^3 + 3 x^5
```

βαθμού $\deg(f) + \deg(g) = 5$. Το ίδιο αποτέλεσμα βρίσκουμε και με την καινούργια μας συνάρτηση

```
polyFFTMultiply[f[x], g[x]]
```

```
-49 + 49 x + 21 x^2 - 28 x^3 + 3 x^5
```

Ας δούμε λεπτομερώς πως δουλεύει. Επειδή ο μετασχηματισμός FFT δουλεύει για διανύσματα που το μήκος τους είναι δύναμη του 2, θεωρούμε πως ο βαθμός του γινομένου είναι μικρότερος από $n = 8$.

Έχοντας ορίσει την τιμή του n, ορίζουμε τις λίστες των συντελεστών των f και g — τις οποίες "γεμίζουμε" με 0's μέχρις ότου το μήκος τους γίνει $n = 8$.

```
n = 8;
flist = CoefficientList[f[x], x]; flist = AppendTo[flist,
   Table[0, {i, 1, n - (Exponent[f[x], x] + 1)}]] // Flatten
```

```
{7, -7, 0, 1, 0, 0, 0, 0}
```

```
glist = CoefficientList[g[x], x]; glist = AppendTo[glist,
   Table[0, {i, 1, n - (Exponent[g[x], x] + 1)}]] // Flatten
```

```
{-7, 0, 3, 0, 0, 0, 0, 0}
```

Στις λίστες των συντελεστών εφαρμόζουμε τον μετασχηματισμό FFT (*συνάρτηση* `Fourier[]`) *και τα αποτελέσματα τα πολλαπλασιασιάζουμε σημείο προς σημείο.*

```
productValues = Fourier[flist] Fourier[glist] // Chop
```

```
{-0.5, 0.415738 + 4.21599 i, -8.75 + 10. i, -12.6657 - 1.03401 i,
 -6.5, -12.6657 + 1.03401 i, -8.75 - 10. i, 0.415738 - 4.21599 i}
```

Από *τον ορισμό του γρήγορου μετασχηματισμού* Fourier *έπεται πως αυτό που κάναμε μέχρις εδώ είναι ισοδύναμο με*: (**α**) *τον υπολογισμό της* "*λίστας τιμών*" *κάθε πολυωνύμου στα σημεία* $u_i = \omega^i$, *όπου* $\omega = e^{\frac{2\pi i}{n}}$, $n = 8$, *και* (**β**) *τον σημείο προς σημείο πολλαπλασιασμό των αποτελεσμάτων.*

Κάνοντας *παρεμβολή στο αποτέλεσμα με την συνάρτηση* `InverseFourier[]`, *και λαμβάνοντας υπ' όψη τον παράγοντα* $\sqrt{n}$, *βρίσκουμε την λίστα των συντελεστών του γινομένου.*

```
productCoefficients =
 √n InverseFourier[productValues] // Chop // Rationalize
```

```
{-49, 49, 21, -28, 0, 3, 0, 0}
```

Ακριβώς *αυτό που βρήκαμε με τον κλασσικό τρόπο.*

Το *κόστος του πολλαπλασιασμού πολυωνύμων με τον* FFT *είναι* $O(n \log n)$ *πράξεις στον δακτύλιο* R, *που στην ουσία είναι το κόστος υπολογισμού του*

μετασχηματισμού FFT και του αντιστρόφου του. Αυτό αποτελεί σημαντική βελτίωση σε σχέση με το κόστος $O(n^2)$ του κλασσικού αλγορίθμου.

Στο σημείο αυτό χρειάζεται η εξής διευκρίνιση. Ο αλγόριθμος αυτός είναι γρηγορότερος από τον κλασσικό στην περίπτωση που τα πολυώνυμα είναι του ιδίου περίπου βαθμού. Αν $m = \deg(f) \ll \deg(g) = n$, δηλαδή ο βαθμός του f είναι κατά πολύ μικρότερος του βαθμού του g, τότε το κόστος του κλασσικού αλγορίθμου είναι $O(m \cdot n)$ και για $m = 1$, γίνεται $O(n)$. Ο πολλαπλασιασμός με FFT έχει πάντα κόστος $O(n \log n)$.

Ο παραπάνω αλγόριθμος εύκολα τροποποιείται για τους ακεραίους. Όπως ξέρουμε, κάθε ακέραιος παρίσταται σαν πολυώνυμο ως προς μία βάση β. Δηλαδή, για τυχόντα ακέραιο a έχουμε $a = (-1)^s \sum_{0 \le i \le n} a_i \beta^i$.

Συνεπώς, ο πολλαπλασιασμός ακεραίων μπορεί να θεωρηθεί σαν πολλαπλασιασμός πολυωνύμων, με μόνη διαφορά ότι στο τελικό αποτέλεσμα αντικαθιστούμε την μεταβλητή x με την βάση β.

Αλγόριθμος: Γρήγορος πολλαπλασιασμός ακεραίων (με FFT)

Είσοδος: Οι ακέραιοι $a, b \in \mathbb{Z}$ με μήκος $\lambda(a)$ και $\lambda(b)$, αντίστοιχα, ως προς την βάση β. Στον $\mathbb{Z}$ υπάρχουν αρχέγονες ρίζες της μονάδας, τάξης n, για κάθε $n \in \mathbb{N}$.
Έξοδος: Ο ακέραιος $c = a \cdot b \in \mathbb{Z}$.

==========

1. Επιλέγουμε $n = 2^k$, έτσι ώστε $n > \lambda(a) + \lambda(b)$ και ω μία αρχέγονη ρίζα της μονάδας, τάξης n. (Η ρίζα ω χρειάζεται στον FFT.)

2. Θεωρούμε τους ακεραίους ως πολυώνυμα βαθμού $n - 1$, $a = (-1)^s \sum_{0 \le i \le \lambda(a)-1} a_i \beta^i$, $b = (-1)^s \sum_{0 \le i \le \lambda(b)-1} b_i \beta^i$, "γεμίζοντας" τις λίστες των συντελεστών τους με μηδενικά αν χρειάζεται, και εφαρμόζουμε σε αυτές τον FFT.

(α) $A = \text{FFT}(\{a_0, a_1, \ldots, a_{\lambda(a)-1}, 0, \ldots, 0\})$;

(β) $B = \text{FFT}(\{b_0, b_1, \ldots, b_{\lambda(b)-1}, 0, \ldots, 0\})$;

3. Τέλος, πολλαπλασιάζουμε τα A, B σημείο προς σημείο και κάνοντας παρεμβολή στο αποτέλεσμα — με τον αντίστροφο μετασχηματισμό iFFT — και

αντικατάσταση του x με την βάση β, παίρνουμε το αποτέλεσμα

(γ) C(*x*) = iFFT(*AB*)**;** c = C(*β*).

==========

Το *κόστος του πολλαπλασιασμού ακεραίων με* FFT *είναι επίσης* $O(n \log n)$ *πράξεις στον δακτύλιο* $\mathbb{Z}$.

Ακολουθεί *η συνάρτηση* `integerFFTMultiply[]`, *στο Mathematica που υπολογίζει το γινόμενο δύο ακεραίων υποθέτοντας ότι η βάση είναι* $\beta = 10$.

```
integerFFTMultiply[f_Integer, g_Integer, b_: 10] := Module[
  {flist, glist, k = 1, m0, n, n0, productValues, x},
  flist = IntegerDigits[f, b] // Reverse;
  n0 = Length[flist];
  glist = IntegerDigits[g, b] // Reverse;
  m0 = Length[glist];
  While[2^k ≤ m0 + n0, ++k ]; n = 2^k;
  flist = AppendTo[flist, Table[0, {i, 1, n - n0}]] // Flatten;
  glist = AppendTo[glist, Table[0, {i, 1, n - m0}]] // Flatten;
  productValues = Fourier[flist] Fourier[glist] // Chop;
  (√n InverseFourier[productValues] // Chop // Rationalize).
    Table[x^i, {i, 0, n - 1}] /. x → b
 ]
```

Έτσι *έχουμε για παράδειγμα*

```
integerFFTMultiply[123456789, 987654321]
```

```
121932631112635269
```

■ **Δ2: Ο FFT *είναι η βάση της γρήγορης προσέγγισης* Fourier (FFF)**

Στην *συνέχεια στρέφουμε την προσοχή μας στην* ***γρήγορη προσέγγιση*** **Fourier** ή fast Fourier fit (FFF), *δηλαδή στην προσέγγιση συναρτήσεων με ημίτονα* ή/*και συνημίτονα*.

Ορισμός: Συναρτήσεις $f : \mathbb{R} \to \mathbb{C}$, *μιας μεταβλητής με τιμές στο μιγαδικό επίπεδο, μπορούν να προσεγγισθούν από μιγαδικά τριγωνομετρικά πολυώνυμα της μορφής*

$$f(t) = \sum_{k=-n}^{n} c_k e^{k\omega i t}$$

$$= \frac{\alpha_0}{2} + \sum_{k=1}^{n} (\alpha_k \cos(k\omega t) + \beta_k \sin(k\omega t)),$$

όπου c_k είναι οι συντελεστές της γρήγορης προσέγγισης Fourier που ικανοποιούν τις εξισώσεις

$$c_0 = \frac{\alpha_0}{2}, c_k = \frac{(\alpha_k - i\beta_k)}{2}, c_{-k} = \frac{(\alpha_k + i\beta_k)}{2}$$

και

$$\alpha_0 = 2c_0, \alpha_k = c_k + c_{-k}, \beta_k = i(c_k - c_{-k})$$

για $k = 1, \ldots, n,$ και $\omega = \frac{2\pi}{L}$ με $L > 0$. Βλέπε και την βιβλιογραφία.

Για να υπολογίσουμε τα προσεγγιστικά μιγαδικά τριγωνομετρικά πολυώνυμα που αναφέρονται στον παραπάνω ορισμό θα χρησιμοποιήσουμε την συνάρτηση `Fast-Fourierfit[]` από το βιβλίο των Bill Davis και Jerry Uhl "Differential Equations & Mathematica".

```
Clear[FastFourierfit, Fourierfitters, F, Fvalues, n, k,
   jump, num, numtab, coeffs, t, L];
jump[n_] := jump[n] = N[1/(2 n)];

Fvalues[F_, L_, n_] :=
 N[Table[F[L t], {t, 0, 1 - jump[n], jump[n]}]];

numtab[n_] := numtab[n] = Table[k, {k, 1, n}];

Fourierfitters[L_, n_, t_] := Table[E^(2 π I k t/L),
    {k, -n + 1, n - 1}];

coeffs[n_, list_] := Join[Reverse[Part[Fourier[list], numtab[n]]],
    Part[InverseFourier[list], Drop[numtab[n], 1]]] /
       N[Sqrt[Length[list]]]
```

```
FastFourierfit[F_, L_, n_, t_] :=
  Chop[Fourierfitters[L, n, t].coeffs[n, Fvalues[F, L, n]]];
```

Το *πρόγραμμα δουλεύει ως εξής*: *από την συνάρτηση f(t) που θέλουμε να προσεγγίσουμε, η συνάρτηση* `Fvalues[]` (*με την βοήθεια της* `jump[]`) *παράγει μία λίστα από* $2n - 1$ *ισοαπέχοντα σημεία* ***ανάμεσα*** *στο* $t = 0$ *και* $t = L$. *Τα σημεία αυτά μαζί με το* 0 *αποτελούν μία λίστα από* $2n$ *σημεία που θα μετασχηματισθούν με τον* FFT.

Κατόπιν η συνάρτηση `numtab[]` *δημιουργεί την λίστα των ακεραίων από το* 1 *μέχρι το n, η οποία λίστα χρησιμοποιείται από την συνάρτηση* `coeffs[]` *για να ενώσει δύο λίστες*.

Η *πρώτη λίστα που ενώνει η* `coeffs[]` *είναι η υπολίστα* (*σε* ***αντίστροφη*** *διάταξη*) *των πρώτων n στοιχείων από τον μετασχηματισμό* Fourier *των* $2n$ *σημείων της* `Fvalues[]`, *ενώ η δεύτερη λίστα είναι η υπολίστα των πρώτων n στοιχείων από τον* ***αντίστροφο*** *μετασχηματισμό* Fourier *των* $2n$ *σημείων της* `Fvalues[]` — ***χωρίς*** *το πρώτο στοιχείο*. *Δηλαδή, η "ενωμένη" λίστα που παράγεται από την* `coeffs[]` *έχει συνολικά* $2n - 1$ *σημεία*.

Τέλος *η συνάρτηση* `FastFourierFit[]` *μας δίνει την προσέγγιση σχηματίζοντας το εσωτερικό γινόμενο της λίστας* $\{e^{(-n+1)2\pi i t/L}, \ldots, 1, \ldots, e^{(n-1)2\pi i t/L}\}$, *που δημιουργείται από την συνάρτηση* `Fourierfitters[]`, *και της "ενωμένης" λίστας που δημιουργείται από την* `coeffs[]`. (*Όσοι αριθμοί είναι* (*σε μέγεθος*) *μικρότεροι από* 10^{-10} *στρογγυλεύονται στο* 0.)

Η *συνάρτηση* `FastFourierFit[]` *παίρνει τέσσερις παραμέτρους*. *Οι πρώτες δύο παράμετροι είναι, αντίστοιχα, η περιοδική συνάρτηση f(t) που θέλουμε να προσεγγίσουμε, και η περίοδός της L*. *Η τρίτη παράμετρος είναι ο αριθμός n που καθορίζει τα ισοαπέχοντα* $2n - 1$ *μη μηδενικά σημεία και η τελευταία παράμετρος είναι η μεταβλητή που θέλουμε να χρησιμοποιήσουμε*. *Προσέξτε πως η* `FastFourierFit[]` *χρησιμοποιεί τις συναρτήσεις* `Fourier[]` *και* `InverseFourier[]`, *με υπολογιστικό κόστος* $n \log n$.

Παράδειγμα χρήσης της `FastFourierfit[]`: Έστω η περιοδική συνάρτηση $f(x) = \cos(2\pi x)\sin(1 - \cos(3\pi x))$ με περίοδο $L = 2$. Το σχεδιάγραμμά της φαίνεται στο Figure 1.

```
f[x_] := Cos[2 π x] Sin[1 - Cos[3 π x]];
L = 2;
cycles = 2;
Plot[f[x], {x, 0, cycles L},
  AxesLabel → {"x", "f(x)"},
  PlotStyle → {{Thickness[0.007], RGBColor[0, 0, 1]}},
  PlotLabel → "cycles" cycles, Epilog →
   {{RGBColor[1, 0, 0], Thickness[0.007], Line[{{0, 0}, {L, 0}}]},
    {Text["One Period", {L/2, 0.1}]}}];
```

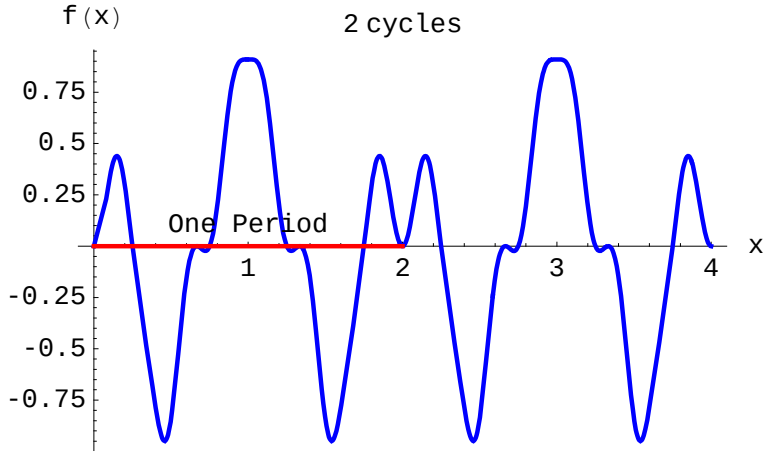


Figure 1

Προσεγγίζοντας την $f(x)$ με $7 = 2n - 1$ σημεία ανάμεσα στο 0 και το 2, δηλαδή $n = 4$, έχουμε

```
L = 2; n = 4;
fApproximation[t_] = FastFourierfit[f, L, n, t]
```

$$-0.0967056 - 0.113662\, e^{-i\pi t} - 0.113662\, e^{i\pi t} + 0.32403\, e^{-2i\pi t} + 0.32403\, e^{2i\pi t} - 0.113662\, e^{-3i\pi t} - 0.113662\, e^{3i\pi t}$$

ή την μη-μιγαδική μορφή της

```
fApproximationReal[t_] = Chop[ComplexExpand[fApproximation[t]]]

-0.0967056 - 0.227324 Cos[π t] + 0.64806 Cos[2 π t] - 0.227324 Cos[3 π t]
```

Π*ροσέξτε πως οι συντελεστές της fApproximation*(*t*) *και fApproximationReal*(*t*) *ικανοποιούν τις σχέσεις που αναφέρονται στον ορισμό*. Επιπλέον, *η f*(*x*) *προσεγγίζεται μόνον από συνημίτονα* (***συνημιτονική προσέγγιση***). Αυτό ήταν αναμενόμενο γιατί *η συνάρτηση f*(*x*) = cos(2π*x*) sin(1 - cos(3π*x*)) *είναι* ***άρτια***, *δηλαδή για την συνάρτηση evenf*(*x*), *που ορίζεται στο* ***επεκταμένο*** *διάστημα* $0 \le x \le 2L$, *έχουμε evenf*(*x*) = *f*(*x*), $0 \le x \le L$, *και evenf*(*x*) = *f*(2*L* - *x*), $L < x \le 2L$. Βλέπε *και το* Figure 1. Αργότερα *θα συναντήσουμε* ***περιττές*** *συναρτήσεις που προσεγγίζονται μόνον από ημίτονα* (***ημιτονική προσέγγιση***).

Τα *σχεδιαγράμματα των συναρτήσεων f*(*x*) *και fApproximationReal*(*t*) *φαίνονται στο* Figure 2.

```
fplot = Plot[f[x], {x, 0, L},
   PlotStyle → {Thickness[0.008], RGBColor[0, 0, 1]},
   AspectRatio → 1/GoldenRatio, DisplayFunction :→ Identity];

fapproxPlot = Plot[fApproximationReal[t], {t, 0, L}, PlotStyle →
    {{Thickness[0.008], RGBColor[1, 0, 0], Dashing[{0.03, 0.03}]}},
   AspectRatio → 1/GoldenRatio, DisplayFunction :→ Identity];

Show[fplot, fapproxPlot, DisplayFunction :→ $DisplayFunction,
  Epilog → {{RGBColor[1, 0, 0], Thickness[0.007],
     Line[{{0, 0}, {L, 0}}]}, {Text["One Period", {L/2, 0.1}]}}];
```

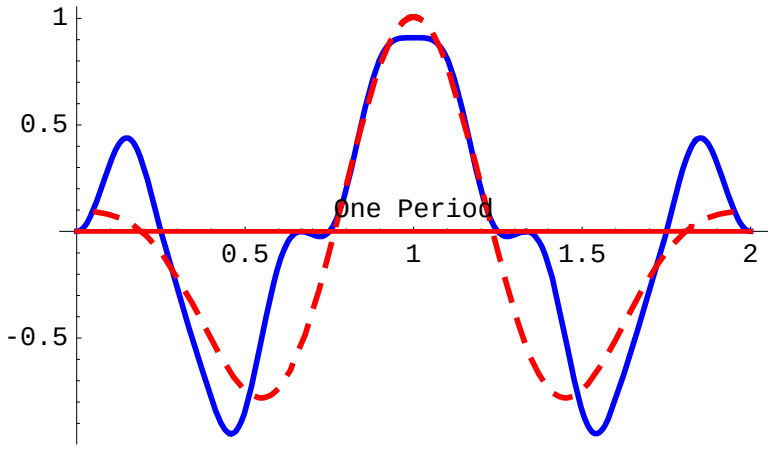


Figure 2

Η *διακεκομμένη* (*κόκκινη*) *γραμμή είναι το σχεδιάγραμμα της προσεγγιστικής συνάρτησης*. *Αυξάνοντας την τιμή του n προκύπτουν καλλίτερες προσεγγίσεις*.

Όπως βλέπουμε στο Figure 3, *η* `FastFourierfit[]` *διαλέγει* 2*n* - 1 *ισοαπέχοντα σημεία από την f*(*x*), *ανάμεσα στο x* = 0 *και x* = *L*, *και κατόπιν προσπαθεί να τα προσεγγίσει με έναν συνδιασμό από μιγαδικά εκθετικά*.

```
fdata = Table[N[{x, f[x]}], {x, 0, L - L/(2 n - 1), L/(2 n - 1)}];
fdataplot = ListPlot[fdata,
   PlotStyle → PointSize[0.02], DisplayFunction → Identity];
Show[fplot, fapproxPlot, fdataplot,
  DisplayFunction → $DisplayFunction];
```

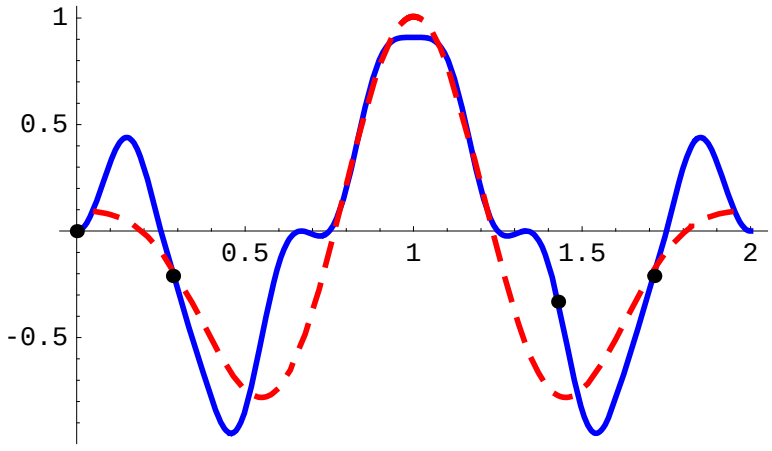


Figure 3

Η *διακεκομμένη* (*κόκκινη*) *γραμμή είναι το σχεδιάγραμμα της προσεγγιστικής συνάρτησης*.

Όπως *έχουμε ήδη αναφέρει*, *οι συντελεστές* c_k *του προσεγγιστικού πολυωνύμου στον ορισμό υπολογίζονται με τον γρήγορο μετασχηματισμό* Fourier *και τον αντίστροφό του — ενσωματωμένους στην συνάρτηση* `FastFourierfit[]`.

Ένας *άλλος τρόπος υπολογισμού των συντελεστών* c_k *είναι με την χρήση ολοκληρωμάτων*

$$c_k = \frac{1}{L}\int_0^L f(t)\, e^{-\frac{i\,k\,(2\pi)\,t}{L}}\, dt.$$

Η *μέθοδος αυτή λέγεται* ***ολοκληρωτική προσέγγιση*** **Fourier** ή integral Fourier fit.

Ο *παραπάνω τύπος με το ολοκλήρωμα προκύπτει αν* ***υποθέσουμε*** *ότι για κάποιο σταθερό* n, *η συνάρτηση* $f(t)$ *προσεγγίζεται από την συνάρτηση*

$$complexApproximation(t) = \sum_{k=-n}^{n} c_k\, e^{\frac{k\,(2\pi)\,i\,t}{L}},$$

όπου $L > 0$, *οπότε θέτουμε*

$$f(t) = complexApproximation(t).$$

Τότε *θα έχουμε*

$$\int_0^L complexApproximation(t)\, e^{-\frac{j\,(2\pi)\,i\,t}{L}}\, dt = \int_0^L f(t)\, e^{-\frac{j\,(2\pi)\,i\,t}{L}}\, dt.$$

Έχουμε *όμως*

$$\int_0^L complexApproximation(t)\, e^{-\frac{j\,(2\pi)\,i\,t}{L}}\, dt = Lc_j$$

απ' όπου και ο τύπος για τους συντελεστές.

Οι *προσεγγιστικές συναρτήσεις που προκύπτουν από την γρήγορη και ολοκληρωτική προσέγγιση* Fourier "*μοιάζουν*" *αρκετά και σχεδόν ταυτίζονται για μεγάλες τιμές του* n.

Το μειονέκτημα της ολοκληρωτικής προσέγγισης Fourier είναι ότι (αρκετά συχνά) τα ολοκληρώματα που πρέπει να υπολογισθούν είναι πολύ δύσκολα και "ακατάλληλα" ακόμα και για αριθμητική ολοκλήρωση. Παρ' όλα αυτά η μέθοδος είναι χρήσιμη για υπολογισμούς με το χέρι — πράγμα αδιανόητο για την γρήγορη προσέγγιση Fourier.

Το πλεονέκτημα της ολοκληρωτικής προσέγγισης Fourier είναι ότι σε θεωρητικά προβλήματα μας παρέχει έναν τύπο για τους υπολογισμούς μας. Όμως, ακόμα και σε αυτήν την περίπτωση, αφού αναπτύξουμε την θεωρία, στρεφόμαστε στην γρήγορη προσέγγιση Fourier.

- **Δ3: Η FFF "συναντά" τον μετασχηματισμό Laplace**

Στην υποενότητα αυτή συνδιάζουμε την γρήγορη προσέγγιση Fourier, `FastFourierfit[]` (FFF) και τον μετασχηματισμό Laplace για να βρούμε καλούς προσεγγιστικούς τύπους σε περιοδικά εξαναγκασμένους ταλαντωτές. Με μόνη εξαίρεση το βιβλίο των Bill Davis και Jerry Uhl, το θέμα αυτό "λάμπει δια της απουσίας" του από όλα τα βιβλία Διαφορικών Εξισώσεων. Όταν δε γίνεται αναφορά στον μετασχηματισμό Fourier αυτό πάντα σχετίζεται με τις εξίσωση θερμότητας και την κυματική εξίσωση (heat and wave equations).

Υπενθυμίζουμε πως η διαφορική εξίσωση της μορφής $y''(x) + by'(x) + cy(x) = f(x)$, με αρχικές τιμές $y(0)$ και $y'(0)$, και όπου $f(x)$ είναι περιοδική συνάρτηση, μπορεί να λυθεί ***είτε*** υπολογίζοντας ένα ολοκλήρωμα συνέλιξης της $f(x)$ ***είτε*** με εφαρμόζοντας τον μετασχηματισμό Laplace. Όμως, και στις δύο περιπτώσεις, τα ολοκληρώματα της $f(x)$ μπορεί να είναι πολύ δύσκολα και να μην μπορούν να λυθούν από κανένα σύστημα υπολογιστικής άλγεβρας.

Στις περιπτώσεις αυτές πρώτα βρίσκουμε μία καλή γρήγορη προσέγγιση Fourier (FFF) της $f(x)$ (με ημίτονα ή/και συνημίτονα) και κατόπιν εφαρμόζουμε οποιαδήποτε από τις δύο μεθόδους για να βρούμε έναν προσεγγιστικό τύπο της λύσης. Με άλλα λόγια, αντί να λύσουμε την εξίσωση $y''(x) + by'(x) + cy(x) = f(x)$ λύνουμε την εξίσωση $y''(t) + by'(t) + cy(t) = fApproximationReal(t)$ με αρχικές τιμές $y(0)$ and $y'(0)$.

Παράδειγμα: Έστω ότι θέλουμε να λύσουμε την διαφορική εξίσωση $y''(x) + 2y'(x) + 20y(x) = 1 - e^{\sin(\pi x)}$ με $y(0) = 2$ και $y'(0) = -4$. Η περιοδική συνάρτηση $f(x) = 1 - e^{\sin(\pi x)}$ φαίνεται στο Figure 4.

```
f[x_] := 1 - e^Sin[π x];
L = 2;
cycles = 2;
Plot[f[x], {x, 0, cycles L},
 AxesLabel → {"x", "f(x) = 1-e^sin(π x)"},
 PlotStyle → {{Thickness[0.007], RGBColor[0, 0, 1]}},
 PlotLabel → "cycles" cycles, Epilog →
  {{RGBColor[1, 0, 0], Thickness[0.007], Line[{{0, 0}, {L, 0}}]},
   {Text["One Period", {L/2, 0.1}]}}];
```

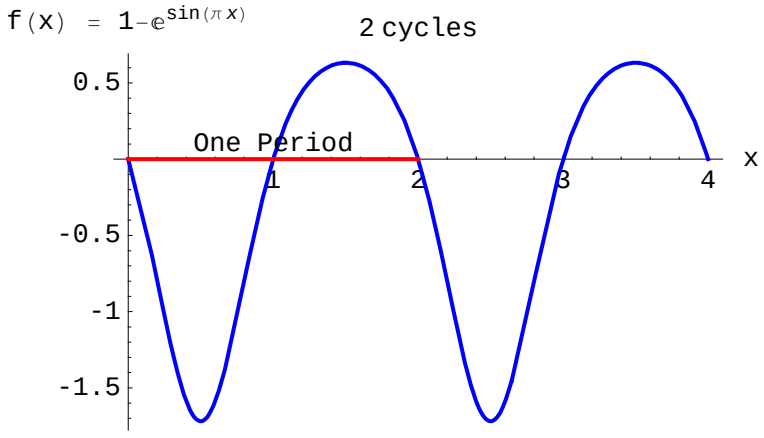


Figure 4

Είναι αδύνατο να βρούμε μία ακριβή λύση της $y''(x) + 2y'(x) + 20y(x) = 1 - e^{\sin(\pi x)}$. Η συνάρτηση `DSolve[]` του Mathematica "κολλάει" γιατί τα ολοκληρώματα είναι πολύ πολύπλοκα.

```
DSolve[{y''[x] + 2 y'[x] + 20 y[x] == 1 - E^Sin[π x], y[0] == 2 , y'[0] == -4},
  y, x] // AbsoluteTiming
```

Όμως, με την βοήθεια της συνάρτησης `NDSolve[]` σχεδιάζουμε την προσεγγιστική λύση στο Figure 5 — χωρίς να την ξέρουμε.

```
sol = NDSolve[{y''[x] + 2 y'[x] + 20 y[x] == 1 - E^Sin[π x],
   y[0] == 2 , y'[0] == -4}, y, {x, 0, 5}]
```

```
{{y → InterpolatingFunction[{{0., 5.}}, <>]}}
```

```
Plot[y[x] /. sol, {x, 0, 5}, PlotRange → All,
  PlotStyle → {{Thickness[0.008], RGBColor[0, 0, 1]}}];
```

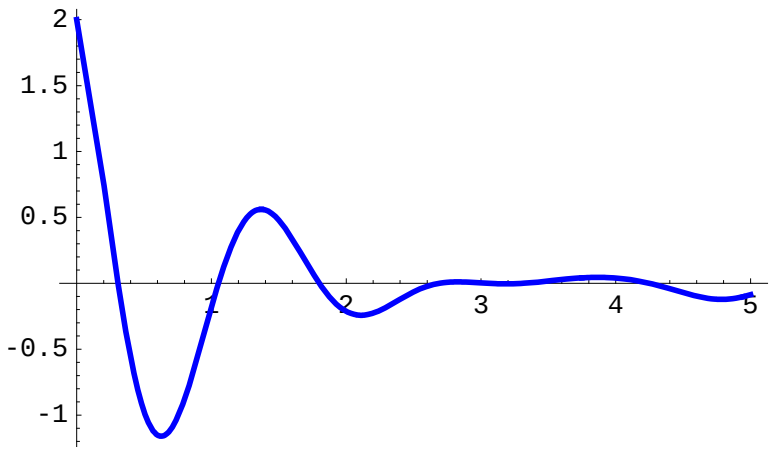


Figure 5

Όπως είπαμε, στην περίπτωση αυτή βρίσκουμε πρώτα την *fApproximationReal*(*t*), μία καλή γρήγορη προσέγγιση Fourier (FFF) της $f(x) = 1 - e^{\sin(\pi x)}$ (με ημίτονα ή/και συνημίτονα).

```
fApproximationReal[t_] =
 Chop[ComplexExpand[FastFourierfit[f, L = 2, n = 4, t] ]]
```

```
-0.266066 + 0.27154 Cos[2 π t] - 1.13032 Sin[π t] + 0.0448798 Sin[3 π t]
```

Κατόπιν εύκολα βρίσκουμε την *LTyApproximation*(*s*), τον μετασχηματισμό Laplace του προσεγγιστικού τύπου της λύσης της εξίσωσης $y''(t) + 2y'(t) + 20y(t) = fApproximationReal(t)$ με $y(0) = 2$ and $y'(0) = -4$.

```
b = 2; c = 20;
ystarter = 2; yprimestarter = -4; LTyApproximation[s_] =
  1/(s^2 + b s + c) (LaplaceTransform[fApproximationReal[t], t, s] +
    2 ystarter + s ystarter + yprimestarter)
```

$$\frac{1}{20 + 2\,s + s^2}\left(-\frac{0.266066}{s} + 2\,s - \frac{3.55101}{\pi^2 + s^2} + \frac{0.27154\,s}{4\,\pi^2 + s^2} + \frac{0.422982}{9\,\pi^2 + s^2}\right)$$

Τέλος, *yApproximation*(*t*), *ο τύπος για την προσεγγιστική λύση, βρίσκεται με την βοήθεια του αντίστροφου μετασχηματισμού* Laplace.

```
yApproximation[t_] = Chop[ComplexExpand[
   InverseLaplaceTransform[LTyApproximation[s], s, t]
  ]]
```

```
-0.0133033 + 0.0499778 Cos[3.14159 t] + 1.97334 e^(-1. t) Cos[4.3589 t] -
 0.00984358 Cos[6.28319 t] - 0.000166123 Cos[9.42478 t] -
 0.0805794 Sin[3.14159 t] - 0.414715 e^(-1. t) Sin[4.3589 t] +
 0.00635052 Sin[6.28319 t] - 0.000606575 Sin[9.42478 t]
```

Στο Figure 6 *συγκρίνουμε το σχεδιάγραμμα της* "άγνωστης" *λύσης που αποκτήσαμε με την* `NDSolve[]` *με εκείνο του προσεγγιστικού τύπου για την λύση — η (κόκκινη) παχύτερη διακεκομμένη γραμμή.* Ταυτίζονται πλήρως!

```
Plot[{y[t] /. sol, yApproximation[t]}, {t, 0, 5}, PlotRange → All,
  PlotStyle → {{Thickness[0.008], RGBColor[0, 0, 1]},
    {Thickness[0.014], RGBColor[1, 0, 0], Dashing[{0.05, 0.07}]}},
  AxesLabel → {"t", "y(t)"}, AspectRatio → 1/GoldenRatio];
```

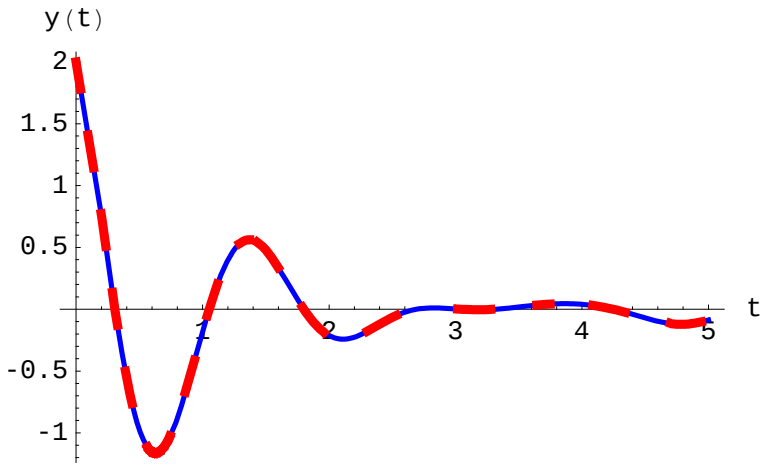


Figure 6

- **Δ4: Η FFF "*ανακαλύπτει*" *τριγωνομετρικές ταυτότητες***

Μια *άλλη ενδιαφέρουσα εφαρμογή της* `FastFourierfit[]` (FFF) *είναι στην* "*ανακάλυψη*" *τριγωνομετρικών ταυτοτήτων*.

Ξέρουμε *για παράδειγμα την τριγωνομετρική ταυτότητα* $2\sin(a)\sin(b) = \cos(a - b) - \cos(a + b)$. Ας *υποθέσουμε προς στιγμήν ότι η ταυτότητα αυτή μας είναι άγνωστη και μας δίνεται η έκφραση* $\sin(3t)\sin(7t)$. Πως *θα την απλοποιήσουμε*; Ένας *τρόπος είναι να χρησιμοποιήσουμε την συνάρτηση* `TrigReduce[]` *του Mathematica*'s

```
TrigReduce[Sin[3 t] Sin[7 t]]
```

$$\frac{1}{2}\,(\text{Cos}[4\,t] - \text{Cos}[10\,t])$$

αλλά εμείς θα γράψουμε την `trigIdentityFinder[]`, *μία νέα συνάρτηση που χρησιμοποιεί την* FFF.

Η *συνάρτησή μας* `trigIdentityFinder[]` *χρησιμοποιεί την* `FastFourierfit[]` *και προσεγγίζει την* $\sin(3t)\sin(7t)$ *για διάφορες τιμές του* n, *μέχρις ότου το αποτέλεσμα δεν αλλάζει*. Αυτό *είναι και η ζητούμενη ταυτότητα*. Έτσι *έχουμε*

```
trigIdentityFinder[f_] := Module[{L = 2 π, old = 0, n = 2, new},
  new = Chop[ComplexExpand[FastFourierfit[f, L, n, t]]];
  While[! Chop[new - old] === 0, old = new; ++n;
   new = Chop[ComplexExpand[FastFourierfit[f, L, n, t]]]];
  Print[n - 1, "  iterations and the identity is:   ", f[t], " =  "];
  Factor[Rationalize[new]]]
```

και η ταυτότητα για το παράδειγμά μας βρίσκεται με 11 *προσεγγίσεις*.

```
f[t_] = Sin[3 t] Sin[7 t];
trigIdentityFinder[f]
```

```
11  iterations and the identity is:   Sin[3 t] Sin[7 t] =

1/2 (Cos[4 t] - Cos[10 t])
```

Τ*ελειώνουμε το θέμα αυτό με ένα ακόμα παράδειγμα*. Όπ*ως και πριν συγκρίνουμε το αποτέλεσμά μας με εκείνο του Mathematica*.

```
f[t_] = Sin[t]^12; trigIdentityFinder[f]
```

```
13  iterations and the identity is:   Sin[t]^12 =

1/2048 (462 - 792 Cos[2 t] + 495 Cos[4 t] -
   220 Cos[6 t] + 66 Cos[8 t] - 12 Cos[10 t] + Cos[12 t])
```

```
TrigReduce[f[t]]
```

```
1/2048 (462 - 792 Cos[2 t] + 495 Cos[4 t] -
   220 Cos[6 t] + 66 Cos[8 t] - 12 Cos[10 t] + Cos[12 t])
```

- **Δ5: Η FFF "*αποδεικνύει*"*την εξίσωση θερμότητας και την κυματική εξίσωση***

Σ*την υποενότητα αυτή αποδεικνύουμε την χρησιμότητα και δύναμη της* FFF *αποδεικνύοντας την εξίσωση θερμότητας και την κυματική εξίσωση*. Π*ροηγούνται όμως μερικές προκαταρκτικές έννοιες*.

Προκαταρκτικές έννοιες

Όπως ξέρουμε, *χρησιμοποιώντας την* FFF *συνήθως παίρνουμε ένα μίγμα από ημίτονα και συνημίτονα*. Μερικές *φορές όμως παίρνουμε μόνο ημίτονα*, *όπως στο παράδειγμα*

```
f[x_] = x (1 - x) (2 - x);
Chop[ComplexExpand[FastFourierfit[f, L = 2, n = 4, t]]]
```

```
0.386374 Sin[π t] + 0.046875 Sin[2 π t] + 0.0113738 Sin[3 π t]
```

και άλλες φορές παίρνουμε μόνο συνημίτονα, *όπως στο παράδειγμα*

```
f[x_] = x (1 - x)^2 (2 - x);
Chop[ComplexExpand[FastFourierfit[f, L = 2, n = 4, t]]]
```

```
0.123047 + 0.0662913 Cos[π t] - 0.09375 Cos[2 π t] - 0.0662913 Cos[3 π t]
```

Θ*α προσέξουμε τις προσεγγίσεις με ημίτονα μόνο* (*ημιτονικές*), *οι οποίες προσεγγίζουν* ***περιττές*** *συναρτήσεις*. (Τ*ις άρτιες συναρτήσεις τις αναφέραμε νωρίτερα*.)

Γ*ια να προσεγγίσουμε μόνο με ημίτονα μία συνάρτηση* $f(x)$ *στο διάστημα* $0 \le x \le L$, *πρέπει να ικανοποιούνται δύο πράγματα*: (**a**) $f(0) = f(L) = 0$, *και* (**b**) *για την συνάρτηση* $oddf(x)$, *που ορίζεται στο* ***επεκταμένο*** *διάστημα* $0 \le x \le 2L$, *έχουμε* $oddf(x) = f(x)$, $0 \le x \le L$, *και* $oddf(x) = -f(2L - x)$, $L < x \le 2L$.

Με *άλλα λόγια*, *το μέρος του σχεδιαγράμματος δεξιά της* (*νοητής*) *κεντρικής γραμμής είναι το αντεστραμμένο είδωλο του σχεδιαγράμματος αριστερά της* (*νοητής*) *κεντρικής γραμμής*.

Η *σχέση* $f(0) = f(L) = 0$, *που πρέπει να ικανοποιείται*, *προέρχεται από την γρήγορη προσέγγιση* Fourier *της* $oddf(x)$ *στο διάστημα* $0 \le x \le 2L$. Αυτό *σημαίνει πως η συνάρτηση* $oddf(x)$ *προσεγγίζεται από τις συναρτήσεις* $\sin(\frac{k\pi t}{L})$, *που όλες*

μηδενίζονται στα σημεία $t = 0$ *και* $t = L$. Επομένως, *αν θέλουμε μία καλή ημιτονική προσέγγιση της* $f(x)$ *στο διάστημα* $0 \le x \le L$ *πρέπει να έχουμε* $f(0) = f(L) = 0$.

Αν *μία συνάρτηση* $f(x)$ *προσεγγίζεται από ένα μίγμα ημιτόνων και συνημιτόνων, υπάρχει τρόπος να προσεγγισθεί μόνο από ημίτονα. Συγκεκριμένα, αν η* $f(x)$ *ορίζεται στο διάστημα* $0 \le x \le L$, *και* $f(0) = f(L) = 0$, *ορίζουμε την νέα συνάρτηση,* $oddf(x)$ *στο* ***επεκταμένο*** *διάστημα* $0 \le x \le 2L$ *ως εξής*:

$$\begin{aligned} oddf(x) &= f(x), && 0 \le x \le L, \\ oddf(x) &= -f(2L - x), && L < x \le 2L. \end{aligned}$$

Προφανώς, *αυτή η νέα συνάρτηση,* $odd(x)$, *θα προσεγγίζεται μόνο από ημίτονα στο διάστημα* $0 \le x \le 2L$.

Παράδειγμα: Ας *θεωρήσουμε την συνάρτηση* $f(x) = 6x\,(4 - x)\,e^{-x}$, *η οποία, στο διάστημα* $0 \le x \le L = 4$, *προσεγγίζεται από ημίτονα και συνημίτονα*.

```
f[x_] = 6 x (4 - x) E^-x;
Chop[ComplexExpand[FastFourierfit[f, L = 4, n = 3, t]]]
```

$$2.94583 - 1.04677\,\text{Cos}\left[\frac{\pi\,t}{2}\right] - 1.32181\,\text{Cos}[\pi\,t] + 3.03426\,\text{Sin}\left[\frac{\pi\,t}{2}\right] + 0.643404\,\text{Sin}[\pi\,t]$$

Επειδή *ισχύει* $f(0) = f(L) = 0$, *μπορούμε να ορίσουμε μία καινούργια συνάρτηση* $oddf(x)$, *η οποία έχει ημιτονική προσέγγιση στο διάστημα* $0 \le x \le 2L$.

```
oddf[x_] := f[x] /; 0 ≤ x ≤ L;
oddf[x_] := -f[2 L - x] /; L < x ≤ 2 L;
Chop[ComplexExpand[FastFourierfit[oddf, 2 L, 3, t]]]
```

$$4.10249\,\text{Sin}\left[\frac{\pi\,t}{4}\right] + 2.39086\,\text{Sin}\left[\frac{\pi\,t}{2}\right]$$

Στην *περίπτωση που* ***δεν ισχύει*** $f(0) = f(L) = 0$ *τότε η συνάρτηση* $f(x)$ *χρειάζεται μία "προεργασία" για να προσεγγισθεί μόνο από ημίτονα. Συγκεκριμένα, περνάμε την ευθεία* $line(x) = f(0) + \frac{f(L) - f(0)}{L}\,x$ *από τα ακραία σημεία* $f(0)$ *και* $f(L)$, *και*

κατόπιν ορίζουμε την "προσαρμοσμένη" συνάρτηση adjustedf(x) = *f*(x) - *line*(x), *για την οποία ισχύει* which *adjustedf*(0) = *adjustedf*(L) = 0.

Παράδειγμα: *Έστω η συνάρτηση f*(x) = 3|0.25x - ⟨0.25x⟩| + 1, *ορισμένη στο διάστημα* 0 ≤ x ≤ L = 3, *και όπου* ⟨0.25x⟩ *δηλώνει τον ακέραιο πλησιέστερα στον αριθμό* 0.25x. *Για την συνάρτηση αυτή έχουμε* 0 ≠ *f*(0) ≠ *f*(L) ≠ 0. *Η γρήγορη προσέγγιση* Fourier *περιλαμβάνει ημίτονα και συνημίτονα*.

```
f[x_] = 3 Abs[0.25 x - Round[0.25 x]] + 1; {f[0], f[L = 3]}
```

```
{1, 1.75}
```

```
Chop[ComplexExpand[FastFourierfit[f, L = 3, n = 4, t]]]
```

$$1.82031 - 0.446978\,\text{Cos}\left[\frac{2\pi t}{3}\right] - 0.1875\,\text{Cos}\left[\frac{4\pi t}{3}\right] - 0.115522\,\text{Cos}[2\pi t] - 0.419519\,\text{Sin}\left[\frac{2\pi t}{3}\right] - 0.046875\,\text{Sin}\left[\frac{4\pi t}{3}\right] - 0.0445194\,\text{Sin}[2\pi t]$$

Επειδή *f*(0) ≠ *f*(L) *δεν μπορούμε προφανώς να προσεγγίσουμε την f*(x) *μόνο με ημίτονα*. *Για να διορθώσουμε το πρόβλημα περνάμε την ευθεία από τα ακραία σημεία και ορίζουμε την "προσαρμοσμένη" συνάρτηση adjustedf*(x) = *f*(x) - *line*(x). *Όπως αναφέραμε ισχύει adjustedf*(0) = *adjustedf*(L) = 0. *Βλέπε* Figure 7.

```
line[x_] = (f[L] - f[0]) x / L + f[0]; adjustedf[x_] = f[x] - line[x];
Plot[adjustedf[x], {x, 0, L},
  PlotStyle → {{Thickness[0.01], RGBColor[0, 0, 1]}},
  AxesLabel → {"x", "adjustedf"}];
```

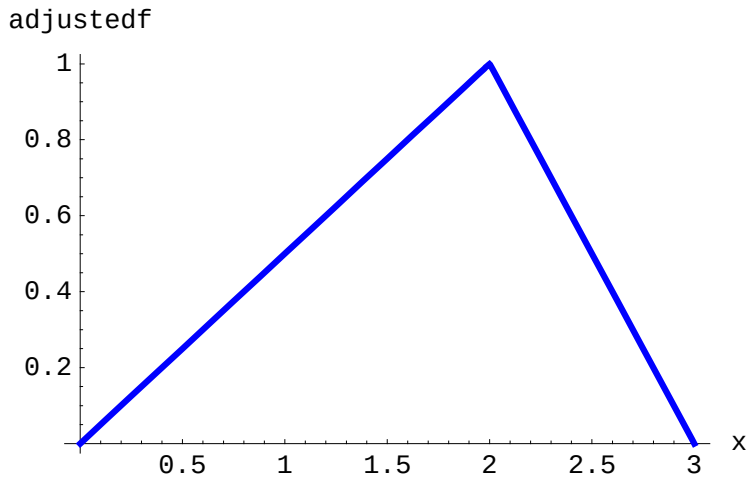


Figure 7

Στην "προσαρμοσμένη" συνάρτηση *adjustedf*(*x*) εφαρμόζουμε τώρα την γνωστή μας τεχνική και την προσεγγίζουμε μόνο με ημίτονα.

```
oddAdjustedf[x_] := adjustedf[x] /; 0 ≤ x ≤ L;
oddAdjustedf[x_] := -adjustedf[2 L - x] /; L < x ≤ 2 L;

Chop[ComplexExpand[FastFourierfit[oddAdjustedf, 2 L, n = 4, t]]]
```

$$0.772748\,\text{Sin}\left[\frac{\pi t}{3}\right] - 0.1875\,\text{Sin}\left[\frac{2\pi t}{3}\right] + 0.0227476\,\text{Sin}[\pi t]$$

Με αυτά τα προκαταρκτικά είμαστε τώρα έτοιμοι για το κύριο θέμα μας.

Η εξίσωση θερμότητας $\frac{\partial^2 temp(x,t)}{\partial x^2} = \frac{\partial temp(x,t)}{\partial t}$

Πρόβλημα: Αρχίζουμε με ένα θερμασμένο σύρμα μήκους *L* μονάδες, στο οποίο η θερμοκρασία μπορεί να διαφέρει από σημείο σε σημείο. Η συνάρτηση *startertemp*(*x*) δίνει την θερμοκρασία στο σημείο *x*, $0 \le x \le L$, του σύρματος στην αρχή του πειράματος. Εξ αιτίας των όσων είπαμε προηγουμένως, θεωρούμε χωρίς περιορισμό της γενικότητας συναρτήσεις *startertemp*(*x*) για τις οποίες *startertemp*(0) = *startertemp*(*L*) = 0, ώστε να μπορούμε να τις προσεγγίσουμε μόνο με ημίτονα.

Θεωρούμε ότι το σύρμα είναι το διάστημα $0 \le x \le L$. Στην αρχή του πειράματος κρυώνουμε τα άκρα στα σημεία $x = 0$ και $x = L$ και διατηρούμε τις θερμοκρασίες

τους στους 0 βαθμούς. Επίσης φροντίζουμε να απομονώσουμε εντελώς το υπόλοιπο σύρμα.

Θέλουμε να βρούμε τον τύπο για την συνάρτηση θερμότητας *temp*(*x*, *t*), ώστε να παίρνουμε την θερμοκρασία του σύρματος στο σημείο *x*, στην χρονική στιγμή *t* μετά την έναρξη του πειράματος.

Λύση με την FFF: Χρησιμοποιώντας την FFF βρίσκουμε εύκολα τον τύπο για την συνάρτηση θερμότητας *temp*(*x*, *t*), ως εξής:

α. Επειδή ισχύει *startertemp*(0) = *startertemp*(*L*) = 0 κάνουμε την συνάρτηση αυτή περιττή (με τον γνωστό μας τρόπο) και προσεγγίζουμε την τελευταία μόνο με ημίτονα — με τον γρήγορο μετασχηματισμό Fourier, για κάποια τιμή *n*.

β. Επιλέγουμε τους συντελεστές *A*(*k*) των όρων $\sin(\frac{k\pi x}{L})$ που προσεγγίζουν την περιττή συνάρτηση του βήματος (**α**), και

γ. Ο τύπος για την συνάρτηση *temp*(*x*, *t*) είναι:

$$temp(x, t) = \sum_{k=1}^{n} A(k)\, e^{-(\frac{k\pi}{L})^2 t} \sin(\frac{k\pi x}{L})$$

Σημείωση: Οι όροι $\sin(\frac{k\pi t}{L})$ στην προσέγγιση της περιττής συνάρτησης του βήματος (**α**) γράφονται σαν $\sin(\frac{k\pi x}{L})$, δηλαδή τα *t* αντικαθίστανται με τα *x*.

Η κεντρική ερώτηση που πρέπει να απαντηθεί είναι από που προέρχονται οι όροι $e^{-(\frac{k\pi}{L})^2 t}$. Η απάντηση ακολουθεί.

Μελέτες έδειξαν πως μετά από κατάλληλες προσαρμογές μονάδων η συνάρτηση *temp*(*x*, *t*) ικανοποιεί την μερική διαφορική εξίσωση $\frac{\partial^2 temp(x,t)}{\partial x^2} = \frac{\partial temp(x,t)}{\partial t}$, γνωστή και σαν **εξίσωση θερμότητας**. Δηλαδή, η δεύτερη παράγωγος της *temp*(*x*, *t*) ως προς *x* ισούται με την πρώτη παράγωγο της *temp*(*x*, *t*) ως προς *t*.

Οι συνοριακές συνθήκες αυτής της διαφορικής εξίσωσης είναι:

$$temp(x, 0) = startertemp(x),$$
$$temp(0, t) = 0 \text{ και } temp(L, t) = 0, \forall t.$$

Από *αυτές, οι βασικές συνοριακές συνθήκες είναι*

$$temp(0, t) = 0 \text{ και } temp(L, t) = 0, \forall t,$$

που συμφωνούν με το γεγονός ότι $\sin(\frac{k\pi x}{L}) = 0$ *για* $x = 0$ *και* $x = L$ *για όλους τους θετικούς ακεραίους* k. Αυτό *σημαίνει πως για κάθε σταθερή τιμή* t, *και οποιαδήποτε τιμή* n, *προσεγγίζουμε την* $temp(x, t)$ *μόνο με ημίτονα με την βούθεια της συνάρτησης*

$$approxtemp(x, t) = \sum_{k=1}^{n} u(t, k) \sin(\frac{k\pi x}{L}),$$

όπου οι συντελεστές της προσέγγισης Fourier, $u(t, k)$, *πρέπει να υπολογισθούν*. Π*ροσέξτε πως αυτοί οι συντελεστές εξαρτώνται από το* t *και το* k *διοτι περιμένουμε διαφορετική ημιτονική προσέγγιση σε διάφορες χρονικές στιγμές* t.

Η ***εξίσωση θερμότητας*** *λέει* $\frac{\partial^2 temp(x, t)}{\partial x^2} = \frac{\partial temp(x, t)}{\partial t}$. Αντί *για την* $temp(x, t)$ *χρησιμοποιούμε την προσέγγισή της,* $approxtemp(x, t)$, *στην εξίσωση θερμότητας, και βλέπουμε πως*

$$\sum_{k=1}^{n} u(t, k) (\frac{k\pi}{L})^2 (-\sin(\frac{k\pi x}{L})) = \sum_{k=1}^{n} \frac{\partial u(t, k)}{\partial t} \sin(\frac{k\pi x}{L}).$$

Η *εξίσωση αυτή όμως ισχύει μόνο στην περίπτωση που ισχύει η διαφορική εξίσωση*

$$\frac{\partial u(t, k)}{\partial t} = - u(t, k) (\frac{k\pi}{L})^2.$$

Αλλά *ξέρουμε πως η λύση αυτής της διαφορικής εξίσωσης είναι*

$$u(t, k) = A(k) e^{-(\frac{k\pi}{L})^2 t}.$$

Έ*τσι για υπολογίσουμε τους συντελεστές της προσέγγισης* Fourier *πρέπει να υπολογίσουμε τους συντελεστές* $A(k)$.

Α*ντικαθιστώντας αυτά τα* $u(t, k)$ *στην* $approxtemp(x, t)$ *προκύπτει*

$$approxtemp(x, t) = \sum_{k=1}^{n} A(k) e^{-(\frac{k\pi}{L})^2 t} \sin(\frac{k\pi x}{L}),$$

η οποία για $t = 0$ *γίνεται*

$$approxtemp(x, 0) = \sum_{k=1}^{n} A(k) \sin(\frac{k \pi x}{L}).$$

Αυτή *είναι η προσέγγιση της αρχικής θερμότητας, και έτσι παίρνουμε τα* $A(k)$'s *από την ημιτονική προσέγγιση της* $startertemp(x)$ — *αφού βέβαια την κάνουμε πρώτα περιττή*.

Παράδειγμα: *Στην αρχή ενός πειράματος, η θερμοκρασία του σύρματος στην θέση* x (*για* $0 \le x \le L = 3$) *δίνεται από την συνάρτηση* $startertemp(x) = 0.2 \sin^2(2x)(x - 3)$.

Για αυτήν την συνάρτηση έχουμε $startertemp(0) = startertemp(L) = 0$ *και το σχεδιάγραμμά της είναι στο* Figure 8.

```
L = 3;
startertemp[x_] = 0.2 Sin[2 x]^2 (x - 3)^2;
{startertemp[0], startertemp[L]}
```

```
{0, 0}
```

```
Plot[startertemp[x], {x, 0, L}, PlotStyle →
   {{Thickness[0.007], RGBColor[0, 0, 1]}}, AxesLabel → {"x", ""}];
```

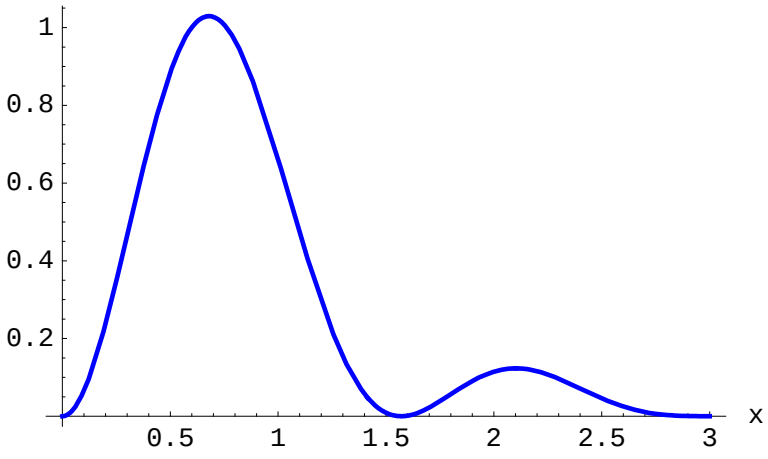


Figure 8

Τ*ην συνάρτηση startertemp*(*x*) *την κάνουμε πρώτα περιττή και μετά αυθαίρετα θέτοντες n* = 8 *υπολογίζουμε με την* FFF *την ημιτονική της προσέγγιση*. *Δηλαδή έχουμε*

```
oddStartertemp[x_] := startertemp[x] /; 0 ≤ x ≤ L;
oddStartertemp[x_] := -startertemp[2 L - x] /; L < x ≤ 2 L;

sinesOnlyfit[t_] =
 Chop[ComplexExpand[FastFourierfit[oddStartertemp, 2 L, n = 8, x]]]
```

$$0.379996 \sin\left[\frac{\pi x}{3}\right] + 0.395139 \sin\left[\frac{2\pi x}{3}\right] + 0.298906 \sin[\pi x] + 0.0691479 \sin\left[\frac{4\pi x}{3}\right] - 0.0908022 \sin\left[\frac{5\pi x}{3}\right] - 0.0548269 \sin[2\pi x] - 0.0186735 \sin\left[\frac{7\pi x}{3}\right]$$

Κ*ατόπιν παίρνουμε τους συντελεστές των* $\sin(\frac{k\pi t}{L})$:

```
A[k_] := Coefficient[sinesOnlyfit[t], Sin[(k π x)/L]];
Table[A[k], {k, 1, n}]
```

```
{0.379996, 0.395139, 0.298906,
 0.0691479, -0.0908022, -0.0548269, -0.0186735, 0}
```

και η συνάρτηση temp(*x*, *t*) *είναι*:

$$\text{temp}[x\_, t\_] = \sum_{k=1}^{n} A[k]\, E^{-(\frac{k\pi}{L})^2 t} \sin\left[\frac{(k\pi)\, x}{L}\right]$$

$$0.379996\, e^{-\frac{\pi^2 t}{9}} \sin\left[\frac{\pi x}{3}\right] + 0.395139\, e^{-\frac{4\pi^2 t}{9}} \sin\left[\frac{2\pi x}{3}\right] + 0.298906\, e^{-\pi^2 t} \sin[\pi x] + 0.0691479\, e^{-\frac{16\pi^2 t}{9}} \sin\left[\frac{4\pi x}{3}\right] - 0.0908022\, e^{-\frac{25\pi^2 t}{9}} \sin\left[\frac{5\pi x}{3}\right] - 0.0548269\, e^{-4\pi^2 t} \sin[2\pi x] - 0.0186735\, e^{-\frac{49\pi^2 t}{9}} \sin\left[\frac{7\pi x}{3}\right]$$

Έ*τσι έχουμε για παράδειγμα*

```
temp[2.5, 1]
```

```
0.0592159
```

Η *κυματική εξίσωση* $\frac{\partial^2 \, position(x,t)}{\partial \, x^2} = \frac{\partial^2 \, position(x,t)}{\partial \, t^2}$

Ας *δούμε τώρα την κυματική εξίσωση*.

Πρόβλημα: Α*ρχίζουμε με μία ελαστική χορδή στερεωμένη στα σημεία* 0 *και* L *του άξονα των* x. Η *χορδή τεντώνεται μέχρι μία αρχική θέση και κατόπιν αφήνεται να πάλλεται μόνο της, με αρχική ταχύτητα* 0. Η *συνάρτηση starterposition*(x) *δίνει την θέση της χορδής στο σημείο* x, *για* $0 \le x \le L$. *Όπως και στην περίπτωση της εξίσωσης θερμότητας, θεωρούμε χωρίς περιορισμό της γενικότητας συναρτήσεις starterposition*(x) *για τις οποίες ισχύει starterposition*(0) = *starterposition*(L) = 0, *ώστε να μπορούμε να τις προσεγγίσουμε μόνο με ημίτονα*.

Θέλουμε *να βρούμε τον τύπο για την συνάρτηση θέσης position*(x, t), *ώστε να παίρνουμε την θεση της χορδής στο σημείο* x, *στην χρονική στιγμή* t *μετά την έναρξη του πειράματος*.

Λύση *με την* FFF: Χ*ρησιμοποιώντας την* FFF *βρίσκουμε εύκολα τον τύπο για την συνάρτηση θεσης position*(x, t), *ως εξής*:

α. Ε*πειδή ισχύει starterposition*(0) = *starterposition*(L) = 0 *κάνουμε την συνάρτηση αυτή περιττή* (*με τον γνωστό μας τρόπο*) *και προσεγγίζουμε την τελευταία μόνο με ημίτονα — με τον γρήγορο μετασχηματισμό* Fourier, *για κάποια τιμή* n.

β. Ε*πιλέγουμε τους συντελεστές* $A(k)$ *των όρων* $\sin(\frac{k\pi x}{L})$ *που προσεγγίζουν την περιττή συνάρτηση του βήματος* (***α***), *και*

γ. Ο *τύπος για την συνάρτηση position*(x, t) *είναι*:

$$position(x, t) = \sum_{k=1}^{n} A(k) \cos(\frac{k\pi t}{L}) \sin(\frac{k\pi x}{L})$$

Σ*ημείωση*: Οι *όροι* $\sin(\frac{k\pi t}{L})$ *στην προσέγγιση της περιττής συνάρτησης του βήματος* (***α***) *γράφονται σαν* $\sin(\frac{k\pi x}{L})$, *δηλαδή τα* t *αντικαθίστανται από τα* x.

Η *κεντρική ερώτηση που πρέπει να απαντηθεί είναι από που προέρχονται οι όροι* $\cos(\frac{k\pi t}{L})$. Η *απάντηση ακολουθεί*.

Μελέτες *έδειξαν πως μετά από κατάλληλες προσαρμογές μονάδων η συνάρτηση position*(x, t) *ικανοποιεί την μερική διαφορική εξίσωση* $\frac{\partial^2 position(x,t)}{\partial x^2} = \frac{\partial^2 position(x,t)}{\partial t^2}$, *γνωστή και σαν* ***κυματική εξίσωση***. Δηλαδή, *η δεύτερη παράγωγος της position*(x, t) *ως προς x ισούται με την δεύτρη παράγωγο της position*(x, t) *ως προς t*.

Οι *συνοριακές συνθήκες αυτής της διαφορικής εξίσωσης είναι*:

$$position(x, 0) = starterposition(x),$$
$$position(0, t) = 0 \text{ και } position(L, t) = 0, \forall t,$$

επειδή τα άκρα της χορδής είναι στερεωμένα. Επίσης *έχουμε*

$$\frac{\partial position(x,t)}{\partial t} = 0 \text{ για } t = 0,$$

επειδή δίνουμε στην χορδή αρχική ταχύτητα 0.

Από *τις παραπάνω, οι βασικές συνοριακές συνθήκες είναι οι*

$$position(0, t) = 0 \text{ και } position(L, t) = 0, \forall t,$$

που συμφωνούν με το γεγονός ότι $\sin(\frac{k\pi x}{L}) = 0$ *για* $x = 0$ *και* $x = L$ *για όλους τους θετικούς ακεραίους k*. Αυτό *σημαίνει πως για κάθε σταθερή τιμή t, και οποιαδήποτε τιμή n, προσεγγίζουμε την position*(x, t) *μόνο με ημίτονα με την βοήθεια της συνάρτησης*

$$approxposition(x, t) = \sum_{k=1}^{n} u(t, k) \sin(\frac{k\pi x}{L}),$$

όπου οι συντελεστές της προσέγγισης Fourier, $u(t, k)$, *πρέπει να υπολογισθούν*. Π*ροσέξτε πως αυτοί οι συντελεστές εξαρτώνται από το t και το k διοτι περιμένουμε διαφορετική ημιτονική προσέγγιση σε διάφορες χρονικές στιγμές t*.

Η ***κυματική εξίσωση*** λέει $\frac{\partial^2 position(x,t)}{\partial x^2} = \frac{\partial^2 position(x,t)}{\partial t^2}$. Αντί για την *position*(*x*, *t*) χρησιμοποιούμε την προσέγγισή της, *approxposition*(*x*, *t*), στην κυματική εξίσωση, και βλέπουμε πως

$$\sum_{k=1}^{n} u(t,\ k)\,(\tfrac{k\pi}{L})^2\,(-\sin(\tfrac{k\pi x}{L})) \;=\; \sum_{k=1}^{n} \tfrac{\partial^2 u(t,k)}{\partial t^2}\,\sin(\tfrac{k\pi x}{L})$$

Η εξίσωση αυτή όμως ισχύει μόνο στην περίπτωση που ισχύει η εκθετική διαφορική εξίσωση

$$\tfrac{\partial^2 u(t,k)}{\partial t^2} = -\,u(t,\ k)\,(\tfrac{k\pi}{L})^2.$$

Αλλά ξέρουμε πως η λύση αυτής της διαφορικής εξίσωσης (που είναι μη εξαναγκασμένος ταλαντωτής χωρίς απόσβεση) είναι

$$u(t, k) = A(k)\cos(\tfrac{k\pi t}{L}) + B(k)\sin(\tfrac{k\pi t}{L}).$$

Έτσι για υπολογίσουμε τους συντελεστές της προσέγγισης Fourier πρέπει να υπολογίσουμε τους συντελεστές $A(k)$ και $B(k)$.

Εφαρμόζοντας την συνθήκη $\frac{\partial position(x,t)}{\partial t} = 0$ για $t = 0$, στην συνάρτηση $approxposition(x, t) = \sum_{k=1}^{n} u(t,\ k)\sin(\frac{k\pi x}{L})$, όπου οι συντελεστές είναι $u(t, k) = A(k)\cos(\frac{k\pi t}{L}) + B(k)\sin(\frac{k\pi t}{L})$, προκύπτει $\frac{\partial u(x,t)}{\partial t} = 0$ από όπου παίρνουμε $\frac{k\pi B(k)}{L} = 0$ και τελικά $B(k) = 0$. Επομένως, $u(t, k) = A(k)\cos(\frac{k\pi t}{L})$.

Αντικαθιστώντας τα $u(t, k)$ στην *approxposition*(*x*, *t*) έχουμε

$$approxtemp(x, t) = \sum_{k=1}^{n} A(k)\cos(\tfrac{k\pi t}{L})\sin(\tfrac{k\pi x}{L}),$$

που για $t = 0$ γίνεται

$$approxposition(x, 0) = \sum_{k=1}^{n} A(k)\cos(\tfrac{k\pi 0}{L})\sin(\tfrac{k\pi x}{L})$$

$$= \sum_{k=1}^{n} A(k)\sin(\tfrac{k\pi x}{L}).$$

Αυτή είναι η προσέγγιση της αρχικής θεσης, και έτσι παίρνουμε τα $A(k)$'s από την ημιτονική προσέγγιση της *starterposition*(*x*) — αφού βέβαια την κάνουμε πρώτα περιττή.

Παράδειγμα: *Στην αρχή του πειράματος η θέση της χορδής στο σημείο* x $(0 \le x \le L = 3)$ *δίνεται από τηνη συνάρτηση* $starterposition(x) = 0.2\sin^2(2x)(x - 3)$ — *συνάρτηση που χρησιμοποιήσαμε σε προηγούμενο παράδειγμα*.

Γ*ια αυτήν την συνάρτηση έχουμε* $starterposition(0) = starterposition(L) = 0$ *και το σχεδιάγραμμά της είναι στο* Figure 8.

```
L = 3;
starterposition[x_] = 0.2 Sin[2 x]^2 (x - 3)^2;
{starterposition[0], starterposition[L]}
```

```
{0, 0}
```

Τ*ην συνάρτηση* $starterposition(t)$ *την κάνουμε πρώτα περιττή και μετά αυθαίρετα θέτοντες* $n = 8$ *υπολογίζουμε με την* FFF *την ημιτονική της προσέγγιση. Δηλαδή έχουμε*

```
oddStarterposition[x_] := starterposition[x] /; 0 ≤ x ≤ L;
oddStarterposition[x_] := -starterposition[2 L - x] /; L < x ≤ 2 L;

sinesOnlyfit[x_] = Chop[
  ComplexExpand[FastFourierfit[oddStarterposition, 2 L, n = 8, x]]]
```

$$0.379996\,\text{Sin}\left[\frac{\pi x}{3}\right] + 0.395139\,\text{Sin}\left[\frac{2\pi x}{3}\right] + 0.298906\,\text{Sin}[\pi x] + 0.0691479\,\text{Sin}\left[\frac{4\pi x}{3}\right] - 0.0908022\,\text{Sin}\left[\frac{5\pi x}{3}\right] - 0.0548269\,\text{Sin}[2\pi x] - 0.0186735\,\text{Sin}\left[\frac{7\pi x}{3}\right]$$

Κ*ατόπιν παίρνουμε τους συντελεστές των όρων* $\sin(\frac{k\pi t}{L})$:

```
A[k_] := Coefficient[sinesOnlyfit[x], Sin[(k π x)/L]];
Table[A[k], {k, 1, n}]
```

```
{0.379996, 0.395139, 0.298906,
 0.0691479, -0.0908022, -0.0548269, -0.0186735, 0}
```

και η συνάρτηση $position(x, t)$ *είναι*:

$$\text{position}[x\_, t\_] = \sum_{k=1}^{n} A[k]\,\text{Cos}\left[\frac{(k\pi)\,t}{L}\right]\,\text{Sin}\left[\frac{(k\pi)\,x}{L}\right]$$

$$0.379996\,\text{Cos}\left[\frac{\pi t}{3}\right]\,\text{Sin}\left[\frac{\pi x}{3}\right] +$$
$$0.395139\,\text{Cos}\left[\frac{2\pi t}{3}\right]\,\text{Sin}\left[\frac{2\pi x}{3}\right] + 0.298906\,\text{Cos}[\pi t]\,\text{Sin}[\pi x] +$$
$$0.0691479\,\text{Cos}\left[\frac{4\pi t}{3}\right]\,\text{Sin}\left[\frac{4\pi x}{3}\right] - 0.0908022\,\text{Cos}\left[\frac{5\pi t}{3}\right]\,\text{Sin}\left[\frac{5\pi x}{3}\right] -$$
$$0.0548269\,\text{Cos}[2\pi t]\,\text{Sin}[2\pi x] - 0.0186735\,\text{Cos}\left[\frac{7\pi t}{3}\right]\,\text{Sin}\left[\frac{7\pi x}{3}\right]$$

Έτσι έχουμε για παράδειγμα

```
position[2.5, 1]
```

```
-0.0208967
```

- **Δ6: Συμπίεση εικόνων με τον διακριτό συνημιτονικό μετασχηματισμό (DCT)**

Ο 21ος αιώνας θα είναι ο αιώνας της συμπίεσης ηλεκτρονικών πληροφοριών. Οι πληροφορίες αυτές αυξάνονται ραγδαία και επειδή δεν υπάρχει διαθέσιμη μνήμη για την αποθήκευσή τους καταφεύγουμε στην "συμπίεσή" τους, η οποία απαιτεί πολύ λιγότερη μνήμη.

Σαν παράδειγμα αναφέρουμε τα δακτυλικά αποτυπώματα που αποθηκεύει ηλεκτρονικά το Ομοσπονδιακό Γραφείο Ανακρίσεων των Ηνωμένων Πολιτειών της Αμερικής, γνωστό και σαν FBI. Η αποθήκευση γίνεται αφού πρώτα συμπιεσθούν τα δακτυλικά αποτυπώματα με μία μορφή του μετασχηματισμού wavelets που θα γνωρίσουμε στο επόμενο κεφάλαιο.

Στην υποενότητα αυτή θα γνωρίσουμε τον ***διακριτό συνημιτονικό μετασχηματισμό*** (discrete cosine transform ή DCT) — που χρησιμοποιείται για την συμπίεση ψηφιακών εικόνων και που σχετίζεται με τον γρήγορο μετασχηματισμό Fourier (FFT).

Βασικές έννοιες

Ορισμός (Σημάτων): Ένα ***συνεχές*** ή ***αναλογικό σήμα*** (continuous or analog signal) είναι μία συνάρτηση

$$f : D \to \mathbb{R}^n, \text{ όπου } D \subseteq \mathbb{R}^m \quad m,n \in \mathbb{N}.$$

Ένα ***διακριτό σήμα*** (discrete-time signal) είναι μία συνάρτηση

$$f : D \to \mathbb{R}^n, \text{ όπου } D \subseteq \mathbb{Z}^m \quad m,n \in \mathbb{N}.$$

Αν επί πλέον $f(D) \subseteq \mathbb{Z}^n$, τότε η f λέγεται ***ψηφιακό σήμα*** (digital signal).

Παράδειγμα συνεχούς σήματος είναι ο ήχος που είναι μία συνάρτηση $f : \mathbb{R} \to \mathbb{R}$, όπου το πεδίο τιμών της είναι η ένταση.

Άλλο παράδειγμα συνεχούς σήματος είναι τα χρώματα ψηφιακών εικόνων. Στην περίπτωση ασπρόμαυρης εικόνας το σήμα αντιστοιχεί σε κάθε εικονοστοιχείο

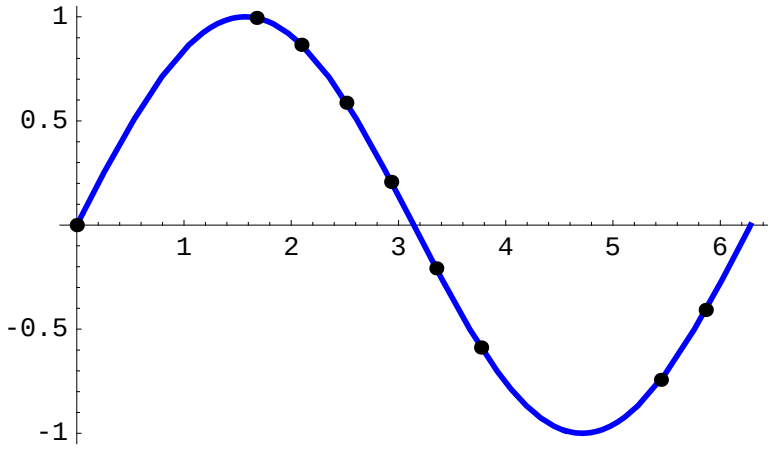
(pixel) *μία τιμή φωτισμού, έτσι ώστε* $f : D \subseteq \mathbb{Z}^2 \to \mathbb{R}$, *ενώ στην περίπτωση έγχρωμης εικόνας έχουμε* $f : D \subseteq \mathbb{Z}^2 \to \mathbb{R}^3$ *ή* $f : D \subseteq \mathbb{Z}^2 \to \mathbb{R}^4$ *ανάλογα με το αν χρησιμοποιούμε* 3 *ή* 4 *βασικά χρώματα*. (*Ενδιαφέρον έχουν οι συναρτήσεις* `GrayLevel[]`, `RGBColor[]` *και* `CMYKColor[]` *στο Mathematica.*)

Τα περισσότερα σήματα αρχίζουν την ζωή τους σε αναλογική μορφή. Γίνονται *διακριτά* (ή *ψηφιακά*) *με* ***δειγματοληψία*** (sampling) *σε ίσα χρονικά διαστήματα, δηλαδή έχουμε* $f_{\text{digital}}(n) = f_{\text{analog}}(nT)$, *όπου* $n = 0, \pm 1, \pm 2, \ldots$ *και* T *είναι ο* ***χρόνος δειγματοληψίας*** (sampling interval).

Παράδειγμα:

Αν υποθέσουμε πως το συνεχές σήμα είναι $f(t) = \sin(t)$ *τότε το διακριτό προκύπτει από την δειγματοληψία όπως φαίνεται στο ακόλουθο σχεδιάγραμμα*

```
L = 2 π; n = 8;
fplot = Plot[Sin[t], {t, 0, L},
   PlotStyle → {Thickness[0.008], RGBColor[0, 0, 1]},
   AspectRatio → 1/GoldenRatio, DisplayFunction :→ Identity];
fdata = Table[N[{t, Sin[t]}], {t, 0, L - L/(2 n - 1), L/(2 n - 1)}];
fdataplot = ListPlot[fdata,
   PlotStyle → PointSize[0.02], DisplayFunction → Identity];
Show[fplot, fdataplot, DisplayFunction → $DisplayFunction];
```



Ορισμός (Συνεχής FT): Ο ***συνεχής μετασχηματισμός* Fourier** (continuous Fourier Transform) *ενός περιοδικού σήματος* $f : \mathbb{R} \to \mathbb{C}$, *με περίοδο* $L \in \mathbb{N}$, *είναι* $\hat{f} : \mathbb{Z} \to \mathbb{C}$ *όπου*

$$\hat{f}(k) = \int_0^L f(t)\, e^{-i\,k\,(2\pi)\,t/L}\, dt,$$

με $k \in \mathbb{Z}$, $i = \sqrt{-1} \in \mathbb{C}$ (*και συνήθως* $L = 2\pi$). *Υπάρχει και ο αντίστροφος τύπος*

$$f(t) = \tfrac{1}{L} \sum_{k \in \mathbb{Z}} \hat{f}(k)\, e^{i\,k\,(2\pi)\,t/L},\ \forall t \in \mathbb{R},$$

που εκφράζει την συνάρτηση f με την βοήθεια του μετασχηματισμού Fourier. *Η σειρά συγκλίνει ομοιόμορφα στην συνάρτηση f και λέγεται* ***σειρά* Fourier**. *Για* $k \in \mathbb{Z}$ *οι αριθμοί* $c_k = \tfrac{1}{L}\hat{f}(k) = \tfrac{1}{L}\int_0^L f(t)\, e^{-i\,k\,(2\pi)\,t/L}\, dt$ *είναι οι* ***συντελεστές* Fourier** *που είδαμε και στην ολοκληρωτική προσέγγιση* Fourier. *Επίσης για* $k \in \mathbb{Z}$, *οι ειδικές συναρτήσεις* $e^{i\,k\,(2\pi)\,t/L}$ *αποτελούν κατά κάποιο τρόπο την "****βάση****" του μιγαδικού διανυσματικού χώρου των L-περιοδικών συναρτήσεων.*

Τονίζουμε πως ενώ το αρχικό σήμα f αντιστοιχεί σε κάθε $t \in [0, L]$ *την τιμή του σήματος* $f(t)$ *σε εκείνη την χρονική στιγμή, ο μετασχηματισμός* Fourier $\hat{f}$ *αντιστοιχεί σε κάθε συχνότητα* $k \in \mathbb{Z}$ *την συνεισφορά* $\hat{f}(k)$ *της συχνότητα αυτής στο f, όπως εκφράζεται από τον αντίστροφο τύπο.*

Με άλλα λόγια ο μετασχηματισμός Fourier "***συμπιέζει***" *τις "αμέτρητες" τιμές* $f(t)$ *στις μετρήσιμες τιμές* $\hat{f}(k)$.

Ορισμός (Διακριτός FT): *Για* $k \in \mathbb{Z}$ *και* $\omega = e^{-2\pi i/L} \in \mathbb{C}$ *μια αρχέγονη ρίζα της μονάδας, τάξης L, ο* ***διακριτός μετασχηματισμός* Fourier** (discrete Fourier Transform) *ενός περιοδικού σήματος* $f : \mathbb{Z} \to \mathbb{C}$, *με περίοδο* $L \in \mathbb{N}$, *ορίζεται από* $\hat{f} : \mathbb{Z} \to \mathbb{C}$ *όπου*

$$\hat{f}(k) = \sum_{0 \le n < L} f(n)\, e^{-2\pi\, i\, k\, n/L} = \sum_{0 \le n < L} f(n)\, \omega^{k\,n}.$$

Με άλλα λόγια, ο διακριτός μετασχηματισμός Fourier *είναι μια προσέγγιση του ολοκληρώματος που ορίζει τον συνεχή μετασχηματισμό* Fourier, *αν στο τελευταίο κάνουμε δειγματοληψία σε ισοαπέχοντα σημεία* $2\pi n/L$, $0 \le n < L$. *Σε αντίθεση με*

τον συνεχή μετασχηματισμό Fourier, *ο διακριτός μετασχηματισμός* $\hat{f}(k)$ *είναι επίσης* ***περιοδικός*** *με περίοδο L και ισχύει ο αντίστροφος τύπος*

$$f(n) = \frac{1}{L}\sum_{0 \le k < L} \hat{f}(k)\, e^{2\pi i k n / L} = \frac{1}{L}\sum_{0 \le k < L} \hat{f}(n)\, \omega^{-kn}.$$

Αυτός *ο ορισμός του διακριτού μετασχηματισμού* Fourier ***σχετίζεται*** *με τον ορισμό της* Ενότητας 4.6 *ως εξής*: Αν *συσχετίσουμε με το πολυώνυμο* $f \in \mathbb{C}[x]$, *βαθμού* $< n$, *ένα διακριτό σήμα* $g : \mathbb{Z} \to \mathbb{C}$ *με περίοδο n που αντιστοιχεί στους* $\{0, 1, \ldots, n - 1\}$ *τους συντελεστές του f, τότε* $V_\omega(f)$, *με* $\omega = e^{-2\pi i/n}$, *είναι ο διακριτός μετασχηματισμός του g.*

Όπως *είδαμε*, *έχουμε* ***δύο ισοδύναμες παραστάσεις ενός σήματος***. Συγκεκριμένα, *αν f είναι ένα σήμα* (*συνεχές* ή *διακριτό*) *και* $\hat{f}(k)$ *ο αντίστοιχος μετασχηματισμός* Fourier, *τότε f*(*t*) *είναι η τιμή του σήματος την χρονική στιγμή t, ενώ* $\hat{f}(k)$ *είναι η συνεισφορά της συχνότητας k στο σήμα f.*

Τα *ακουστικά και οπτικά σήματα που προορίζονται για τους ανθρώπους μεταβάλλονται* "*αργά*" *στον χρόνο πράγμα που σημαίνει ότι οι συχνότητές τους είναι μικρές*. Συνεπώς, *οι* ***ψηλές συχνότητες έχουν μικρή συνεισφορά*** *και επομένως για μεγάλες τιμές του k έχουμε μικρές τιμές του* $|\hat{f}(k)|$.

Με *βάση την τελευταία παρατήρηση η βασική ιδέα της συμπίεσης είναι να πετάμε τις τιμές του* $|\hat{f}(k)|$ *που είναι* "*κοντά στο* 0" *και να κρατάμε τις υπόλοιπες*. Επειδή *το ανθρώπινο αυτί ή μάτι δουλεύει καλλίτερα σε χαμηλές* (*παρά σε ψηλές*) *συχνότητες ο ακροατής ή θεατής δεν παρατηρεί το χάσιμο πληροφοριών*. Έτσι *επιτυγχάνεται μεγάλη συμπίεση*!

Σημειώνουμε *πως στην σταθερή τηλεφωνία χρησιμοποιούνται συχνότητες μόνο μέχρι* 4000 Hertz *αν και το* ***μέσο*** *ανθρώπινο αυτί αντιλαμβάνεται συχνότητες από* 20 *μέχρι* 15000 Hertz. (Το *πάνω όριο εξαρτάται και από την ηλικία και μπορεί να πάει μέχρι* 22000 Hertz).

Επειδή *τα ακουστικά και οπτικά σήματα είναι πραγματικά σήματα, αντί του μετασχηματισμού* Fourier *χρησιμοποιείται μία παραλλαγή του — ο διακριτός*

συνημιτονικός μετασχηματισμός — που αντιστοιχεί σε πραγματικά σήματα πάλι πραγματικά σήματα.

Ο *διακριτός συνημιτονικός μετασχηματισμός* (DCT)

Έστω $f : \{0, 1, \ldots, L - 1\} \to \mathbb{R}$ *ένα διακριτό πραγματικό σήμα πεπερασμένης διάρκειας* $L \in \mathbb{N}$, *όπως για παράδειγμα μία σειρά μιας ψηφιακής εικόνας. Θεωρούμε ότι η f επεκτείνεται σε περιοδικό σήμα σε όλο τον δακτύλιο* $\mathbb{Z}$ *θέτοντες* $f(n) = f(n \bmod L)\ \forall n \in \mathbb{Z}$. *Τότε ο* ***διακριτός συνημιτονικός μετασχηματισμός*** (discrete cosine transorm ή DCT) *είναι*

$$\mathrm{DCT}(f)(k) = \frac{1}{\sqrt{L}}\, c(k) \sum_{0 \le n \le L-1} f(n) \cos \frac{(2n+1)\pi k}{2L},\ 0 \le k < L,$$

όπου $c(k) = 1$ *αν* $k = 0$ *και* $c(k) = \sqrt{2}$ *αν* $k \neq 0$. *Αυτό σημαίνει πως κάθε στοιχείο της μετασχηματισμένης λίστας είναι το εσωτερικό γινόμενο της αρχικής λίστας* $f(n)$ *και ενός από τα* L *διανύσματα* $\delta_k(n) = \cos \frac{(2n+1)\pi k}{2L}$, $0 \le k \le L - 1$, *που αποτελούν την "****βάση****" του χώρου. Οι σταθερές διαλέγονται έτσι ώστε τα διανύσματα βάσης είναι ορθοκανονικά.*

Επομένως, ο μετασχηματισμός DCT *ορίζεται και σαν το γινόμενο ενός διανύσματος και ενός* $L \times L$ *πίνακα του οποίου οι σειρές είναι τα διανύσματα* $\delta_k(n)$, $0 \le k \le L - 1$, *της βάσης. Για* $L = 8$, *ο πίνακας ορίζεται στο Mathematica ως εξής*:

```
DCTMatrix[L_: 8] :=
 Table[If[k == 0, Sqrt[1/L], Sqrt[2/L] Cos[ ((2 n + 1) π k)/(2 L)]],
   {k, 0, L - 1}, {n, 0, L - 1}] // N
```

Κάθε *σειρά του πίνακα είναι και ένα από τα διανύσματα βάσης*:

```
DCTMatrix[] // MatrixForm
```

```
0.353553    0.353553    0.353553    0.353553    0.353553    0.353553    0.353553    0.353553
0.490393    0.415735    0.277785    0.0975452  -0.0975452  -0.277785   -0.415735   -0.490393
 0.46194     0.191342   -0.191342   -0.46194    -0.46194    -0.191342    0.191342    0.46194
0.415735   -0.0975452  -0.490393   -0.277785    0.277785    0.490393    0.0975452  -0.415735
0.353553   -0.353553   -0.353553    0.353553    0.353553   -0.353553   -0.353553    0.353553
0.277785   -0.490393    0.0975452   0.415735   -0.415735   -0.0975452   0.490393   -0.277785
0.191342   -0.46194     0.46194    -0.191342   -0.191342    0.46194    -0.46194     0.191342
0.0975452  -0.277785    0.415735   -0.490393    0.490393   -0.415735    0.277785   -0.0975452
```

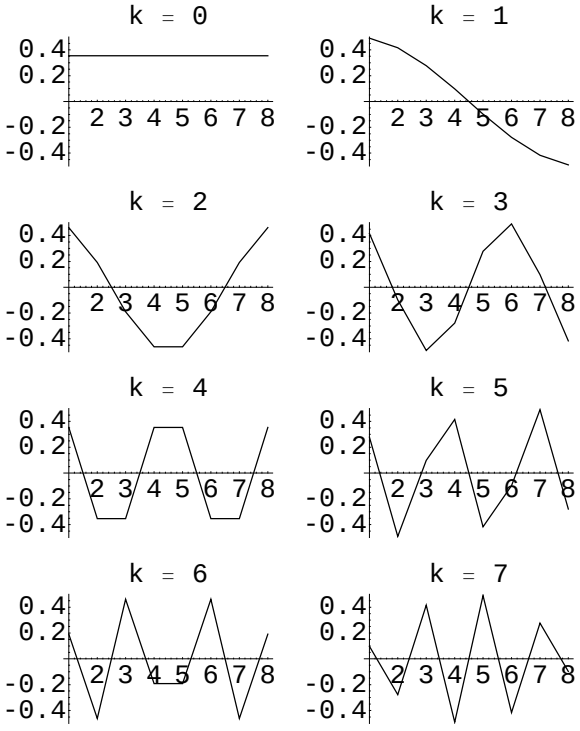
Επειδή *τα διανύσματα της βάσης είναι ορθοκανονικά ξέρουμε ότι ο αντίστροφος πίνακας του* DCTMatrix[] *είναι ο ανάστροφός του.* Συνεπώς *ισχύει*:

```
DCTMatrix[].Transpose[DCTMatrix[]] // MatrixForm // Chop
```

$$\begin{pmatrix} 1. & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1. & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1. & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1. & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1. & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1. & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1. & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1. \end{pmatrix}$$

Τα *οχτώ διανύσματα της βάσης φαίνονται στο ακόλουθο σχεδιάγραμμα*

```
k = 0;
Show[GraphicsArray[Partition[Map[(ListPlot[#, PlotRange → {-.5, .5},
      PlotJoined → True, PlotLabel → "k = " <> ToString[k++],
      DisplayFunction → Identity]) &, DCTMatrix[]], 2]]];
```



Προσέξτε ότι *σε μεγαλύτερες τιμές του k αντιστοιχεί γρηγορότερη* "*μεταβολή*" *του διανύσματος βάσης*.

Σημειώνουμε *πως ο παραπάνω ορισμός αυτός είναι ο συνηθέστερος στην βιβλιογραφία*. Στο *Mathematica ο διακριτός συνημιτονικός μετασχηματισμός ορίζεται διαφορετικά*. Υπάρχει δε στο πακέτο

```
<< LinearAlgebra`FourierTrig`
```

η συνάρτηση `FourierCos[ ]` (*ίδια με την αντίστροφό της*) *την οποία όμως δεν θα χρησιμοποιήσουμε*.

```
? FourierCos
```

```
FourierCos[list] computes the discrete
   cosine transform of a list of real numbers. More...
```

Ανάλογα *ορίζεται και ο* ***αντίστροφος διακριτός συνημιτονικός μετασχηματισμός*** (inverse discrete cosine transorm ή iDCT)

$$\mathrm{iDCT}(f)(n) = \frac{1}{\sqrt{L}} \sum_{0 \le k \le L-1} c(k)\, f(k) \cos \frac{(2n+1)\pi k}{2L},\ 0 \le n < L,$$

όπου *οι τιμές του* $c(k)$ *είναι*, όπως *και πριν*, $c(k) = 1$ *αν* $k = 0$ *και* $c(k) = \sqrt{2}$ *αν* $k \neq 0$. Προσέξτε *πως ο* $\mathrm{iDCT}(f)(n)$ *εκφράζεται σαν γραμμικός συνδιασμός των διανυσμάτων της βάσης*. Οι *συντελεστές είναι στοιχεία του μετασχηματισμού* DCT *και παριστάνουν την συνεισφορά κάθε συχνότητας στον* $\mathrm{iDCT}(f)(n)$.

Παράδειγμα:

Παίρνουμε *ένα τυχαίο διάνυσμα με* 8 *στοιχεία*

```
L = 8;
vector1 = Table[Random[Real, {-1, 1}], {L}]
```

```
{0.873374, -0.358588, 0.848067, 0.13191,
 0.467967, 0.751282, -0.524331, 0.00638497}
```

Ο *διακριτός συνημιτονικός μετασχηματισμός* DCT *του διανύσματος αυτού υπολογίζεται με τον* "*πίνακα* DCT" *και είναι*

```
out1 = DCTMatrix[].vector1
```

```
{0.776427, 0.488175, -0.345673,
 0.390159, 0.269834, 0.0292881, 1.20021, 0.243567}
```

Ο *αντίστροφος διακριτός συνημιτονικός μετασχηματισμός* iDCT *του διανύσματος* out1 *υπολογίζεται με τον αντίστροφο* "*πίνακα* DCT" *και είναι*

```
Inverse[DCTMatrix[]].out1
```

```
{0.873374, -0.358588, 0.848067, 0.13191,
 0.467967, 0.751282, -0.524331, 0.00638497}
```

```
% - vector1 // Chop
```

```
{0, 0, 0, 0, 0, 0, 0, 0}
```

Ι*σοδύναμα*, *επειδή ο πίνακας είναι ορθοκανονικός*, *ο αντίστροφος διακριτός συνημιτονικός μετασχηματισμός* iDCT *του διανύσματος* out1 *υπολογίζεται και με τον* ***ανάστροφο*** "*πίνακα* DCT" *και είναι*

```
Transpose[DCTMatrix[]].out1
```

```
{0.873374, -0.358588, 0.848067, 0.13191,
 0.467967, 0.751282, -0.524331, 0.00638497}
```

```
% - vector1 // Chop
```

```
{0, 0, 0, 0, 0, 0, 0, 0}
```

Όπως *αναφέραμε*, *ο μετασχηματισμός* DCT *σχετίζεται με τον* (*γρήγορο*) *διακριτό μετασχηματισμό* Fourier (FFT). *Συγκεκριμένα*, *μπορούμε να υπολογίσουμε τον μετασχηματισμό* DCT *μιας λίστας με τον* FFT *ως εξής*: **(*α*)** *δημιουργούμε μία καινούργια λίστα παίρνοντας πρώτα τα άρτια στοιχεία και έπειτα τα περιττά στοιχεία της αρχικής λίστας*, **(*β*)** *πολλαπλασιάζουμε στοιχείο προς στοιχείο τον*

αντίστροφο (*γρήγορο*) *μετασχηματισμό* Fourier, iFFT, *της καινούργιας λίστας με την λίστα των εκθετικών συναρτήσεων* $\{1, \sqrt{2}\, e^{-i\pi\frac{1}{2L}}, \ldots, \sqrt{2}\, e^{-i\pi\frac{L-1}{2L}}\}$ *και* **(γ)** *τέλος από το αποτέλεσμα παίρνουμε το πραγματικό μέρος κάθε στοιχείου.*

Με *το Mathematica ορίζουμε πρώτα τις εκθετικές συναρτήσεις που τις ονομάζουμε* `DCTfitters[]`

```
DCTfitters[L_: 8] := Join[{1}, Table[√2 e^(-i π k/(2 L)), {k, L - 1}]] // N
```

και κατόπιν την συνάρτηση `DCT[]`. (*Προσέξτε την χρήση της συνάρτησης* `InverseFourier[]`!)

```
DCT[list_, L_: 8] :=
 Re[DCTfitters[L] * InverseFourier[list[[Join[Table[i + 1,
        {i, 0, L - 1, 2}], Reverse[Table[i + 1, {i, 1, L - 1, 2}]]]]]]]
```

Έτσι *για το διάνυσμα του παραδείγματός μας έχουμε*

```
DCT[vector1]
```

```
{0.776427, 0.488175, -0.345673,
 0.390159, 0.269834, 0.0292881, 1.20021, 0.243567}
```

και το αποτέλεσμα είναι το ίδιο με εκείνο που βρήκαμε πριν.

```
out1
```

```
{0.776427, 0.488175, -0.345673,
 0.390159, 0.269834, 0.0292881, 1.20021, 0.243567}
```

Ο *αντίστροφος μετασχηματισμός* iDCT *μιας λίστας υπολογίζεται τώρα με τον* (*γρήγορο*) *μετασχηματισμό* Fourier, FFT, *αφού*:

(α) *ορίσουμε τις αντίστροφες εκθετικές συναρτήσεις* `iDCTfitters[]` *παίρνοντας τις συζυγείς τους*

```
iDCTfitters[L_: 8] := Conjugate[DCTfitters[L]]
```

(β) *πολλαπλασιάσουμε στοιχείο προς στοιχείο τον μετασχηματισμό* FFT *της αρχικής λίστας με την λίστα των εκθετικών συναρτήσεων* `iDCTfitters[]`, *πάρουμε το πραγματικό μέρος κάθε στοιχείου της λίστας που προκύπτει και τέλος αναδιατάξουμε τα στοιχεία αυτά.*

```
iDCT[list_, L_: 8] :=
 Re[Fourier[iDCTfitters[L] * list]][[indeces = Table[i, {i, 1, L}];
   indecesOrdered = {};
   For[i = 1, i ≤ L/2, i++, AppendTo[indecesOrdered, indeces[[i]]];
    AppendTo[indecesOrdered, indeces[[L - i + 1]]]]; indecesOrdered]]
```

Έτσι *έχουμε*

```
iDCT[DCT[vector1]] - vector1 // Chop
```

```
{0, 0, 0, 0, 0, 0, 0, 0}
```

Π*ροσέξτε πως λειτουργούν οι αναδιατάξεις στους μετασχηματισμούς* DCT *και* iDCT. Η *λίστα* {1, 2, …, 8} *με τον* DCT *γίνεται*

```
L = 8; indeces = Join[Table[i + 1, {i, 0, L - 1, 2}],
  Reverse[Table[i + 1, {i, 1, L - 1, 2}]]]
```

```
{1, 3, 5, 7, 8, 6, 4, 2}
```

και αυτή μετην σειρά της με τον iDCT *γίνεται πάλι η αρχική*

```
indecesOrdered = {};
For[i = 1, i ≤ L/2, i++, AppendTo[indecesOrdered, indeces[[i]]];
 AppendTo[indecesOrdered, indeces[[L - i + 1]]]]; indecesOrdered
```

```
{1, 2, 3, 4, 5, 6, 7, 8}
```

Έ*χοντας πλήρως ορίσει τον διακριτό συνημιτονικό μετασχηματισμό ας δούμε πως μπορεί να είναι ένας* ***αλγόριθμος συμπίεσης ψηφιακών εικόνων*** *— για την* "*οικονομική*" *αποθήκευσή τους.*

Υπενθυμίζουμε πως σε μία ασπρόμαυρη ψηφιακή εικόνα αντιστοιχεί ένας πίνακας, κάθε στοιχείο του οποίου παριστάνει την φωτεινότητα του αντίστοιχου εικονοστοιχείου. (Στις έγχρωμες εικόνες αντιστοιχούν περισσότεροι πίνακες — ένας για την φωτεινότητα κάθε βασικού χρώματος του εικονοστοιχείου.)

Εφαρμόζουμε λοιπόν πρώτα σε κάθε σειρά και μετά σε κάθε στήλη του πίνακα της εικόνας τον μετασχηματισμό DCT και αποκτούμε έναν μετασχηματισμένο πίνακα. Κατόπιν διαιρούμε τα στοιχεία αυτού του μετασχηματισμένου πίνακα με μία σταθερά $q \in \mathbb{R}_{>1}$ (quantization) και τα στρογγυλεύουμε προς τον πλησιέστερο ακέραιο.

Σκοπός αυτής της διαίρεσης και στρογγύλευσης είναι να μηδενίσουμε τις τιμές που είναι κοντά στο 0 (σε απόλυτη τιμή) και να μειώσουμε τον αριθμό των διαφορετικών τιμών — και συνεπώς και τον αριθμό των δυαδικών ψηφίων που χρειάζονται για την κωδικοποίησή τους.

Στο στάδιο αυτό έχουμε μόνιμες "απώλειες" πληροφοριών που δεν μπορούμε να επανακτήσουμε. Αυτό γίνεται για τα στοιχεία με υψηλή συχνότητα, δηλαδή για μεγάλες τιμές του k. Επομένως μεγάλες τιμές της σταθεράς q αντιστοιχούν σε υψηλή συμπίεση — αλλά και σε κακή ποιότητα εικόνας.

Στο τέλος, ένας συνδιασμός από κωδικοποιήσεις **Huffman** και **run length** συμπιέζουν τον στρογγυλεμένο πίνακα χωρίς "απώλειες" πληροφοριών (lossless compression).

Αναφέρουμε πως η κωδικοποίηση Huffman (1952) συμπιέζει αντιστοιχώντας "μικρούς κωδικούς" (λίγα δυαδικά ψηφία) σε σύμβολα που χρησιμοποιούνται συχνά και "μεγάλους κωδικούς" σε σύμβολα που σπάνια χρησιμοποιούνται (όπως για παράδειγμα ο κώδικας Morse). Για περισσότερα σχετικά με τον κώδικα Huffman βλέπε και παρακάτω. Όσον αφορά την κωδικοποίηση run length, αυτή συμπιέζει κάθε ακολουθία (run) από 0 σε δύο αριθμούς: ο πρώτος είναι 0 και σημαδεύει την θέση της ακολουθίας ενώ ο δεύτερος δηλώνει το μήκος της

ακολουθίας. (Για παράδειγμα, η ακολουθία 1, 0, 0, 0, 0, 2 συμπιέζεται στην 1, 0, 4, 2.)

Η συμπίεση πληροφοριών χωρίς "απώλειες" επιτυγχάνεται στο Mathematica 5 με την βοήθεια της συνάρτησης `SparseArray[]`. Εμείς θα ασχοληθούμε με τα υπόλοιπα στάδια της συμπίεσης.

Για την αποσυμπίεση της εικόνας εργαζόμαστε αντίστροφα: αφού κάνουμε αποσυμπίεση πληροφοριών — με την βοήθεια της συνάρτησης `Normal[SparseArray[]]` στο Mathematica 5 — πολλαπλασιάζουμε τα στοιχεία με q και εφαρμόζουμε τον αντίστροφο μετασχηματισμό iDCT σε κάθε σειρά και στήλη του πίνακα.

Αβεβαιότητα και εντροπία

Στην υποενότητα αυτή θα ασχοληθούμε με την εντροπία, η οποία "ποσολογεί" (quantifies) την αβεβαιότητα. Όπως θα δούμε η εντροπία σχετίζεται με τον **λόγο συμπίεσης** (compression ratio) μίας εικόνας.

Έστω A ένα σύνολο με ℓ διαφορετικά στοιχεία και έστω ότι επιλέγουμε τυχαία ένα από τα στοιχεία αυτά. Η ***αβεβαιότητα*** αυτής της επιλογής είναι ℓ. Για τεχνικούς όμως λόγους, που σχετίζονται με την αθροιστικότητα, ***ορίζουμε την αβεβαιότητα*** της επιλογής σαν $\log_2 \ell$.

Η αθροιστικότητα της αβεβαιότητας εμφανίζεται όταν, για παράδειγμα, έχουμε δύο σύνολα, A, B, με ℓ_1 και ℓ_2 διαφορετικά στοιχεία αντίστοιχα. Παρότι το Καρτεσιανό γινόμενο $A \times B = \{(a, b), a \in A, b \in B\}$ των δύο αυτών συνόλων έχει $\ell_1 \ell_2$ στοιχεία, η αβεβαιότητα της τυχαίας επιλογής από αυτό ***δεν*** καθορίζεται από το πλήθος $\ell_1 \ell_2$. Και τούτο διότι τυχαία επιλογή από το $A \times B$ αντιστοιχεί σε μία τυχαία επιλογή από το A ***και*** σε μία τυχαία επιλογή από το B.

Αυτή η αθροιστικότητα της αβεβαιότητας ικανοποιείται μόνο με τον λογάριθμο, διότι $\log_2 \ell_1 + \log_2 \ell_2 = \log_2 \ell_1 \ell_2$. Δηλαδή, η σύνδεση ενός μέτρου αβεβαιότητας με το πλήθος στοιχείων του συνόλου επιτυγχάνεται μόνο με τον ***λογάριθμο του***

πλήθους των στοιχείων. Η *βάση του λογαρίθμου* (2 *στην προκειμένη περίπτωση*) *καθορίζει τις μονάδες* (bits).

Από *την θεωρία πιθανοτήτων αναγνωρίζουμε ότι οι επιλογές στα παραπάνω παραδείγματα αντιστοιχούν στην* ***ομοιόμορφη τυχαία μεταβλητή***.

Γ*ενικά, για κάθε τυχαία μεταβλητή* X *υπάρχει η* ***κατανομή*** *της, δηλαδή ένα σύνολο* $A = \{a_0, a_1, \ldots, a_{\ell-1}\}$ *των δυνατών* ***τιμών*** ή ***αποτελεσμάτων*** *της καθώς επίσης και οι* ***πιθανότητες*** $p_i = P(X = a_i)$, $0 \le i \le \ell$-1, *με τις οποίες παίρνει αυτά τα αποτελέσματα* a_i, $0 \le i \le \ell$-1. Υ*πενθυμίζουμε πως* $p_i \ge 0\ \forall i$ *και* $\sum_{i=0}^{\ell-1} p_i = 1$.

Έ*στω τώρα* C *η κωδικοποίηση μιας τυχαίας μεταβλητής* X (*των συμβόλων του συνόλου* A) *και έστω* $\lambda(C)$ *η* ***μέση τιμή*** *του μήκους* (*αριθμός* bits) *ενός κωδικοποιημένου συμβόλου*. Επιπλέον, *έστω* $H(X)$ ή $H(p_0, \ldots, p_{\ell-1})$ *η εντροπία της* X. Η ***σημασία της εντροπίας*** *έγκειται στο γεγονός ότι* $\lambda(C) \in [H(X), H(X) + 1)$. Το *σημαντικό αυτό αποτέλεσμα αποδεικνύεται στην θεωρία της πληροφορίας*. Δ*ηλαδή η εντροπία* $H(X)$ *είναι ένα* ***κάτω φράγμα*** *για την* $\lambda(C)$, *τον* "*συμπιεσμένο*" *λόγο* bits/*σύμβολο*.

Α*ς υπολογίσουμε τώρα την εντροπία* H *μιας τυχαίας μεταβλητής* X. Όπως *είδαμε, αν* ℓ *είναι ο αριθμός των διαφορετικών στοιχείων* (ή *συμβόλων*) *τότε η αβεβαιότητα επιλογής είναι* $H(X) = \log_2 \ell$. Αυτό *γράφεται και ως εξής*:

$$H(X) = \log_2 \ell = - \log_2 \ell^{-1} = - \log_2 \tfrac{1}{\ell} = - \log_2 p,$$

όπου $p = \frac{1}{\ell}$ *είναι η πιθανότητα επιλογής ενός συμβόλου*. Π*ροσέξτε πως στην περίπτωση αυτή τα στοιχεία επιλέγονται με την ίδια πιθανότητα*.

Θ*α γενικεύσουμε τον τελευταίο τύπο για την περίπτωση που τα σύμβολα επιλέγονται με διαφορετικές πιθανότητες* p_i, $\sum_{i=0}^{\ell-1} p_i = 1$. Π*ρος τούτο, με τον τύπο*

$$\nu_i = - \log_2 p_i$$

ορίζουμε τον ***βαθμό έκπληξης*** *για την επιλογή του* i-*στού συμβόλου*. Α*ν η πιθανότητα επιλογής του* i-*στού συμβόλου είναι πολύ μικρή, δηλαδή* $p_i \approx 0$, *η*

έκπληξή μας θα είναι πολύ μεγάλη αν επιλεγεί — δηλαδή $\nu_i \approx \infty$ γιατί το σύμβολο αυτό θα πρέπει να εμφανίζεται πολύ σπάνια. Αντίθετα, αν η πιθανότητα επιλογής του i-στού συμβόλου είναι πολύ μεγάλη, δηλαδή $p_i \approx 1$, τότε η έκπληξή μας θα είναι πολύ μικρή αν επιλεγεί — δηλαδή $\nu_i \approx 0$ γιατί το σύμβολο αυτό θα πρέπει να εμφανίζεται πολύ συχνά.

Η εντροπία είναι **ο μέσος βαθμός έκπληξης** για μία άπειρη ακολουθία που σχηματίζεται από τα ℓ σύμβολα $\{a_0, a_1, \ldots, a_{\ell-1}\}$. Ας βρούμε όμως τον μέσο βαθμό έκπληξης για μία πεπερασμένη ακολουθία μήκους m από αυτά τα ℓ σύμβολα. Έστω ότι το i-στό σύμβολο εμφανίζεται m_i φορές, οπότε

$$m = \sum_{i=0}^{\ell-1} m_i .$$

Τότε ο μέσος βαθμός έκπληξης για τα ℓ σύμβολα είναι

$$\frac{\sum_{i=0}^{\ell-1} m_i \nu_i}{m} = \sum_{i=0}^{\ell-1} \frac{m_i}{m} \nu_i .$$

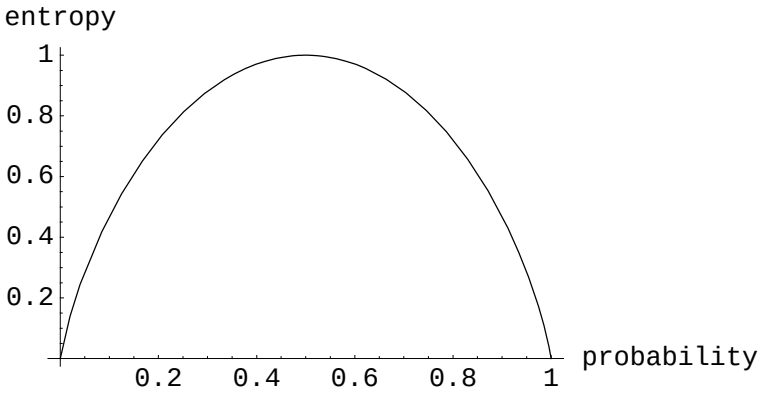
Για μία άπειρη ακολουθία, η συχνότητα $\frac{m_i}{m}$ του i-στού συμβόλου γίνεται η πιθανότητά του p_i, οπότε αντικαθιστώντες έχουμε τον φημισμένο τύπο του Shannon για την εντροπία

$$H(p_0, \ldots, p_{\ell-1}) = - \sum_{i=0}^{\ell-1} p_i \log_2 p_i ,$$

σε bits/σύμβολο.

Για την περίπτωση δύο συμβόλων με πιθανότητες p και $1 - p$ η γραφική παράσταση της εντροπίας είναι

```
Plot[-(p Log[2, p] + (1 - p) Log[2, 1 - p]),
  {p, 0, 1}, AxesLabel → {"probability", "entropy"}];
```



Από *το σχήμα βλέπουμε πως η μεγαλύτερη εντροπία είναι όταν τα* $\ell = 2$ *σύμβολα έχουν την ίδια πιθανότητα* ($p = \frac{1}{2}$) *επιλογής*. Αυτό *συμφωνεί με τον αρχικό μας τύπο* $H(X) = \log_2 \ell = -\log_2 p = 1$. Στις *ασκήσεις αποδεικνύεται πως γενικά* $0 \le H(p_0, \ldots, p_{\ell-1}) \le \log_2 \ell$ *και η δεξιά ισότητα ισχύει εάν και μόνον εάν* $p_0 = \cdots = p_{\ell-1} = \frac{1}{\ell}$.

Στο *Mathematica προγραμματίσαμε τον τύπο του* Shannon *λίγο διαφορετικά με την συνάρτηση* `entropy[]`. Με *την βοήθεια ενός στατιστικού πακέτου η συνάρτηση αυτή υπολογίζει την εντροπία μιας ακολουθίας χαρακτήρων αφού πρώτα υπολογίσει την συχνότητα εμφάνισής τους*.

```
<<Statistics`DataManipulation`
entropy[list_List]:=-Apply[Plus,
       Map[#
Log[2,#]&,First[Transpose[Frequencies[list]]]/Length[list]]]//N
```

Στο *ακόλουθο παράδειγμα υπολογίζουμε την εντροπία της λέξης* mississippi *και των* 4 *διαφορετικών γραμμάτων που περιλαμβάνει*.

```
entropy[{"m", "i", "s", "p"}]
```

```
2.
```

```
entropy[Characters["mississippi"]]
```

```
1.82307
```

Η *εντροπία είναι διαφορετική*. Τι *σημαίνει όμως να λέμε ότι η εντροπία της ακολουθίας των χαρακτήρων* {m,i,s,s,i,s,s,i,p,p,i} *είναι* 1.82307; Η *απάντηση είναι πως ειδικά για την λέξη* mississippi *μπορούμε να κωδικοποιήσουμε τους* 4 *χαρακτήρες κατά τέτοιο τρόπο ώστε κατά μέσο όρο να χρειαστούν* 1.82307 bits/*χαρακτήρα*. (Επαναλαμβάνουμε *πως ο αριθμός αυτός είναι ένα κάτω φράγμα*.)

Η *κωδικοποίηση αυτή λέγεται* ***μεταβλητού μήκους*** (variable length coding) *διότι σε χαρακτήρες που εμφανίζονται συχνά αντιστοιχούν λιγότερα δυαδικά ψηφία* (bits) *απ' ότι σε χαρακτήρες που εμφανίζονται σπάνια*. Τέτοιοι *είναι οι λεγόμενοι* **Huffman *κώδικες***.

Π*αράδειγμα κώδικα* Huffman:
Σ*την λέξη* mississippi *έχουμε* 4 *διαφορετικά γράμματα που εμφανίζονται με διαφορετικές συχνότητες*.

```
Frequencies[Characters["mississippi"]]
```

```
{{4, i}, {1, m}, {2, p}, {4, s}}
```

Γ*ια να βρούμε τον κώδικα* Huffman *για την λέξη* mississippi *δημιουργούμε ένα δυαδικό δένδρο κάθε κλαδί του οποίου* "*σημαδεύεται*" *με* 0 ή 1, *και τα φύλλα του οποίου είναι οι χαρακτήρες μαζί με τις συχνότητες*. Π*ροσέξτε πως οι συχνότητες των γραμμάτων i*, *m*, *p και s για την λέξη* mississippi *είναι* $\frac{4}{11}$, $\frac{1}{11}$, $\frac{2}{11}$ *και* $\frac{4}{11}$, *αντίστοιχα αλλά χρησιμοποιούμε μόνο τους αριθμητές*.

Η *δημιουργία του δένδρου γίνεται ως εξής*:

Από *τους* "*κόμβους*" {{4, i}, {1, m}, {2, p}, {4, s}} *επιλέγουμε τους δύο με τις μικρότερες συχνότητες*, {1, m}, {2, p}, *και δημιουργούμε την ρίζα τους* {m, p} *με συχνότητα* 3. Το *κλαδί από την ρίζα* {*m*, *p*} *προς τον κόμβο* {1, m} *σημαδεύεται με* 1 *ενώ το κλαδί προς τον κόμβο* {2, p} *σημαδεύεται με* 0.

Στην συνέχεια, από τους κόμβους {{4, i}, {3, {m, p}}, {4, s}} επιλέγουμε πάλι τους δύο με τις μικρότερες συχνότητες, έστω τους {4, i}, και {3, {m, p}} και δημιουργούμε την ρίζα τους {i, m, p} με συχνότητα 7. Το κλαδί από την ρίζα {i, m, p} προς τον κόμβο {4, i} σημαδεύεται με 1 ενώ το κλαδί προς τον κόμβο {3, {m, p} σημαδεύεται με 0.

Τέλος για τους τελευταίους κόμβους {{7, {i, m, p}}, {4, s}} δημιουργούμε την ρίζα τους {i, m, p, s} με συχνότητα 11. Το κλαδί από την ρίζα {i, m, p, s} προς τον κόμβο {7, {i, m, p}} σημαδεύεται με 1 ενώ το κλαδί προς τον κόμβο {4, s} σημαδεύεται με 0.

Η κωδικοποίηση τώρα κάθε γράμματος προκύπτει από τα σήματα των κλαδιών που συναντάμε στην πορεία από την ρίζα του δένδρου {11, {i, m, p, s}} προς το αντίστοιχο φύλλο. Για παράδειγμα, για να πάμε από την ρίζα του δένδρου {11, {i, m, p, s}} στο φύλλο {4, i} περνάμε από δύο κλαδιά σημαδεμένα με το 1 και έτσι το i κωδικοποιείται με 11. Έτσι έχουμε τον κώδικα

```
code = {"i" → "11", "m" → "101", "p" → "100", "s" → "0"};
```

Η μέση τιμή του μήκους του κώδικα είναι $\frac{9}{4} = 2.25$ bits/σύμβολο $\in [H(X), H(X) + 1) = [1.82307, 2.82307)$. Όπως είπαμε η σημασία της εντροπίας έγκειται στο γεγονός ότι μας δίνει ένα κάτω φράγμα της μέσης τιμής του μήκους του κώδικα.

Η κωδικοποίηση της λέξης mississippi είναι λοιπόν

```
("mississippi" // Characters) /. code // StringJoin // ToExpression
```

```
10111001100111001001 1
```

με 21 δυαδικά ψηφία.

```
Apply[Plus, DigitCount[%]]
```

```
21
```

Προσέξτε όμως πως αν διαιρέσουμε τον αριθμό 21 των δυαδικών ψηφίων με τον αριθμό 11 των συμβόλων, που αποτελούν την λέξη mississippi, βρίσκουμε 1.90909 bits/σύμβολο — αριθμό μεγαλύτερο της εντροπίας, που είναι το κάτω φράγμα.

Στο παράδειγμα αυτό δεν μπορέσαμε να φτάσουμε το κάτω φράγμα διότι τα δυαδικά ψηφία δεν μεταφέρουν πληροφορία ενός bit. Αυτό σημαίνει ότι τα σύνολα που αντιπροσωπεύονται από τα δυαδικά ψηφία δεν είναι όλα ισοπίθανα!

Για παράδειγμα το πρώτο δυαδικό ψηφίο στην κωδικοποίηση της λέξης mississippi ξεχωρίζει τα σύνολα των γραμμάτων {i, m, p} και {s} οι πιθανότητες των οποίων είναι 7/11 και 4/11 αντίστοιχα.

Όπως είπαμε και στην αρχή αυτής της υποενότητας, η εντροπία σχετίζεται με τον λόγο συμπίεσης (compression ratio) μίας εικόνας Ο **λόγος συμπίεσης** είναι απλά ο λόγος του μεγέθους του αρχείου της εικόνας πριν και μετά την συμπίεση. Ισοδύναμα, ο λόγος συμπίεσης είναι ο λόγος των bits/εικονοστοιχείο πριν και μετά την συμπίεση.

Επειδή αρχικά έχουμε συνήθως 8 bits/εικονοστοιχείο και η εντροπία είναι ο συμπιεσμένος λόγος bits/εικονοστοιχείο έπεται πως

$$\text{λόγος συμπίεσης} = \frac{8}{\text{εντροπία}}.$$

Ακριβέστερα, $\frac{8}{\text{εντροπία}}$ είναι ένα **πάνω φράγμα** στον λόγο συμπίεσης.

Ψηφιοποίηση εικόνας

Θα εφαρμόσουμε τα προηγούμενα στην συμπίεση της εικόνας του Pedro (r.i.p.) — ενός πολύ αγαπημένου σκύλου — που βρίσκεται υπό μορφή encapsulated postscript στο αρχείο "pedro.eps".

Για να επεξεργαστούμε μία ασπρόμαυρη εικόνα πρέπει πρώτα να την κάνουμε ψηφιακή. Αυτό σημαίνει πως πρέπει να την μετατρέψουμε σε έναν πίνακα $m \times n$, κάθε στοιχείο του οποίου είναι η ένταση του αντίστοιχου εικονοστοιχείου. Για την περίπτωσή μας, που θέλουμε να εφαρμόσουμε τον συνημιτονικό

μετασχηματισμό DCT (*και συνεπώς τον* FFT), *απαιτούμε* $m = n = 2^k$, *για κάποια τιμή* $k \in \mathbb{N}$.

Γ*ια να κάνουμε λοιπόν την εικόνα ψηφιακή*, *διαβάζουμε κατ' αρχάς σαν* string *το περιεχόμενο του αρχείου* "pedro.eps".

```
string1 = ReadList[
   ToFileName[{"AddOns", "Applications"}, "pedro.eps"], String];
```

Από *το* string *αυτό αφαιρούμε τις πρόσθετες πληροφορίες που βρίσκονται στην αρχή και το τέλος του αρχείου και παίρνουμε το* string *που αντιστοιχεί στην εικόνα αυτή καθ' εαυτή*. Το *μήκος αυτού του* string *είναι* 314, *και φανερώνει τον αριθμό των σειρών της εικόνας μας*.

```
(***  For some versions of Mma we have to use
   string2=Drop[Drop[string1,17],-2];   ***)

string2 = Drop[Drop[string1, 18], -3];
Length[string2]
```

```
314
```

Κ*ατόπιν μετατρέπουμε το* string *σε λίστα χαρακτήρων*, *την οποία χωρίζουμε σε ζεύγη*, *π.χ*. {7, A}.

```
chars = Flatten[Characters[ string2]];
chars2 = Partition[chars, 2];
Length[chars2]
```

```
51496
```

Τ*α ζεύγη αυτά τα μετατρέπουμε σε* string *της μορφής* "16^^7A" (*βλέπε και την συνάρτηση του Mathematica* `BaseForm[]`) *και αυτά με την σειρά τους σε* "*έκφραση*. Ε*πειδή η μέγιστη τιμή*, 16^^FF, *είναι* 255, *προκύπτει μία λίστα τιμών* $\in$ [0, 255], *μήκους* 51496.

```
string3 = Map[StringJoin[Join[{"16^^"}, #]] &, chars2];
list1 = Map[ToExpression, string3];
Length[list1]
```

```
51496
```

Στην συνέχεια πρέπει να χωρίσουμε αυτήν την λίστα τιμών σε ομάδες στοιχείων (σειρές) κατά τέτοιο τρόπο ώστε να προκύψει ένας πίνακας με 314 γραμμές — όσες είναι και οι σειρές της εικόνας. Αν θέλουμε ο πίνακας της εικόνας μας να είναι τάξης 256×256 πρέπει το μήκος της λίστας των τιμών ∈ [0, 255] να είναι μεγαλύτερο του 65536 (= 256×256). Στην περίπτωσή μας το μήκος της λίστας αυτής είναι 51496 < 65536, οπότε θα χρειαστεί να κάνουμε ένα τρικ.

Με δοκιμές βλέπουμε πως πρέπει να χωρίσουμε την λίστα των τιμών ∈ [0, 255] σε ομάδες των 164 στοιχείων. Μόνο τότε προκύπτει πίνακας με 314 γραμμές και, φυσικά, 164 στήλες.

```
FactorInteger[51496]
```

```
{{2, 3}, {41, 1}, {157, 1}}
```

```
list2 = Partition[list1, 164];
Dimensions[list2]
```

```
{314, 164}
```

Για να αποκτήσουμε τον επιθυμητό πίνακα τάξης 256×256 στήλες, πρέπει να αφαιρέσουμε από τον παραπάνω πίνακα, τάξης 314×164, τις πλεονάζουσες 58 (= 314 - 256) γραμμές και να προσθέσουμε 92 (= 256 - 164) στήλες. Στο παράδειγμά μας αφαιρούμε τις πρώτες 58 γραμμές (θα μπορούσαμε κάλλιστα να είχαμε αφαιρέσει τις 58 τελευταίες — ανάλογα με την εικόνα) και προσθέτουμε 92 "άσπρες" στήλες — 46 αριστερά και 46 δεξιά.

```
whiteColumn = Table[255, {256}];
```

```
data = Transpose[
   Join[Table[whiteColumn, {46}], #, Table[whiteColumn, {46}]] &[
    Transpose[Drop[list2, -58]]]];
Dimensions[
 data]
```

```
{256, 256}
```

Αυτός *είναι ο πίνακας που αντιστοιχεί στην ψηφιακή εικόνα του Pedro — που φαίνεται στο* Figure *Pedro*-1 *— έτοιμος για συμπίεση.*

```
originalPicture = ListDensityPlot[data, Mesh → False];
```

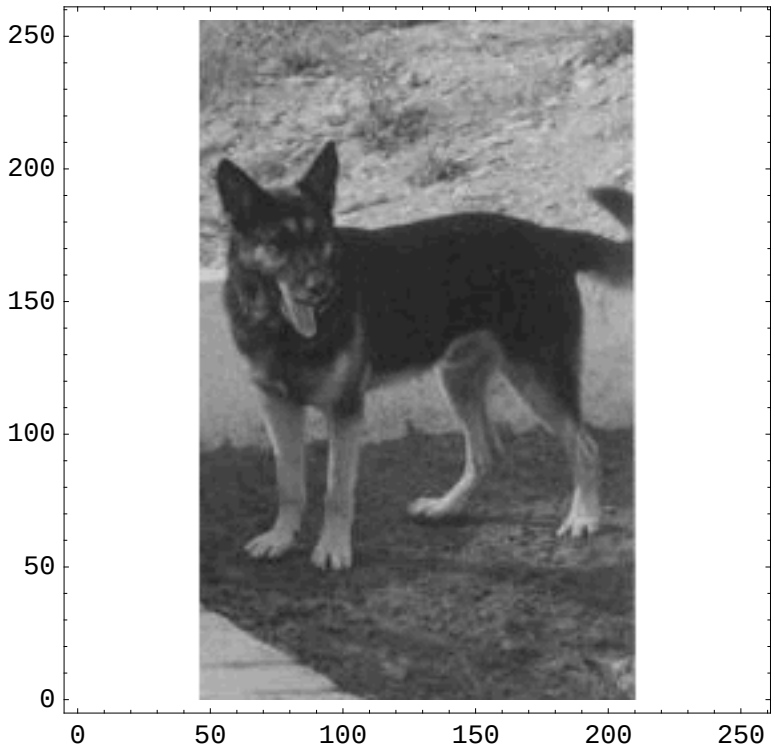


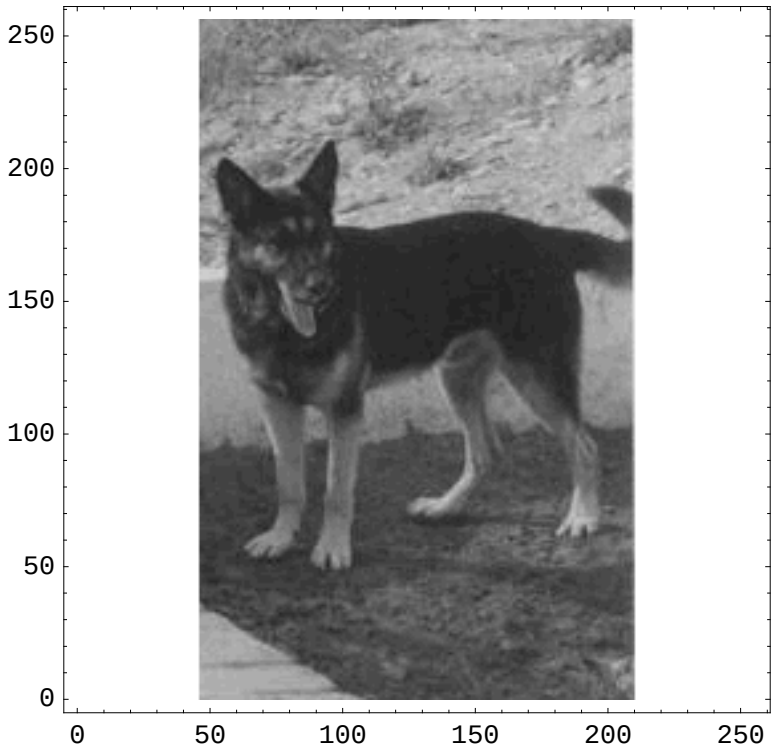
Figure *Pedro*-1: Η *αρχική εικόνα του* Pedro. Οι "*άσπρες*" *στήλες αριστερά και δεξιά της εικόνας είναι προφανείς*.

Αναφέρουμε *πως μπορούμε να αποθηκεύσουμε τον πίνακα τάξης* 256×256 *με την ψηφιακή εικόνα του* Pedro *στο αρχείο pedro.data*

```
Write[ToFileName[{"AddOns", "Applications"}, "pedro.data"], data];
```

απ΄όπου μπορούμε πάλι να τον διαβάσουμε και να τον αποδώσουμε στην μεταβλητή pedroDigitalData, την οποία και επεξεργαζόμαστε

```
pedroDigitalData =
  ReadList[ToFileName[{"AddOns", "Applications"}, "pedro.data"],
    Expression] // Last;

originalPicture = ListDensityPlot[pedroDigitalData, Mesh → False];
```



Συμπίεση εικόνας

Στην *συνέχεια θα πάρουμε την εικόνα του* Pedro, *η οποία βρίσκεται σε μορφή πίνακα τάξης* 256×256 (*ψηφιακή μορφή*) *στην μεταβλητή pedroDigitalData*, *και θα την συμπιέσουμε*.

Ακολουθώντας *τον αλγόριθμο συμπίεσης που αναφέραμε πιο πάνω*, ***επεκτείνουμε*** *τους ορισμούς των συναρτήσεων* DCT *και* iDCT *έτσι ώστε να εφαρμόζονται πρώτα σε κάθε σειρά και ύστερα σε κάθε στήλη ενός πίνακα*. Από *τον ορισμό των*

συναρτήσεων DCT *και* iDCT *συνεπάγεται πως οι σειρές και οι στήλες του πίνακα αυτού έχουν μήκος* 8.

```
DCT[array_?MatrixQ] :=
 Transpose[Map[DCT, Transpose[Map[DCT, array]]]]
iDCT[array_?MatrixQ] :=
 Transpose[Map[iDCT, Transpose[Map[iDCT, array]]]]
```

Γ*ια πίνακες στους οποίους το μήκος των σειρών και των στηλών είναι* > 8 (*όπως και του παραδείγματός μας*) ***επεκτείνουμε ακόμα μία φορά*** *τους ορισμούς των συναρτήσεων* DCT *και* iDCT *έτσι ώστε να εφαρμόζονται σε* "*μπλόκα*" (ή *υπολίστες*) *μήκους* 8:

```
DCT[list_?(Length[#] > 8 &)] :=
 Apply[Join, Map[DCT, Partition[list, 8]]]
iDCT[list_?(Length[#] > 8 &)] :=
 Apply[Join, Map[iDCT, Partition[list, 8]]]
```

Π*ροσέξτε στο σημείο αυτό τους* 3 *ορισμούς της συνάρτησης* DCT:

```
?DCT

Global`DCT

DCT[array_?MatrixQ] := Transpose[DCT /@ Transpose[DCT /@ array]]

DCT[list_?(Length[#1] > 8 &)] := Join @@ DCT /@ Partition[list, 8]

DCT[list_, L_ : 8] :=
 Re[DCTfitters[L] InverseFourier[list⟦Join[Table[i + 1,
       {i, 0, L - 1, 2}], Reverse[Table[i + 1, {i, 1, L - 1, 2}]]]⟧]]
```

Κ*ατόπιν*, *σύμφωνα πάντα με τον αλγόριθμο συμπίεσης*, *θα διαιρέσούμε τα στοιχεία αυτού του μετασχηματισμένου πίνακα με μία σταθερά* $q \in \mathbb{R}_{>1}$ (quantization) *και θα στρογγυλεύσουμε τα αποτελέσματα προς τον πλησιέστερο ακέραιο*. Το *αποτέλεσμα της στρογγύλευσης αυτής είναι ότι ελαττώνονται οι πιθανές τιμές φωτεινότητας των εικονοστοιχείων της εικόνας και συνεπώς και ο αριθμός των δυαδικών ψηφίων που χρειάζονται για την παράστασή τους*.

Η στρατηγική που ακολουθείται (ακολουθείτο) στο στάνταρτ JPEG είναι (ήταν) να διαιρούμε έναν 8×8 πίνακα με τον πίνακα φωτεινότητας *qValues*, όπου οι σταθερές *q* δεν είναι ίδιες, αλλά είναι μεγαλύτερες για τις υψηλές συχνότητες (μεγάλες τιμές του *k*). Βλέπε και το σχεδιάγραμμα των 8 διανυσμάτων της βάσης.

```
(qValues =
   {{16, 11, 10, 16, 24, 40, 51, 61},
    {12, 12, 14, 19, 26, 58, 60, 55}, {14, 13, 16, 24, 40, 57, 69, 56},
    {14, 17, 22, 29, 51, 87, 80, 62}, {18, 22, 37, 56, 68, 109, 103, 77},
    {24, 35, 55, 64, 81, 104, 113, 92}, {49, 64, 78, 87, 103,
     121, 120, 101}, {72, 92, 95, 98, 112, 100, 103, 99}});
```

Για να διαιρέσουμε στοιχείο προς στοιχείο έναν πίνακα τάξης $2^k \times 2^k$, $k \in \mathbb{N}_{>3}$, με τον πίνακα *qValues* πρέπει να χωρίσουμε τον πρώτο σε υποπίνακες τάξης 8×8, να κάνουμε την διαίρεση (και στρογγύλευση) και μετά να "επανενώσουμε" τους υποπίνακες που προκύπτουν με τα αποτελέσματα των πράξεων αυτών.

Ο χωρισμός ενός πίνακα σε υποπίνακες και η επανένωσή τους γίνονται με τις συναρτήσεις `blockMatrix[]` και `unBlockTensor[]`.

```
blockMatrix[pic_?MatrixQ, blocksize_: {8, 8}] :=
 Partition[pic, blocksize] /;
  Apply[And, Map[IntegerQ, (Dimensions[pic] / blocksize)]]

unBlockTensor[blocks_] :=
 Partition[Flatten[Transpose[blocks, {1, 3, 2}]],
  {Apply[Times, Dimensions[blocks][[{2, 4}]]]}]
```

Η διαίρεση ενός πίνακα τάξης 8×8 με τον πίνακα *qValues* και η στρογγύλευση του αποτελέσματος γίνεται με την συνάρτηση `DCTQ[]`. Εδώ λαμβάνουν χώρα μόνιμες "απώλειες" πληροφοριών, δηλαδή δεν είναι ανατρέψιμες με την αντίστροφη διαδικασία που γίνεται με την συνάρτηση `iDCTQ[]`.

```
DCTQ[pic_?MatrixQ, qValues_?MatrixQ] :=
 Map[(# / qValues) &, blockMatrix[pic, Dimensions[qValues]], {2}] //
   unBlockTensor // Round
```

```
iDCTQ[pic_?MatrixQ, qValues_?MatrixQ] :=
 Map[(# qValues) &, blockMatrix[pic, Dimensions[qValues]], {2}] //
  unBlockTensor
```

Είμαστε τώρα έτοιμοι για την συμπίεση της εικόνας του Pedro. Ο πίνακας της εικόνας έχει αποδωθεί στην μεταβλητή *pedroDigitalData*.

```
pedroDigitalData = data;
```

Ξεκινάμε με μερικά στατιστικά στοιχεία της μεταβλητής *pedroDigitalData*, όπως ελάχιστη, μέγιστη, και μέση τιμή, καθώς επίσης και τον αριθμό bytes που χρησιμοποιεί.

```
initialStatistics =
 {Min[pedroDigitalData], Max[pedroDigitalData], Mean[
    Map[Mean, pedroDigitalData]], ByteCount[pedroDigitalData]} // N
```

```
{31., 255., 158.575, 1.31997×10^6}
```

Στην συνέχεια θα εφαρμόζουμε μία-μία τις συναρτήσεις του αλγορίθμου συμπίεσης και θα εξετάζουμε τα αντίστοιχα στατιστικά στοιχεία με έμφαση τον αριθμό bytes που χρησιμοποιούνται.

Μετά τον συνημιτονικό μετασχηματισμό DCT (απαιτείται αρκετός χρόνος!)

```
pedroDigitalDataT = DCT[pedroDigitalData];

transformedStatistics =
 {Min[pedroDigitalDataT], Max[pedroDigitalDataT],
  Mean[Map[Mean, pedroDigitalDataT]], ByteCount[pedroDigitalDataT]}
```

```
{-393.197, 2040., 19.577, 524348}
```

βλέπουμε πως για τον μετασχηματισμένο πίνακα *pedroDigitalDataT* μειώθηκε αρκετά η μέση τιμή του πίνακα της εικόνας και τώρα χρησιμοποιούνται περίπου 500K bytes.

Τον μετασχηματισμένο πίνακα *pedroDigitalDataT* χωρίζουμε σε υποπίνακες τάξης 8×8 κάθε έναν από τους οποίους διαιρούμε στην συνέχεια στοιχείο προς στοιχείο με τον πίνακα *qValues* και στρογγυλεύουμε τα αποτελέσματα. (Εδώ λαμβάνουν χώρα οι μόνιμες "απώλειες" πληροφοριών που αναφέραμε.)

```
pedroDigitalDataTQ = DCTQ[pedroDigitalDataT, qValues];
```

```
roundedStatistics = {Min[pedroDigitalDataTQ],
   Max[pedroDigitalDataTQ], Mean[Map[Mean, pedroDigitalDataTQ]],
   ByteCount[pedroDigitalDataTQ]} // N
```

```
{-36., 128., 1.21906, 262204.}
```

Παρατηρούμε πως η μέση τιμή του πίνακα της εικόνας έχει δραστικά μειωθεί και τώρα χρησιμοποιούνται περίπου 250K bytes.

Μετρώντας τα bytes που χρησιμοποιούνται βλέπουμε πως **μέχρι το σημείο αυτό** ο λόγος συμπίεσης (δηλαδή ο λόγος του μεγέθους του πίνακα, σε bytes, πριν και μετά τον μετασχηματισμό) είναι περίπου 5.

```
ByteCount[pedroDigitalData] / ByteCount[pedroDigitalDataTQ] // N
```

```
5.03413
```

Στο σημείο αυτό της διαδικασίας μετράμε την εντροπία και βλέπουμε πως είναι 1.0041.

```
entropy[pedroDigitalDataTQ // Flatten]
```

```
1.0041
```

Στις ασκήσεις μελετάμε την ποιότητα της εικόνας σε συνάρτηση με την εντροπία της εικόνας που προκύπτει μετά την στρογγυλοποίηση.

Ακολουθεί τέλος συμπίεση χωρίς απώλειες με την συνάρτηση `SparseArray[]`

```
pedroCompressed = SparseArray[pedroDigitalDataTQ];
```

```
compressedStatistics = {Min[pedroCompressed], Max[pedroCompressed],
   Mean[Map[Mean, pedroCompressed]], ByteCount[pedroCompressed]} // N
```

```
{-36., 128., 1.21906, 64032.}
```

και βλέπουμε πως ο πίνακας pedroCompressed αποθηκεύεται χρησιμοποιώντας μόνο 64K bytes! (Τ*α άλλα στατιστικά δεδόμενα παρέμειναν όπως ήταν*.)

Ο *τελικός λόγος συμπίεσης είναι περίπου* 20.

```
ByteCount[pedroDigitalData] / ByteCount[pedroCompressed] // N
```

```
20.6142
```

Ας *δούμε τώρα την εικόνα του* Pedro *όπως προκύπτει από την αποσυμπίεση*. Α*κολουθούμε την αντίστροφη πορεία*: Π*ρώτα κάνουμε την αποσυμπίεση χωρίς* "*απώλειες*",

```
pedroDigitalDataTQDecom = Normal[pedroCompressed];
```

μετά κάνουμε την αποσυμπίεση με "*απώλειες*"

```
pedroDigitalDataTDecom = iDCTQ[pedroDigitalDataTQDecom, qValues];
```

και τέλος τον αντίστροφο συνημιτονικό μετασχηματισμό

```
pedroDecompressed = iDCT[pedroDigitalDataTDecom];
```

για να πάρουμε την αποσυμπιεσμένη εικόνα του Pedro.

```
quantizedPicture = ListDensityPlot[pedroDecompressed, Mesh → False];
```

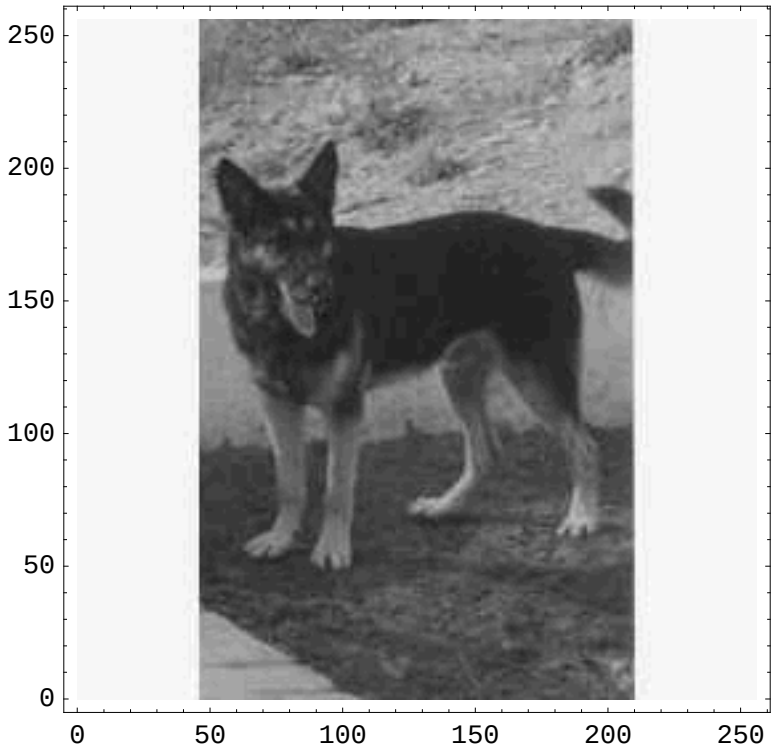
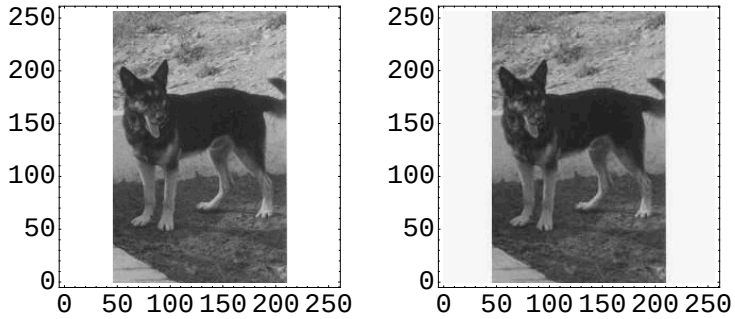


Figure *Pedro*-2: Η *εικόνα του* Pedro *όπως φαίνεται μετά την αποσυμπίεση*.

Συγκρίνοντας τις δύο εικόνες — την αρχική και αυτήν που προκύπτει μετά την αποσυμπίεση — βλέπουμε πως σχεδόν ταυτίζονται.

```
Show[GraphicsArray[{originalPicture, quantizedPicture}]];
```



Η όλη *διαδικασία της συμπίεσης*, *αποθήκευσης*, *επαναφοράς και αποσυμπίεσης* "*συμπιέζεται*" *σε μία γραμμή*. Πράγματι έχουμε:

```
pedroDecompressed =
  pedroDigitalData // DCT // DCTQ[#, qValues] & // SparseArray //
     Normal // iDCTQ[#, qValues] & // iDCT;
```

Αν *δεν μας ενδιαφέρει η αποθήκευση και επαναφορά της εικόνας παρά μόνον η μελέτη της εντροπίας των εικόνων μετά την στρογγύλευση η* όλη *διαδικασία* "*συμπιέζεται*" *ως εξής*:

```
pedroDecompressed = pedroDigitalData // DCT // DCTQ[#, qValues] & //
    iDCTQ[#, qValues] & // iDCT;
```

Ασκήσεις

Α1--με λύση: Στην απόδειξη του Κινεζικού θεωρήματος υπολοίπων πρέπει να δείξουμε πως η απεικόνηση φ είναι μονοσήμαντη και επί. Εξηγήστε γιατί αρκεί να δείξουμε πως για κάθε μοναδιαίο διάνυσμα $e_i = \{0, \ldots, 0, 1, 0, \ldots, 0\} \in R/\langle m_0 \rangle \times \cdots \times R/\langle m_{n-1} \rangle$ υπάρχει ένα στοιχείο $L_i \in R$, έτσι ώστε $\varphi(L_i) = e_i$.

Α2--με λύση: Προγραμματίστε τον Κινεζικό αλγόριθμο υπολοίπων, τόσο για τους ακεραίους όσο και για τα πολυώνυμα, χωρίς την εντολή `For[]`, αλλά με τον επεκταμένο Ευκλείδειο αλγόριθμο.

Α3: Αποδείξτε το Λήμμα 4.4.2 για τον χρόνο υπολογισμού του Κινεζικού αλγορίθμου υπολοίπων για ακεραίους.

Α4--με λύση: Προγραμματίστε τον Κινεζικό αλγόριθμο υπολοίπων για τους ακεραίους ώστε να υπολογίζει την λύση σταδιακά (αλγόριθμοι CRA2 και CRAn στην ενότητα 4.5).

Α5: Υπολογίστε τον χρόνο υπολογισμού της παρεμβολής του Νεύτωνα.

Α6: Αποδείξτε πως $\sum_{j=0}^{n-1} L_j(x) = 1$, όπου έχουμε ορίσει $L_j(x) = \prod_{i \neq j,\, 0 \leq i \leq n-1} \frac{x - u_i}{u_j - u_i}$.

Α7: Αποδείξτε πως για $R = \mathbb{Z}[x]$ και $u \in \mathbb{N}$, $\frac{f^{(i)}(u)}{i!} \in \mathbb{Z}$ για όλα τα i.

Α8--με λύση: Προγραμματίστε την αναστροφή των δυαδικών ψηφίων όταν θέλουμε να υπολογίσουμε τον μετασχηματισμό Fourier ενός διανύσματος για $R = \mathbb{C}$.

Α9--με λύση: Προγραμματίστε αναγωγικά τον γρήγορο μετασχηματισμό Fourier (FFT) με υπόλοιπα χωρίς διαιρέσεις, για $R = \mathbb{C}$, καθώς επίσης και την αναδιάταξη του αποτελέσματος με βάση την norm του $\mathbb{C}$.

A10: Α*ποδείξτε ότι ο τύπος του* Shannon *ικανοποιεί την σχέση* $0 \le H(p_0, \ldots, p_{\ell-1}) \le \log_2 \ell$. Ε*πιπλέον αποδείξτε ότι αριστερά η ισότητα ισχύει εάν και μόνον εάν* $\ell = 1$ *ενώ δεξιά η ισότητα ισχύει εάν και μόνον εάν* $p_0 = \cdots = p_{\ell-1} = \frac{1}{\ell}$.

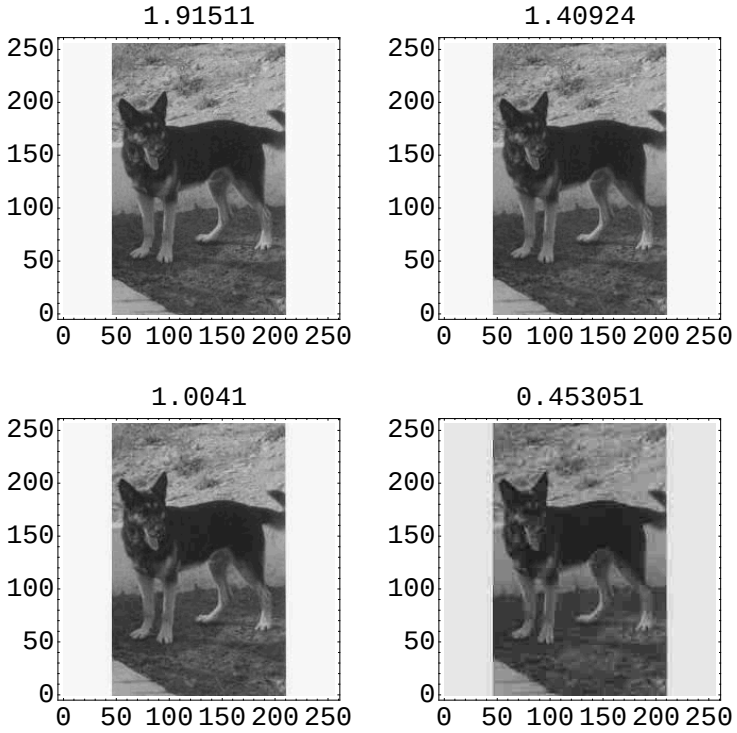
Υπόδειξη: Χ*ρησιμοποιείστε την ανισότητα* $\ln x \le x - 1$ *που ισχύει για όλα τα* $x \in \mathbb{R}_{>0}$ (*η ισότητα ισχύει εάν και μόνον εάν* $x = 1$) *και εφαρμόστε την στον τύπο του* Shannon *εκφρασμένο με τον φυσικό λογάριθμο* $\sum_{i=0}^{\ell-1} p_i \frac{\ln \frac{1}{p_i}}{\ln 2}$

A11: Χ*ρησιμοποιήστε τα παρακάτω προγράμματα για να δείτε την ποιότητα της εικόνας του* Pedro *για διάφορες τιμές της εντροπίας της εικόνας μετά την στρογγύλευση.* Οι *διαφορετικές τιμές της εντροπίας προκύπτουν πολλαπλασιάζοντας τον πίνακα qValues με διάφορες τιμές.*

Σ*το παράδειγμά μας πολλαπλασιάζουμε τον πίνακα qValues με τις τιμές* $\frac{1}{4}$, $\frac{1}{2}$, 1 *και* 4 (*στην μεταβλητή test*).

```
entropyAndQuantizedPic[pic_?MatrixQ, qValues_?MatrixQ] :=
 ({entropy[Flatten[#]], iDCT[iDCTQ[#, qValues]]} & )[
  DCTQ[DCT[pic], qValues]]

test = Map[entropyAndQuantizedPic[pedroDigitalData, qValues #] & ,
   {1 / 4, 1 / 2, 1, 4}];
```

```
Show[GraphicsArray[
   Partition[Map[ListDensityPlot[Last[#], Mesh → False, PlotLabel →
        ToString[First[#]], DisplayFunction → Identity] &, test], 2]
  ]
 ];
```



πρότυπο

- **Λύσεις *ορισμένων ασκήσεων***

Λύση της άσκησης A1:

Αυτό *αρκεί διότι για ένα άλλο τυχαίο διάνυσμα*

$$w = \{w_0 \bmod m_0, \ldots, w_{n-1} \bmod m_{n-1}\} \in R/\langle m_0\rangle \times \cdots \times R/\langle m_{n-1}\rangle$$

με $w_0, \ldots, w_{n-1} \in R$ *εύκολα βλέπουμε πως υπάρχει το στοιχείο* $\sum_{i=0}^{n-1} L_i w_i \in R$ *για το οποίο ισχύει* $\varphi(\sum_{i=0}^{n-1} L_i w_i) = w$. *Πράγματι έχουμε*

$$\varphi\left(\sum_{i=0}^{n-1} L_i w_i\right) = \sum_{i=0}^{n-1} \varphi(L_i)\varphi(w_i)$$

$$= \sum_{i=0}^{n-1} e_i.\{w_i \bmod m_0, \ldots, w_i \bmod m_{n-1}\}$$
$$= \sum_{i=0}^{n-1} \{0, \ldots, 0, w_i \bmod m_i, 0, \ldots, 0\} = w$$

Εδώ *πρέπει να προσέξουμε το εσωτερικό γινόμενο των διανυσμάτων* $\varphi(L_i)$ (*που ισούται με* e_i) *και* $\varphi(w_i)$ *καθώς επίσης και την πρόσθεση διανυσμάτων στο τέλος*.

Λύση της άσκησης Α2:

Για *τους ακεραίους*:

```
ourInteger2CRA[w_List, m_List] /; Length[w] == Length[m] :=
 Module[{mProduct = Apply[Times, m]},
  mm = mProduct/m;
  Mod[Apply[Plus, w mm Map[First[Last[#]] &,
      MapThread[ExtendedGCD[#1, #2] &, {mm, m}]]], mProduct]
 ]

ourInteger2CRA[{1, 2, 5}, {2, 5, 7}]
```

```
47
```

Για *τα πολυώνυμα*:

```
<< Algebra`PolynomialExtendedGCD`;

ourPoly2CRA[w_List, m_List] /; Length[w] == Length[m] :=
 Module[{mProduct = Apply[Times, m]},
  mm = mProduct/m; PolynomialMod[
   Apply[Plus, w mm Map[First[Last[#]] &, MapThread[
       PolynomialExtendedGCD[#1, #2] &, {mm, m}]]], mProduct]
 ]

ourPoly2CRA[{x + 5, x^2 + 2, 8}, {x^2 - 1, x^3 - 2, x - 4}]
```

$$\frac{968}{155} + \frac{8\,x}{3} + \frac{41\,x^2}{465} - \frac{329\,x^3}{155} - \frac{4\,x^4}{3} + \frac{212\,x^5}{465}$$

Λύση της άσκησης **A4**:

```
ourIntegerCRA2[w_List, m_List] /; Length[w] == Length[m] := Module[
  {g, q, r, s, t},
  r = Mod[w[[1]], m[[1]]];
  {g, {s, t}} = ExtendedGCD[m[[1]], m[[2]]];
  q = Mod[s (w[[2]] - r), m[[2]]];
  r = r + q m[[1]]
 ]
```

```
ourIntegerCRA2[{7, 5}, {10, 7}]
```

```
47
```

```
ourIntegerCRAn[w_List, m_List] /; Length[w] == Length[m] :=
 Module[{mProd = 1, V, W = w[[1]]},
  For[i = 0, i < Length[w] - 1, i++,
   mProd = mProd m[[i + 1]];
   V = ourIntegerCRA2[{W, w[[i + 2]]}, {mProd, m[[i + 2]]}]; W = V
  ]; W
 ]
```

```
ourIntegerCRAn[{1, 2, 5}, {2, 5, 7}]
```

```
47
```

Λύση της άσκησης **A8**:

```
resuffleList[list_] /; Log[2, Length[list]] ∈ Integers := Module[
  {dig, digLength, i, indeces, indecesReversed,
   indecesResuffled, listResuffled = {}, n = Log[2, Length[list]]},
  Table[i, {i, 0, 2^n - 1}];
indeces = Map[
    (dig = IntegerDigits[#, 2]; digLength = Length[dig]; If[digLength <
        n, Flatten[PrependTo[dig, Table[0, {i, n - digLength}]]],
       IntegerDigits[#, 2]]) &, Table[i, {i, 0, 2^n - 1}]];
indecesReversed = Map[Reverse, indeces];
indecesResuffled = Map[FromDigits[#, 2] &, indecesReversed];
  For[i = 1, i < 2^n + 1, i++, AppendTo[listResuffled,
    list[[indecesResuffled[[i]] + 1]]]]; listResuffled
 ]
```

Λύση της άσκησης Α9:

Για να λειτουργήσει η `fourierRWD[]` πρέπει να είναι ενεργή η συνάρτηση `remainderWithoutDivision[]`.

```
fourierRWD[list_, ωList_] /;
  Log[2, Length[list]] ∈ Integers && Length[list] == Length[ωList] :=
 Module[
  {evenPowersW, f, i, n = Length[list],
   oddPowersW, powersX, powersW = ωList, r0, r1, r1S},
  powersX = Table[x^i, {i, 0, n - 1}];
  evenPowersW = Table[powersW[[i + 1]], {i, 0, n - 1, 2}];
  oddPowersW = Table[powersW[[i + 1]], {i, 1, n, 2}];
  f = list.powersX;

  If[Exponent[f, x] == 1,

   (*then*)
   Return[Thread[
       {f /. x → evenPowersW, f /. x → oddPowersW}] // Flatten // N],

   (*else*)
   r0 = remainderWithoutDivision[f, x^(Length[list]/2) - 1, Length[list]];
   r1 = remainderWithoutDivision[f, x^(Length[list]/2) + 1, Length[list]];

   If[Exponent[r1, x] > 2,

    (*then*)
    ω = First[oddPowersW];
    r1S = r1 /. x → x ω;
    Thread[{fourierRWD[CoefficientList[r0, x], evenPowersW],
      fourierRWD[CoefficientList[r1S, x], evenPowersW]}],

    (*else*)
    Return[Thread[{r0 /. x → evenPowersW, r1 /. x → oddPowersW}]] //
      Flatten // N
   ]
  ]
 ]
```

```
resuffleList2[list_] /; Log[2, Length[list]] ∈ Integers := Module[
  {firstHalf, i, lista = {}, listb = {}, n = Length[list],
   realFirstHalf, secondHalf, setPairs, sortedPairs},
  If[n == 2, Return[list]]; i = 1;
  setPairs = Map[{Norm[#], i++} &, list];
  sortedPairs = Sort[setPairs, First[#1] > First[#2] &];
  firstHalf = Table[sortedPairs[[i]], {i, 3, n, 2}];
  realFirstHalf = PrependTo[firstHalf, First[sortedPairs]];
  Map[AppendTo[lista, list[[Last[#]]]] &, realFirstHalf];
  secondHalf = Table[sortedPairs[[i]], {i, 2, n, 2}];
  Map[AppendTo[listb, list[[Last[#]]]] &, secondHalf];
  Join[lista, Reverse[listb]]
 ]
```

Βιβλιογραφία

1. Akritas, A.G.: *Elements of Computer Algebra with Applications.* Wiley Interscience, New York, 1989.

2. Cajori, F.: Horner's method of approximation anticipated by Ruffini. *American Mathematical Society Bulletin* **17**, 409–414, 1911.

3. Dummit, D.S. and R.M. Foote: *Abstract Algebra*. Prentice-Hall, NJ 1991.

Wagon, S.: *Mathematica in Action.* W. H. Freeman and Company, NY 1991.

Cooley, J.W.: The re-discovery of the Fast Fourier Transform algorithm. *Mikrochimika Acta* **3**, 33–45, 1987.

Cooley, J.W. and J.W. Tuckey: An algorithm for the machine calculation of complex Fourier series. *Mathematics of Computation* **19**, 297–301, April 1965.

Heideman, M.T., Johnson, D.H. *και* C.S. Burrus: Gauss and the history of the Fast Fourier Transform. *IEEE ASSP Magazine* 14–21, 1984.

Watson, A. B.: Image compression using the Discrete Cosine Transform. *Mathematica Journal* **4(1)**, pp 81–88, 1994.

Schneider, T.D.: Information Theory Primer. National Cancer Institute, Frederick Cancer Research. (//www.lecb.ncifcrf.gov/~toms/)

Πρότυπο

C. Mulcahy: Plotting and Scheming with Wavelets, *Mathematics Magazine* **69,** pp 323–343, 1996.

G. Nakos and D. Joyner: Linear Algebra with Applications. Brooks/Cole Publishing Company. New York, 1998.

G. Strang: Signal Processing for Everyone. November 1, 2000. Report downloaded from the internet.

1. William E. Boyce and Richard C. DiPrima: *Elementary Differential Equations and Boundary Value Problems*, John Wiley & Sons, New York, NY, 1997.

2. Bill Davis and Jerry Uhl: *Differential Equations & Mathematica*, Math Everywhere, Inc., 1999 (Part of the "Calculus & Mathematica" series of books).

3. David Kahaner, Cleve Moler and Stephen Nash: *Numerical Methods and Software*, Prentice Hall, Englewood Cliffs, New Jersey, 1989.

4. Walter Strampp, Victor Ganzha and Evgenij Vorozhtsov: *Höhere Mathematik mit Mathematica*, Vieweg Lehrbuch Computeralgebra, Braunschweig/Wiesbaden, 1997.

5. H. Joseph Weaver: *Applications of Discrete and Continuous Fourier Analysis*, John Wiley & Sons, New York, NY, 1983.

Κεφάλαιο 5: Μετασχηματισμοί Wavelets και συμπίεση εικόνων

Wavelet Transforms trace their origin both to Signal Processing and Theoretical Mathematics. Since their introduction they have found applications in many areas — most notably in fingerprinting by the FBI, where they are used to compress fingerprint data before storing it. In this review article, using *Mathematica* as a paradigm, we first establish the relation of wavelet transforms to adoptive plotting of functions and then show their development with filter banks and multiresolution analysis. The book by George Nakos and the papers by Colm Mulcahy and Gilbert Strang were inspirational and material was taken from them at will.

5.1 Uniform and Adaptive Plotting of Functions

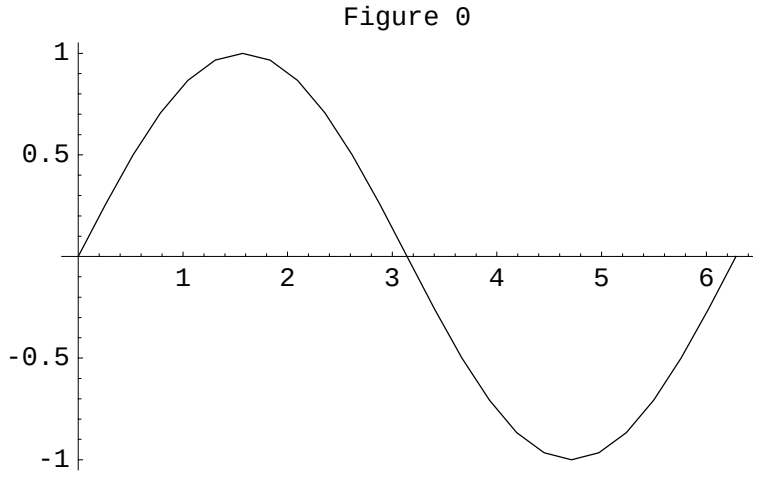
Plotting various functions with a computer algebra system like *Mathematica* is an activity quite similar to signal processing; to wit, in both cases we take samples. In this section we review standard ways of plotting discrete data sets and two dimensional digital images. The inherent difficulties of plotting functions by uniform sampling will lead us to adoptive plotting techniques (the main idea of which is at the heart of wavelet transforms) and to techniques based on wavelets (see the section on Image Compression with the Haar Transform). We also indicate the need for data compression.

■ Uniformly distributed sample points

Suppose we have a set of n planar data points $\{x_i, y_i\}$, i = 1, ..., n, which are samples of the function $y = f(x)$. A common way to plot $f(x)$ from this set of points is to plot these points individually and then join adjacent points with line segments. This can be achieved with the *Mathematica* function ListPlot — used with option PlotJoined→True**.**

Earlier versions of *Mathematica* used — in their Plot function — a fixed number of (just 25) points x_i uniformly distributed in the interval $[a, b]$. As an example, in Figure 0 below we see a "darn good" plot of sin(x) using ListPlot.

```
xPoints0 = Table[x, {x, 0, 2 π, 2 π / 24}];
yPoints0 = Map[Sin[ #] &, xPoints0];
ListPlot[Transpose[{xPoints0, yPoints0}],
  PlotJoined → True, PlotLabel → "Figure 0"];
Print["Sin[x] in the interval [0, 2π];
    25 sample points were used"];
```



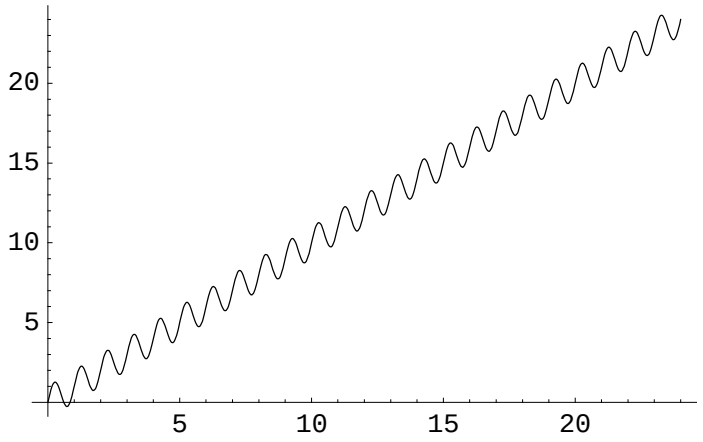

```
Sin[x] in the interval [0, 2π];  25 sample points were used
```

However, this practice of using a fixed number of points x_i uniformly distributed in the interval [a, b] leads to serious problems as we see in the following examples. Please note that the plots of Figures 2 and 4 below (obtained using the function ListPlot) could be obtained using the function Plot of older versions of *Mathematica* — drawing, at the time, a lot of flak.

Consider, first, the function $f(x) = x + \sin(2\pi x)$, in the interval $0 \le x \le 24$. Its plot is shown in Figure 1 below:

```
Plot[x + Sin[2 π x], {x, 0, 24}, PlotLabel → "Figure 1"];
Print["The function x+Sin[2πx] in the interval [0, 24]"];
```

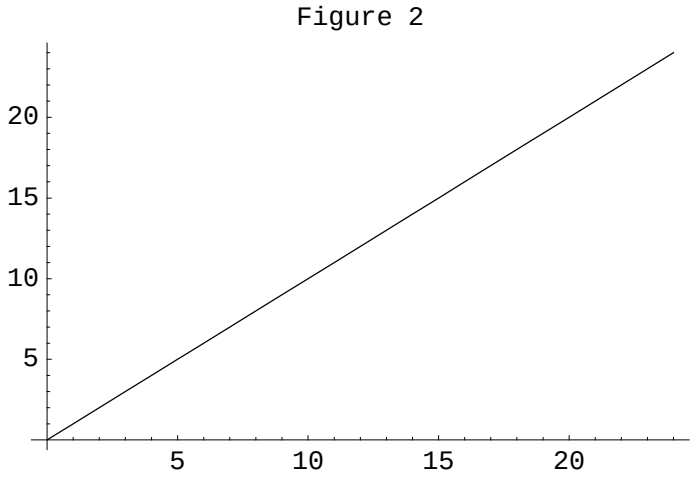
Figure 1




```
The function x+Sin[2πx] in the interval [0, 24]
```

Using the uniform sampling approach mentioned above, and sampling at the points x = 0, 1, 2, ... 24, we get the straight line shown in Figure 2.

```
xPoints1 = Table[x, {x, 0, 24}];
yPoints1 = Map[# + Sin[2 π #] &, xPoints1];
ListPlot[Transpose[{xPoints1, yPoints1}],
  PlotJoined → True, PlotLabel → "Figure 2"];
Print["Aliased version of the function x+
    Sin[2πx]; 25 sample points were used"];
```

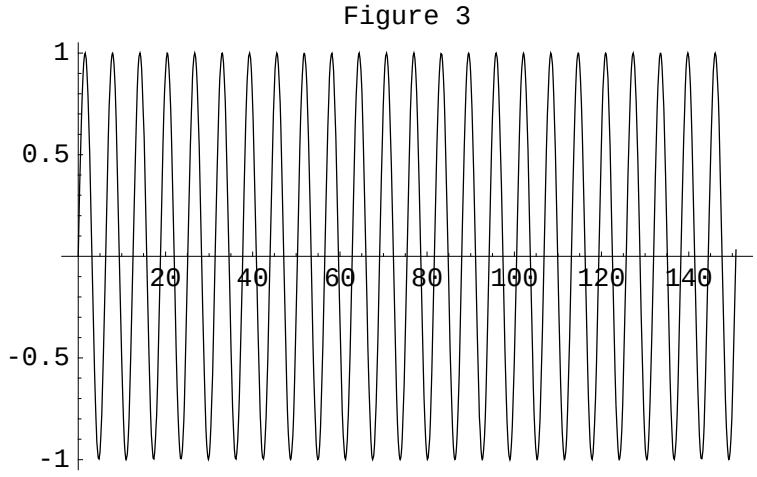



```
Aliased version of the function
  x+Sin[2πx]; 25 sample points were used
```

In the vernacular of signal analysis this is called *aliasing*, whereby a slow frequency appears instead of a higher frequncy because the discrete samples are the same. In our example we get an "aliased" version of the desired function. The reason for aliasing is that we did *not* use enough sample points (and at the points where we did sample, i.e. $x = 0, 1, 2, \ldots 24$, we have $x = x + \sin(2\pi x)$).

As a second example of aliasing consider the function $\sin(x)$, in the (storied) interval $0 \le x \le 48.01\pi$. Its true nature is shown in Figure 3.

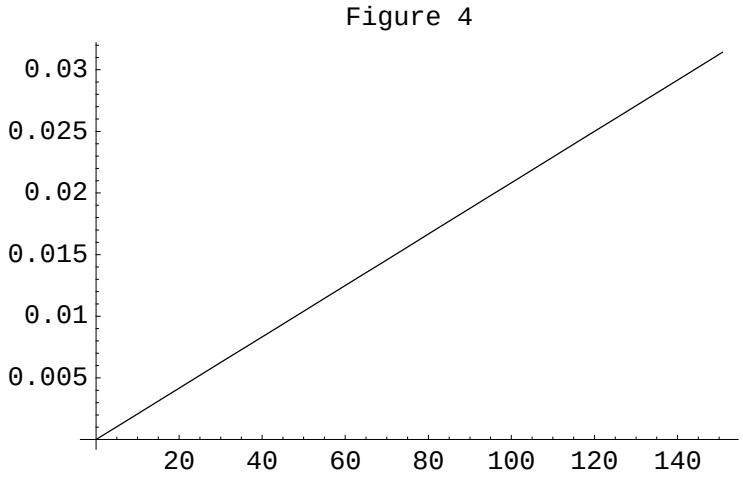
```
Plot[Sin[x], {x, 0, 48.01 π}, PlotLabel → "Figure 3"];
Print["The function Sin[x] in the interval [0, 48.01π]"];
```



The function Sin[x] in the interval [0, 48.01π]

However, using only 25 uniformly sampled points in the interval [0, 48.01π] we obtain the alias of Figure 4.

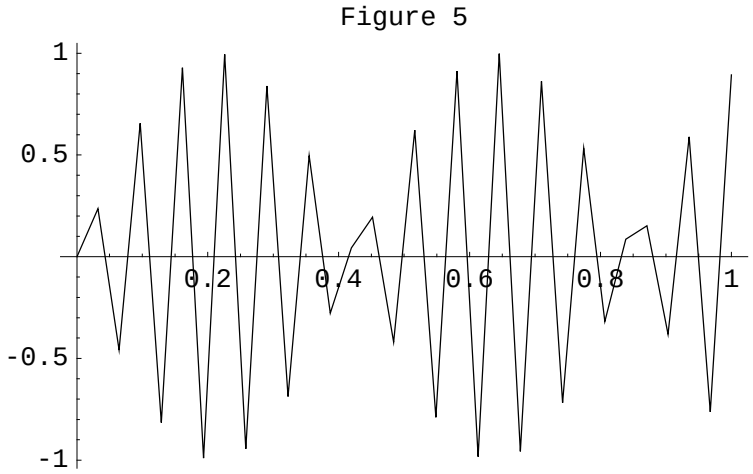
```
xPoints2 = Table[x, {x, 0, 48.01 π, 48.01 π / 24}];
yPoints2 = Map[Sin[ #] &, xPoints2];
ListPlot[Transpose[{xPoints2, yPoints2}],
  PlotJoined → True, PlotLabel → "Figure 4"];
Print["Aliased version of the function Sin[x] in the
    interval [0, 48.01π];  25 sample points were used"];
```




```
Aliased version of the function Sin[x] in the
  interval [0, 48.01π];  25 sample points were used
```

A final example of aliasing is obtained when we try to plot the function sin(90*x*) using only 32 uniformly spaced sample points. Instead of a horizontally telescoped version of the sin(*x*) function we obtain the pulsing pattern shown in Figure 5.

```
xPoints3 = Table[x, {x, 0, 1, 1 / 31}];
yPoints3 = Map[Sin[90 #] &, xPoints3];
ListPlot[Thread[{xPoints3, yPoints3}],
  PlotJoined → True, PlotLabel → "Figure 5"];
Print["Aliased version of the function Sin[90x] in the
   interval [0, 1]; 32 uniformly sampled points were used"]
```




```
Aliased version of the function Sin[90x] in the
  interval [0, 1]; 32 uniformly sampled points were used
```

Some definitions are needed now in order to state the famous theorem explaining alias. Consider the sinusoid $x(t) = \sin(\omega t)$. This repeats whenever the time t is increased by $T = \frac{2\pi}{\omega}$, its *period*. Suppose T is measured in seconds. Then the numbers of cycles in one second is $f = \frac{1}{T}$. Thus, the frequency in Hertz is $f = \frac{1}{T} = \frac{\omega}{2\pi}$. We can now state the following:

Shannon's Sampling Theorem: Every analog signal $x(t)$ with frequencies not exceeding $\omega_{\max}$ can be perfectly reconstructed from its discrete samples $x(nT)$, provided the sampling rate $\frac{1}{T}$ exceeds $2 f_{\max} = \frac{\omega_{\max}}{\pi}$ (Nyquist rate).

Note that the critical Nyquist rate has two samples per oscillation! Also, reconstruction requires a strict inequality -- the Nyquist rate cuts it too close.

According to Shannon's theorem, the first two examples fail because we have one sample per oscillation. In the third example, we observe that we have enough sample points for the function $\sin(90x)$, since $32 > 2 f_{\max} = \frac{\omega_{\max}}{\pi} = \frac{90}{\pi} = 28.6479$ (Nyquist rate), but nonetheless aliasing appears. This is not unusual, and using a few more points yields the anticipated plot.

■ Adoptive plotting

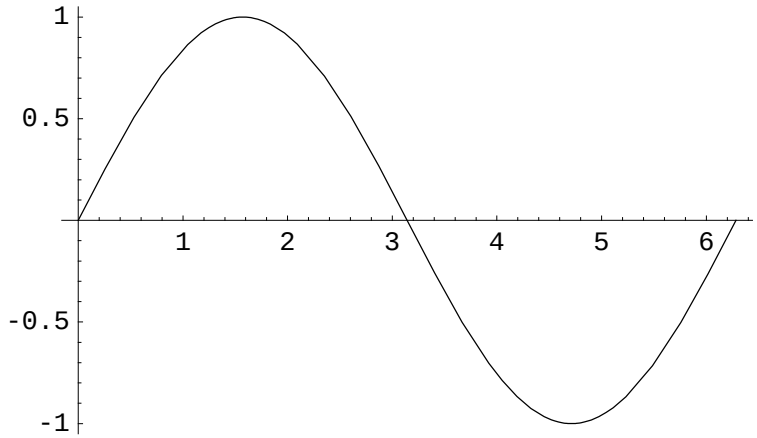
From the above examples it is obvious that uniformly spaced sample points is not the right way to go, unless we are prepared to use a lot of points; this is done in the next section where we use a large number of uniformly spaced sample points together with wavelet transforms for "data compression".

To correct the errors described in the above examples, adoptive plotting (or sampling) techniques are used with points clustered where the function exhibits great variation. These routines initially generate an internal plot based on uniform sampling and they examine the angles between the connecting line segments in order to identify regions of great variation. Having identified regions of great variation, they next subdivide the corresponding intervals further before producing the visible plot.

This is precisely what happens when the current version *Mathematica* plots the function $\sin(x)$, $0 \le x \le 2\pi$. As we can see 82 sample points were used to make the plot of Figure 6.

```
n = 0;
xlow = 0;
xhigh = 2 π;

Plot[++n; Sin[x], {x, xlow, xhigh}, PlotLabel → "Figure 6"];
Print["For this plot ", n,
 " sample points were used in the interval [",
 xlow, ", ", xhigh, "]"]
```

Figure 6




```
For this plot 82 sample points were used in the interval [0, 2 π]
```

Note that these 82 points are *not* uniformly distributed in the interval $[0, 2\pi]$. Letting *xhigh* vary successively from 1 to 6 we see that in the interval (0, 1] there were 25 sample points, in the interval (1, 2]

there were 29 sample points, in *each* of the intervals (2, 3], and (3, 4] there was 1 sample point, in the interval (4, 5] there were 24 sample points and finally, in the interval (5, 6] there were 2 sample points.

We can also look at these sample points. For example to plot sin(*x*) in the interval [0, 0.5] the following points were used:

```
n = 0;
xlow = 0;
xhigh = 0.5;

Plot[Print["n = ", ++n, ", x[", n, "] = ", x]; Sin[x],
  {x, xlow, xhigh}, PlotPoints → 5, DisplayFunction → Identity];
```

```
n = 1, x[1] = 1.25 × 10^-7

n = 2, x[2] = 0.121701

n = 3, x[3] = 0.254426

n = 4, x[4] = 0.379078

n = 5, x[5] = 0.498955

n = 6, x[6] = 0.5
```

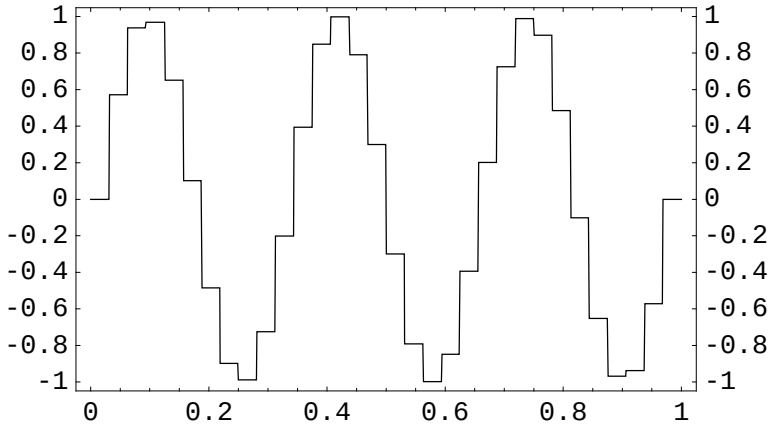
To recap: When there is no variation we need few points to recover the plot of a function and, conversly, when there is variation we need many points. This point will be used in the wavelet transforms; to wit, if a digital image is "smooth" (mostly black or white) then we can recover the image keeping very few of the "details" nonzero.

■ Step functions

Finally, as a prelude to the discussion on wavelets we mention that linear interpolation of uniformly sampled points is *not* the only way of plotting; instead of joining the points with line segments, we could use the *y*-values as step levels for a staicase effect. The function sin(6πx) is plotted this way in Figure 7.

```
fi[x_] := Which[0 ≤ x < 1, 1, True, 0]; fi[x_, i_, k_] := fi[2^i x - k];
xPoints4 = Table[x, {x, 0, 1, 1 / 31}];
yPoints4 = Map[Sin[6 π #] &, xPoints4];
yStep4 = Table[fi[t, 5, k], {k, 0, 31}];
Plot[yPoints4.yStep4, {t, 0, 1}, PlotLabel → "Figure 7",
 Axes → {False, False} , Frame → True, FrameTicks →
  {Automatic, {-1, -0.8, -0.6, -0.4, -0.2, 0, 0.2, 0.4, 0.6, 0.8, 1}}];
Print["Step function version of the function Sin[6πx]
    in the interval [0, 1]; 32 sample points were used"];
```

Figure 7




```
Step function version of the function Sin[6πx]
  in the interval [0, 1]; 32 sample points were used
```

This should come as no surprise since we know from Calculus that continuous functions on [0, 1] can be approximated to any degree of precision with linear functions or by step functions. These staircase approximations lead to better and better plots as the number of sample points increases, although a lot more are needed to obtain the continuous effect. In this context questions of data storage and transmission arise. These questions become crucial when we explore digital images which are higher-dimensional analogues of data points in the plane.

Black and white digital images are derived from two-dimensional arrays of *pixels.* A pixel is the smallest unit of space in a digital image. Associated with each pixel there is a number, in the range [0, 1], representing gray levels ranging from black (0) to white (1). So now our data points are triplets $\{x, y, z\}$, where z measures the gray level at pixel position $\{x, y\}$. Viewed from above we have two-dimensional step functions, where the steps are shaded according to their height. (Color images can be dealt with by decomposing them into their red, green and blue components and treating each of these like grayscales.) When 256 ($= 2^8$) levels of gray are used, we are talking about 8-*bit images.*

Digital images come in various resolutions. An image with resolution 1600×1200 pixels is derived from a matrix of 1600×1200 = 1,920,000 pieces of data, each representing a gray level. Clearly, images with such high resolution are difficult to store and time-consuming to transmit over low band line. Therefore, we should try to come up with a more economical way to store the matrices that represent these images. This is achieved by taking advantage of regions in the image where there is little or no variation, (for example, a black coat in the picture).

5.2 Signals and Wavelet Transforms

Most signals start their lives in analog form. They become digital by sampling at equal time intervals, $x_{\text{digital}}(n) = x_{\text{analog}}(nT)$, where $n = 0, \pm1, \pm2, \ldots$ and T is the *sampling interval.*

The Discrete Time Fourier Transform turns the samples $x_{\text{digital}}(n) = x(n)$ into the coefficients of a 2π -periodic function $\chi(\omega) = \sum_{n=-\infty}^{\infty} x(n)\, e^{-i n \omega}$. We refer to ω as the *frequency*, and we plot the transform $\chi(\omega)$ in the interval $\omega = -\pi$ and $\omega = \pi$. Then "low frequencies" refer to frequencies near zero, and "high frequencies" have $|\omega| \approx \pi$.

Notice that the signal $x(n) = \{..., 1, 1, 1, 1, 1, ...\}$ has exactly zero frequency ($\omega = 0$, the lowest frequency), whereas the alteration between 1 and -1 gives the signal $x(n) = \{..., 1, -1, 1, -1, 1, ...\}$ with the highest frequency $\omega = \pm\pi$. In between these two extremes we have the family of pure sinusoidal signals $x(n) = e^{i n \omega}$ with frequency $0 < |\omega| < \pi$.

■ Lowpass and Highpass Filters

Sampling, *transforming* and *filtering* constitute the basic signal processing operations. Of the three, filtering (and filters) will be of importance to us.

Filters are also called *linear time-invariant* systems. When presented with the input $x_\omega(n)$ these systems produce the output $H(\omega)\, x_\omega(n)$, where the amplifying factor $H(\omega)$ is the *frequency response.* Hence, filters act on signals to produce modified signals and this is the most important operation in signal processing. Below we examine filters that are: (a) *finite impulse response* (FIR), meaning each output is a linear combination of a *finite* number of input samples, (b) *time-invariant*, meaning their coefficients are constant for all time, and (c) *causal*, meaning they involve no future samples like $x(n+1)$. (Note that for images the situation is different because n refers to position, not time. Therefore, once the complete image is available, filters in image processing need not be causal.)

Lowpass Filter: The first example is the moving average filter

$$yLP[n\_] := \frac{1}{2}\, x[\![n-1]\!] + \frac{x[\![n]\!]}{2}$$

We are going to consider three special inputs, the impulse, the constant and the alternating signal. For the impulse $x(n) = \{..., 0, 0, 1, 0, 0, ...\}$ the output is $y_{\text{LP}}(n) = \{..., 0, 0, \frac{1}{2}, \frac{1}{2}, 0, ...\}$; that is, the impulse response contains the filter coefficients.

```
Clear[x]; x = Insert[Table[0, {n, 1, 19}], 1, 10]
```

```
{0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}
```

```
Table[yLP[n], {n, 2, 20}]
```

$$\left\{0, 0, 0, 0, 0, 0, 0, 0, \frac{1}{2}, \frac{1}{2}, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\right\}$$

For the constant signal $x(n) = \{..., 1, 1, 1, 1, 1, ...\}$ the output is $y_{LP}(n) = \{..., 1, 1, 1, 1, 1, ...\}$. That is, the response exactly equals the input; the frequency $\omega = 0$ is in the *passband*. Taking a look at a finite number of samples we have:

```
x = Table[1, {n, 1, 20}]
```

```
{1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1}
```

```
Table[yLP[n], {n, 2, 20}]
```

```
{1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1}
```

On the other hand, for the alternating signal $x(n) = \{..., 1, -1, 1, -1, 1, ...\}$ the output is $y_{LP}(n) = \{..., 0, 0, 0, 0, 0, ...\}$. In other words, the response is zero; the frequency $\omega = \pm\pi$ is in the *stopband*.

```
x = Table[(-1)^n, {n, 1, 20}]
```

```
{-1, 1, -1, 1, -1, 1, -1, 1, -1, 1, -1, 1, -1, 1, -1, 1, -1, 1, -1, 1}
```

```
Table[yLP[n], {n, 2, 20}]
```

```
{0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}
```

For the in between frequencies, i.e. $0 < \omega < \pi$, consider the input signal $x(n) = e^{i n \omega}$. The output then is

$$y_{LP}(n) = \frac{1}{2} e^{i n \omega} + \frac{1}{2} e^{i(n-1)\omega} = \left(\frac{1}{2} + \frac{1}{2} e^{-i\omega}\right) e^{i n \omega}$$

In this case, the output frequency is the same ω, but the filter multiplies each frequency component of the input by the *frequency response* function $H(\omega) = \frac{1}{2} + \frac{1}{2} e^{-i\omega}$.

As we have seen,, at $\omega = 0$ the frequency response is $H = 1$; i.e. the constant signal passes unchanged through the filter. At $\omega = \pi$ the frequency response is $H = 0$; that is, the alternating signal is completely

blacked out. Writing the response function as

$$H(\omega) = \frac{1}{2} + \frac{1}{2} e^{-i\omega} = e^{-i\omega/2}\left(\frac{1}{2} e^{i\omega/2} + \frac{1}{2} e^{-i\omega/2}\right) = e^{-i\omega/2} \cos\frac{\omega}{2}.$$

we see that the amplitude is $|H(\omega)| = \cos\frac{\omega}{2}$, which drops from one at $\omega = 0$ to zero at $\omega = \pi$. This is the *transition band* for the filter.

Highpass Filter: The second example is the moving difference filter

```
yHP[n_] := 1/2 x[[n]] - 1/2 x[[n - 1]]
```

For the impulse $x(n) = \{..., 0, 0, 1, 0, 0, ...\}$ the output is $y_{HP}(n) = \{..., 0, 0, \frac{1}{2}, -\frac{1}{2}, 0, ...\}$; that is, the impulse response again contains the filter coefficients.

```
Clear[x]; x = Insert[Table[0, {n, 1, 19}], 1, 10]
```

```
{0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0}
```

```
Table[yHP[n], {n, 2, 20}]
```

$$\left\{0, 0, 0, 0, 0, 0, 0, 0, \frac{1}{2}, -\frac{1}{2}, 0, 0, 0, 0, 0, 0, 0, 0\right\}$$

For the constant signal $x(n) = \{..., 1, 1, 1, 1, 1, ...\}$ the output is $y_{HP}(n) = \{..., 0, 0, 0, 0, 0, ...\}$. That is, the lowest frequency $\omega = 0$ is stopped. Taking a look at a finite number of samples we have:

```
x = Table[1, {n, 1, 20}]
```

```
{1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1}
```

```
Table[yHP[n], {n, 2, 20}]
```

```
{0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}
```

On the other hand, for the alternating signal $x(n) = (-1)^n$ the output is $y_{HP}(n) = (-1)^n$. In other words, the frequency $\omega = \pi$ passes without any change.

```
x = Table[(-1)^n, {n, 1, 20}]
```

```
{-1, 1, -1, 1, -1, 1, -1, 1, -1, 1, -1, 1, -1, 1, -1, 1, -1, 1, -1, 1}
```

```
Table[yHP[n], {n, 2, 20}]
```

```
{1, -1, 1, -1, 1, -1, 1, -1, 1, -1, 1, -1, 1, -1, 1, -1, 1, -1}
```

For the in between frequencies, i.e. $0 < \omega < \pi$, consider again the input signal $x(n) = e^{i n \omega}$. The output then is

$$y_{\mathrm{HP}}(n) = \frac{1}{2} e^{i n \omega} - \frac{1}{2} e^{i(n-1)\omega} = (\frac{1}{2} - \frac{1}{2} e^{-i\omega}) e^{i n \omega}.$$

The multiplying factor is the frequency response function $H(\omega) = \frac{1}{2} - \frac{1}{2} e^{-i\omega}$. Again we write as

$$\mathrm{H}(\omega) = \frac{1}{2} - \frac{1}{2} e^{-i\omega} = e^{-i\omega/2}(\frac{1}{2} e^{i\omega/2} - \frac{1}{2} e^{-i\omega/2}) = e^{-i\omega/2}\, i \sin \frac{\omega}{2},$$

in which case the amplitude is $|H(\omega)| = \sin \frac{\omega}{2}$, which rises from zero at $\omega = 0$ to one at $\omega = \pi$. This is the transition band for the filter.

In summary, a lowpass filter preserves the *smooth* part of a signal, whereas a highpass filter preserves the *rough and noisy* part. In wavelet language, a lowpass filter gives averages whereas a highpass filter gives details. In some applications those details are important to keep (like edges in an image) whereas in other applications the high frequencies are mostly noise from measurements.

■ Convolutions

A good lowpass filter has many uses, so we look now at the (FIR, linear time-invariant and causal) filter with N+1 coefficients $h(0)$, $h(1)$, ..., $h(N)$. This is a filter of *order N* with $N + 1$ coefficients. At each time step the $N + 1$ coefficients multiply $N + 1$ samples from the input signal – starting with the current sample $x(n)$ and going back to the sample $x(n - N)$. Hence, the output is:

$$y(n) = h(0)\, x(n) + h(1)\, x(n-1) + \cdots + h(N)\, x(n-N).$$

We may write the action of the filter in the time domain compactly as the sum:

$$y(n) = \sum_{k=0}^{N} h(k)\, x(n-k).$$

Convolution is the fundamental operation of discrete time-invariant systems and *filters are discrete convolutions!* This convolution is easily implemented in hardware using three building blocks: *unit delays, multipliers and adders.*

Convolution is also easily implemented in softwrae. For example, the filter $h = \{\frac{1}{4}, \frac{1}{2}, \frac{1}{4}\}$ is lowpass, since the alternating signal $\omega = \pi$ is eliminated

```
yLP[n_] := 1/4 x[[n]] + 1/2 x[[n - 1]] + 1/4 x[[n - 2]]
```

```
Clear[x]; x = Table[(-1)^n, {n, 1, 20}]
```

```
{-1, 1, -1, 1, -1, 1, -1, 1, -1, 1, -1, 1, -1, 1, -1, 1, -1, 1, -1, 1}
```

```
Table[yLP[n], {n, 3, 20}]
```

```
{0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}
```

whereas the constant signal is preserved. (Note that the filter coefficients sum up to 1.)

```
x = Table[1, {n, 1, 20}]
```

```
{1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1}
```

```
Table[yLP[n], {n, 3, 20}]
```

```
{1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1}
```

Note again how the impulse response contains the filter coefficients.

```
x = Insert[Table[0, {n, 1, 19}], 1, 10]
```

```
{0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}
```

```
Table[yLP[n], {n, 3, 20}]
```

$$\left\{0, 0, 0, 0, 0, 0, 0, \frac{1}{4}, \frac{1}{2}, \frac{1}{4}, 0, 0, 0, 0, 0, 0, 0, 0\right\}$$

The interesting part starts when the input is the linear signal x = {0, 1, 2, 3, 4}. Padding the signal with two zeros on both ends (since the filter is of order $N = 2$) we have:

```
x = {0, 0, 0, 1, 2, 3, 4, 0, 0}; Table[yLP[n], {n, 3, 9}]
```

$$\left\{0, \frac{1}{4}, 1, 2, 3, \frac{11}{4}, 1\right\}$$

Note that the signal is preserved at the center!

We can also easily implement lowpass filters using the ListConvolve function of *Mathematica* (see also the Help Browser for this function). As expected, the result is the same as the one obtained above.

```
ListConvolve[{1/4, 1/2, 1/4}, {0, 1, 2, 3, 4}, {1, -1}, 0]
```

$$\left\{0, \frac{1}{4}, 1, 2, 3, \frac{11}{4}, 1\right\}$$

Observe that a unit *delay* is introduced between input and output. Notice that five terms convolved with three terms produce seven terms. We convolved an input signal of length $L = 5$, with a second order filter, $N = 2$. The output is of length $L + N = 7$, which is reasonable given that a fourth degree polynomial, $0 + x + 2\,x^2 + 3\,x^3 + 4\,x^4$, times a second degree polynomial, $\frac{1}{4} + \frac{1}{2}\,x + \frac{1}{4}\,x^2$ yields a sixth degree polynomial, $0 + \frac{1}{4}\,x + x^2 + 2\,x^3 + 3\,x^4 + \frac{11}{4}\,x^5 + x^6$.

It is extermely interesting to see how ListConvolve works. The linear signal is *inverted* and padded with $N = 2$ zeros at both ends to obtain the list {0, 0, 4, 3, 2, 1, 0, 0, 0}. Starting from the right end we form the dot products $\{\frac{1}{4}, \frac{1}{2}, \frac{1}{4}\}.\{0, 0, 0\} = 0$, $\{\frac{1}{4}, \frac{1}{2}, \frac{1}{4}\}.\{1, 0, 0\} = \frac{1}{4}$, $\{\frac{1}{4}, \frac{1}{2}, \frac{1}{4}\}.\{2, 1, 0\} = 1$, $\{\frac{1}{4}, \frac{1}{2}, \frac{1}{4}\}.\{3, 2, 1\} = 2$, $\{\frac{1}{4}, \frac{1}{2}, \frac{1}{4}\}.\{4, 3, 2\} = 3$, $\{\frac{1}{4}, \frac{1}{2}, \frac{1}{4}\}.\{0, 4, 3\} = \frac{11}{4}$, and $\{\frac{1}{4}, \frac{1}{2}, \frac{1}{4}\}.\{0, 0, 4\} = 1$ which result in the coefficients of the sixth degree polynomial. The thing to note here is that inversion is associated with convolution!

Note the end effects at both ends because we do not have the samples $x(-2)$ and $x(-1)$ for $y(0)$ and $y(1)$ on the left and $x(5)$ and $x(6)$ for $y(5)$ and $y(6)$ on the right.

The above remarks apply when the input is the sinusoidal signal x = {0, 1, 0, -1, 0} with frequency $\omega = \frac{\pi}{2}$. We observe that it, too, is preserved with a unit delay between input and output; additionally, the amplitude of the output has changed to $\frac{1}{2}$.

```
x = {0, 0, 0, 1, 0, -1, 0, 0, 0}; Table[yLP[n], {n, 3, 9}]
```

$$\left\{0, \frac{1}{4}, \frac{1}{2}, 0, -\frac{1}{2}, -\frac{1}{4}, 0\right\}$$

or

```
ListConvolve[{1/4, 1/2, 1/4}, {0, 1, 0, -1, 0}, {1, -1}, 0]
```

```
{0, 1/4, 1/2, 0, -1/2, -1/4, 0}
```

```
Clear[x];
```

To see that the amplitude is reduced by a half let us look at the frequency response function $H(\omega)$. For the filter under consideration we have

$$H(\omega) = \frac{1}{4} + \frac{1}{2} e^{-i\omega} + \frac{1}{4} e^{-2i\omega}.$$

Evaluating at $\omega = \frac{\pi}{2}$ we obtain $H = \frac{-1}{2} i$, from which we have that the magnitude of the frequency is $\frac{1}{2}$. (Recall that from Fourier transforms we have that convolutions in the time domain correspond to multiplications in the frequency domain.)

■ Wavelet Transforms

As we said, lowpass filters greatly reduce the high frequency components; when these components represent noise in signals, this reduction is the right thing to do. However, if we want to *reconstruct* the signal, then we use two filters, a highpass as well as a lowpass filter. This generates a *filter bank* structure, which leads to a *Discrete Wavelet Transform* (DWT) – in itself a *lossless* transform. Its inverse, (IDWT), is another transform of the same type – two filters that are fast to compute. Between the DWT and the IDWT compression and transmission of the signal can take place, and this is the key to more and more applications.

Wavelet transforms are associated with *multiresolution into different scales.* The simplest change of scale comes from *downsampling* a signal, to wit keeping only its even-numbered components $y(2n)$. This operation is denoted by the symol ↓2:

$$(\downarrow 2)\begin{pmatrix} y(1) \\ y(2) \\ y(3) \\ y(4) \end{pmatrix} = \begin{pmatrix} y(2) \\ y(4) \end{pmatrix}.$$

In *Mathematica* this is easily programmed:

```
downSample[x_List] :=
 Module[{i = 1, y = {}}, While[i <= Length[x],
   If[Mod[i, 2] == 0, AppendTo[y, x[[i]]]; i = ++i, i = ++i]]; y]
```

```
downSample[{2, 3, 4, 5}]
```

```
{3, 5}
```

Obviously, with downsampling information is lost. However, we can *double* the length of the input signal by using two filters, and then *halve* each output to obtain a lossless transform. The input is at one time scale and the two half-length outputs are at another scale (an octave lower).

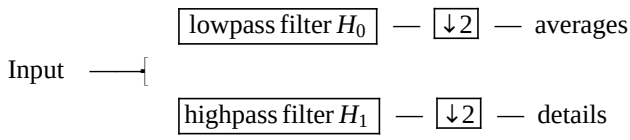


Figure 8a: Filter Bank Structure — Averages and details in the Discrete Wavelet Transform

If the input signal is of length L, the output of both H_0 and H_1 is originally of length L (provided we deal with the samples at the ends). The redundancy from $2L$ outputs is removed by (↓2), since after downsampling the length of each output becomes $\frac{L}{2}$.

```
(*  !for the next diagram expand to full screen !*)
```

The main idea of wavelet transforms is to *repeat* the filter bank structure. The *lowpass output* in the Filter Bank Structure above becomes the *input* to a second filter bank. The computation is cut in half because this second input is half the original length. Typically, this process is repeated four or five times.

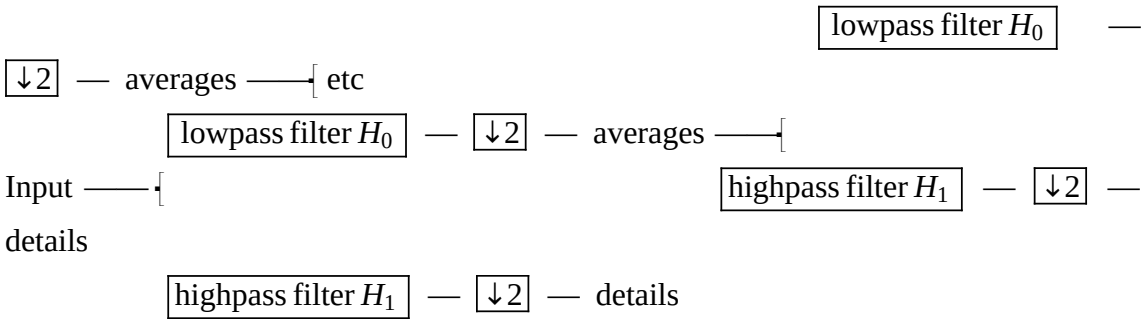


Figure 8b: Wavelet Transform Structure

Notice that the basic wavelet transform structure assumes that the downsampled coefficients of the highpass filter are small enough and not worth the additional effort to run them through another filter bank structure. They are candidates for compression.

5.3 The Haar Wavelet Transform and Its *Mathematica* Implementation

Below we choose the simplest of the wavelet transforms, named after Alfred Haar. It naturally combines the simplest examples of lowpass and highpass filters that we have previously seen (to wit, the moving average filter and the moving difference filter) into the most basic example of a filter bank.

■ Analysis

We denote the two filters by H_0 (lowpass) and H_1 (highpass). For convenience we reverse the sign of H_1 and we have:

$$\begin{array}{lll} y_0 = H_0\, x & \text{is the averaging filter} & y_0(n) = \frac{1}{2}\, x(n-1) + \frac{1}{2}\, x(n) \\ y_1 = H_1\, x & \text{is the differencing filter} & y_1(n) = \frac{1}{2}\, x(n-1) - \frac{1}{2}\, x(n). \end{array}$$

or as we have already defined in *Mathematica*:

```
yLP[n_] := 1/2 x[[n - 1]] + 1/2 x[[n]]
```

```
yHP[n_] := 1/2 x[[n - 1]] - 1/2 x[[n]]
```

Let the signal be the vector $x = \{7, 5, 6, 2\}$. To take care of the left end, we prepend 0 to the vector x.

```
Clear[x]; x = {0, 7, 5, 6, 2};
```

We next feed this vector to the "first stage" of the Wavelet Transform Structure to obtain the following outputs. From the lowpass filter we obtain the averages:

```
y0 = Table[yLP[n], {n, 2, 5}]
```

$$\left\{\frac{7}{2},\ 6,\ \frac{11}{2},\ 4\right\}$$

whereas from the highpass filter we obtain the differences:

```
y1 = Table[yHP[n], {n, 2, 5}]
```

$$\left\{-\frac{7}{2},\ 1,\ -\frac{1}{2},\ 2\right\}$$

Notice that the differences tend to be smaller than the averages. For smooth inputs this is even more true.

At this point we have eight coefficients in the vectors y_0 and y_1. They are redundant, since they came from only four samples of the vector *x*. We now downsample and keep the even-numbered components.

```
{y0[[2]], y0[[4]]} = downSample[y0]
```

```
{6, 4}
```

```
{y1[[2]], y1[[4]]} = downSample[y1]
```

```
{1, 2}
```

This is the first step in our analysis. We have found the four "first-level" wavelet coefficients of the signal, or we have separated the signal into wavelets. As we will see in the synthesis, the inverse transform uses these coefficients (that is, adds up wavelets times coefficients) to reconstruct the signal vector *x*.

The thing to note here is that, appart from downsampling, the coefficients {6, 4} can be obtained in yet another way: namely, by splitting the signal input vector into pairs *x* = {{7, 5},{6, 2}} and taking the average of each pair; i.e. 6 = $\frac{7+5}{2}$ and 4 = $\frac{6+2}{2}$. Similarly the coefficients {1, 2} can be obtained from these pairs by subtraction; to wit, 1 = $\frac{7-5}{2}$ and 2 = $\frac{6-2}{2}$. These observations are fully exploited in programming the Haar transform.

Now the downsampled output {6, 4} of the lowpasss filter becomes our new input vector *x* into the "second stage" of the Wavelet Transform Structure. Proceeding in a similar fashion we obtain the following outputs.

```
x = {0, 6, 4};
```

```
z0 = Table[yLP[n], {n, 2, 3}]
```

```
{3, 5}
```

```
z1 = Table[yHP[n], {n, 2, 3}]
```

```
{-3, 1}
```

```
Clear[x];
```

The "second-level" coefficients of the signal are

```
z0[[2]] = downSample[z0]
```

```
{5}
```

```
z1[[2]] = downSample[z1]
```

```
{1}
```

Just as in the first stage, we have $5 = \frac{6+4}{2}$ and $1 = \frac{6-4}{2}$. This completes the iteration of the Haar analysis bank, which is graphically represented below:

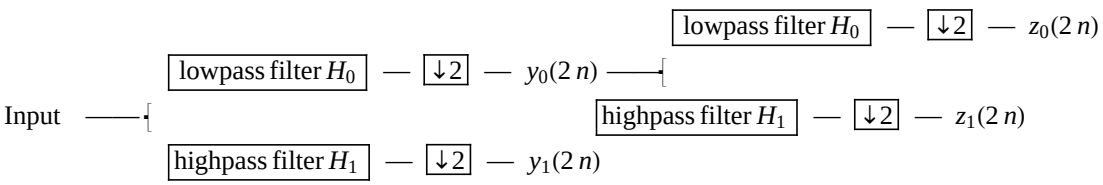


Figure 9: Haar analysis bank.

Summarizing: For the input vector $x = \{7, 5, 6, 2\}$ the wavelet coefficients are $\{5, 1, 1, 2\} = \{z_0(2\,n), z_1(2\,n), y_1(2\,n)\}$. To wit, $z_0(2\,n) = 5$ is the overall average and the rest are the details, $z_1(2\,n) = 1$, and $y_1(2\,n) = \{1, 2\}$.

■ Synthesis

Notice that in the analysis above we obtained two outputs from two successive inputs, to wit:

$$y_0(2\,n) = \tfrac{1}{2}\,x(2\,n-1) + \tfrac{1}{2}\,x(2\,n), \text{ and}$$
$$y_1(2\,n) = \tfrac{1}{2}\,x(2\,n-1) - \tfrac{1}{2}\,x(2\,n).$$

It is now easy to recover $x(2\,n-1)$ and $x(2\,n)$ by the sums and differences of the outputs:

$$x(2\,n-1) = y_0(2\,n) + y_1(2\,n), \text{ and}$$
$$x(2\,n) = y_0(2\,n) - y_1(2\,n).$$

Hence, in the synthesis the same operations are used as in the analysis. The sequence of inverse operations is *upsampling* (↑2) followed by filtering:

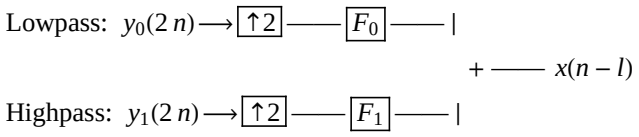


Figure 10: A stage of the inverse wavelet transform

With upsampling we double the length of a list by inserting zeros at its even positions. From the above equations we conclude that the coefficients of the lowpass filter F_0 are {1, 1} and those of the highpass filter F_1 are {1, -1}.

```
xLP[n_] := y[[n]] + y[[n - 1]]
```

```
xHP[n_] := y[[n]] - y[[n - 1]]
```

Moreover, the program to upsample is straightforward:

```
upSample[x_List] :=
 Module[{i = 1, y = x}, While[i <= 2 Length[x],
   If[Mod[i, 2] == 0, y = Insert[y, 0, i]; i = ++i, i = ++i]]; y]
```

```
upSample[{5}]
```

```
{5, 0}
```

To recover (with a unit delay) the input vector x = {7, 5, 6, 2} from the wavelet coefficients {5, 1, 1, 2} we proceed as follows: In the first stage the inputs are the coefficients {5} and {1}, which were computed last in the analysis process; we upsample them, take care of the left end of each vector, and run them through the lowpass and highpass filters F_0 and F_1 respectively. The output vectors are: *lp*, obtained from {5}, and *hp*, obtained from {1}. Then, the inputs to the second stage are: *lp*+*hp* and the last two of the wavelet coefficients {5, 1, 1, 2}. They are subjected to the same treatment.

The first stage going back is

```
Clear[y]; y = upSample[{5}]; PrependTo[y, 0];
```

```
lp = Table[xLP[n], {n, 2, 3}]
```

```
{5, 5}
```

```
Clear[y]; y = upSample[{1}]; PrependTo[y, 0];
```

```
hp = Table[xHP[n], {n, 2, 3}]
```

```
{1, -1}
```

```
lp + hp
```

```
{6, 4}
```

Here also, appart from upsampling, the coefficients {6, 4} can be recovered from the first two of the wavelet coefficients {5, 1, 1, 2} in yet another way: namely, by adding 1 to 5 and by subtracting 1 from 5. In the second stage we recover the input vector x = {7, 5, 6, 2} from {6, 4} and the last two of the wavelet coefficients {5, 1, 1, 2} by adding 1 to 6 and by subtracting 1 from 6 and by adding 2 to 4 and by subtracting 2 from 4. These observations are again fully exploited in programming the Haar transform.

For the second stage we have:

```
Clear[y]; y = upSample[{6, 4}]; PrependTo[y, 0];
```

```
lp = Table[xLP[n], {n, 2, 5}]
```

```
{6, 6, 4, 4}
```

```
Clear[y] ; y = upSample[{1, 2}] ; PrependTo[y, 0] ;
```

```
hp = Table[xHP[n], {n, 2, 5}]
```

```
{1, -1, 2, -2}
```

```
lp + hp
```

```
{7, 5, 6, 2}
```

■ *Mathematica* Implementation of the Haar Transform

In this implementation we do not use down/up-sampling. Instead we take advantage of the speedups discussed in the analysis/synthesis and compute averages and differences.

```
discreteWaveletTransform::usage =
  "This routine has at most 2 arguments.  The first
    argument is a list of ANY length; the second argument
    is an OPTIONAL flag with default value 0. Called with
    one argument it does NOT print the results of the
    intermediate steps; to print them set the flag to 1.
  Regarding the first argument (list), its length should be 2^i,
    i>0. If this is not the case, zeros are appended to fix the
    situation. The  averaging-differencing algorithm (LowPass/
    HighPass Filtering) is performed on the (new) list. After i (
    or i+1)iterations the output is a list of the same length.";


discreteWaveletTransform[list_List, flag_Integer: 0] :=
 Module[{tempList = list, list2,
   tempInteger = i = Log[2, Length[list]] // N // IntegerPart,
   pairedRow, averages, differences, fl = flag, nextRow, temp},
  If[i == 0, Print["Adjust the list; its length should be 2^i, i>0"];
   Return[Null]];
  If[2^i ≠ Length[list], list2 =
    Append[tempList, Table[0, {i, 1, 2^(i+1) - Length[list]}]] // Flatten;
   tempInteger = i = i + 1; tempList = list2];
  nextRow = {}; If[fl == 1, Print[tempList // N]];
  While[tempInteger ≠ 0,
   pairedRow = pairedList[tempList];
   averages = Apply[aver, pairedRow, 1];
   differences = diff[pairedRow, averages];
   nextRow = Prepend[nextRow, {averages, differences}] // Flatten;
   If[fl == 1, Print[nextRow]];
   tempInteger = tempInteger - 1; temp = 2^tempInteger;
   tempList = Drop[nextRow, -2^i + temp];
   nextRow = Drop[nextRow, temp];
  ]; If[fl == 1, Prepend[nextRow, First[tempList]];,
   Prepend[nextRow, First[tempList]]]
 ]
```

```
(*          2nd function                *)

discreteInverseWaveletTransform::usage =
  "This routine has at most 2 arguments.  The first
    argument is a list of ANY length; the second argument
    is an OPTIONAL flag with default value 0. Called with
    one argument it does NOT print the results of the
    intermediate steps; to print them set the flag to 1.
  Regarding the first argument (list), its length should
    be 2^i , i>0. If this is not the case, zeros are
    appended to fix the situation. The  INVERSE averaging-
    differencing algorithm (LowPass/HighPass Filtering)
    is performed on the (new) list. After i (or i+1)
    iterations the output is a list of the same length.";

discreteInverseWaveletTransform[list_List, flag_Integer: 0] :=
 Module[{tempList = list, tempList2 = list, list2, av, di, fl = flag,
   i = Log[2, Length[list]] // N // IntegerPart, j, m = 1, temp = 1},
  If[i == 0, Print["Adjust the list; its length should be 2^i, i>0"];
   Return[Null]];
  If[2^i ≠ Length[list], list2 =
    Append[tempList, Table[0, {i, 1, 2^(i+1) - Length[list]}]] // Flatten;
   i = i + 1; tempList = tempList2 = list2];
  m = 1;
  While[m ≤ i, j = 0; k = 1;
   While[k <= temp,  av = tempList[[k]]; di = tempList[[k + temp]];
    tempList2 = ReplacePart[tempList2, av + di, k + j];
    tempList2 = ReplacePart[tempList2, av - di, k + j + 1];
    j = j + 1; k = k + 1
   ]; If[fl == 1,  Print[tempList]]; temp = 2 * temp;
   tempList = tempList2; m = m + 1
  ]; tempList
 ]


(*    3d function; computes the average value of 2 numbers     *)

aver[a_, b_] := (a + b) / 2.


(*    4th function; splits a list in pairs    *)

pairedList[list_] := Partition[list, 2]
```

```
(*          5 th function                    *)
diff::usage =
  "The inputs are two lists of equal length--- 2^k, for some k.
    The first list is a list of pairs, whereas the second list
    is a simple list (of averages). The output is a simple list
    of the same length as the input lists; each element of the
    output list is the difference of the first element of each
    pair (of the first list) and the corresponding element
    of the second list. For example, if the input lists are
    {{a,b},{c,d}} and {e,f} the output is {a-e,c-f}. ";

diff[list1_, list2_] :=
 Fold[Append[#1, #2[[1, 1]] - #2[[2]]] &, {}, Thread[{list1, list2}]]
```

Example.

```
discreteWaveletTransform[{7, 5, 6, 2}, 1]
```

```
{7., 5., 6., 2.}
```

```
{6., 4., 1., 2.}
```

```
{5., 1., 1., 2.}
```

5 is the general average and the coefficients 1, 1, 2 are the details. This transformation is reversible:

```
discreteInverseWaveletTransform[%, 1]
```

```
{5., 1., 1., 2.}
```

```
{6., 4., 1., 2.}
```

```
{7., 5., 6., 2.}
```

5.4 Image Compression with the Haar Transform

We saw that in adoptive plotting of functions few samples are taken when there is little or no variation. The same idea will be used in image processing to take advantage of regions of little or no color variation (as in example in the color of a shirt, pants, hair etc). The Haar wavelet transform with the averaging and differencing scheme is ideally suited for this purpose and results in a method of image compression (storing much less information than would otherwise be necessary). At first though we examine some of the advantages of these transforms.

■ Wavelet Transform Benefits

Let us look at the advantages of using the Haar wavelet transform.

```
data = {74, 58, 26, 42, 66, 66, 58, 34};
discreteWaveletTransform[data]
```

```
{53., -3., 16., 10., 8., -8., 0., 12.}
```

53 is the general average and the coefficients -3, 16, 10, 8, -8, 0, 12 are the details. We have transformed a set of eight numbers into another set of eight numbers and, moreover, this transformation is reversible, as we have seen. In doing this transformation our advantage is that we can "play around" with the detail coefficients. By taking advantage of regions of little variation, we can change the set of coefficient details and use this new set (with the inverse wavelet transform) to approximate the original one. This approximation should be very close to the original.

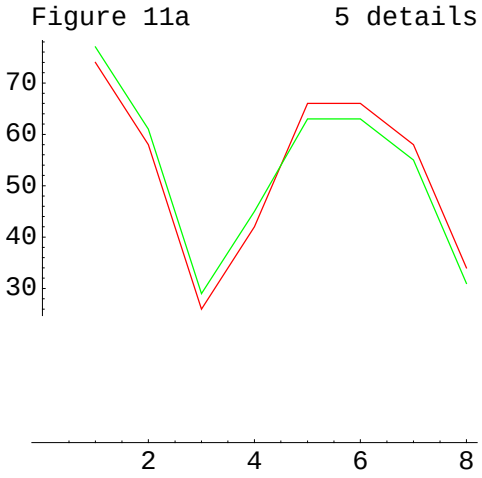
In the above example, one detail coefficient is zero indicating a region of little variation (due to the two adjacent numbers 66). The next smallest detail coefficient is -3. We change -3 to 0 and applying the inverse wavelet transform we obtain:

```
dataApproximated =
 discreteInverseWaveletTransform[{53, 0, 16, 10, 8, -8, 0, 12}]
```

```
{77, 61, 29, 45, 63, 63, 55, 31}
```

The plot of the original (red) against the approximated (green) set of numbers is seen in Figure 11a.

```
plot1 = ListPlot[data, PlotJoined → True, AxesOrigin → {0, 0},
   PlotStyle → RGBColor[1, 0, 0], DisplayFunction :→ Identity];
plot2 = ListPlot[dataApproximated, PlotJoined → True, AxesOrigin →
    {0, 0}, PlotStyle → RGBColor[0, 1, 0], DisplayFunction :→ Identity];
Show[plot1, plot2, DisplayFunction :→ $DisplayFunction,
  PlotLabel → "Figure 11a          5 details"];
```

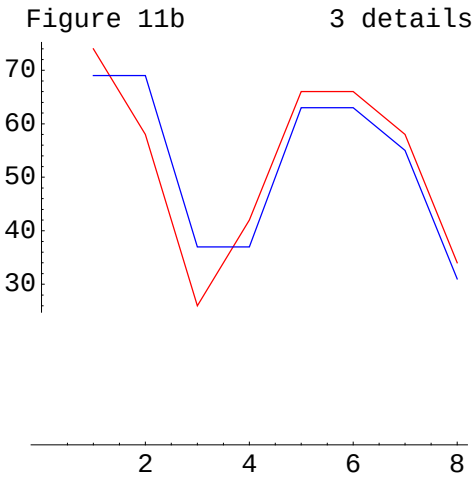


A very good approximation. Changing additionally -8 *and* 8 to zero we obtain Figure 11b in which the approximated set is in blue:

```
dataApproximatedAgain =
 discreteInverseWaveletTransform[{53, 0, 16, 10, 0, 0, 0, 12}]
```

```
{69, 69, 37, 37, 63, 63, 55, 31}
```

```
plot3 = ListPlot[dataApproximatedAgain,
   PlotJoined → True, AxesOrigin → {0, 0},
   PlotStyle → RGBColor[0, 0, 1], DisplayFunction :→ Identity];
Show[plot1, plot3, DisplayFunction :→ $DisplayFunction,
  PlotLabel → "Figure 11b          3 details"];
```



Considering that this is based only on 3 detail coefficients, the approximation is surprisingly good!

Overall, this scheme can be applied to a number string of any length. We can always pad at the end with zeros until the length of the string is a power of two. As we are going to see next, the full potential of this transformation comes when we apply it to strings of big length.

■ Plotting with Wavelet Transforms — Lossless and Lossy Compression

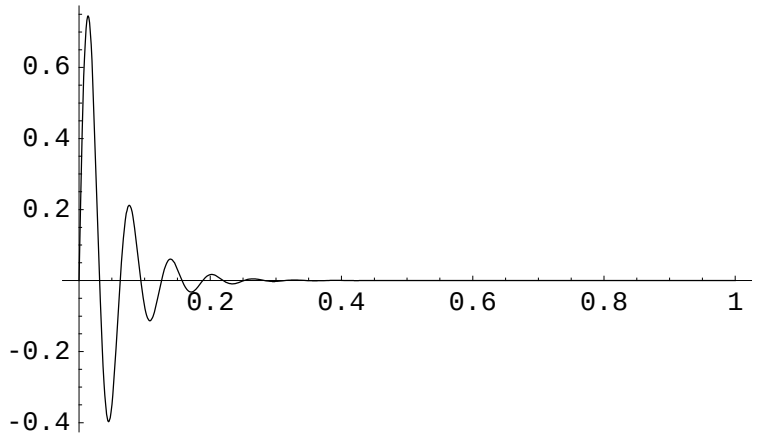
The compression process works as follows: We start with a data string of any length and using the wavelet transform we obtain another string of the same length. We can now use the inverse transform algorithm in two ways: (a) We can use all the detail coefficients to obtain the original list in which case the transform is called *lossless compression.* That is, we lost nothing. (b) We can define a *threshold* value ϵ, replace by zero all the detail coefficients with magnitude $\leq \epsilon$ and, based on the altered version of the transformed string, obtain an approximation of the original string. This is referred to as *lossy compression.* The surprising thing is that we can zero a sizable proportion of the detail coefficients and still get a decent approximation to the original string.

Let us apply the above in the function $e^{-10\,x}\,\text{Sin}[100\,x]$ which we are going to plot with uniform sampling in the interval [0, 1]. We picked this particular function because it has a great region of no variation (low activity) as can be seen from Figure 12.

```
f[x_] := E^(-20 x) Sin[100 x];
```

```
Plot[f[x], {x, 0, 1}, PlotRange → All, PlotLabel → "Figure 12"];
```

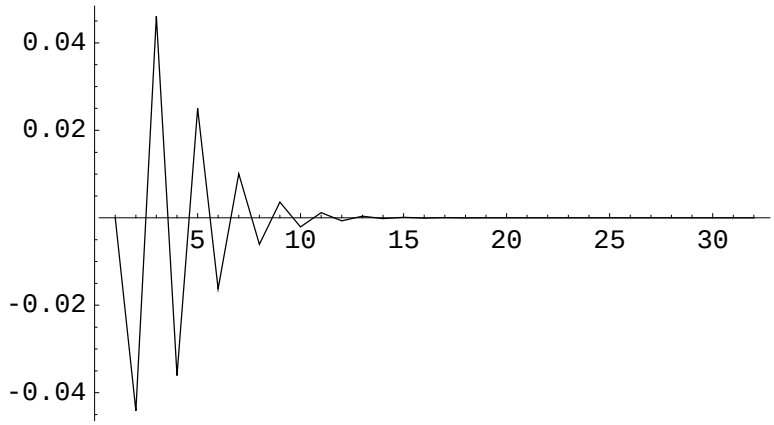
Figure 12



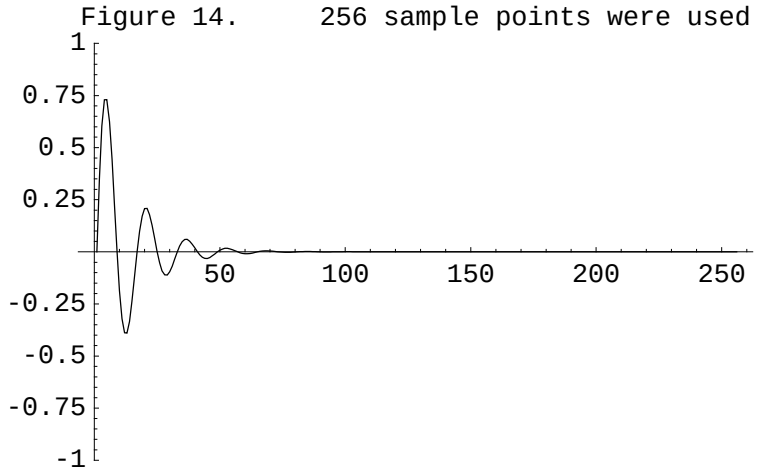
We now plot e^{-10x} Sin[100 x] using 32 and 255 uniformly sampled points in the interval [0, 1].

```
xPoints5 = Table[x, {x, 0, 1, 1 / 31}];
yPoints5 = Map[f, xPoints5];
ListPlot[yPoints5, PlotJoined → True,
  PlotRange → All, AxesOrigin → {0, 0},
  PlotLabel → "Figure 13.  32 sample points were used"];
```

Figure 13. 32 sample points were used



```
xPoints6 = Table[x, {x, 0, 1, 1 / 255}];
yPoints6 = Map[f, xPoints6];
plot15 = ListPlot[yPoints6,
   PlotJoined → True, AxesOrigin → {0, 0}, PlotRange → {-1, 1},
   PlotLabel → "Figure 14.     256 sample points were used"];
```



We see that the approximation with 256 sample points is excellent, whereas the one with 32 sample points is quite poor. In both cases though half of the points plotted were essentially wasted!

Consider the string of 256 elements, *yPoints6,* used to plot this function. Their values range from -0.39 to 0.73.

```
{Length[yPoints6], Max[yPoints6], Min[ yPoints6]} // N
```

```
{256., 0.730718, -0.39016}
```

After eight cycles of averaging and differencing we get a transformed string of the same length, *yPoints6WT*, with values ranging from -0.245 to 0.355.

```
yPoints6WT = discreteWaveletTransform[yPoints6];
{Length[yPoints6WT], Max[yPoints6WT], Min[ yPoints6WT]}
```

```
{256, 0.355324, -0.245031}
```

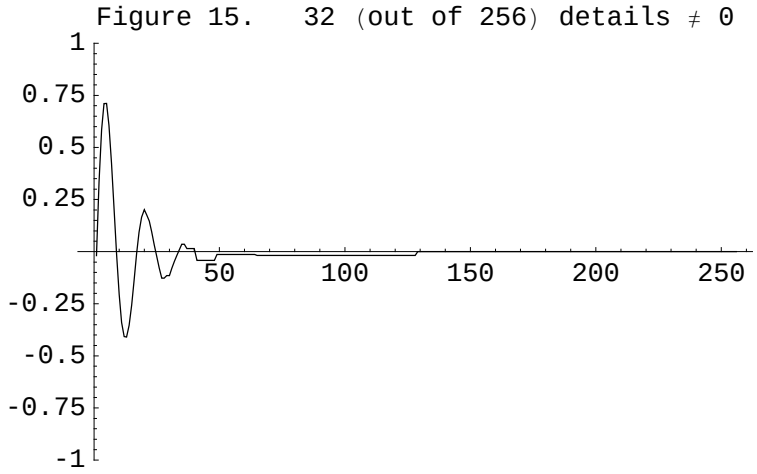
We next pick a threshold value of $\epsilon = 0.014$ and zero all values of *yPoints6WT* (the detail coefficients) $\leq \epsilon$. This way we get an altered transformed list, *yPoints6WTSparse*, with 32 nonzero entries.

```
yPoints6WTSparse =
  Fold[If[Abs[#2] <= 0.014, Append[#1, 0], Append[#1, #2]] &,
   {}, yPoints6WT];
Select[yPoints6WTSparse, # != 0 &] // Length
```

```
32
```

From this sparse list, *yPoints6WTSparse*, we obtain an approximation, *yPoints6WTApprox*, of the original list and we plot it in Figure 15.

```
yPoints6WTApprox =
  discreteInverseWaveletTransform[yPoints6WTSparse];
ListPlot[yPoints6WTApprox, PlotJoined → True,
  AxesOrigin → {0, 0}, PlotRange → {-1, 1},
  PlotLabel → "Figure 15.   32 (out of 256) details ≠ 0"];
```

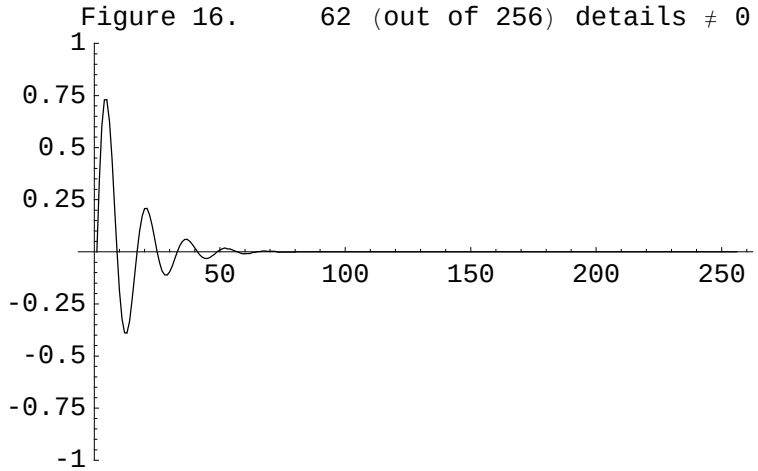


Despite its limitations this does a better job of approximating the actual graph of e^{-10x} Sin[100 x], than the plot with 32 uniformly sampled points in Figure 13. Finally, Figure 16 shows the even better approximation of E^{-10x} Sin[100 x] obtained by using the threshold value of $\epsilon = 0.0014$, in which case the altered transformed list, *yPoints6WTSparse*, has 62 nonzero entries.

```
yPoints6WTSparse =
  Fold[If[Abs[#2] <= 0.0014, Append[#1, 0], Append[#1, #2]] &,
   {}, yPoints6WT];
Select[yPoints6WTSparse, # != 0 &] // Length
```

```
62
```

```
yPoints6WTApprox =
  discreteInverseWaveletTransform[yPoints6WTSparse];
ListPlot[yPoints6WTApprox, PlotJoined → True,
  AxesOrigin → {0, 0}, PlotRange → {-1, 1},
  PlotLabel → "Figure 16.     62 (out of 256) details ≠ 0"];
```



It should be noted that the threshold values are very low compared to the range of values. Yet, despite this fact, we obtain good results using few detail coefficients. The reason for this behavior is that there is no variation in more than half of the interval of interest. Using 32 detail coefficients, out of 256, gives a compression ration of 8:1, whereas using 62 gives a compression ratio of 4.1:1.

Also, note that Figures 15 and 16 were generated using 32 and 62 nonzero detail coefficients respectively, in sparse strings of 256 elements; however, the plots themselves used all 256 mostly nonzero numbers obtained from those strings by applying the *inverse* Haar wavelet transform.

Lossy compression comes into play once we realize that it takes less space to store *sparse* strings of length 256 — with 32 or 62 nonzero entries — than arbitrary strings of length 256. Using *Mathematica*'s special function SparseArray[], we see that storing the 256-element list *yPoints6WT* requires about 5.5K bytyes of memory:

```
(*   Memory used by a non-sparse string of length 256   *)

a = MemoryInUse[]; yPoints6WT = discreteWaveletTransform[yPoints6];
b = MemoryInUse[]; b - a
```

```
5472
```

whereas, for a sparse array about half that memory is needed for storing. Note that, to be able to use the SparseArray[] function of *Mathematica*, we first take the list *yPoints6WTSparse*, generated above with 32 or 62 nonzero entries, compute the positions where these nonzero entries are and turn the position lists into rules of the form "position → value". The list of these rules, *positionAndValueRules*, is the input to the function SparseArray[].

```
yPoints6WTSparse;
res = {};
Map[If[# ≠ 0, AppendTo[res, {Position[yPoints6WTSparse, #], #}]] &,
 yPoints6WTSparse];
positions = Map[{Flatten[First[#]], Last[#]} &, res];
positionAndValueRules = Map[Apply[Rule, #] &, positions];
```

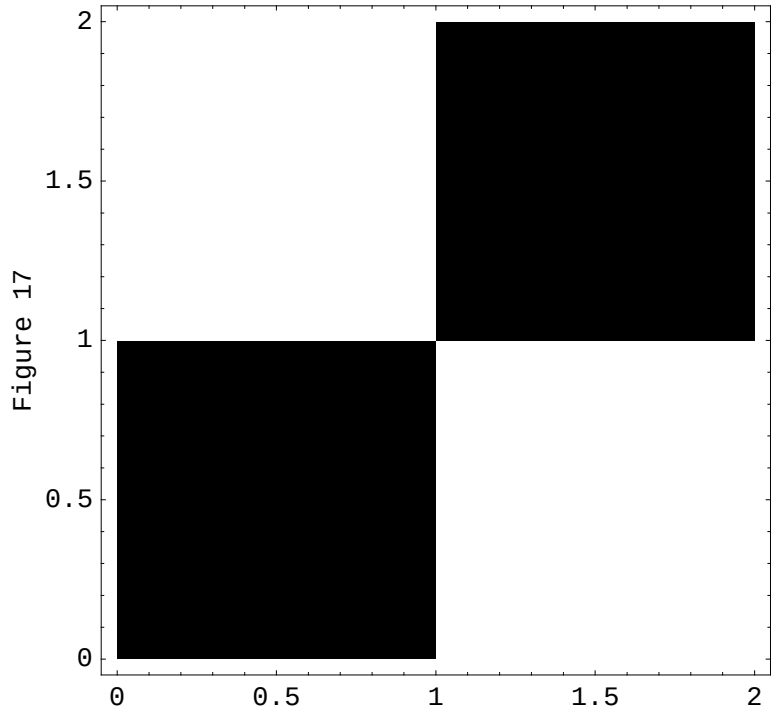
```
(*   Memory used by a sparse string of length 256   *)

a = MemoryInUse[];
SparseArray[positionAndValueRules];
b = MemoryInUse[]; b - a
```

```
2384
```

■ Primitive Image Compression

Suppose we have the image shown in Figure 17. Please note that 0 corresponds to black and 1 to white; moreover, the rows in the Raster function are printed in reverse order.

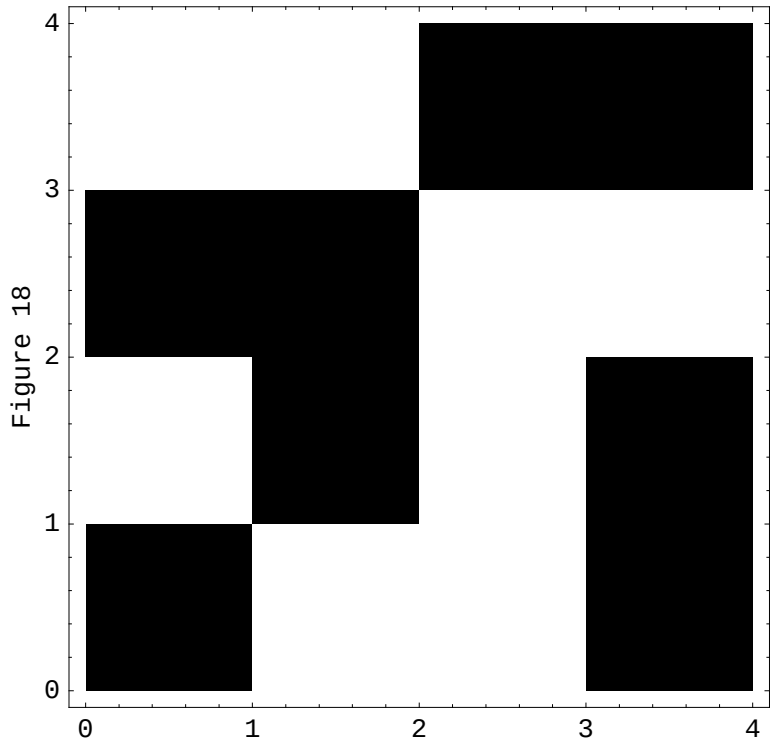
```
Show[Graphics[Raster[{{0, 1}, {1, 0}}],
   Frame → True, FrameLabel → "Figure 17", AspectRatio → 1]];
```



The above image can be considered as a $2^i \times 2^i$ pixel image which has 4 pixels if $i = 1$. What information is needed to store it? If we assume that we have black unless specified otherwise, then we need only say where there is white. In the case $i = 1$, we obviously need 2 pieces of information, namely that pixels {1, 1} and {2, 2} are white. We will show the quite interesting fact that, for *any* value of $i > 1$ we need *only* 2 pieces of information.

Next consider the more complex image shown in Figure 18.

```
Show[Graphics[
   Raster[{{0, 1, 1, 0}, {1, 0, 1, 0}, {0, 0, 1, 1}, {1, 1, 0, 0}}],
   Frame → True, FrameLabel → "Figure 18", AspectRatio → 1]];
```



Again, we can treat it as $4^i \times 4^i$ pixel image which has 16 pixels if $i = 1$. Arguing as above, in the case $i = 1$ we need 8 pieces of information. However, we can do better if we use the fact that several of the white blocks are adjacent to each other. We will show that for any value of i (i.e. no matter what the resolution) we only need 5 pieces of information.

We assume that both images of Figures 17 and 18 have resolution 256×256; that is, each image is represented by a 256×256 matrix with entries 0 or 1. Using the technique of lossy compression we will show that to store the first image we need 2 pieces of information whereas for the second we need 5. In each case we apply the Haar transform on each row separately, and then do the same on each column of the resulting matrix. The final result is a new 256×256 matrix with an overall average at the top left hand corner and a lot of detail coefficients. Regions of little variation in the original image manifest themselves as numerous small or zero elements in the transformed matrix.

We start with the image of Figure 17. Each one of the first 128 rows is made up of 128 ones (1→white) followed by 128 zeros (0→black).

```
matrixRowF = Join[Table[1, {i, 1, 128}], Table[0, {i, 1, 128}]];
```

When we apply the Haar wavelet transform to each one of these 128 rows the result is a row with all, but the first two elements, zero.

```
discreteWaveletTransform[matrixRowF];
```

Hence after the transformation the *first* 128 rows of the matrix are identical; in each row, the first two elements are 0.5, 0.5 and the rest 254 are 0.

Likewise, after the transformation, the *last* 128 rows of the matrix are identical; in each row, the first two elements are 0.5, -0.5 and the rest 254 are 0.

```
matrixRowL = Join[Table[0, {i, 1, 128}], Table[1, {i, 1, 128}]];
discreteWaveletTransform[matrixRowL];
```

At this point the columns of the 256×256 matrix are all zero except for the first two. The first column has 256 identical entries of 0.5 whereas the second column consists of 128 rows of 0.5 followed by 128 rows of -0.5. Transforming the first column we obtain another one with all entries zero, except for the *first* entry which is 0.5; likewise, from the second column we obtain another one with all entries zero, except for the *second* entry which is 0.5.

```
matrixColumn1 = Table[0.5, {i, 1, 256}];
discreteWaveletTransform[matrixColumn1];
```

```
matrixColumn2 =
  Join[Table[0.5, {i, 1, 128}], Table[-0.5, {i, 1, 128}]];
discreteWaveletTransform[matrixColumn2];
```

Since our wavelet transform is invertible the image of Figure 17 can be stored in a 2×2 matrix form with only two nonzero pieces of information

$$\begin{pmatrix} 0.5 & 0 \\ 0 & 0.5 \end{pmatrix}.$$

The image compression process can be summarized as follows: (a) form the original matrix, (b) transform all the rows to obtain the transformed matrix, (c) transform the columns of the transformed matrix and (d) print the first few rows and columns. (To do (c) in *Mathematica* we have to transpose the

transformed matrix so that the columns become rows, apply the wavelet transform on the rows and transpose it again to obtain the final matrix.)

```
originalMatrix = Join[
   Table[matrixRowF, {i, 1, 128}], Table[matrixRowL, {i, 1, 128}]];

transformedMatrix = Map[discreteWaveletTransform, originalMatrix];

transformedTransposedMatrix = Transpose[transformedMatrix];

finalMatrix = Map[discreteWaveletTransform,
    transformedTransposedMatrix] // Transpose;

Table[finalMatrix[[i, j]], {i, 1, 7}, {j, 1, 7}] // MatrixForm
```

$$\begin{pmatrix} 0.5 & 0. & 0. & 0. & 0. & 0. & 0. \\ 0. & 0.5 & 0. & 0. & 0. & 0. & 0. \\ 0. & 0. & 0. & 0 & 0. & 0. & 0 \\ 0. & 0. & 0 & 0. & 0 & 0 & 0. \\ 0. & 0. & 0. & 0 & 0. & 0. & 0 \\ 0. & 0. & 0. & 0 & 0. & 0. & 0 \\ 0. & 0. & 0 & 0. & 0 & 0 & 0. \end{pmatrix}$$

Applying the same process to the image of Figure 18 we see that it can be stored in a 4×4 matrix using 5 nonzero pieces of information

$$\begin{pmatrix} 0.5 & 0 & 0 & 0.25 \\ 0 & 0 & 0 & -0.25 \\ 0 & 0.5 & 0 & 0 \\ 0 & 0 & 0.5 & 0 \end{pmatrix}$$

- **Compression of the Image of Figure 18**

The reader is encouraged to apply the same process to the image of Figure 18. We work on blocks of 64 rows; for the 1st block of rows we have:

```
matrixRowFQ = Join[Table[1, {i, 1, 128}], Table[0, {i, 1, 128}]];
```

For the second block we have:

```
matrixRowSQ = Join[Table[0, {i, 1, 128}], Table[1, {i, 1, 128}]];
```

For the third block we have:

```
matrixRowTQ = Join[Table[1, {i, 1, 64}],
   Table[0, {i, 1, 64}], Table[1, {i, 1, 64}], Table[0, {i, 1, 64}]];
```

For the last block we have:

```
matrixRowLQ = Join[Table[0, {i, 1, 64}],
   Table[1, {i, 1, 128}], Table[0, {i, 1, 64}]];
```

The original matrix then is:

```
originalMatrix =
  Join[Table[matrixRowFQ, {i, 1, 64}], Table[matrixRowSQ, {i, 1, 64}],
   Table[matrixRowTQ, {i, 1, 64}], Table[matrixRowLQ, {i, 1, 64}]];

transformedMatrix = Map[discreteWaveletTransform, originalMatrix];

transformedTransposedMatrix = Transpose[transformedMatrix];

finalMatrix = Map[discreteWaveletTransform,
    transformedTransposedMatrix] // Transpose;

Table[finalMatrix[[i, j]], {i, 1, 7}, {j, 1, 7}] // MatrixForm
```

$$\begin{pmatrix} 0.5 & 0. & 0. & 0.25 & 0. & 0. & 0. \\ 0. & 0. & 0. & -0.25 & 0. & 0. & 0. \\ 0. & 0.5 & 0. & 0. & 0. & 0. & 0. \\ 0. & 0. & 0.5 & 0. & 0. & 0. & 0. \\ 0. & 0. & 0. & 0 & 0. & 0. & 0 \\ 0. & 0. & 0 & 0. & 0 & 0 & 0. \\ 0. & 0. & 0. & 0. & 0. & 0 & 0. \end{pmatrix}$$

■ Compression of arbitrary images (Pedro!)

Παράρτημα Δ6. Ψηφιοποίηση εικόνας. Εντροπία!!!

We add parenthetically that the digital information about Pedro — that is, the matrix data1 — can be stored in the seperate file pedro.data using the function Write

```
Write[ToFileName[{"AddOns", "Applications"}, "pedro.data"], data1];
```

from which it can be read into the variable pedroDigitalData with the function ReadList

```
pedroDigitalData =
  ReadList[ToFileName[{"AddOns", "Applications"}, "pedro.data"],
    Expression] // Last;
```

Once we have the picture in digital form, all the fun begins. We run it through the wavelet transform to compress it; rows are transformed first and then columns as was done for the case of the primitive images.

```
originalMatrix = data1;
transformedMatrix = Map[discreteWaveletTransform, originalMatrix];
transformedTransposedMatrix = Transpose[transformedMatrix];
finalMatrix = Map[discreteWaveletTransform,
    transformedTransposedMatrix] // Transpose;
```

Compression can be achieved by zeroing (for example) the detail coefficients which are smaller than 3 in absolute value. This way we obtain a matrix with only 6629 nonzero entries (about 10% of the entries)!

```
a = MemoryInUse[];
```

```
finalMatrixSparse =
  Map[Fold[If[Abs[#2] ≤ 3.0, Append[#1, 0], Append[#1, #2]] &, {}, #] &,
   finalMatrix];
Select[Flatten[finalMatrixSparse], # != 0 &] // Length
```

```
6629
```

```
b = MemoryInUse[]; b - a
```

```
275408
```

Stored in the usual way — with the zero entries included — it takes about 270K bytes of memory. However, *Mathematica* 5 allows us to store this matrix as a sparse array, and doing so it takes about 80K bytes of memory. The improvement in memory storage is quite impressive.

```
sparseFinalMatrixSparse = SparseArray[finalMatrixSparse]
```

```
SparseArray[<6629>, {256, 256}]
```

```
c = MemoryInUse[]; c - b
```

```
83664
```

Compression achieved!

We then reverse the process and obtain an *approximation* to the original matrix! Notice that the sparse array has to be converted to a "normal" matrix first.

```
backUpOneStep = Map[discreteInverseWaveletTransform,
   Transpose[sparseFinalMatrixSparse // Normal]];
backUpOneStepTransposed = Transpose[backUpOneStep];
approximatedMatrix =
  Map[discreteInverseWaveletTransform, backUpOneStepTransposed];
```

Note that the approximated matrix has 65536 (= 256×256) nonzero entries, just like the original matrix.

```
Select[Flatten[approximatedMatrix], # != 0 &] // Length
```

```
65536
```

When we plot the approximated matrix, and consider the fact that we used only 10% of the entries in the original matrix, the result is not at all bad!

```
ListDensityPlot[approximatedMatrix, Mesh → False];
```

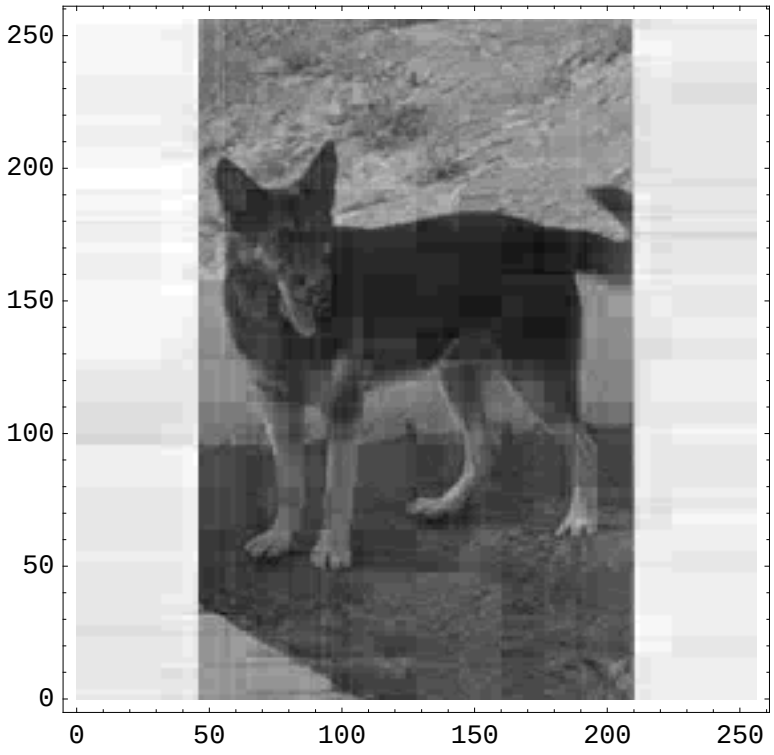


Figure *Pedro-2*: Picture of Pedro obtained using only 10% of the details.

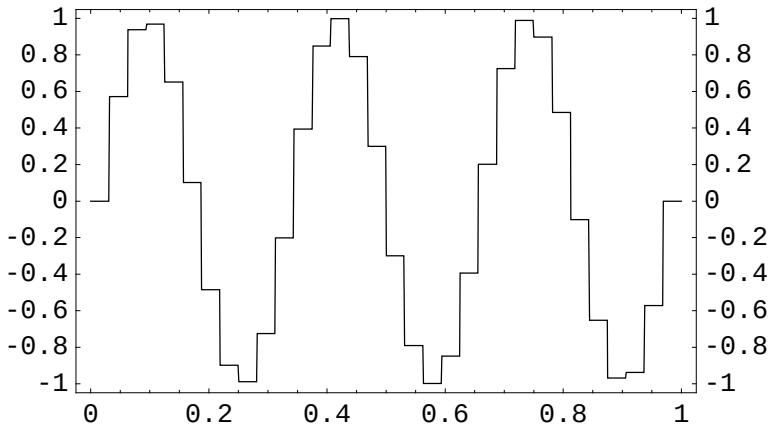
5.5 Wavelets and Multiresolution Analysis

In the spirit of Shannon (at least two samples per cycle) we are going over wavelets a second time and introduce the general concepts and notations used in their study.

■ Scaling Functions

We start by associating a list of k elements with a step function on [0, 1]; this step function could change at $k-1$ equally spaced points and uses the list elements as its step heights. As an example consider Figure 7 (shown below again) where the step function plotted is associated with the list of the 32 $\sin(6\pi x)$-values obtained by uniformly subdividing the interval [0, 1].

Figure 7




```
Step function version of the function Sin[6πx]
  in the interval [0, 1]; 32 sample points were used
```

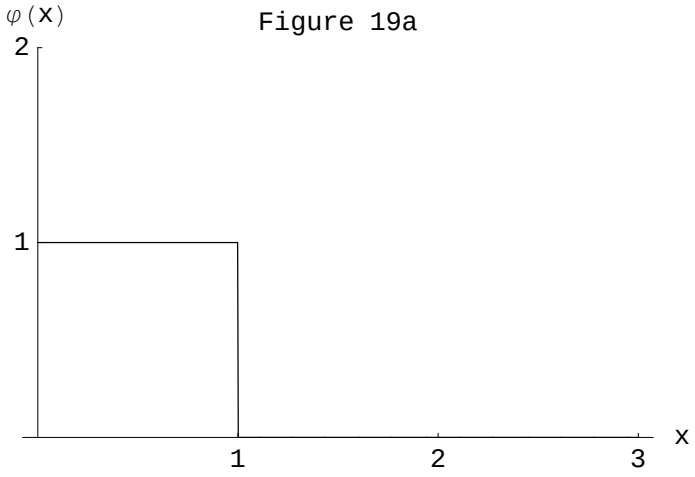
In turn, these step functions can be thought of as linear combinations of dyadically dilated and translated unit step functions on [0, 1). To explain the last statement we define the *Haar scaling function*:

$$\varphi(x) = \begin{cases} 1 & \text{for } 0 \le x < 1 \\ 0 & \text{otherwise} \end{cases}$$

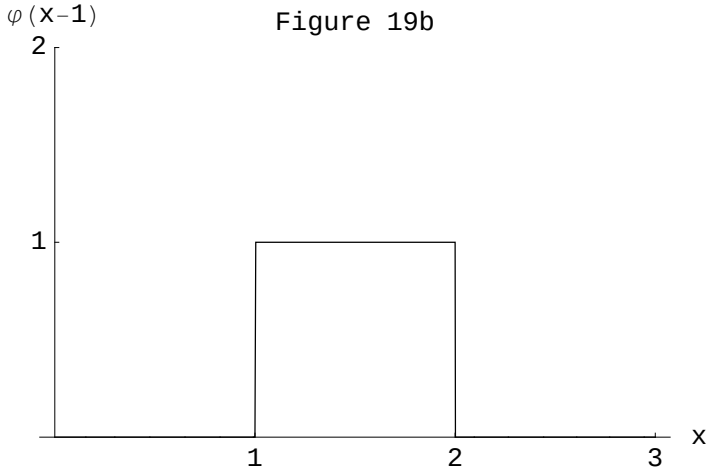
```
Clear[x];
φ[x_] := Which[0 ≤ x < 1, 1, True, 0];
```

and look at two of its plots for better understanding.

```
Plot[φ[x], {x, 0, 3}, PlotRange → {0, 2}, Ticks → {{1, 2, 3}, {1, 2}},
  PlotLabel → "Figure 19a", AxesLabel → {"x", "φ(x)"}];
```

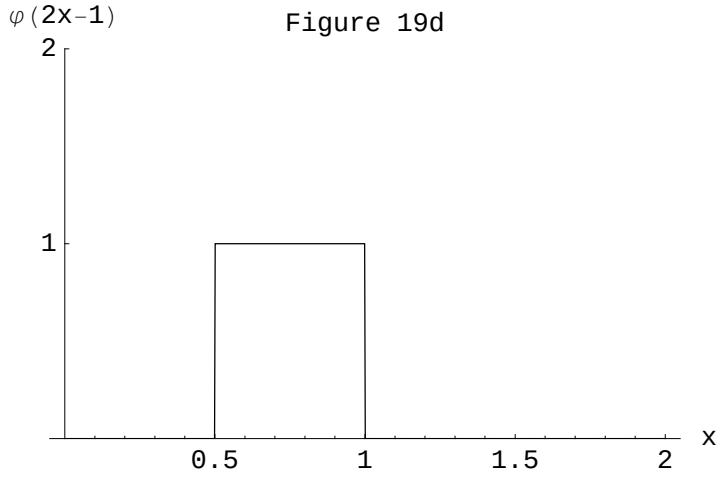



```
Plot[φ[x - 1], {x, 0, 3}, PlotRange → {0, 2}, Ticks → {{1, 2, 3}, {1, 2}},
  PlotLabel → "Figure 19b", AxesLabel → {"x", "φ(x-1)"}];
```



φ satisfies a scaling equation of the form $\varphi(x) = \sum_{i=0}^{\infty} c_i \varphi(2x - i)$, where in our case the only nonzero coefficients are $c_0 = c_1 = 1$; that is, we have $\varphi(x) = \varphi(2x) + \varphi(2x - 1)$. Note that the equations $h(i) = \frac{c_i}{2}$, $i = 0, 1$ define the filter coefficients.

```
Plot[φ[2 x - 1], {x, 0, 2},
  PlotRange → {0, 2}, Ticks → {Automatic, {1, 2}},
  PlotLabel → "Figure 19d", AxesLabel → {"x", "φ(2x-1)"}];
```



We now define, for each nonnegative integer i, the *approximating* vector space $\mathcal{V}^i$ of piecewise constant functions on [0, 1), with possibble breaks at $\frac{1}{2^i}$, $\frac{2}{2^i}$, ..., $\frac{2^i-1}{2^i}$. Then the 2^i dyadically dilated and translated scaling functions $\varphi_k^i(x) = \varphi(2^i x - k)$, $0 \le k \le 2^i\text{-}1$, form a basis for $\mathcal{V}^i$. We thus have the (theoretically infinite) chain of ascending vector spaces $\mathcal{V}^0 \subset \mathcal{V}^1 \subset \mathcal{V}^2 \subset \mathcal{V}^3 \subset \ldots$; however, since we are dealing with sampled signals, the resolution is finite and hence this chain stops at some finite value i.

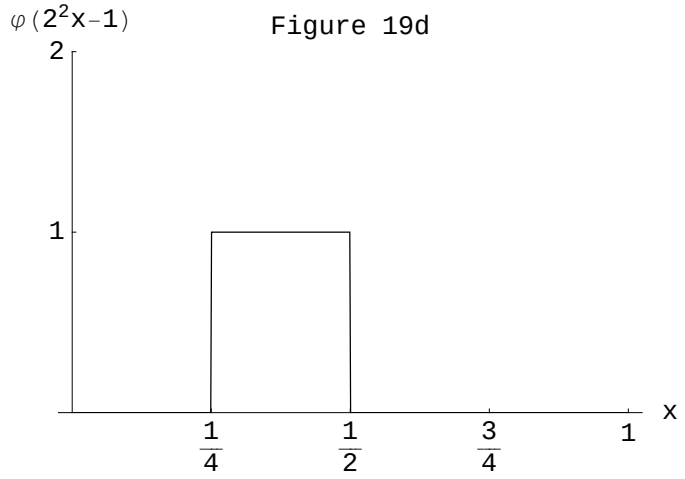
```
φ[x_, i_, k_] := φ[2^i x - k];
```

Taking for example $i = 2$, we have the set of 4 functions $\varphi(2^2 x - k)$, $k = 0, 1, 2, 3$.

Note that the function $\varphi(2^2 x)$ is 1 on the interval $\left[0, \frac{1}{4}\right)$,

the function $\varphi(2^2 x - 1)$ is 1 on the interval $\left[\frac{1}{4}, \frac{2}{4}\right)$ etc.

```
Plot[φ[2^2 x - 1], {x, 0, 1}, PlotRange → {0, 2},
  Ticks → {{0, 1 / 4, 2 / 4, 3 / 4, 1}, {1, 2}},
  PlotLabel → "Figure 19d", AxesLabel → {"x", "φ(2^2x-1)"}];
```



These four functions form a basis for the *approximating* vector space $\mathcal{V}^2$ of piecewise constant functions on [0, 1), with possibble breaks at 1/4, 2/4, 3/4. Taking a random list of four numbers $\{e_0,\ e_1,\ e_2,\ e_3\}$ we can think of them as the element

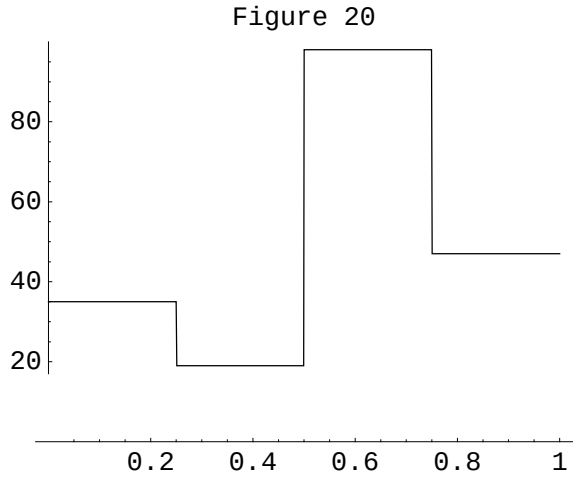
$$e_0\,\varphi[2^3\,x]\ +e_1\,\varphi[2^3\,x-1]\ +\ e_2\,\varphi[2^3\,x-2]\ +e_3\,\varphi[2^3\,x-3]$$

of the space $\mathcal{V}^2$. Given a specific list of length 4, its plot is shown in Figure 20:

```
data = Table[Random[Integer, {10, 99}], {4}]
```

```
{35, 19, 98, 47}
```

```
yStep = Table[φ[x, 2, k], {k, 0, 3}];
Plot[data.yStep, {x, 0, 1},
  PlotLabel → "Figure 20", AxesOrigin → {0, 0}];
```



In other words, we have a step function representation of our data list. Similarly, any list of length 4 can be associated with an element of the approximating space $\mathcal{V}^2$.

We can explain the Haar wavelet transform (lowpass/highpass filters) in terms of this version of data lists, but we need additional (vector) approximating spaces. As above, the two functions defined by $\varphi_k^1(x) = \varphi(2^1 x - k)$, $k = 0, 1$ form a basis of the approximating space $\mathcal{V}^1$ of the piecewise constant functions on [0, 1) with possible break at the point 1/2. Finally, the function $\varphi_0^0(x) = \varphi(x)$ itself is a basis of the approximating space $\mathcal{V}^0$ of the constant functions on [0, 1). We have $\mathcal{V}^0 \subset \mathcal{V}^1 \subset \mathcal{V}^2$

If we think of the various averages — that we computed in the Haar wavelet transform — as lower-resolution versions of the original data list, we can associate them with elements of these new vector spaces. The higher the dimension of the vector space $\mathcal{V}^i$ the higher the resolution. Let us come back to the example we did earlier, where now we print the intermediate results:

```
discreteWaveletTransform[{7, 5, 6, 2}, 1]
```

```
{7., 5., 6., 2.}
```

```
{6., 4., 1., 2.}
```

```
{5., 1., 1., 2.}
```

The first output line (the original data list) is associated with the element $7\varphi(2^2\,x-0)+5\varphi(2^2\,x-1)+6\varphi(2^2\,x-2)+2\varphi(2^2\,x-3)$ of the approximating space $\mathcal{V}^2$ and has the highest resolution. The average values 6 and 4 of the second output line are associated with the element $6\varphi(2^1\,x-0)+4\varphi(2^1\,x-1)$ of the approximating space $\mathcal{V}^1$ (the resolution is cut in half) and the overall average value 5 is associated with the element $5\varphi(2^0\,x)$ of the approximating space $\mathcal{V}^0$ (the resolution is cut *yet again* in half). ("Going down" the approximating spaces the resolution goes down as well. This is the dyadical dilation we mentioned above.)

- **Wavelets — at last!**

It only remains to associate the detail coefficients with "something". This is where wavelets come in as the *details* or *wavelet* space!

For any nonnegative integer i, consider the inner product

$$\langle \mathrm{f}\,,\,\mathrm{g}\rangle = \int_0^1 f[t]\,g[t]\,dt$$

defined in the vector space $\mathcal{V}^i$. Two functions are orthogonal if and only if their product on [0, 1] encloses equal areas above and below the horizontal axis. In $\mathcal{V}^{i+1}$ we define the wavelet space $\mathcal{W}^i$ as the orthogonal complement of the approximating space $\mathcal{V}^i$. Hence, we have the following (orthogonal) direct sum decomposition:

$$\begin{aligned}\mathcal{V}^{i+1} &= \mathcal{V}^i \oplus \mathcal{W}^i = \mathcal{V}^{i-1} \oplus \mathcal{W}^{i-1} \oplus \mathcal{W}^i \\ &\vdots \\ &= \mathcal{V}^0 \oplus \mathcal{W}^0 \oplus \mathcal{W}^1 \oplus \ldots \oplus \mathcal{W}^i\,.\end{aligned}$$

Note that at every resolution (approximating space) there exist details corresponding to *all* the previous resolutions.

Every wavelet space $\mathcal{W}^i$ has a basis $\chi_k^i(x) = \chi(2^i x - k)$, $k = 0, 1, ..., 2^i - 1$, which is used to associate the details, obtained during the Haar transform, with elements of the wavelet space; the details are called *wavelet coefficients*.

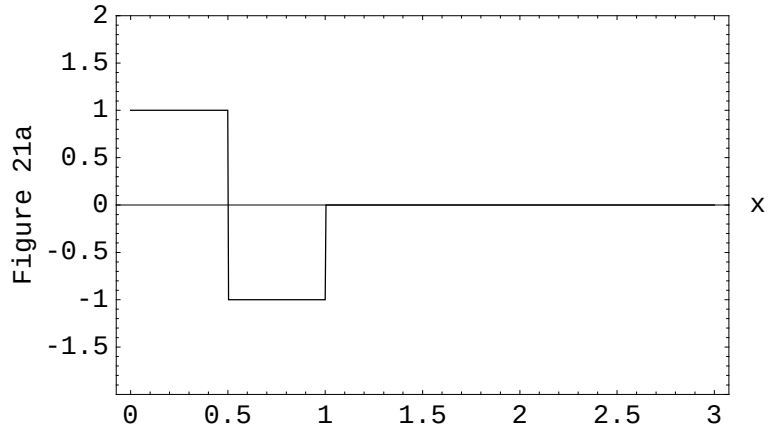
Equivalent to the Haar scaling function is the *mother Haar wavelet*, defined by:

$$\chi(x) = \begin{cases} 1 & \text{for } 0 \le x < \frac{1}{2} \\ -1 & \text{for } \frac{1}{2} \le x < 1 \\ 0 & \text{otherwise} \end{cases}$$

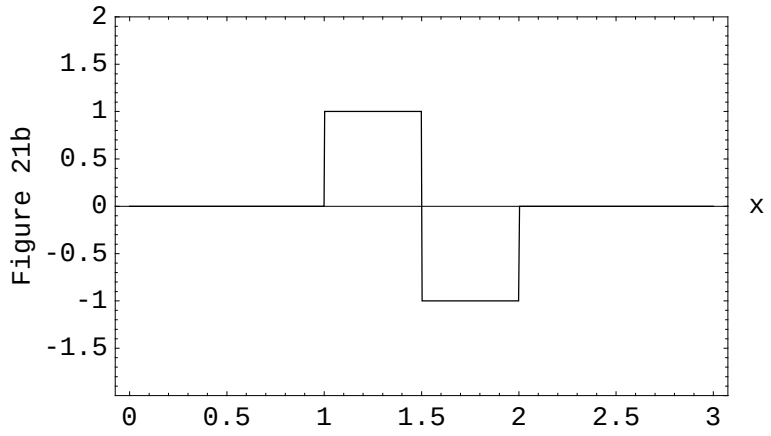
```
Clear[x];
χ[x_] := Which[0 ≤ x < 1 / 2, 1, 1 / 2 ≤ x < 1, -1, True, 0];
```

along with two plots for better understanding.

```
Plot[χ[x], {x, 0, 3}, PlotRange → {-2, 2},
  Ticks → {{0, 1, 2, 3}, {1, 2}}, Frame → True,
  FrameLabel → "Figure 21a", AxesLabel → {"x", "χ(x)"}];
```



```
Plot[χ[x - 1], {x, 0, 3}, PlotRange → {-2, 2},
  Ticks → {{0, 1, 2, 3}, {1, 2}}, Frame → True,
  FrameLabel → "Figure 21b", AxesLabel → {"x", "χ(x)"}];
```



The function $\chi(x)$ is orthogonal to $\varphi(x)$ and forms a basis for $\mathcal{W}^0$. Likewise, the functions

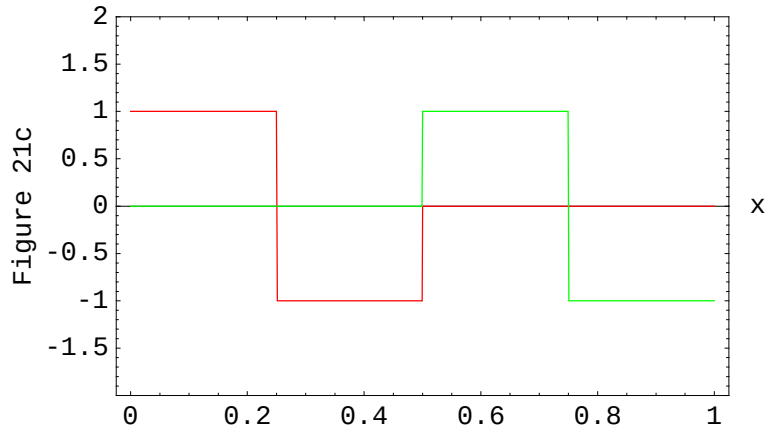
$$\chi_k^i(x) = \chi(2^i x - k)\ ,\ k = 0,\ 1,\ ...,\ 2^i - 1$$

are visibly orthogonal to each other and to the corresponding functions $\varphi_k^i(x)$; hence, they form a basis for $\mathcal{W}^i$.

```
χ[x_, i_, k_] := χ[2^i x - k];
```

Moreover, for $i = 1$ we have the 2 functions $\chi_k^1(x) = \chi(2^1 x - k)$, $k = 0,\ 1$ which are visibly orthogonal to each other and to the corresponding functions $\varphi_k^1(x)$ which form a basis of the resolution space $\mathcal{V}^1$. Hence they form a basis of $\mathcal{W}^1$.

```
Plot[{χ[2 x - 0], χ[2 x - 1]}, {x, 0, 1},
  PlotRange → {-2, 2}, Ticks → {{0, 1 / 4, 2 / 4, 3 / 4, 1}, {1, 2}},
  PlotStyle → {RGBColor[1, 0, 0], RGBColor[0, 1, 0]}, Frame → True,
  FrameLabel → "Figure 21c", AxesLabel → {"x", "χ(2x)"}];
```



Taking another look at our example

```
discreteWaveletTransform[{7, 5, 6, 2}, 1]
```

```
{7., 5., 6., 2.}
```

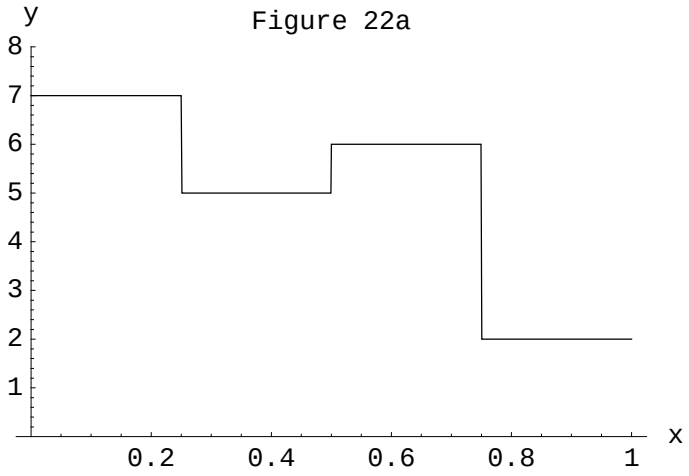
```
{6., 4., 1., 2.}
```

```
{5., 1., 1., 2.}
```

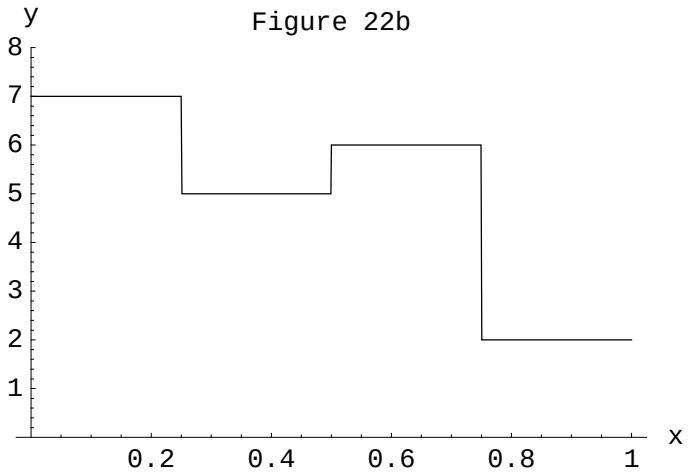
we see that each output line is associated with the corresponding line of the following relations:

$$
\begin{aligned}
&7\varphi(2^2 x - 0) + 5\varphi(2^2 x - 1) + 6\varphi(2^2 x - 2) + 2\varphi(2^2 x - 3) \\
&= 6\varphi[2^1 x - 0] + 4\varphi(2^1 x - 1) + 1\chi(2^1 x - 0) + 2\chi(2^1 x - 1) \\
&= 5\varphi(2^0 x) + 1\chi(2^0 x) + 1\chi(2^1 x - 0) + 2\chi(2^1 x - 1).
\end{aligned}
$$

```
Plot[7 φ[2^2 x - 0] + 5 φ[2^2 x - 1] + 6 φ[2^2 x - 2] + 2 φ[2^2 x - 3] ,
 {x, 0, 1}, PlotRange → {0, 8},
 PlotLabel → "Figure 22a", AxesLabel → {"x", "y"}];
```




```
Plot[5 φ[2^0 x] + 1 χ[2^0 x] + 1 χ[2^1 x - 0] + 2 χ[2^1 x - 1], {x, 0, 1},
 PlotRange → {0, 8}, PlotLabel → "Figure 22b", AxesLabel → {"x", "y"}];
```



The images that we compressed in the previous section had a resolution of 256×256. Thus, working with lists of length 256 was equivalent to working in the larger space $\mathcal{V}^8$ and using the identity

$$\mathcal{V}^0 \oplus \mathcal{W}^0 \oplus \mathcal{W}^1 \oplus \ldots \oplus \mathcal{W}^7$$

one overall average and 255 details.

There are two-dimensional analogs of those ideas providing a theoretical framework for the digital image representation and compression ideas discussed in the last section. Details can be found elsewhere.

- **Wavelet Transform Benefits — Revisited**

As we have seen in an example in the previous section, we set the smallest detail coefficients equal to zero and achived a pretty good approximation of the original list (of eight elements). This amounts to setting some of the wavelet coefficients to zero. Let us look again at the same list *data2*:

```
data2 = {74, 58, 26, 42, 66, 66, 58, 34};
discreteWaveletTransform[data2]
```

```
{53., -3., 16., 10., 8., -8., 0., 12.}
```

Setting the wavelet coefficient -3 to 0 we have the approximated list *data2Approximated*.

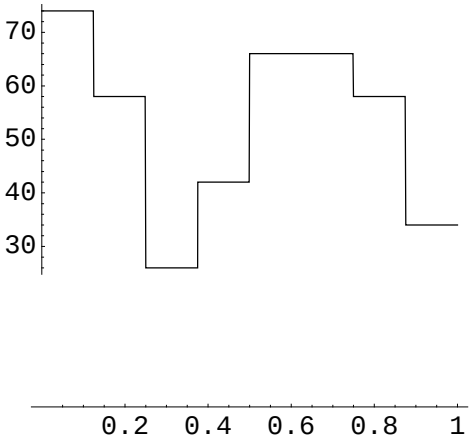
```
data2Approximated =
 discreteInverseWaveletTransform[{53, 0, 16, 10, 8, -8, 0, 12}]
```

```
{77, 61, 29, 45, 63, 63, 55, 31}
```

Using step functions — this time — we plot these two lists in Figures 23a and 23b and ask the reader to spot the difference!

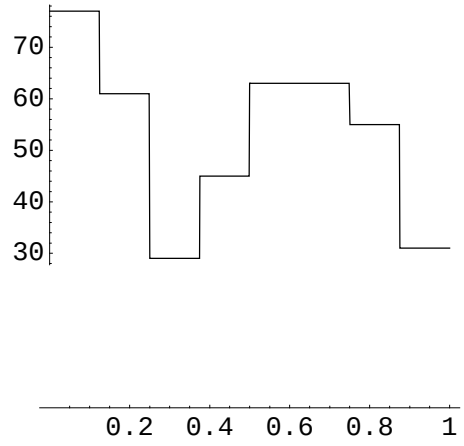
```
Clear[x];
yStep2 = Table[φ[x, 3, k], {k, 0, 7}];
Plot[data2.yStep2, {x, 0, 1},
  PlotLabel → "Figure 23a", AxesOrigin → {0, 0}];
```

Figure 23a




```
Plot[data2Approximated.yStep2, {x, 0, 1},
  PlotLabel → "Figure 23b", AxesOrigin → {0, 0}];
```

Figure 23b



5.5 Filters and Filter Banks with Matrices

In this section we give a matrix formulation of the Haar wavelet transform. Consider the, by now, well known example with the list {7, 5, 6, 2}. We apply the Haar transform and below we see again the intermediate and final results :

```
row1 = {7, 5, 6, 2};
discreteWaveletTransform[row1, 1]
```

```
{7., 5., 6., 2.}
```

```
{6., 4., 1., 2.}
```

```
{5., 1., 1., 2.}
```

■ Matrix formulation of the Haar transform

The same results can be obtained using the following matrices A_1 , A_2 (filter banks):

$$\text{A1} = \begin{pmatrix} \frac{1}{2} & \frac{1}{2} & 0 & 0 \\ 0 & 0 & \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & \frac{-1}{2} & 0 & 0 \\ 0 & 0 & \frac{1}{2} & \frac{-1}{2} \end{pmatrix};$$

$$\text{A2} = \begin{pmatrix} \frac{1}{2} & \frac{1}{2} & 0 & 0 \\ \frac{1}{2} & \frac{-1}{2} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix};$$

Matrix A_1 is associated with the first pair of lowpass/highpass filter. The intermediate result, row2 = {6, 4, 1, 2}, is obtained from the product *A1.data1*:

```
row2 = A1.row1
```

```
{6, 4, 1, 2}
```

Matrix A_2 is associated with the second pair of lowpass/highpass filter; only the first two elements of *row2*, i.e. {6, 4}, are affected by the filter bank. The result, *row3* = {5, 1, 1, 2}, is obtained from the product *A2.row2*:

```
row3 = A2.row2
```

```
{5, 1, 1, 2}
```

Therefore, in this case the discrete wavelet transform matrix is the matrix $A_2.A_1$

```
(discreteWaveletTransformMatrix = A2.A1) // MatrixForm
```

$$\begin{pmatrix} \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \\ \frac{1}{4} & \frac{1}{4} & -\frac{1}{4} & -\frac{1}{4} \\ \frac{1}{2} & -\frac{1}{2} & 0 & 0 \\ 0 & 0 & \frac{1}{2} & -\frac{1}{2} \end{pmatrix}$$

```
discreteWaveletTransformMatrix.row1
```

```
{5, 1, 1, 2}
```

Likewise, the discrete inverse wavelet transform matrix in this case is the matrix Inverse[A_1].Inverse[A_2].

```
(discreteInverseWaveletTransformMatrix =
   Inverse[discreteWaveletTransformMatrix]) // MatrixForm
```

$$\begin{pmatrix} 1 & 1 & 1 & 0 \\ 1 & 1 & -1 & 0 \\ 1 & -1 & 0 & 1 \\ 1 & -1 & 0 & -1 \end{pmatrix}$$

```
discreteInverseWaveletTransformMatrix.row3
```

```
{7, 5, 6, 2}
```

```
Inverse[A1].Inverse[A2].row3
```

```
{7, 5, 6, 2}
```

It is fairly obvious how to construct the $2^i \times 2^i$ matrices A_1, A_2, ..., A_i needed to work with lists of length 2^i.

Observe now that the rows of the discrete wavelet transform matrix are orthogonal, but *not* orthonormal:

```
Table[discreteWaveletTransformMatrix[[i]].
  discreteWaveletTransformMatrix[[j]], {i, 1, 4}, {j, 1, 4}]
```

$$\left\{\left\{\frac{1}{4}, 0, 0, 0\right\}, \left\{0, \frac{1}{4}, 0, 0\right\}, \left\{0, 0, \frac{1}{2}, 0\right\}, \left\{0, 0, 0, \frac{1}{2}\right\}\right\}$$

We know from Linear Algebra that if the rows of a matrix are orthonormal, then its inverse is the same as its transpose. Consequently, the inverse in these cases can be very easily computed; in our example above, the inverse discrete wavelet transform matrix would be the matrix Transpose[A_1].Transpose[A_2] — instead of the matrix Inverse[A_1].Inverse[A_2].

Hence, to speed up computations, we need to normalize the rows of matrices A_1 and A_2. This is achieved by dividing *each* row of A_1 and the *first two* rows of A_2 by $\frac{1}{\sqrt{2}}$, their measure. We thus obtain the orthonormal matrices $A_1\,N$ and $A_2\,N$:

```
A1N = {{1/Sqrt[2], 1/Sqrt[2], 0, 0}, {0, 0, 1/Sqrt[2], 1/Sqrt[2]},
  {1/Sqrt[2], -1/Sqrt[2], 0, 0}, {0, 0, 1/Sqrt[2], -1/Sqrt[2]}};
```

$$\text{A1N} = \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 & 0 \\ 0 & 0 & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} & 0 & 0 \\ 0 & 0 & \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} \end{pmatrix};$$

```
A2N = {{1/Sqrt[2], 1/Sqrt[2], 0, 0}, {1/Sqrt[2], -1/Sqrt[2], 0, 0},
  {0, 0, 1, 0}, {0, 0, 0, 1}};
```

$$\text{A2N} = \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 & 0 \\ \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix};$$

This way the *normalized* discrete wavelet transform matrix and its inverse are computed as follows:

```
( discreteWaveletTransformMatrixN = A2N.A1N ) // MatrixForm
```

$$\begin{pmatrix} \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} & -\frac{1}{2} & -\frac{1}{2} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & 0 & 0 \\ 0 & 0 & \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{pmatrix}$$

```
(discreteInverseWaveletTransformMatrixN =
   Transpose[discreteWaveletTransformMatrixN]) // MatrixForm
```

$$\begin{pmatrix} \frac{1}{2} & \frac{1}{2} & \frac{1}{\sqrt{2}} & 0 \\ \frac{1}{2} & \frac{1}{2} & -\frac{1}{\sqrt{2}} & 0 \\ \frac{1}{2} & -\frac{1}{2} & 0 & \frac{1}{\sqrt{2}} \\ \frac{1}{2} & -\frac{1}{2} & 0 & -\frac{1}{\sqrt{2}} \end{pmatrix}$$

and both are orthonormal as can be easily verified:

```
Table[discreteWaveletTransformMatrixN[[i]].
  discreteWaveletTransformMatrixN[[j]], {i, 1, 4}, {j, 1, 4}]
```

```
{{1, 0, 0, 0}, {0, 1, 0, 0}, {0, 0, 1, 0}, {0, 0, 0, 1}}
```

The results of the transform are somewhat different now. To wit, we have:

```
discreteWaveletTransformMatrixN.row1
```

$$\left\{10,\ 2,\ \sqrt{2},\ 2\sqrt{2}\right\}$$

```
discreteInverseWaveletTransformMatrixN.%
```

```
{7, 5, 6, 2}
```

This process is called the *normalization process* and is equivalent to working with the *normalized* Haar scaling and wavelet functions; that is, the functions $\varphi_k^i(x) = \sqrt{2^i}\ \varphi(2^i x - k)$, and $\chi_k^i(x) = \sqrt{2^i}\ \chi(2^i x - k)$, $k = 0, 1, ..., 2^i - 1$, are now used as bases of the spaces $\mathcal{V}^i$ and $\mathcal{W}^i$ respectively. In other words, the average value is computed by $\frac{(a+b)}{\sqrt{2}}$ whereas the difference is computed by $\frac{(a-b)}{\sqrt{2}}$. Strange as it might seem,

the normalization process results in compressed images that are more pleasing to the eye as we can see below.

- ***Normalized* Haar wavelet transform algorithms.**

```
discreteWaveletTransformNormalized::usage =
  "This routine has at most 2 arguments.  The first
    argument is a list of ANY length; the second argument
    is an OPTIONAL flag with default value 0. Called with
    one argument it does NOT print the results of the
    intermediate steps; to print them set the flag to 1.
  Regarding the first argument (list), its length should be 2^i,
    i>0. If this is not the case, zeros are appended to fix the
    situation. The  averaging-differencing algorithm (LowPass/
    HighPass Filtering) is performed on the (new) list. After i (
    or i+1)iterations the output is a list of the same length.";


discreteWaveletTransformNormalized[list_List, flag_Integer: 0] :=
 Module[{tempList = list, list2,
   tempInteger = i = Log[2, Length[list]] // N // IntegerPart,
   pairedRow, averages, differences, fl = flag, nextRow, temp},
  If[i == 0, Print["Adjust the list; its length should be 2^i, i>0"];
   Return[Null]];
  If[2^i ≠ Length[list], list2 =
    Append[tempList, Table[0, {i, 1, 2^(i+1) - Length[list]}]] // Flatten;
   tempInteger = i = i + 1; tempList = list2];
  nextRow = {}; If[fl == 1, Print[tempList // N]];
  While[tempInteger ≠ 0,
   pairedRow = pairedList[tempList];
   averages = Apply[averNormalized, pairedRow, 1];
   differences = diffNormalized[pairedRow, averages];
   nextRow = Prepend[nextRow, {averages, differences}] // Flatten;
   If[fl == 1, Print[nextRow]];
   tempInteger = tempInteger - 1; temp = 2^tempInteger;
   tempList = Drop[nextRow, -2^i + temp];
   nextRow = Drop[nextRow, temp];
  ]; If[fl == 1, Prepend[nextRow, First[tempList]];,
   Prepend[nextRow, First[tempList]]]
 ]
```

```
(*          2 nd function                *)

discreteInverseWaveletTransformNormalized::usage =
  "This routine has at most 2 arguments.  The first
    argument is a list of ANY length; the second argument
    is an OPTIONAL flag with default value 0. Called with
    one argument it does NOT print the results of the
    intermediate steps; to print them set the flag to 1.
  Regarding the first argument (list), its length should
    be 2^i\\\\\\\\, i>0. If this is not the case, zeros are
    appended to fix the situation. The  INVERSE averaging-
    differencing algorithm (LowPass/HighPass Filtering)
    is performed on the (new) list. After i (or i+1)
    iterations the output is a list of the same length.";

discreteInverseWaveletTransformNormalized[
  list_List, flag_Integer: 0] :=
 Module[{tempList = list, tempList2 = list, list2, av, di, fl = flag,
   i = Log[2, Length[list]] // N // IntegerPart, j, m = 1, temp = 1},
  If[i == 0, Print["Adjust the list; its length should be 2^i, i>0"];
   Return[Null]];
  If[2^i ≠ Length[list], list2 =
    Append[tempList, Table[0, {i, 1, 2^(i+1) - Length[list]}]] // Flatten;
   i = i + 1; tempList = tempList2 = list2];
  m = 1;
  While[m ≤ i, j = 0; k = 1;
   While[k <= temp,  av = tempList[[k]]; di = tempList[[k + temp]];
    tempList2 = ReplacePart[tempList2, (av + di)/Sqrt[2.], k + j];
    tempList2 = ReplacePart[tempList2, (av - di)/Sqrt[2.], k + j + 1];
    j = j + 1; k = k + 1
   ]; If[fl == 1,  Print[tempList]]; temp = 2 * temp;
   tempList = tempList2; m = m + 1
  ]; tempList
 ]

(*    3 d function; computes the average value of 2 numbers     *)

averNormalized[a_, b_] := (a + b) / Sqrt[2.]
```

```
(*    4 th function; splits a list in pairs    *)

pairedList[list_] := Partition[list, 2]
```

```
(*          5 th function                    *)
diffNormalized::usage =
  "The inputs are two lists of equal length--- 2^k, for some k.
    The first list is a list of pairs, whereas the second list
    is a simple list (of averages). The output is a simple list
    of the same length as the input lists; each element of the
    output list is the difference of the first element of each
    pair (of the first list) and the corresponding element
    of the second list. For example, if the input lists are
    {{a,b},{c,d}} and {e,f} the output is {a-e,c-f}. ";

diffNormalized[list1_, list2_] :=
 Fold[Append[#1, (#2[[1, 1]] - #2[[1, 2]])/Sqrt[2.]] &,
  {}, Thread[{list1, list2}]]
```

Example.

```
discreteWaveletTransformNormalized[row1, 1]
```

```
{7., 5., 6., 2.}
```

```
{8.48528, 5.65685, 1.41421, 2.82843}
```

```
{10., 2., 1.41421, 2.82843}
```

10 is the general average and the coefficients 2, 1.41421, 2.82843 are the details. This transformation is reversible:

```
discreteInverseWaveletTransformNormalized[%, 1]
```

```
{10., 2., 1.41421, 2.82843}
```

```
{8.48528, 5.65685, 1.41421, 2.82843}
```

```
{7., 5., 6., 2.}
```

■ Compression of arbitrary images — revisited

Here we work again on the Pedro picture using 10% and 3% of the detail coefficients. We see that in the first case the result is a picture almost identical to the original picture

```
originalMatrix = data1;
transformedMatrix =
  Map[discreteWaveletTransformNormalized, originalMatrix];
transformedTransposedMatrix = Transpose[transformedMatrix];
finalMatrix = Map[discreteWaveletTransformNormalized,
    transformedTransposedMatrix] // Transpose;
```

```
a = MemoryInUse[];
```

```
finalMatrixSparseNormalized = Map[
   Fold[If[Abs[#2] ≤ 15.0, Append[#1, 0], Append[#1, #2]] &, {}, #] &,
   finalMatrix];
Select[Flatten[finalMatrixSparseNormalized], # != 0 &] // Length
```

```
6826
```

```
b = MemoryInUse[]; b - a
```

```
276024
```

```
sparseFinalMatrixSparseNormalized =
 SparseArray[finalMatrixSparseNormalized]
```

```
SparseArray[<6826>, {256, 256}]
```

```
c = MemoryInUse[]; c - b
```

```
86080
```

```
backUpOneStep = Map[discreteInverseWaveletTransformNormalized,
   Transpose[sparseFinalMatrixSparseNormalized // Normal]];
backUpOneStepTransposed = Transpose[backUpOneStep];
approximatedMatrixNormalized =
  Map[discreteInverseWaveletTransformNormalized,
   backUpOneStepTransposed];
```

```
Select[Flatten[approximatedMatrixNormalized], # != 0 &] // Length
```

```
65536
```

```
ListDensityPlot[approximatedMatrixNormalized, Mesh → False];
```

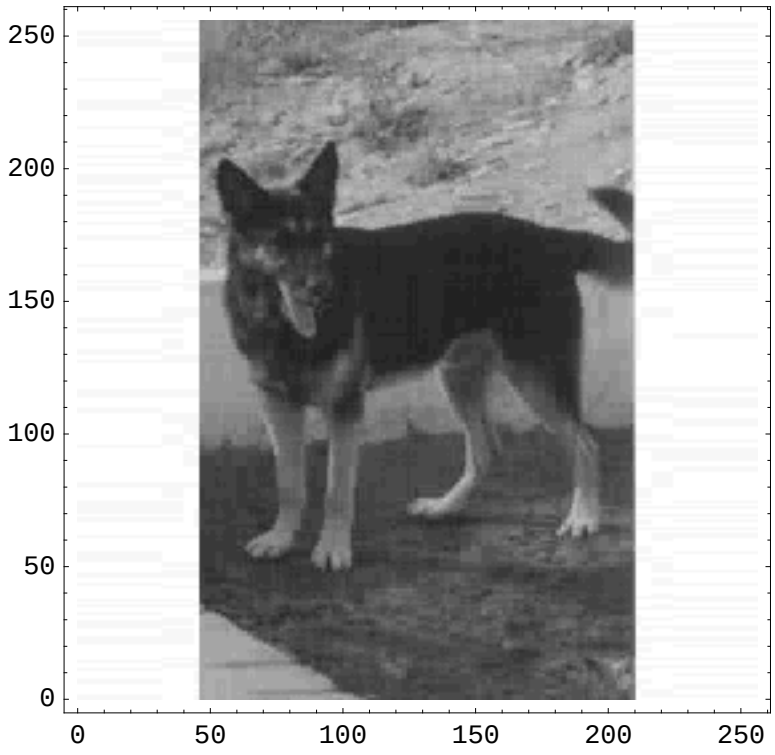


Figure *Pedro-3*: Picture of Pedro obtained using 10% of the details in the normalized wavelet transform; almost identical to *Pedro-1* (the original).

Let us see how far we can compress the picture now. Below we use 2216 nonzero elements and the sparse array can be stored in only 30K bytes of memory!

```
a = MemoryInUse[];
```

```
finalMatrixSparseNormalized = Map[
   Fold[If[Abs[#2] ≤ 35.0, Append[#1, 0], Append[#1, #2]] &, {}, #] &,
   finalMatrix];
Select[Flatten[finalMatrixSparseNormalized], # != 0 &] // Length
```

```
2216
```

```
b = MemoryInUse[]; b - a
```

```
276056
```

```
sparseFinalMatrixSparseNormalized =
 SparseArray[finalMatrixSparseNormalized]
```

```
SparseArray[<2216>, {256, 256}]
```

```
c = MemoryInUse[]; c - b
```

```
30592
```

```
backUpOneStep = Map[discreteInverseWaveletTransformNormalized,
   Transpose[sparseFinalMatrixSparseNormalized // Normal]];
backUpOneStepTransposed = Transpose[backUpOneStep];
approximatedMatrixNormalized =
  Map[discreteInverseWaveletTransformNormalized,
   backUpOneStepTransposed];
```

```
Select[Flatten[approximatedMatrixNormalized], # != 0 &] // Length
```

```
65536
```

```
ListDensityPlot[approximatedMatrixNormalized, Mesh → False];
```

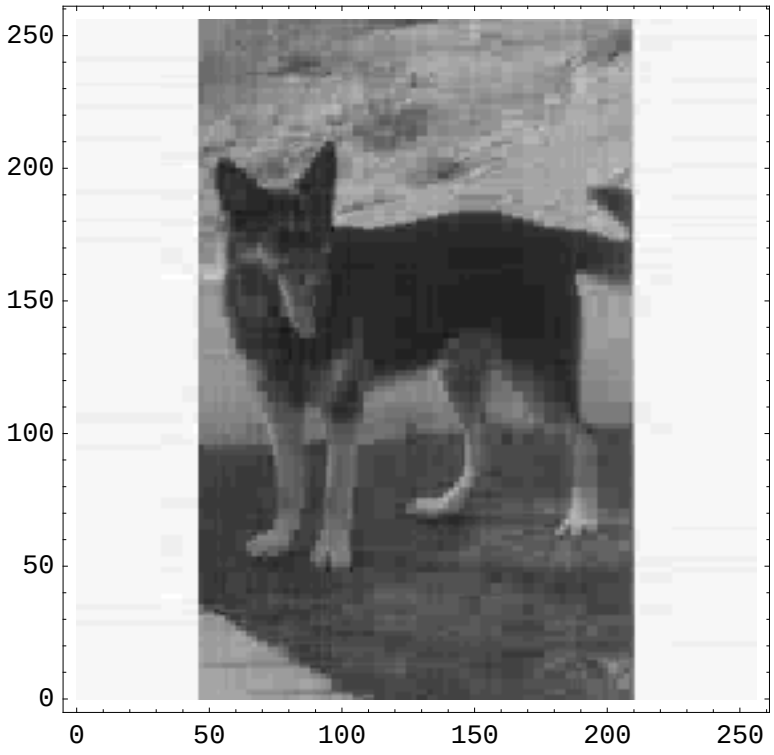


Figure *Pedro-4*: Picture of Pedro obtained using about 3% of the details in the normalized wavelet transform; almost identical to *Pedro-2* (obtained using 10% of the details in the *non*-normalized wavelet transform).

The advantages of normalization are obvious!

Ευρετήριο