ΕΙΣΑΓΩΓΗ ΣΤΗΝ ΑΝΑΛΥΣΗ ΑΛΓΟΡΙΘΜΩΝ

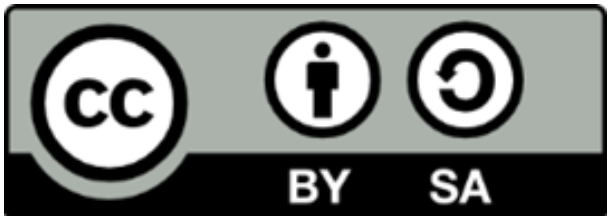
Ενότητα 13: Αλγόριθμοι-Μεγάλων ακεραίων-Εκθετοποίηση- Πολλαπλασιασμός πινάκων -Strassen

Μαρία Σατρατζέμη
Τμήμα Εφαρμοσμένης Πληροφορικής

Ευρωπαϊκή Ένωση
Ευρωπαϊκό Κοινωνικό Ταμείο
ΕΠΙΧΕΙΡΗΣΙΑΚΟ ΠΡΟΓΡΑΜΜΑ ΕΚΠΑΙΔΕΥΣΗ ΚΑΙ ΔΙΑ ΒΙΟΥ ΜΑΘΗΣΗ
επένδυση στην κοινωνία της γνώσης
ΥΠΟΥΡΓΕΙΟ ΠΑΙΔΕΙΑΣ ΚΑΙ ΘΡΗΣΚΕΥΜΑΤΩΝ
ΕΙΔΙΚΗ ΥΠΗΡΕΣΙΑ ΔΙΑΧΕΙΡΙΣΗΣ
Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης
ΕΣΠΑ 2007-2013
πρόγραμμα για την ανάπτυξη
ΕΥΡΩΠΑΪΚΟ ΚΟΙΝΩΝΙΚΟ ΤΑΜΕΙΟ

Άδειες Χρήσης

- Το παρόν εκπαιδευτικό υλικό υπόκειται σε άδειες χρήσης Creative Commons.
- Για εκπαιδευτικό υλικό, όπως εικόνες, που υπόκειται σε άλλου τύπου άδειας χρήσης, η άδεια χρήσης αναφέρεται ρητώς.



Χρηματοδότηση

- Το παρόν εκπαιδευτικό υλικό έχει αναπτυχθεί στα πλαίσια του εκπαιδευτικού έργου του διδάσκοντα.
- Το έργο «Ανοικτά Ακαδημαϊκά Μαθήματα στο Πανεπιστήμιο Μακεδονίας» έχει χρηματοδοτήσει μόνο τη αναδιαμόρφωση του εκπαιδευτικού υλικού.
- Το έργο υλοποιείται στο πλαίσιο του Επιχειρησιακού Προγράμματος «Εκπαίδευση και Δια Βίου Μάθηση» και συγχρηματοδοτείται από την Ευρωπαϊκή Ένωση (Ευρωπαϊκό Κοινωνικό Ταμείο) και από εθνικούς πόρους.

Πρόσθεση – Πολλαπλασιασμός Ακεραίων

- **Υπάρχουν αποδοτικοί αλγόριθμοι για την πρόσθεση και τον πολλαπλασιασμό των ακεραίων (μας ενδιαφέρουν κυρίως οι μη αρνητικοί ακέραιοι).**
- **Λέγοντας αποδοτικό αλγόριθμο εννοούμε έναν αλγόριθμο του οποίου ο χρόνος εκτέλεσης είναι φραγμένος (άνω) από ένα πολυώνυμο ως προς το μέγεθος των αριθμών στην είσοδό του.**
- **Ως γνωστόν, αναλύουμε τον χρόνο εκτέλεσης ενός αλγορίθμου σαν συνάρτηση του μεγέθους των δεδομένων της εισόδου.**
- **Μέσα στον αλγόριθμο συχνά πρέπει να εκτελέσουμε αριθμητικές πράξεις, όπως πρόσθεση ή πολλαπλασιασμό των δεικτών του πίνακα. Τυπικά υποθέτουμε ότι αυτές έχουν κόστος $O(1)$.**
- **Ο λόγος που αυτή η υπόθεση είναι λογική είναι ότι οι εν λόγω αριθμοί είναι μικροί και το κόστος χειρισμού τους είναι αμελητέο συγκρινόμενο με το κόστος που είναι ανάλογο του μεγέθους του πίνακα ή του γραφήματος με τα οποία δουλεύουμε.**

Πρόσθεση – Πολλαπλασιασμός Ακεραίων

- Υπάρχουν όμως εφαρμογές και προβλήματα όπου οι αριθμοί που προκύπτουν στους αλγορίθμους (π.χ. τους κρυπτογραφικούς) έχουν μεγέθη όπως το 2512 ή το 21024.
- Οι αριθμητικές πράξεις με αυτούς τους αριθμούς είναι το κύριο κόστος του αλγορίθμου και αυτό το κόστος αυξάνει καθώς οι αριθμοί γίνονται όλο και πιο μεγάλοι.
- Οι αριθμοί παρέχονται στον αλγόριθμο συνήθως σε δυαδική μορφή και το μέγεθος των αριθμών της εισόδου είναι συνεπώς το πλήθος των bit στη δυαδική τους αναπαράσταση ή κωδικοποίηση.
- Κωδικοποίηση Ακεραίων. Η μελέτη των αλγορίθμων οι οποίοι έχουν ως είσοδο και έξοδο ακεραίους απαιτεί, όπως είπαμε, την κωδικοποίηση των ακεραίων. Υποθέτουμε πάντα ότι οι ακέραιοι $a \geq 0$, είναι κωδικοποιημένοι στη βάση B ως μη προσημασμένοι ακέραιοι:

$$a = a_0 + a_1 \cdot B^1 + \cdots + a_{k-2} B^{k-2} + a_{k-1} B^{k-1} = \sum_{i=0}^{k-1} a_i \cdot B^i .$$

- Αν το (μεγιστοβάθμιο) ψηφίο a_{k-1} δεν είναι μηδέν, λέμε τον a ακέραιο k-bit και το k λέγεται μήκος του a. Το μήκος του $a \in N$ συμβολίζεται ως $|a|$, ή len(a) αν υπάρχει κίνδυνος να δημιουργηθεί σύγχυση με την απόλυτη τιμή, και είναι ίσο με $\lfloor \log_B a \rfloor + 1$. Οι ακέραιοι με μήκος k είναι οι αριθμοί $a \in \infty$ με $B^{k-1} \leq a \leq B^k - 1$.

Πρόσθεση δύο ακεραίων

- Έστω ότι $a = (a_{n-1} \ldots a_0)_B$ και $b = (b_{n-1} \ldots b_0)_B$ είναι μη προσημασμένοι ακέραιοι. Χρησιμοποιώντας την καθιερωμένη μέθοδο "με χαρτί και μολύβι" (προσαρμοσμένη από τη βάση 10 στη βάση B φυσικά), μπορούμε να υπολογίσουμε την αναπαράσταση του $a + b$ ως προς βάση B, ως εξής:
- Γράφουμε τον έναν κάτω από τον άλλον, με τα λιγότερο σημαντικά ψηφία ευθυγραμμισμένα, και αθροίζουμε τους ακεραίους κατά ψηφίο, μεταφέροντας τα κρατούμενα, από μια θέση στην επόμενη

	a_{n-1}	…	a_1	a_0	πρώτος προσθετέος
	b_{n-1}	…	b_1	b_0	δεύτερος προσθετέος
s_n	s_{n-1}	…	s_1	0	κρατούμενα
c_n	c_{n-1}	…	c_1	c_0	άθροισμα

- όπου s_n έως s_0 είναι η ακολουθία των κρατούμενων και $c = (c_n \ldots c_0)_B$ είναι το άθροισμα. Έχουμε

$$s_0 = 0,\ a_i + b_i + s_i = s_{i+1} \cdot B + c_i \text{ για } 0 \le i < n \text{ και } c_n = s_n.$$

Πρόσθεση δύο ακεραίων

- **Ψευδοκώδικας**

  carry ← 0

  for i ← 0 to n – 1

  tmp ← $a_i + b_i$ + carry // $0 \le tmp \le 2B - 1$

  carry ← $\lfloor tmp/B \rfloor$ // carry = 0 ή 1

  c_i ← tmp mod B

  c_n ← carry //το κρατούμενο ψηφίο από την αριστερότερη στήλη

  //τίθεται μπροστά από το αποτέλεσμα χωρίς

  //επιπλέον υπολογισμό

- **Το άθροισμα $c := a + b$ είναι της μορφής $c = (c_n c_{n-1} \ldots c_0)_B$. Ο χρόνος εκτέλεσης είναι $T(n) = O(n)$.**
- ***Παράδειγμα:***

  6 9 1 7

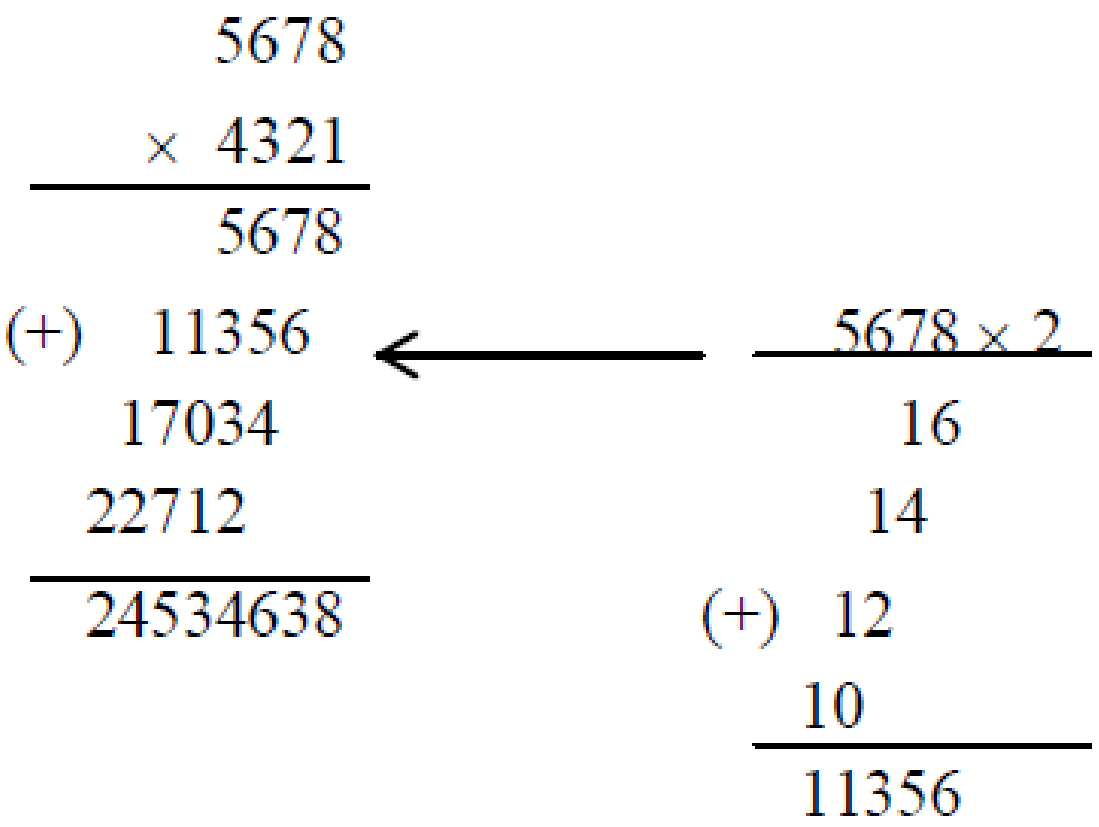
  + 4 2 6 9

  (1 1 0 1 0)

  1 1 1 8 6

Πολλαπλασιασμός δύο ακεραίων

- ***Παράδειγμα:*** **Έστω *a* = 5678 και *b* = 4321.**
- **Έχουμε**



Αλγόριθμος

- **Έστω ότι $a = (a_{k-1}\ldots a_0)_B$ και $b = (b_{\ell-1}\ldots b_0)_B$ είναι μη προσημασμένοι ακέραιοι, με $k \geq 1$ και $\ell \geq 1$.**

---

for $i \leftarrow 0$ **to** $k + \ell - 1$
    $c_i \leftarrow 0$
for $i \leftarrow 0$ **to** $k - 1$
    $carry \leftarrow 0$
    for $j \leftarrow 0$ **to** $\ell - 1$
        $tmp \leftarrow a_i b_j + c_{i+j} + carry$ // $0 \leq tmp \leq B^2 - 1$
        $carry \leftarrow \lfloor tmp/B \rfloor$ // $0 \leq carry \leq B - 1$
        $c_{i+j} \leftarrow tmp \bmod B$
    $c_{i+\ell} \leftarrow carry$

- **Το γινόμενο $c := a \cdot b$ είναι της μορφής $(c_{k+\ell-1}\ldots c_0)_B$ και μπορεί να υπολογιστεί σε χρόνο $O(k\ell)$.**

Πολλαπλασιασμός μεγάλων ακεραίων

- **Κλασσικός πολλαπλασιασμός: πολλαπλασιασμός δύο *n*-ψήφιων ακεραίων**
  - **χρόνος εκτέλεσης $T(n) = O(n^2)$.**
- **"Διαίρει και βασίλευε": πολλαπλασιασμός δύο *n*-ψήφιων ακεραίων**
  - **χρόνος εκτέλεσης $T(n) = \Theta(n^{\lg 3})$.**
- **Πράγματι, ας ξεκινήσουμε με ένα παράδειγμα:**
- **$23 = 2 \cdot 10^1 + 3 \cdot 10^0$**
- **$14 = 1 \cdot 10^1 + 4 \cdot 10^0$**
- **$23 * 14 = (2 \cdot 10^1 + 3 \cdot 10^0) * (1 \cdot 10^1 + 4 \cdot 10^0)$**
  $= (2*1) \cdot 10^2 + (3*1 + 2*4) \cdot 10^1 + (3*4) \cdot 10^0$
  - **4 πολλαπλασιασμοί ψηφίων**
- **Αλλά,**
- **$3*1 + 2*4 = (2 + 3) * (1 + 4) - (2*1) - (3*4)$**

(1 επιπλέον πολλαπλασιασμός)

- **Μια πρώτη γενίκευση του παραδείγματος είναι η ακόλουθη:**

$$a = a_1 a_0 \quad | \quad c = a * b = c_2 10^2 + c_1 10^1 + c_0$$
$$b = b_1 b_0$$

- **όπου:**
- $c_2 = a_1 * b_1$ **(= γινόμενο των πρώτων τους ψηφίων)**
- $c_0 = a_0 * b_0$ **(= γινόμενο των δεύτερών τους ψηφίων)**
- $c_1 = (a_1 + a_0) * (b_1 + b_0) - (c_2 + c_0)$

  (= γινόμενο των αθροισμάτων των ψηφίων τους μείον το άθροισμα των c_2 και c_0)
- **Έστω τώρα ότι πολλαπλασιάζουμε δύο *n*-ψήφιους ακεραίους *a* και *b*, όπου *n* είναι άρτιος. Χωρίζουμε τα ψηφία τους στη μέση και έστω ότι συμβολίζουμε**

$$a = a_1 a_0 = a_1 10^{n/2} + a_0$$
$$b = b_1 b_0 = b_1 10^{n/2} + b_0$$

- **οπότε**

$$c = a * b = \left(a_1 10^{n/2} + a_0\right) * \left(b_1 10^{n/2} + b_0\right)$$
$$= \left(a_1 * b_1\right) 10^n + \left(a_1 * b_0 + a_0 * b_1\right) 10^{n/2} + a_0 * b_0$$
$$= c_2 10^n + c_1 10^{n/2} + c_0$$

- όπου
- $c_2 = a_1 * b_1$ (= γινόμενο των πρώτων μισών τους)
- $c_0 = a_0 * b_0$ (= γινόμενο των δεύτερων μισών τους)
- $c_1 = (a_1 + a_0) * (b_1 + b_0) - (c_2 + c_0)$
- Αν $n/2$ = άρτιος, μπορούμε να εφαρμόσουμε την ίδια μέθοδο για τον υπολογισμό των γινομένων c_2, c_0, και c_1. Έτσι, αν n = δύναμη του 2, έχουμε έναν αναδρομικό αλγόριθμο για τον υπολογισμό του γινομένου δύο n-ψήφιων ακεραίων. Η αναδρομή σταματά όταν ο n γίνει 1 ή όταν ο n αρκετά μικρός ώστε να είναι εφικτός ο πολλαπλασιασμός άμεσα.
- Ας βρούμε τώρα πόσους πολλαπλασιασμούς κάνει αυτός ο αλγόριθμος. Επειδή ο πολλαπλασιασμός των n-ψήφιων ακεραίων απαιτεί 3 πολλαπλασιασμούς $(n/2)$-ψήφιων ακεραίων, έχουμε

$$\begin{cases} M(n) = 3M(n/2), n \geq 2 \\ M(1) = 1 \end{cases}$$

- Για $n = 2^k \Rightarrow M(2^k) = 3\, M(2^{k-1})$ και αν θέσουμε $M(2^k) = a_k$, προκύπτει η αναδρομή $a_k = 3a_{k-1}$, $k \geq 1$. Για $n = 1 \Rightarrow k = 0$, και $M(1) = M(2) = a_0 = 1$, οπότε τελικά έχουμε την αναδρομική σχέση

$$a_k = \begin{cases} 1, & k = 0 \\ 3a_{k-1}, & k \geq 1 \end{cases}$$

- με λύση, $a_k = a_0 3^k = 3^k$. Επειδή τώρα, $n = 2^k \Leftrightarrow k = \lg n$, παίρνουμε,
- $M(n) = 3^{\lg n} = n^{\lg 3}$ (ισχύει $a^{\log_b c} = c^{\log_b a}$) (είναι, $\lg 3 \approx 1.585 < 2$)

- Παίρνοντας υπόψη και τις προσθέσεις τότε για τον συνολικό χρόνο θα έχουμε την αναδρομή

$$T(n) = 3\,T(n/2) + \Theta(n)$$

- Με βάση το Κύριο Θεώρημα (Master Theorem) έχουμε

$$a = 3,\ b = 2 \Rightarrow E = \log_2 3 = \lg 3$$

- Επίσης, $f(n) = \Theta(n) \Rightarrow f(n) = O(n^{E-\varepsilon})$ (περίπτωση 1)

(είναι, $E - \varepsilon = 1 \Leftrightarrow \varepsilon = E - 1 = \lg 3 - 1 > 0$). Συνεπώς,

- $T(n) = \Theta(n^E) = \Theta(n^{\lg 3})$.

Δυαδική ύψωση σε δύναμη («Εκθετοποίηση»)

- **Ένας αποδοτικός αλγόριθμος ύψωσης σε δύναμη προκύπτει αν στηριχθούμε στην ιδέα που είχε ο Horner για τον υπολογισμό της τιμής ενός πολυωνύμου.**
- **Γενικά, έχουμε τον εξής μετασχηματισμό ενός πολυωνύμου**

$$P(x) = ax^n + a_{n-1}x^{n-1} + \cdots + a_1 x + a_0 = (\cdots(a_n x + a_{n-1})x + \cdots)x + a_0$$

όπου μπορούμε να παρατηρήσουμε ότι το πλήθος των πολλαπλασιασμών μειώνεται δραστικά, αποφεύγοντας τους υπολογισμούς των δυνάμεων του *x*.

- **Η εφαρμογή της μεθόδου είναι άμεση στον ακόλουθο αλγόριθμο:**
- **Αλγόριθμος *Horner*(*P*[0..*n*], *x*)**
- // Είσοδος: ένας πίνακας γραμμή *P*[0..*n*] των συντελεστών ενός πολυωνύμου βαθμού *n*,
  // αποθηκευμένων σε αύξουσα σειρά βαθμού και ένας αριθμός *x*
- //Έξοδος: η τιμή του πολυωνύμου στο σημείο *x*
- $p \leftarrow P[n]$
- **for** $1 \leftarrow n - 1$ **downto** 0
- $p \leftarrow x*p + P[i]$
- **return** *p*

- Έστω τώρα $n = b_l \ldots b_i \ldots b_0$ η δυαδική συμβολοσειρά που αναπαριστά τον θετικό ακέραιο n. Αυτό σημαίνει ότι η τιμή του n μπορεί να υπολογιστεί ως η τιμή του πολυωνύμου

$$p(x) = b_\ell x^\ell + \cdots + b_i x^i + \cdots + b_0 x^0$$

στο $x = 2$. Π.χ, αν $n = 13$, η δυαδική αναπαράστασή του είναι 1101, αφού

$$13 = 1 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0$$

- Ας υπολογίσουμε την τιμή του πολυωνύμου $p(x)$ εφαρμόζοντας τον κανόνα του Horner και ας δούμε πως μεταφέρονται οι υπολογισμοί στην περίπτωση υπολογισμού της δύναμης

$$a^n = a^{p(2)} = a^{b_\ell 2^\ell + \cdots + b_i 2^i + \cdots + b_0 2^0}$$

Κανόνας του Horner για το πολυώνυμο $p(2)$

$p \leftarrow 1$ // ο μεγιστοβάθμιος συντελεστής
// είναι πάντα 1 για $n \geq 1$

for $i \leftarrow \ell - 1$ **downto** 0

do $p \leftarrow 2p + b_i$

Εφαρμογή στον υπολογισμό της $a^n = a^{p(2)}$

$a^p \leftarrow a^1$

for $i \leftarrow \ell - 1$ **downto** 0

do $a^p \leftarrow a^{2p+b_i}$

$$a^{2p+b_i} = a^{2p} \cdot a^{b_i} = (a^p)^2 \cdot a^{b_i} = \begin{cases} (a^p)^2, & \text{αν } b_i = 0 \\ (a^p)^2 \cdot a, & \text{αν } b_i = 1 \end{cases}$$

- **Έτσι οδηγούμαστε στον ακόλουθο αλγόριθμο:**
- ***Αλγόριθμος*** *ΔυαδικήΕκθετοποίησηL_R(a, b(n))*
- // Υπολογίζει την δύναμη a^n με ύψωση από αριστερά προς τα δεξιά στη δυαδική // αναπαράσταση
- // Είσοδος: Ένας αριθμός *a* και μια λίστα *b*(*n*) δυαδικών ψηφίων b_ℓ, …, b_0 στο // δυαδικό ανάπτυγμα ενός θετικού ακεραίου *n*
- // Έξοδος: Η τιμή της δύναμης a^n

1. product ← *a*
2. **for** *i* ← *l* – 1 **downto** 0
3. $\begin{cases} \text{product} \leftarrow \text{product} * \text{product} \\ \textbf{if } b_i = 1 \quad \text{product} \leftarrow \text{product} * a \end{cases}$
4. **return** product

- ***Παράδειγμα*: Ας δούμε πως εκτελεί ο αλγόριθμος αυτός τον υπολογισμό της δύναμης a^{13}.**

- **Είναι $n = 13 = 1101_2$, οπότε έχουμε**

i	3	2	1	0
b_i	1	1	0	1
γινόμενα	a	$a^2 \cdot a = a^3$	$(a^3)^2 = a^6$	$(a^6)^2 \cdot a = a^{13}$

- ***Ανάλυση:***
- **Επειδή ο αλγόριθμος εκτελεί έναν ή δύο πολλαπλασιασμούς σε κάθε επανάληψη του βρόχου, για το συνολικό πλήθος των πολλαπλασιασμών $M(n)$ που γίνονται κατά τον υπολογισμό της δύναμης a^n, είναι**

$$(b - 1) \leq M(n) \leq 2\,(b - 1)$$

- **Όπου b είναι το μήκος της δυαδικής συμβολοσειράς που αναπαριστά τον εκθέτη n. Λαμβάνοντας υπόψη ότι b – 1 =lgn συμπεραίνουμε ότι η αποδοτικότητα του αλγόριθμου είναι λογαριθμική.**

- **Μπορούμε αντί να εφαρμόσουμε τον κανόνα του Horner για το $p(2) = n$, να χρησιμοποιήσουμε το γεγονός ότι**

$$a^n = a^{b_\ell 2^\ell + \cdots + b_i 2^i + \cdots + b_0 2^0} = a^{b_\ell 2^\ell} \ldots\ldots a^{b_i 2^i} \ldots\ldots a^{b_0}$$

- **Έτσι η δύναμη a^n μπορεί να υπολογιστεί ως το γινόμενο των όρων**

$$a^{b_i 2^i} = \begin{cases} a^{2^i}, & \text{αν } b_i = 1 \\ 1, & \text{αν } b_i = 0 \end{cases}$$

- **Δηλ. ως το γινόμενο διαδοχικών όρων της μορφής a^{2^i} παραλείποντας εκείνους για τους οποίους το δυαδικό ψηφίο b_i είναι μηδέν. Επιπλέον μπορούμε να υπολογίσουμε το a^{2^i} υψώνοντας απλώς στο τετράγωνο τον ίδιο όρο που υπολογίσαμε για την προηγούμενη τιμή του i διότι $a^{2^i} = (a^{2^{i-1}})^2$. Ετσι υπολογίζουμε όλες τις δυνάμεις του a από τη μικρότερη στη μεγαλύτερη (από δεξιά στα αριστερά) αλλά συμπεριλαμβάναμε στα γινόμενα μόνον εκείνες των οποίων το αντίστοιχο δυαδικό ψηφίο είναι 1.**

Αλγόριθμος *ΔυαδικήΕκθετοποίησηR_L*$(a, b(n))$

term $\leftarrow a$ // αρχικοποίηση

if $b_0 = 1$ product $\leftarrow a$

else product $\leftarrow 1$

for $i \leftarrow 1$ to ℓ

do {
term $\leftarrow$ term * term
if $b_i = 1$ product $\leftarrow$ product * term

return product

Η αποδοτικότητα αυτού του αλγόριθμου είναι ίδια με αυτή του προηγούμενου.

- ***Παράδειγμα:***

Ας δούμε πως εκτελεί ο αλγόριθμος αυτός τον υπολογισμό της δύναμης a^{13}. Είναι πάλι $13 = 1101_2$, οπότε έχουμε τον ακόλουθο πίνακα που τον συμπληρώνουμε από δεξιά στα αριστερά

3	2	1	0	i
1	1	0	1	b_i
a^8	a^4	a^2	a	a^{2^i}
$a^5 \cdot a^8 = a^{13}$	$a \cdot a^4 = a^5$		a	γινόμενα

Πολλαπλασιασμός τετραγωνικών πινάκων

- Έστω A = [a_{ij}] και B = [b_{ij}] δύο *n* x *n* πίνακες. Το γινόμενο των A και B, AB = C = [c_{ij}], είναι ο *n* x *n* πίνακας που ορίζεται από την

$$c_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj} = a_{i1} b_{1j} + a_{i2} b_{2j} + \cdots + a_{in} b_{nj}$$

$$\text{για} \quad i = 1, 2, \ldots, n \quad j = 1, 2, \ldots, n$$

- **Επαναληπτικός Αλγόριθμος**
- Αλγόριθμος MatMul(*A*[0..*n*–1, 0..*n*–1], *B*[0..*n*–1, 0..*n*–1])
- // Είσοδος: Δύο *n*×*n* πίνακες *A*, *B*
- //Έξοδος: Πίνακας *C* = *AB*

```
1.  for i ← 0 to n – 1
2.      for j ← 0 to n – 1
3.          C[i,j] ← 0
4.          for k ← 0 to n – 1
5.              C[i,j] ← C[i,j] + A[i,k]*B[k,j]
6.  return C
```

- **Παράμετρος το *n* (= μέγεθος πινάκων)**
- **Βασική πράξη ο πολλαπλασιασμός (και η πρόσθεση πάει μαζί). Πλήθος πολλαπλασιασμών:**

$$M(n) = \sum_{i=0}^{n-1}\sum_{j=0}^{n-1}\sum_{k=0}^{n-1} 1 = \sum_{i=0}^{n-1}\sum_{j=0}^{n-1} n = \sum_{i=0}^{n-1} n^2 = n^3$$

- **$\Rightarrow T(n) = n^3\Theta(1) = \Theta(n^3)$.**
- **Αναδρομικός Αλγόριθμος (Strassen)**

$$\begin{bmatrix} c_{00} & c_{01} \\ c_{10} & c_{11} \end{bmatrix} = \begin{bmatrix} a_{00} & a_{01} \\ a_{10} & a_{11} \end{bmatrix} * \begin{bmatrix} b_{00} & b_{01} \\ b_{10} & b_{11} \end{bmatrix} = \begin{bmatrix} m_1 + m_4 - m_5 + m_7 & m_3 + m_5 \\ m_2 + m_4 & m_1 + m_3 - m_2 + m_6 \end{bmatrix}$$

- **όπου**
  - $m_1 = (a_{00} + a_{11})*(b_{00} + b_{11})$
  - $m_2 = (a_{10} + a_{11})*b_{00}$
  - $m_3 = a_{00} *(b_{00} - b_{11})$
  - $m_4 = a_{11} *(b_{10} - b_{00})$
  - $m_5 = (a_{00} + a_{01})*b_{11}$
  - $m_6 = (a_{10} - a_{00})*(b_{00} + b_{01})$
  - $m_7 = (a_{01} - a_{11})*(b_{10} + b_{11})$

7 πολλαπλασιασμοί

18 προσθαφαιρέσεις

- **Διαίρει**

  Εστω $n = 2^k$ θεωρούμε ότι οι πίνακες Α και Β αποτελείται ο καθένας από 4 υποπίνακες μεγέθους n/2 x n/2, οπότε:

$$\left[\begin{array}{c|c} C_{00} & C_{01} \\ \hline C_{10} & C_{11} \end{array}\right] = \left[\begin{array}{c|c} A_{00} & A_{01} \\ \hline A_{10} & A_{11} \end{array}\right] * \left[\begin{array}{c|c} B_{00} & B_{01} \\ \hline B_{10} & B_{11} \end{array}\right]$$

- όπου, $C_{00} = M_1 + M_4 - M_5 + M_7$ … και έχουμε τώρα πίνακες $\frac{n}{2} \times \frac{n}{2}$
- Αν τα 7 γινόμενα των $(n/2)\times(n/2)$ πινάκων υπολογιστούν αναδρομικά με την ίδια μέθοδο, προκύπτει ο αλγόριθμος του Strassen για τον πολλαπλασιασμό πινάκων.
- Ανάλυση του αλγορίθμου:
- Έστω $M(n)$ ο αριθμός των πολλαπλασιασμών. Τότε, είναι
  - $M(n) = 7\,M(n/2),\ n \geq 2$
  - $M(1) = 1$
- Θεωρούμε $n = 2^k \Rightarrow M(2^k) = 7M(2^{k-1}) = 7[7M(2^{k-2})] = 7^2M(2^{k-2}) = \ldots = 7^iM(2^{k-i}) = \ldots$
  $= \ldots = 7^kM(2^{k-k}) = 7^k$
- $\Rightarrow a_k = 7a_{k-1},\ a_0 = 1$
- $\Rightarrow a_k = 7^k$
- και επειδή $n = 2^k \Leftrightarrow k = \lg n$
- $\Rightarrow M(n) = 7^{\lg n} = n^{\lg 7} \approx n^{2.807} < n^3$ (ισχύει $a^{\log_b c} = c^{\log_b a}$)

- Παίρνοντας υπόψη και τις προσθαφαιρέσεις, έχουμε
  - $A(n) = 7A(n/2) + 18(n/2)^2, \ n > 1$
  - $A(1) = 0$
- και από το Κύριο Θεώρημα $\Rightarrow A(n) = \Theta(n^{\lg 7})$ (περίπτωση 1).
- Παρατήρηση: Θεωρήσαμε γνωστό ότι για την πρόσθεση δύο τετραγωνικών $m \times m$ πινάκων ο χρόνος εκτέλεσης του αλγορίθμου ασυμπτωτικά είναι $\Theta(m^2)$. Πράγματι, το πρόγραμμα για την πρόσθεση δύο $m \times m$ πινάκων είναι στην ουσία ο διπλός βρόχος

```
1.  for i ← 1 to m
2.      for j ← 1 to m
3.          C[i,j] ← A[i,j] + B[i,j];
```

- Άρα το πλήθος των προσθέσεων (βασική πράξη) είναι

$$\sum_{i=1}^{m} \sum_{j=1}^{m} 1 = \sum_{i=1}^{m} m = mm = m^2$$

Τέλος Ενότητας