Αλγόριθμοι και Πολυπλοκότητα

Διαίρει και Βασίλευε

Δημήτρης Μιχαήλ

Τμήμα Πληροφορικής και Τηλεματικής
Χαροκόπειο Πανεπιστήμιο

Διαίρει και Βασίλευε

Divide and Conquer

Η τεχνική *διαίρει και βασίλευε* αναφέρεται σε μια κατηγορία αλγοριθμικών τεχνικών όπου:

1. ”διασπούμε” την είσοδο σε αρκετά τμήματα
2. επιλύουμε το πρόβλημα σε κάθε τμήμα αναδρομικά
3. συνδυάζουμε τις λύσεις αυτών των υποπροβλημάτων σε μια συνολική λύση

Διαίρει και Βασίλευε

Divide and Conquer

Η τεχνική *διαίρει και βασίλευε* αναφέρεται σε μια κατηγορία αλγοριθμικών τεχνικών όπου:

1. ”διασπούμε” την είσοδο σε αρκετά τμήματα
2. επιλύουμε το πρόβλημα σε κάθε τμήμα αναδρομικά
3. συνδυάζουμε τις λύσεις αυτών των υποπροβλημάτων σε μια συνολική λύση

Η ανάλυση του χρόνου εκτέλεσης ενός αλγορίθμου τύπου ”διαίρει και βασίλευε” γενικά περιλαμβάνει την επίλυση μιας *αναδρομικής σχέσης* (recurrence relation), η οποία φράσσει αναδρομικά το χρόνο εκτέλεσης με βάση τους χρόνους εκτέλεσης μικρότερων στιγμιοτύπων.

Πρώτη Αναδρομή: MergeSort

Ο αλγόριθμος ταξινόμησης *MergeSort* ταξινομεί μια δεδομένη λίστα αριθμών με τον εξής τρόπο:

1. πρώτα χωρίζει την είσοδο σε δύο ίσα τμήματα,
2. ταξινομεί τα τμήματα αναδρομικά,
3. συγχωνεύει τα δυο ταξινομημένα υποπροβλήματα σε ένα.

Πρώτη Αναδρομή: MergeSort

Ο αλγόριθμος ταξινόμησης *MergeSort* ταξινομεί μια δεδομένη λίστα αριθμών με τον εξής τρόπο:

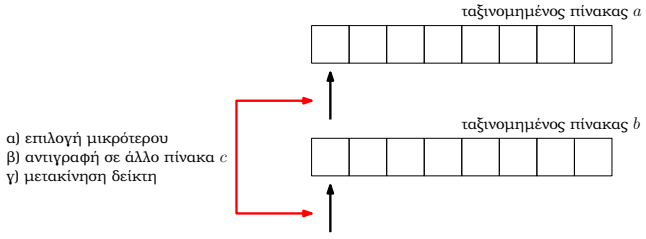
1. πρώτα χωρίζει την είσοδο σε δύο ίσα τμήματα,
2. ταξινομεί τα τμήματα αναδρομικά,
3. συγχωνεύει τα δυο ταξινομημένα υποπροβλήματα σε ένα.

Σε κάθε αλγόριθμο ”διαίρει και βασίλευε” χρειαζόμαστε μια βασική περίπτωση για την αναδρομή.

Στο MergeSort π.χ σταματάμε την αναδρομή όταν έχουμε 2 στοιχεία. Απλά τα συγκρίνουμε και τα επιστρέφουμε στη σωστή σειρά.

Συγχώνευση

Συγχώνευση είναι η διαδικασία συνδυασμού δύο ταξινομημένων αρχείων σε ένα ταξινομημένο αρχείο.



Για εισόδους μεγέθους n γίνεται σε χρόνο $\mathcal{O}(n)$.

Έστω $T(n)$ ο χρόνος εκτέλεσης της χειρότερης περίπτωσης για στιγμιότυπα μεγέθους n. [1]

Ο αλγόριθμος δαπανά το πολύ χρόνο

- $\mathcal{O}(n)$ για να διαιρέσει την είσοδο σε δύο τμήματα μεγέθους $n/2$ το καθένα
- $T(n/2)$ για να επιλύσει το κάθε τμήμα
- $\mathcal{O}(n)$ για να συνδυάσει τις λύσεις από τις δύο αναδρομικές κλήσεις.

[1] Ας υποθέσουμε προς το παρόν πως n είναι άρτιος.

Έστω $T(n)$ ο χρόνος εκτέλεσης της χειρότερης περίπτωσης για στιγμιότυπα μεγέθους n. [1]

Άρα ο χρόνος εκτέλεσης $T(n)$ ικανοποιεί την ακόλουθη **αναδρομική σχέση** (recurrence relation).

Για κάποια σταθερά c,

$$T(n) \leq 2T(n/2) + cn$$

όταν $n > 2$, και

$$T(n) \leq cn.$$

[1] Ας υποθέσουμε προς το παρόν πως n είναι άρτιος.

Λύση Αναδρομής

Υπόθεση Άρτιας Εισόδου. Γενικά είναι ανακριβές να υποθέσουμε πως το μέγεθος της εισόδου n είναι άρτιος ή είναι δύναμη του 2.

Ακριβής Αναδρομή. Για κάποια σταθερά c,

$$T(n) \leq T(\lceil n/2 \rceil) + T(\lfloor n/2 \rfloor) + cn$$

για $n \geq 2$.

Για τις αναδρομές που θα μας απασχολήσουν, τα ασυμπτωτικά όρια δεν επηρεάζονται παραβλέποντας τα άνω και κάτω ακέραια μέρη.

Τεχνικές Λύσης Αναδρομής

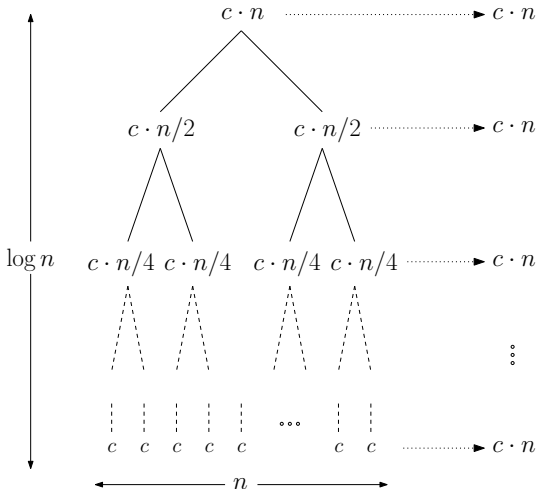
Ξεδιπλώνοντας.

1. ξεδιπλώνουμε την αναδρομή (unroll) υπολογίζοντας τον χρόνο εκτέλεσης για τα πρώτα λίγα επίπεδα
2. αναγνωρίζουμε ένα μοτίβο (pattern) που μπορεί να συνεχιστεί
3. αθροίζουμε τους χρόνους εκτέλεσης για όλα τα επίπεδα της αναδρομής

Τεχνικές Λύσης Αναδρομής

Ανάλυση Πρώτων Επιπέδων.

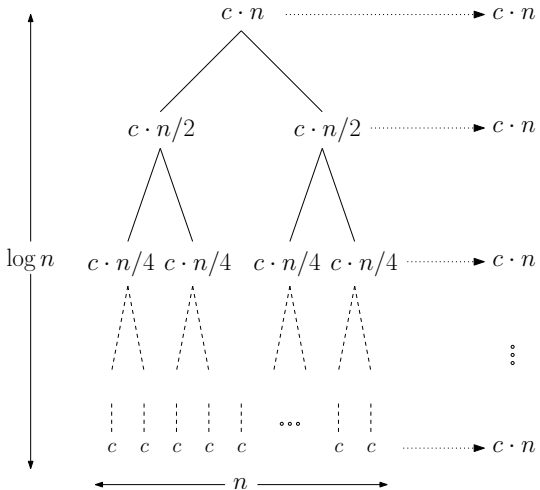
- πρώτο επίπεδο έχουμε ένα υποπρόβλημα μεγέθους n το οποίο χρειάζεται cn χρόνο συν το χρόνο των αναδρομικών κλήσεων
- δεύτερο επίπεδο δύο προβλήματα, το καθένα μεγέθους $n/2$. Καθένα από αυτά χρειάζεται χρόνο το πολύ $cn/2$. Συνολικά χρόνο cn συν το χρόνο των αναδρομικών κλήσεων.
- τρίτο επίπεδο τέσσερα προβλήματα, το καθένα μεγέθους $n/4$. Καθένα από αυτά χρειάζεται χρόνο το πολύ $cn/4$. Συνολικά χρόνο cn συν το χρόνο των αναδρομικών κλήσεων.



Τεχνικές Λύσης Αναδρομής

Αναγνώριση Μοτίβου.

- Στο επίπεδο *j* της αναδρομής ο αριθμός των υποπροβλημάτων έχει διπλασιαστεί *j* φορές και άρα έχουμε 2^j υποπροβλήματα.
- Καθένα από αυτά έχει συρρικνωθεί σε μέγεθος κατά 2 επίσης *j* φορές. Άρα κάθε υποπρόβλημα έχει μέγεθος $n/2^j$. Χρειάζεται λοιπόν χρόνο το πολύ $cn/2^j$.
- Συνολικά όλα τα υποπροβλήματα του επιπέδου *j* χρειάζονται χρόνο το πολύ $2^j(cn/2^j) = cn$.



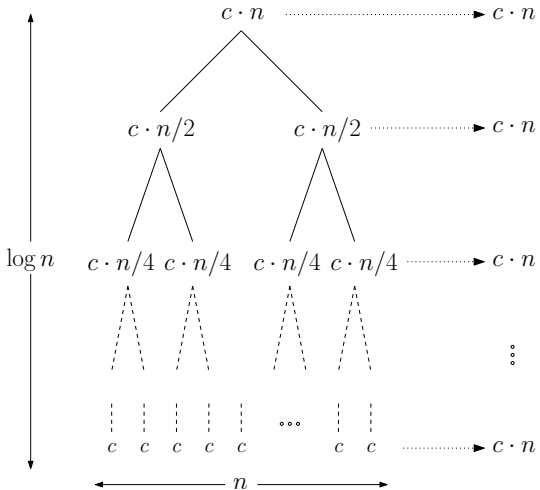
Τεχνικές Λύσης Αναδρομής

Άθροιση Επιπέδων.

- Ο αριθμός των υποδιπλασιασμών της εισόδου έτσι ώστε να μειωθεί το μέγεθος της από n σε 2 είναι $\log_2 n - 1$ αφού ισχύει

$$\frac{n}{2^k} = 2 \Rightarrow n = 2^{k+1} \Rightarrow \log_2 n = k + 1.$$

- Η άθροιση της εργασίας cn για $\log n$ επίπεδα μας δίνει συνολικό χρόνο εκτέλεσης $\mathcal{O}(n \log n)$.



Τεχνικές Λύσης Αναδρομής

Μαντεύοντας!

1. ξεκινάμε με μια πρόβλεψη για τη λύση
2. αντικαθιστούμε την πρόβλεψη στην αναδρομική σχέση και ελέγχουμε αν λειτουργεί
3. αιτιολογούμε αυτή την αντικατάσταση χρησιμοποιώντας ένα επιχείρημα επαγωγής ως προς το n

Τεχνικές Λύσης Αναδρομής

Υποθέστε ότι πιστεύουμε ότι $T(n) \leq cn \log_2 n$ για όλα τα $n \geq 2$.

Θέλουμε να το επαληθεύσουμε με την μέθοδο της επαγωγής.

- για $n = 2$ έχουμε $cn \log_2 n = 2c$ και άρα ισχύει αφού ξέρουμε πως $T(2) \leq c$.
- έστω τώρα ότι $T(m) \leq cm \log_2 m$ για όλες τις τιμές του $m < n$
- για n έχουμε πως

$$\begin{aligned} T(n) &\leq 2T(n/2) + cn \\ &\leq 2c(n/2)\log_2(n/2) + cn \\ &= cn(\log_2 n - 1) + cn \\ &= cn \log_2 n \end{aligned}$$

Πολλαπλασιασμός Ακεραίων

Η μέθοδος του σχολείου

Έστω δύο ακέραιοι αριθμοί x και y οι οποίοι αποτελούνται από n bits. Πόσο χρόνο χρειάζεται ο αλγόριθμος του σχολείου για να πολλαπλασιάσει τους x και y;

Πολλαπλασιασμός Ακεραίων

Η μέθοδος του σχολείου

Έστω δύο ακέραιοι αριθμοί x και y οι οποίοι αποτελούνται από n bits. Πόσο χρόνο χρειάζεται ο αλγόριθμος του σχολείου για να πολλαπλασιάσει τους x και y;

Υποθέστε πως θέλουμε να εκτελέσουμε τον πολλαπλασιασμό 13×11, ή σε δυαδικό $x = 1101$ και $y = 1011$.

				1	1	0	1	
			$\times$	1	0	1	1	
				1	1	0	1	(1101 επί 1)
			1	1	0	1		(1101 επί 1, μετατοπισμένο μια φορά)
		0	0	0	0			(1101 επί 0, μετατοπισμένο δύο φορές)
+	1	1	0	1				(1101 επί 1, μετατοπισμένο τρείς φορές)
1	0	0	0	1	1	1	1	(δυαδικό 143)

Έστω δύο ακέραιοι αριθμοί x και y οι οποίοι αποτελούνται από n bits. Πόσο χρόνο χρειάζεται ο αλγόριθμος του σχολείου για να πολλαπλασιάσει τους x και y;

Υποθέστε πως θέλουμε να εκτελέσουμε τον πολλαπλασιασμό 13×11, ή σε δυαδικό $x = 1101$ και $y = 1011$.

				1	1	0	1	
			$\times$	1	0	1	1	
				1	1	0	1	(1101 επί 1)
			1	1	0	1		(1101 επί 1, μετατοπισμένο μια φορά)
		0	0	0	0			(1101 επί 0, μετατοπισμένο δύο φορές)
+	1	1	0	1				(1101 επί 1, μετατοπισμένο τρείς φορές)
1	0	0	0	1	1	1	1	(δυαδικό 143)

Υπάρχουν n ενδιάμεσες γραμμές με μήκη μέχρι $2n$ bit (με την μετατόπιση). Ο συνολικός χρόνος είναι λοιπόν $\mathcal{O}(n^2)$.

Πολλαπλασιασμός Ακεραίων Αναδρομικά

Λόγω απλότητας υποθέστε πως το n είναι δύναμη του 2. Ας χωρίσουμε τους ακεραίους x και y στο αριστερό και δεξιό τους μέρος.

$$x = \boxed{x_L}\boxed{x_R} = 2^{n/2}x_L + x_R$$
$$y = \boxed{y_L}\boxed{y_R} = 2^{n/2}y_L + y_R$$

Πολλαπλασιασμός Ακεραίων Αναδρομικά

Λόγω απλότητας υποθέστε πως το n είναι δύναμη του 2. Ας χωρίσουμε τους ακεραίους x και y στο αριστερό και δεξιό τους μέρος.

$$x = \boxed{x_L \mid x_R} = 2^{n/2} x_L + x_R$$
$$y = \boxed{y_L \mid y_R} = 2^{n/2} y_L + y_R$$

Για παράδειγμα, αν $x = 10110110_2$ τότε

- $x_L = 1011_2$
- $x_R = 0110_2$

Πολλαπλασιασμός Ακεραίων Αναδρομικά

Λόγω απλότητας υποθέστε πως το n είναι δύναμη του 2. Ας χωρίσουμε τους ακεραίους x και y στο αριστερό και δεξιό τους μέρος.

$$x = \boxed{x_L}\boxed{x_R} = 2^{n/2}x_L + x_R$$
$$y = \boxed{y_L}\boxed{y_R} = 2^{n/2}y_L + y_R$$

Το γινόμενο xy γράφετε ως

$$xy = (2^{n/2}x_L + x_R)(2^{n/2}y_L + y_R) = 2^n x_L y_L + 2^{n/2}(x_L y_R + x_R y_L) + x_R y_R$$

Πολλαπλασιασμός Ακεραίων Αναδρομικά

$$x = \boxed{x_L \mid x_R} = 2^{n/2} x_L + x_R$$
$$y = \boxed{y_L \mid y_R} = 2^{n/2} y_L + y_R$$

$$xy = 2^n x_L y_L + 2^{n/2}(x_L y_R + x_R y_L) + x_R y_R$$

1. οι προσθέσεις χρειάζονται γραμμικό χρόνο
2. οι πολλαπλασιασμοί με δυνάμεις του 2 επίσης γραμμικό χρόνο μιας και είναι απλά μετατοπίσεις προς τα αριστερά
3. οι πολλαπλασιασμοί μπορούν να γίνουν αναδρομικά

Πολλαπλασιασμός Ακεραίων Αναδρομικά

$$x = \boxed{x_L}\boxed{x_R} = 2^{n/2}x_L + x_R$$
$$y = \boxed{y_L}\boxed{y_R} = 2^{n/2}y_L + y_R$$

$$xy = 2^n x_L y_L + 2^{n/2}(x_L y_R + x_R y_L) + x_R y_R$$

Ο αναδρομικός αλγόριθμος πολλαπλασιασμού αριθμών n bits

- υπολογίζει τα γινόμενα $x_L y_L$, $x_L y_R$, $x_R y_L$ και $x_R y_R$
- έπειτα αποτιμά την παραπάνω παράσταση σε χρόνο $\mathcal{O}(n)$

Πολλαπλασιασμός Ακεραίων Αναδρομικά

$$x = \boxed{x_L}\boxed{x_R} = 2^{n/2}x_L + x_R$$
$$y = \boxed{y_L}\boxed{y_R} = 2^{n/2}y_L + y_R$$

$$xy = 2^n x_L y_L + 2^{n/2}(x_L y_R + x_R y_L) + x_R y_R$$

Ο αναδρομικός αλγόριθμος πολλαπλασιασμού αριθμών n bits

- υπολογίζει τα γινόμενα $x_L y_L$, $x_L y_R$, $x_R y_L$ και $x_R y_R$
- έπειτα αποτιμά την παραπάνω παράσταση σε χρόνο $\mathcal{O}(n)$

Εαν $T(n)$ είναι ο χρόνος στη χειρότερη περίπτωση, έχουμε

$$T(n) \leq 4T(n/2) + \mathcal{O}(n)$$

Αναδρομή $T(n) \leq 4T(n/2) + \mathcal{O}(n)$

1. Ας υποθέσουμε για λόγους απλότητας πως το n είναι δύναμη του 2.

Αναδρομή $T(n) \leq 4T(n/2) + \mathcal{O}(n)$

1. Ας υποθέσουμε για λόγους απλότητας πως το n είναι δύναμη του 2.
2. Το μέγεθος των υποπροβλημάτων μειώνεται κατά ένα παράγοντα 2 σε κάθε επίπεδο αναδρομής και επομένως φτάνει στη βασική περίπτωση μετά από $\log_2 n$ επίπεδα.

Αναδρομή $T(n) \leq 4T(n/2) + \mathcal{O}(n)$

1. Ας υποθέσουμε για λόγους απλότητας πως το n είναι δύναμη του 2.
2. Το μέγεθος των υποπροβλημάτων μειώνεται κατά ένα παράγοντα 2 σε κάθε επίπεδο αναδρομής και επομένως φτάνει στη βασική περίπτωση μετά από $\log_2 n$ επίπεδα.
3. Ο παράγοντας διακλάδωσης είναι 4 οπότε στο j-οστό επίπεδο του δέντρου έχουμε 4^j υποπροβλήματα, το καθένα μεγέθους $n/2^j$.

Αναδρομή $T(n) \leq 4T(n/2) + \mathcal{O}(n)$

1. Ας υποθέσουμε για λόγους απλότητας πως το n είναι δύναμη του 2.
2. Το μέγεθος των υποπροβλημάτων μειώνεται κατά ένα παράγοντα 2 σε κάθε επίπεδο αναδρομής και επομένως φτάνει στη βασική περίπτωση μετά από $\log_2 n$ επίπεδα.
3. Ο παράγοντας διακλάδωσης είναι 4 οπότε στο j-οστό επίπεδο του δέντρου έχουμε 4^j υποπροβλήματα, το καθένα μεγέθους $n/2^j$.
4. Άρα στο j επίπεδο έχουμε συνολικό κόστος

$$4^j \times \mathcal{O}(\frac{n}{2^j}) = 2^j \times \mathcal{O}(n)$$

Αναδρομή $T(n) \leq 4T(n/2) + \mathcal{O}(n)$

1. Ας υποθέσουμε για λόγους απλότητας πως το n είναι δύναμη του 2.
2. Το μέγεθος των υποπροβλημάτων μειώνεται κατά ένα παράγοντα 2 σε κάθε επίπεδο αναδρομής και επομένως φτάνει στη βασική περίπτωση μετά από $\log_2 n$ επίπεδα.
3. Ο παράγοντας διακλάδωσης είναι 4 οπότε στο j-οστό επίπεδο του δέντρου έχουμε 4^j υποπροβλήματα, το καθένα μεγέθους $n/2^j$.
4. Άρα στο j επίπεδο έχουμε συνολικό κόστος

$$4^j \times \mathcal{O}(\frac{n}{2^j}) = 2^j \times \mathcal{O}(n)$$

5. Συνολικά

$$\sum_{j=0}^{\log_2 n} 2^j \times \mathcal{O}(n) \approx \mathcal{O}(n) \times 2^{\log_2 n} = \mathcal{O}(n^2)$$

Πολλαπλασιασμός Ακεραίων Αναδρομικά

Πρόοδος; Ο αναδρομικός μας αλγόριθμος τρέχει λοιπόν σε χρόνο $\mathcal{O}(n^2)$. Ίδιο χρόνο με την μέθοδο του σχολείου. Θα χρησιμοποιήσουμε ένα τέχνασμα ώστε να επιταχυνθεί.

Πρόοδος; Ο αναδρομικός μας αλγόριθμος τρέχει λοιπόν σε χρόνο $\mathcal{O}(n^2)$. Ίδιο χρόνο με την μέθοδο του σχολείου. Θα χρησιμοποιήσουμε ένα τέχνασμα ώστε να επιταχυνθεί.

Τέχνασμα του Gauss.
Ο μαθηματικός Carl Friedrich Gauss (1777-1855) παρατήρησε κάποτε ότι παρόλο που το γινόμενο δύο μιγαδικών αριθμών

$$(a + bi)(c + di) = ac - bd + (bc + ad)i$$

φαίνεται να περιλαμβάνει τέσσερις πολλαπλασιασμούς πραγματικών αριθμών, στην πραγματικότητα μπορεί να εκτελεστεί με μόλις τρεις: ac, bd και $(a + b)(c + d)$ αφού

$$bc + ad = (a + b)(c + d) - ac - bd.$$

Πολλαπλασιασμός Ακεραίων Αναδρομικά

Αλγόριθμος των Karatsuba και Ofman

$$x = \boxed{x_L}\boxed{x_R} = 2^{n/2}x_L + x_R$$
$$y = \boxed{y_L}\boxed{y_R} = 2^{n/2}y_L + y_R$$

$$xy = 2^n x_L y_L + 2^{n/2}(x_L y_R + x_R y_L) + x_R y_R$$

Ενώ η παράσταση φαίνεται να χρειάζεται 4 πολλαπλασιασμούς, στην πραγματικότητα τρεις αρκούν

- $x_L y_L$
- $x_R y_R$
- $(x_L + x_R)(y_L + y_R)$

αφού

$$x_L y_R + x_R y_L = (x_L + x_R)(y_L + y_R) - x_L y_L - x_R y_R$$

Πολλαπλασιασμός Ακεραίων Αναδρομικά

multiply(x, y)

Input: θετικοί ακέραιοι x και y των n bits
Output: Το γινόμενο τους
if $n = 1$ **then**
  return xy
end
$x_L, x_R \leftarrow$ αριστερότερα $\lceil n/2 \rceil$, δεξιότερα $\lfloor n/2 \rfloor$ bit του x
$y_L, y_R \leftarrow$ αριστερότερα $\lceil n/2 \rceil$, δεξιότερα $\lfloor n/2 \rfloor$ bit του y
$P_1 \leftarrow$ *multiply*(x_L, y_L)
$P_2 \leftarrow$ *multiply*(x_R, y_R)
$P_3 \leftarrow$ *multiply*$(x_L + x_R, y_L + y_R)$
return $P_1 \times 2^n + (P_3 - P_1 - P_2) \times 2^{n/2} + P_2$

Πολλαπλασιασμός Ακεραίων Αναδρομικά

multiply(x, y)

Input: θετικοί ακέραιοι x και y των n bits
Output: Το γινόμενο τους
if $n = 1$ **then**
  return xy
end
$x_L, x_R \leftarrow$ αριστερότερα $\lceil n/2 \rceil$, δεξιότερα $\lfloor n/2 \rfloor$ bit του x
$y_L, y_R \leftarrow$ αριστερότερα $\lceil n/2 \rceil$, δεξιότερα $\lfloor n/2 \rfloor$ bit του y
$P_1 \leftarrow multiply(x_L, y_L)$
$P_2 \leftarrow multiply(x_R, y_R)$
$P_3 \leftarrow multiply(x_L + x_R, y_L + y_R)$
return $P_1 \times 2^n + (P_3 - P_1 - P_2) \times 2^{n/2} + P_2$

Έχοντας μειώσει σε 3 τις αναδρομικές κλήσεις έχουμε την αναδρομή

$$T(n) \leq 3T(n/2) + \mathcal{O}(n).$$

Επειδή η επίδραση αυτή γίνεται σε κάθε αναδρομικό βήμα, οδηγούμαστε σε ένα δραματικά χαμηλότερο χρονικό φράγμα. Η αναδρομή αυτή λύνει σε

$$T(n) = \mathcal{O}(n^{\log_2 3}) = \mathcal{O}(n^{1.59}).$$

Στην πράξη επειδή οι σύγχρονες αρχιτεκτονικές πολλαπλασιάζουν 32 ή 64 bits πολύ γρήγορα, θα σταματούσαμε την αναδρομή όταν το n φτάσει αυτές τις τιμές.

Κύριο Θεώρημα
Master Theorem

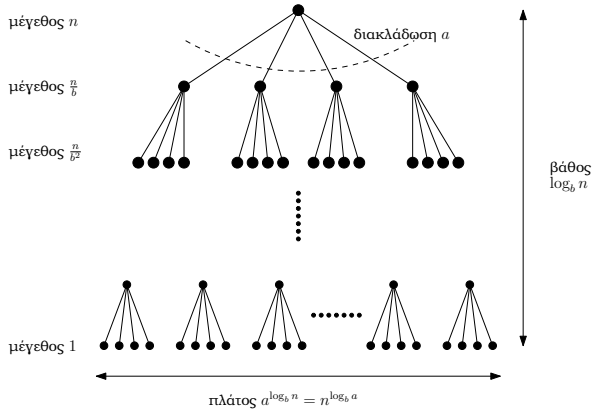
Θεώρημα
Αν $T(n) = a \cdot T(\lceil n/b \rceil) + \mathcal{O}(n^d)$ για κάποιες σταθερές $a > 0$, $b > 1$ και $d \geq 0$, τότε

$$T(n) = \begin{cases} \mathcal{O}(n^d) & \text{αν } d > \log_b a \\ \mathcal{O}(n^d \log n) & \text{αν } d = \log_b a \\ \mathcal{O}(n^{\log_b a}) & \text{αν } d < \log_b a. \end{cases}$$

Κύριο Θεώρημα

Απόδειξη

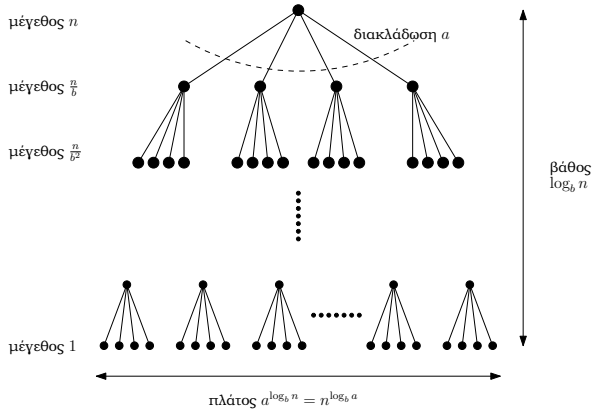
- Ας υποθέσουμε πάλι για λόγους απλότητας ότι το *n* είναι δύναμη του *b*.



Κύριο Θεώρημα

Απόδειξη

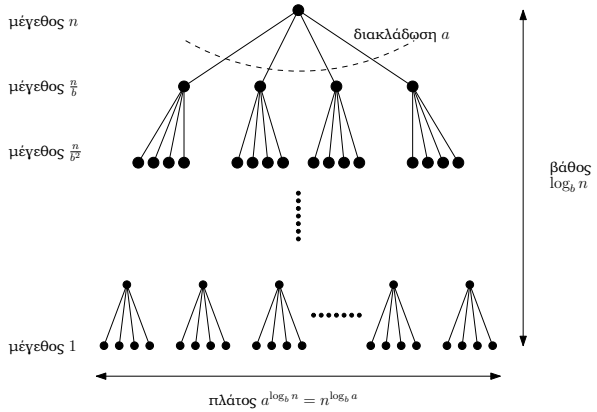
- Ας υποθέσουμε πάλι για λόγους απλότητας ότι το *n* είναι δύναμη του *b*.
- Το μέγεθος των υποπροβλημάτων μειώνεται κατά ένα παράγοντα *b* σε κάθε επίπεδο αναδρομής, και άρα φτάνει στην βασική περίπτωση μετά από $\log_b n$ επίπεδα.



Κύριο Θεώρημα

Απόδειξη

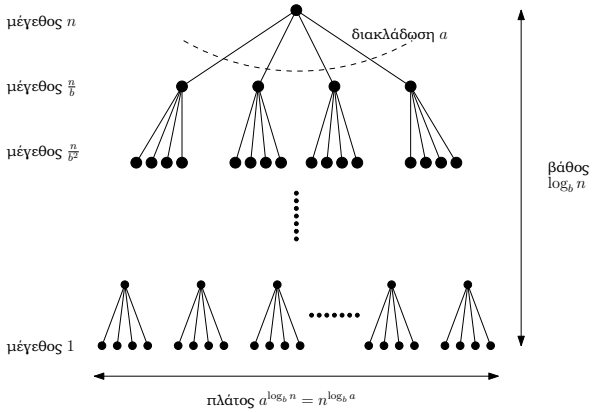
- Ας υποθέσουμε πάλι για λόγους απλότητας ότι το n είναι δύναμη του b.
- Το μέγεθος των υποπροβλημάτων μειώνεται κατά ένα παράγοντα b σε κάθε επίπεδο αναδρομής, και άρα φτάνει στην βασική περίπτωση μετά από $\log_b n$ επίπεδα.
- Παράγοντας διακλάδωσης a και άρα στο j επίπεδο έχουμε a^j υποπροβλήματα με μέγεθος n/b^j το καθένα.



Κύριο Θεώρημα

Απόδειξη

- Ας υποθέσουμε πάλι για λόγους απλότητας ότι το n είναι δύναμη του b.
- Το μέγεθος των υποπροβλημάτων μειώνεται κατά ένα παράγοντα b σε κάθε επίπεδο αναδρομής, και άρα φτάνει στην βασική περίπτωση μετά από $\log_b n$ επίπεδα.
- Παράγοντας διακλάδωσης a και άρα στο j επίπεδο έχουμε a^j υποπροβλήματα με μέγεθος n/b^j το καθένα.
- Η συνολική εργασία στο επιπέδου j είναι $a^j \times \mathcal{O}(\frac{n}{b^j})^d = \mathcal{O}(n^d) \times (\frac{a}{b^d})^j$.

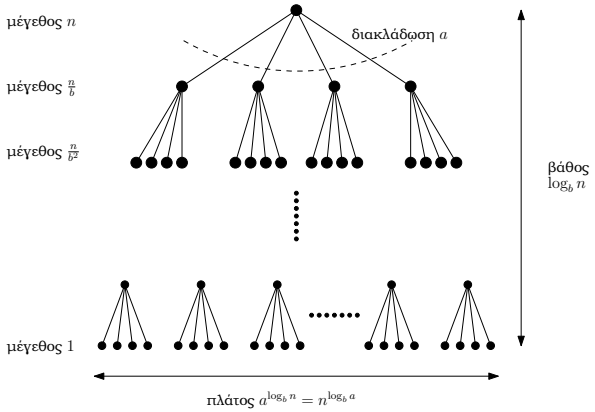


Κύριο Θεώρημα

Απόδειξη

- Ας υποθέσουμε πάλι για λόγους απλότητας ότι το n είναι δύναμη του b.
- Το μέγεθος των υποπροβλημάτων μειώνεται κατά ένα παράγοντα b σε κάθε επίπεδο αναδρομής, και άρα φτάνει στην βασική περίπτωση μετά από $\log_b n$ επίπεδα.
- Παράγοντας διακλάδωσης a και άρα στο j επίπεδο έχουμε a^j υποπροβλήματα με μέγεθος n/b^j το καθένα.
- Η συνολική εργασία στο επιπέδου j είναι $a^j \times \mathcal{O}(\frac{n}{b^j})^d = \mathcal{O}(n^d) \times (\frac{a}{b^d})^j$.
- Καθώς το j πηγαίνει από 0 έως $\log_b n$ έχουμε μια γεωμετρική σειρά με λόγο $\frac{a}{b^d}$

$$\mathcal{O}(n^d) \times \left(1 + \left(\frac{a}{b^d}\right) + \left(\frac{a}{b^d}\right)^2 + \ldots + \left(\frac{a}{b^d}\right)^{\log_b n}\right)$$

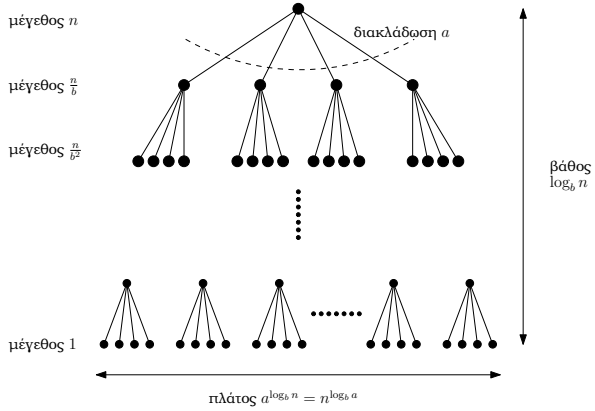


Κύριο Θεώρημα

Απόδειξη

- Ας υποθέσουμε πάλι για λόγους απλότητας ότι το n είναι δύναμη του b.
- Το μέγεθος των υποπροβλημάτων μειώνεται κατά ένα παράγοντα b σε κάθε επίπεδο αναδρομής, και άρα φτάνει στην βασική περίπτωση μετά από $\log_b n$ επίπεδα.
- Παράγοντας διακλάδωσης a και άρα στο j επίπεδο έχουμε a^j υποπροβλήματα με μέγεθος n/b^j το καθένα.
- Η συνολική εργασία στο επιπέδου j είναι $a^j \times \mathcal{O}(\frac{n}{b^j})^d = \mathcal{O}(n^d) \times (\frac{a}{b^d})^j$.
- Καθώς το j πηγαίνει από 0 έως $\log_b n$ έχουμε μια γεωμετρική σειρά με λόγο $\frac{a}{b^d}$

$$\mathcal{O}(n^d) \times \left(1 + \left(\frac{a}{b^d}\right) + \left(\frac{a}{b^d}\right)^2 + \ldots + \left(\frac{a}{b^d}\right)^{\log_b n}\right)$$



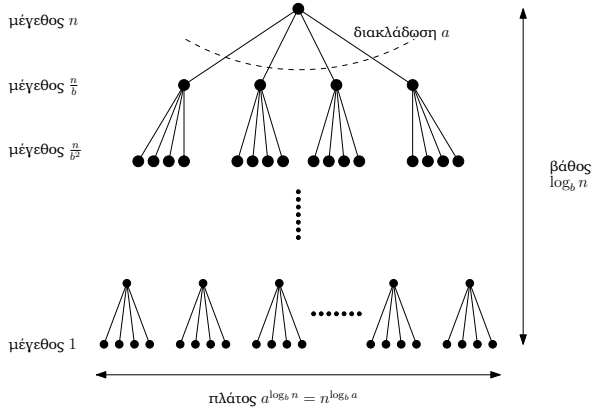
Εαν $\frac{a}{b^d} < 1$, δηλαδή $d > \log_b a$, τότε η σειρά είναι φθίνουσα και το άθροισμα ισούται με τον πρώτο όρο $\mathcal{O}(n^d)$.

Κύριο Θεώρημα

Απόδειξη

- Ας υποθέσουμε πάλι για λόγους απλότητας ότι το n είναι δύναμη του b.
- Το μέγεθος των υποπροβλημάτων μειώνεται κατά ένα παράγοντα b σε κάθε επίπεδο αναδρομής, και άρα φτάνει στην βασική περίπτωση μετά από $\log_b n$ επίπεδα.
- Παράγοντας διακλάδωσης a και άρα στο j επίπεδο έχουμε a^j υποπροβλήματα με μέγεθος n/b^j το καθένα.
- Η συνολική εργασία στο επιπέδου j είναι $a^j \times \mathcal{O}(\frac{n}{b^j})^d = \mathcal{O}(n^d) \times (\frac{a}{b^d})^j$.
- Καθώς το j πηγαίνει από 0 έως $\log_b n$ έχουμε μια γεωμετρική σειρά με λόγο $\frac{a}{b^d}$

$$\mathcal{O}(n^d) \times \left(1 + \left(\frac{a}{b^d}\right) + \left(\frac{a}{b^d}\right)^2 + \ldots + \left(\frac{a}{b^d}\right)^{\log_b n}\right)$$

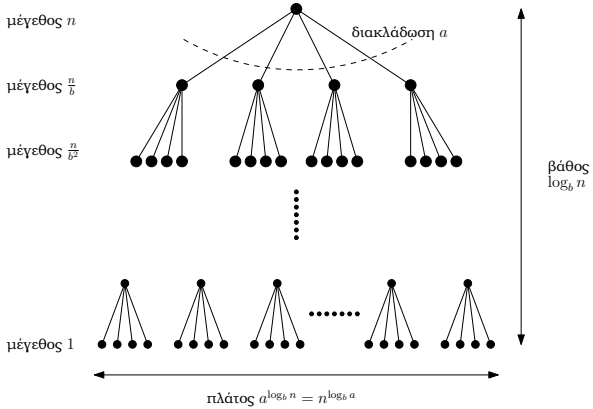


Εάν $\frac{a}{b^d} = 1$, δηλαδή $d = \log_b a$, τότε όλοι οι $\mathcal{O}(\log n)$ όροι της σειράς είναι ίσοι με $\mathcal{O}(n^d)$ και άρα έχουμε $\mathcal{O}(n^d \log n)$.

Κύριο Θεώρημα

Απόδειξη

- Ας υποθέσουμε πάλι για λόγους απλότητας ότι το n είναι δύναμη του b.
- Το μέγεθος των υποπροβλημάτων μειώνεται κατά ένα παράγοντα b σε κάθε επίπεδο αναδρομής, και άρα φτάνει στην βασική περίπτωση μετά από $\log_b n$ επίπεδα.
- Παράγοντας διακλάδωσης a και άρα στο j επίπεδο έχουμε a^j υποπροβλήματα με μέγεθος n/b^j το καθένα.
- Η συνολική εργασία στο επιπέδου j είναι $a^j \times \mathcal{O}(\frac{n}{b^j})^d = \mathcal{O}(n^d) \times (\frac{a}{b^d})^j$.
- Καθώς το j πηγαίνει από 0 έως $\log_b n$ έχουμε μια γεωμετρική σειρά με λόγο $\frac{a}{b^d}$



$$\mathcal{O}(n^d) \times \left(1 + \left(\frac{a}{b^d}\right) + \left(\frac{a}{b^d}\right)^2 + \ldots + \left(\frac{a}{b^d}\right)^{\log_b n}\right)$$

Εάν $\frac{a}{b^d} > 1$, δηλαδή $d < \log_b a$, τότε η σειρά είναι αύξουσα και το άθροισμα της δίνεται από τον τελευταίο της όρο

$$n^d \left(\frac{a}{b^d}\right)^{\log_b n} = n^d \left(\frac{a^{\log_b n}}{(b^{\log_b n})^d}\right) = a^{\log_b n} = a^{(\log_a n)(\log_b a)} = n^{\log_b a}.$$

Πολλαπλασιασμός Ακεραίων

Έχοντας μειώσει σε 3 τις αναδρομικές κλήσεις έχουμε την αναδρομή

$$T(n) \leq 3T(n/2) + \mathcal{O}(n).$$

Επειδή η επίδραση αυτή γίνεται σε κάθε αναδρομικό βήμα, οδηγούμαστε σε ένα δραματικά χαμηλότερο χρονικό φράγμα.

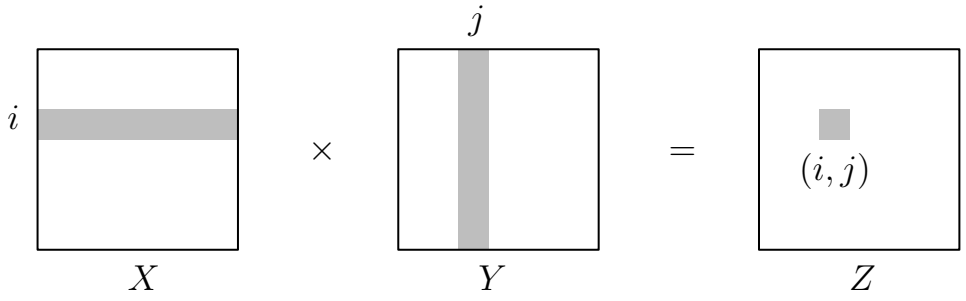
Χρησιμοποιώντας το κύριο θεώρημα με $a = 3$, $b = 2$ και $d = 1$ και άρα $\frac{a}{b^d} = 3/2 > 1$ πέρνουμε πως ο χρόνος είναι

$$T(n) = O(n^{\log_2 3}) = \mathcal{O}(n^{1.59})$$

Πολλαπλασιασμός Πινάκων

Το γινόμενο δύο $n \times n$ πινάκων X και Y είναι ένας τρίτος πίνακας $Z = XY$ πάλι διαστάσεων $n \times n$ όπου το στοιχείο (i, j) είναι

$$Z_{ij} = \sum_{k=1}^{n} X_{ik} Y_{kj}.$$



Πολλαπλασιασμός Πινάκων

$$\begin{pmatrix} Z_{11} & Z_{12} & \cdots & Z_{1n} \\ Z_{21} & Z_{22} & \cdots & Z_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ Z_{n1} & Z_{n2} & \cdots & Z_{nn} \end{pmatrix} = \begin{pmatrix} X_{11} & X_{12} & \cdots & X_{1n} \\ X_{21} & X_{22} & \cdots & X_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ X_{n1} & X_{n2} & \cdots & X_{nn} \end{pmatrix} \cdot \begin{pmatrix} Y_{11} & Y_{12} & \cdots & Y_{1n} \\ Y_{21} & Y_{22} & \cdots & Y_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ Y_{n1} & Y_{n2} & \cdots & Y_{nn} \end{pmatrix}$$

Ο παρακάτω κώδικας υλοποιεί τον αλγόριθμο πολλαπλασιασμού και χρειάζεται $\mathcal{O}(n^3)$ χρόνο.

```
for( i = 0; i < n; i++ )
   for( j = 0; j < n; j++ )
      for( k = 0, c[i][j] = 0.0; k < n; k++ )
         Z[i][j] += X[i][k] * Y[k][j];
```

Πολλαπλασιασμός Πινάκων

$$\begin{pmatrix} Z_{11} & Z_{12} & \cdots & Z_{1n} \\ Z_{21} & Z_{22} & \cdots & Z_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ Z_{n1} & Z_{n2} & \cdots & Z_{nn} \end{pmatrix} = \begin{pmatrix} X_{11} & X_{12} & \cdots & X_{1n} \\ X_{21} & X_{22} & \cdots & X_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ X_{n1} & X_{n2} & \cdots & X_{nn} \end{pmatrix} \cdot \begin{pmatrix} Y_{11} & Y_{12} & \cdots & Y_{1n} \\ Y_{21} & Y_{22} & \cdots & Y_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ Y_{n1} & Y_{n2} & \cdots & Y_{nn} \end{pmatrix}$$

Ο παρακάτω κώδικας υλοποιεί τον αλγόριθμο πολλαπλασιασμού και χρειάζεται $\mathcal{O}(n^3)$ χρόνο.

```
for( i = 0; i < n; i++ )
    for( j = 0; j < n; j++ )
        for( k = 0, c[i][j] = 0.0; k < n; k++ )
            Z[i][j] += X[i][k] * Y[k][j];
```

Το 1969, ο Γερμανός μαθηματικός Volker Strassen ανακοίνωσε έναν αλγόριθμο ”διαίρει και βασίλευε” που είναι σημαντικά γρηγορότερος.

Είναι ιδιαίτερα εύκολο να χωριστεί το πρόβλημα σε υποπροβλήματα αφού μπορεί να εκτελεστεί κατά μπλόκ.

Χωρίζουμε τους πίνακες X και Y σε τέσσερα μπλοκ τον καθένα μεγέθους $n/2 \times n/2$:

$$X = \begin{bmatrix} A & B \\ C & D \end{bmatrix}, Y = \begin{bmatrix} E & F \\ G & H \end{bmatrix}$$

Το γινόμενο τους μπορεί να εκφραστεί συναρτήσει αυτών των μπλόκ σαν να ήταν τα μπλόκ απλά στοιχεία:

$$XY = \begin{bmatrix} A & B \\ C & D \end{bmatrix} \begin{bmatrix} E & F \\ G & H \end{bmatrix} = \begin{bmatrix} AE + BG & AF + BH \\ CE + DG & CF + DH \end{bmatrix}$$

$$XY = \begin{bmatrix} A & B \\ C & D \end{bmatrix} \begin{bmatrix} E & F \\ G & H \end{bmatrix} = \begin{bmatrix} AE + BG & AF + BH \\ CE + DG & CF + DH \end{bmatrix}$$

Ένας αλγόριθμος διαίρει και βασίλευε λοιπόν:

- υπολογίζει αναδρομικά 8 γινόμενα μισού ($n/2$) μεγέθους AE, BG, AF, BH, CE, DG, CF και DH
- και στην συνέχεια κάνει μερικές προσθέσεις

Συνολικά χρειάζεται χρόνο

$$T(n) \leq 8T(n/2) + \mathcal{O}(n^2)$$

που καταλήγει σε $\mathcal{O}(n^3)$, το ίδιο με τον αλγόριθμο του σχολείου.

Ο Strassen με έξυπνη άλγεβρα έδειξε πως το γινόμενο XY μπορεί να υπολογιστεί με μόνο 7 υποπροβλήματα $n/2 \times n/2$ και περισσότερες προσθέσεις.

$$XY = \begin{bmatrix} P_5 + P_4 - P_2 + P_6 & P_1 + P_2 \\ P_3 + P_4 & P_1 + P_5 - P_3 - P_7 \end{bmatrix}$$

όπου

$$P_1 = A(F - H)$$
$$P_2 = (A + B)H$$
$$P_3 = (C + D)E$$
$$P_4 = D(G - E)$$
$$P_5 = (A + D)(E + H)$$
$$P_6 = (B - D)(G + H)$$
$$P_7 = (A - C)(E + F)$$

Συνολικά χρειάζεται χρόνο

$$T(n) \leq 7T(n/2) + \mathcal{O}(n^2)$$

που από το κύριο θεώρημα προκύπτει ότι είναι $\mathcal{O}(n^{\log_2 7}) \approx \mathcal{O}(n^{2.81})$.