Αλγόριθμοι και Πολυπλοκότητα

Αρχοντία Γιαννοπούλου
Όλγα Φουρτουνέλλη

Εθνικό και Καποδιστριακό Πανεπιστήμιο Αθηνών

Διαίρει και Κυρίευε

Ταξινόμηση και η Πολυπλοκότητά της

Τάξη μεγέθους

Έστω $f : \mathbb{N} \to \mathbb{R}^+$.

Ορίζουμε τις παρακάτω κλάσεις συναρτήσεων σε σχέση με την f.

$$\mathcal{O}(f(n)) = \{T : \exists n_0 \in \mathbb{N} \;\; \exists c \in \mathbb{R}^+ \;\; \forall n \geq n_0 \;\; T(n) \leq c \cdot f(n)\}$$

Τάξη μεγέθους

Έστω $f : \mathbb{N} \to \mathbb{R}^+$.

Ορίζουμε τις παρακάτω κλάσεις συναρτήσεων σε σχέση με την f.

$$\Omega(f(n)) = \{T : \exists n_0 \in \mathbb{N} \;\; \exists c \in \mathbb{R}^+ \;\; \forall n \geq n_0 \;\; T(n) \geq c \cdot f(n)\}$$

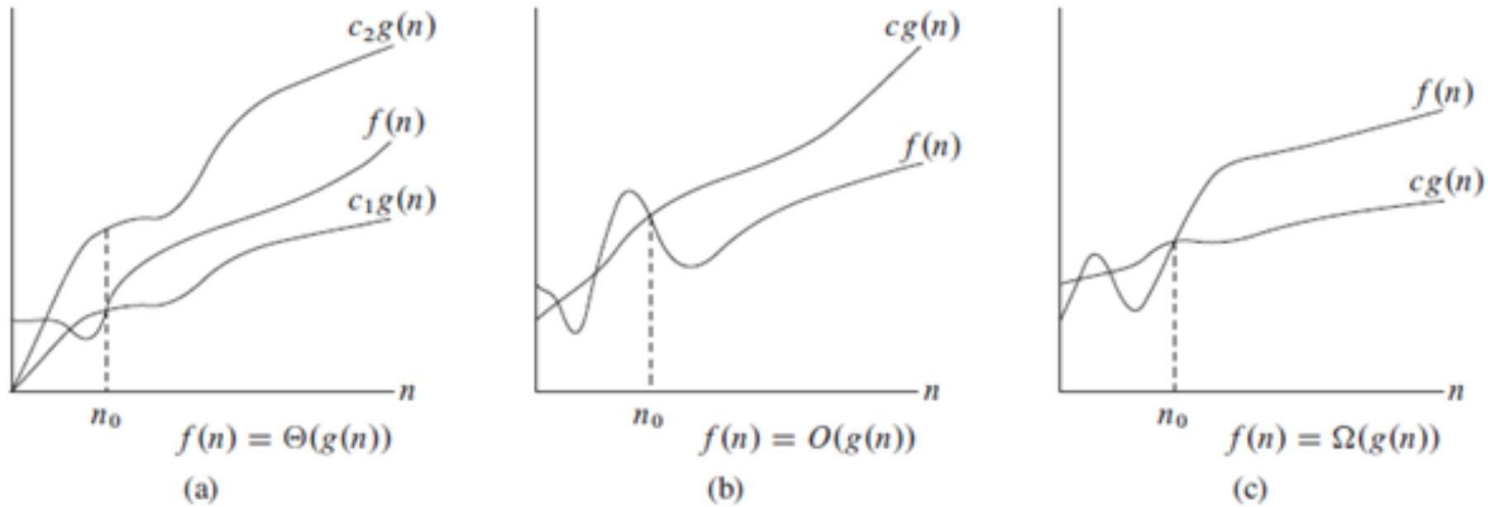
Τάξη μεγέθους

Έστω $f : \mathbb{N} \to \mathbb{R}^+$.

Ορίζουμε τις παρακάτω κλάσεις συναρτήσεων σε σχέση με την f.

$$\Theta(f(n)) = \mathcal{O}(f(n)) \cap \Omega(f(n))$$

Σύγκριση


$c_2g(n)$
$f(n)$
$c_1g(n)$
n
n_0
$f(n) = \Theta(g(n))$
(a)
$cg(n)$
$f(n)$
n
n_0
$f(n) = O(g(n))$
(b)
$f(n)$
$cg(n)$
n
n_0
$f(n) = \Omega(g(n))$
(c)

Συμβολισμοί εναλλακτικά

$f, g : \mathbb{N} \to \mathbb{R}^+$

- $\lim_{n\to\infty} \frac{f(n)}{g(n)} = \alpha \neq 0 \Rightarrow f \in \Theta(g)$
- $\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0 \Rightarrow f \in \mathcal{O}(g)$ και $f \notin \Theta(g)$
- $\lim_{n\to\infty} \frac{f(n)}{g(n)} = \infty \Rightarrow g \in \mathcal{O}(f)$ και $g \notin \Theta(f)$

Συμβολισμοί εναλλακτικά

$f, g : \mathbb{N} \to \mathbb{R}^+$

- $\lim_{n\to\infty} \frac{f(n)}{g(n)} = \alpha \neq 0 \Rightarrow f \in \Theta(g)$
- $\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0 \Rightarrow f \in \mathcal{O}(g)$ και $f \notin \Theta(g)$ $\qquad f \in o(g)$
- $\lim_{n\to\infty} \frac{f(n)}{g(n)} = \infty \Rightarrow g \in \mathcal{O}(f)$ και $g \notin \Theta(f)$

Συμβολισμοί εναλλακτικά

$f, g : \mathbb{N} \to \mathbb{R}^+$

- $\lim_{n\to\infty} \frac{f(n)}{g(n)} = \alpha \neq 0 \Rightarrow f \in \Theta(g)$
- $\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0 \Rightarrow f \in \mathcal{O}(g)$ και $f \notin \Theta(g)$ $\qquad f \in o(g)$
- $\lim_{n\to\infty} \frac{f(n)}{g(n)} = \infty \Rightarrow g \in \mathcal{O}(f)$ και $g \notin \Theta(f)$ $\qquad f \in \omega(g)$

Κλάσεις Πολυπλοκότητας

- $f(n) = 1$ σταθερή πολυπλοκότητα
- $f(n) = \log n$ λογαριθμική πολυπλοκότητα
- $f(n) = n$ γραμμική πολυπλοκότητα
- $f(n) = n \log n$
- $f(n) = n^2$
- $f(n) = n^3$

$\vdots$

- $f(n) = n^p$ πολυωνυμική πολυπλοκότητα

$\vdots$

- $f(n) = 2^n$ εκθετική πολυπλοκότητα
- $f(n) = 3^n$

$\vdots$

Η έννοια του βέλτιστου αλγόριθμου

Για ένα πρόβλημα Π να βρεθεί ο καλύτερος αλγόριθμος που το επιλύει.

Δηλαδή ο αλγόριθμος με την καλύτερη πολυπλοκότητα.

Η έννοια του βέλτιστου αλγόριθμου

Για ένα πρόβλημα Π να βρεθεί ο καλύτερος αλγόριθμος που το επιλύει.

Δηλαδή ο αλγόριθμος με την καλύτερη πολυπλοκότητα.

Αν **οποιοσδήποτε** αλγόριθμος που επιλύει το Π έχει ασυμπτωτική πολυπλοκότητα $\Omega(f(n))$ τότε **ένας** αλγόριθμος με πολυπλοκότητα $\mathcal{O}(f(n))$ είναι (ασυμπτωτικά) βέλτιστος!

Υπολογισμός μέγιστου στοιχείου

Αλγόριθμος 1

$v \leftarrow a_1$
Για $i = 2$ έως n
    Εάν $v < a_i$ τότε $v \leftarrow a_i$
επίστρεψε v

Υπολογισμός μέγιστου στοιχείου

Αλγόριθμος 1

$v \leftarrow a_1$
Για $i = 2$ έως n
    Εάν $v < a_i$ τότε $v \leftarrow a_i$
επίστρεψε v

Αλγόριθμος 2

Για $i = 1$ έως $n - 1$
    Εάν $a_i > a_{i+1}$ τότε ενάλλαξε τις δύο τιμές
επίστρεψε a_n

Ταξινόμηση στοιχείων

Μας δίνεται ένας πίνακας από στοιχεία και το ζητούμενο είναι να τα ταξινομήσουμε (με βάση κάποιο κριτήριο).

Προφανείς εφαρμογές της ταξινόμησης

Ταξινόμηση στοιχείων

Μας δίνεται ένας πίνακας από στοιχεία και το ζητούμενο είναι να τα ταξινομήσουμε (με βάση κάποιο κριτήριο).

Προφανείς εφαρμογές της ταξινόμησης

- Οργάνωση μίας λίστας τραγουδιών (playlist).

Ταξινόμηση στοιχείων

Μας δίνεται ένας πίνακας από στοιχεία και το ζητούμενο είναι να τα ταξινομήσουμε (με βάση κάποιο κριτήριο).

Προφανείς εφαρμογές της ταξινόμησης

- Οργάνωση μίας λίστας τραγουδιών (playlist).
- Αλφαβητική διάταξη μίας λίστας ονομάτων (το κινητό το κάνει αυτόματα όταν προσθέτουμε νέους τηλεφωνικούς αριθμούς).

Ταξινόμηση στοιχείων

Μας δίνεται ένας πίνακας από στοιχεία και το ζητούμενο είναι να τα ταξινομήσουμε (με βάση κάποιο κριτήριο).

Προφανείς εφαρμογές της ταξινόμησης

- Οργάνωση μίας λίστας τραγουδιών (playlist).
- Αλφαβητική διάταξη μίας λίστας ονομάτων (το κινητό το κάνει αυτόματα όταν προσθέτουμε νέους τηλεφωνικούς αριθμούς).
- Απαρίθμηση των αρχείων σε έναν φάκελο αντίστροφα από την ημερομηνία δημιουργίας τους.

Ταξινόμηση στοιχείων

Μας δίνεται ένας πίνακας από στοιχεία και το ζητούμενο είναι να τα ταξινομήσουμε (με βάση κάποιο κριτήριο).

Προφανείς εφαρμογές της ταξινόμησης

- Οργάνωση μίας λίστας τραγουδιών (playlist).
- Αλφαβητική διάταξη μίας λίστας ονομάτων (το κινητό το κάνει αυτόματα όταν προσθέτουμε νέους τηλεφωνικούς αριθμούς).
- Απαρίθμηση των αρχείων σε έναν φάκελο αντίστροφα από την ημερομηνία δημιουργίας τους.
- κ.ο.κ.

Ταξινόμηση στοιχείων

Χρησιμότητα ταξινομημένης λίστας

Ταξινόμηση στοιχείων

Χρησιμότητα ταξινομημένης λίστας

- Εύρεση μέσου στοιχείου.

Ταξινόμηση στοιχείων

Χρησιμότητα ταξινομημένης λίστας

- Εύρεση μέσου στοιχείου.
- Εύρεση κοντινού ζευγαριού.

Ταξινόμηση στοιχείων

Χρησιμότητα ταξινομημένης λίστας

- Εύρεση μέσου στοιχείου.
- Εύρεση κοντινού ζευγαριού.
- Εφαρμογή δυαδικής αναζήτησης.

Ταξινόμηση στοιχείων

Χρησιμότητα ταξινομημένης λίστας

- Εύρεση μέσου στοιχείου.
- Εύρεση κοντινού ζευγαριού.
- Εφαρμογή δυαδικής αναζήτησης.
- Εντοπισμός ακραίων τιμών (outliers).

Ταξινόμηση στοιχείων

Χρησιμότητα ταξινομημένης λίστας

- Εύρεση μέσου στοιχείου.
- Εύρεση κοντινού ζευγαριού.
- Εφαρμογή δυαδικής αναζήτησης.
- Εντοπισμός ακραίων τιμών (outliers).
- Εντοπισμός επαναλαμβανόμενων στοιχείων.

Ταξινόμηση στοιχείων

Μη προφανής χρησιμότητα ταξινομημένης λίστας

Ταξινόμηση στοιχείων

Μη προφανής χρησιμότητα ταξινομημένης λίστας

- Συμπίεση δεδομένων.

Ταξινόμηση στοιχείων

Μη προφανής χρησιμότητα ταξινομημένης λίστας

- Συμπίεση δεδομένων.
- Γραφικά Η/Υ.

Ταξινόμηση στοιχείων

Μη προφανής χρησιμότητα ταξινομημένης λίστας

- Συμπίεση δεδομένων.
- Γραφικά Η/Υ.
- Υπολογιστική βιολογία.

Ταξινόμηση στοιχείων

Μη προφανής χρησιμότητα ταξινομημένης λίστας

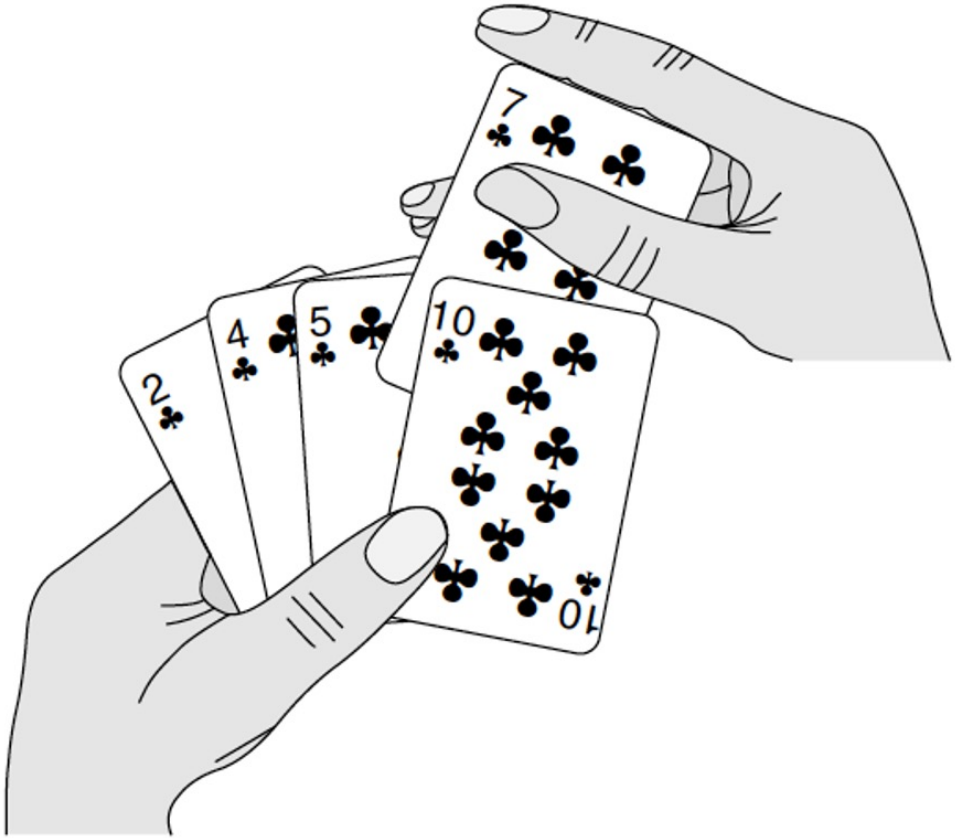
- Συμπίεση δεδομένων.
- Γραφικά Η/Υ.
- Υπολογιστική βιολογία.
- Ελάχιστα επικαλύπτοντα δέντρα.

Αλγόριθμος ταξινόμησης (φυσαλίδας)

Διαδικασία bubblesort$(a_1, a_2, \ldots, a_n)$
Για $i = 1$ ως $n - 1$
    Για $j = 1$ ως $n - i$
        Αν $a_j > a_{j+1}$ **τότε** ενάλλαξε τα a_j και a_{j+1}

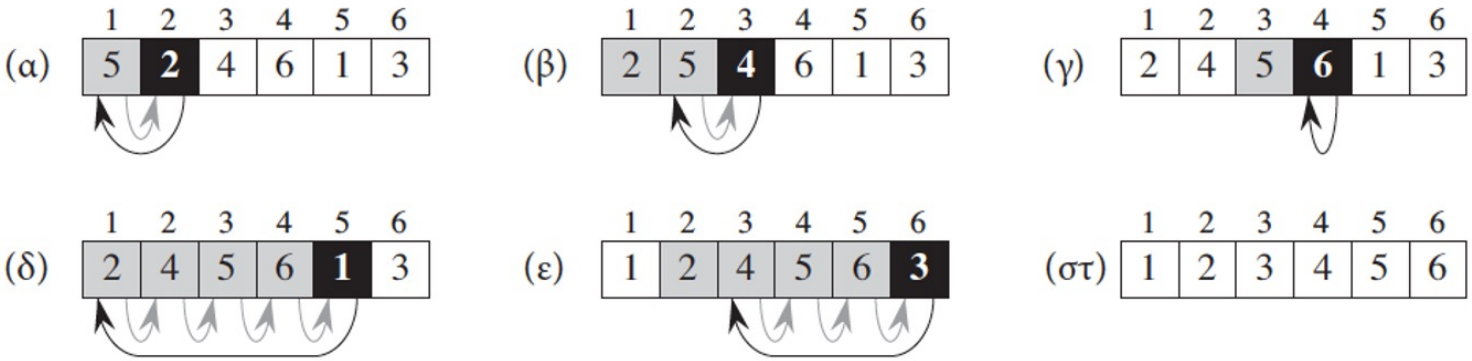
Πόσα βήματα χρειαζόμαστε για να ταξινομήσουμε τα στοιχεία;

Αλγόριθμος ταξινόμησης (ενθετική)



Σχήμα 2.1 Ταξινόμηση ενός συνόλου από τραπουλόχαρτα με την ενθετική μέθοδο.

Αλγόριθμος ταξινόμησης (ενθετική)



Σχήμα 2.2 Η λειτουργία της ΕΝΘΕΤΙΚΗΣ ΤΑΞΙΝΟΜΗΣΗΣ στη συστοιχία $A = \langle 5, 2, 4, 6, 1, 3 \rangle$. Οι αριθμοί επάνω από τα τετραγωνίδια είναι οι αύξοντες αριθμοί των στοιχείων της συστοιχίας, ενώ οι αριθμοί μέσα στα τετραγωνίδια είναι οι τιμές που είναι αποθηκευμένες στις θέσεις της συστοιχίας. **(α)**–**(ε)** Οι διαδοχικές επαναλήψεις του βρόχου **για** (γραμμές 1–8). Σε κάθε επανάληψη, το μαύρο τετραγωνίδιο περιέχει το εξεταζόμενο κλειδί $A[j]$, το οποίο συγκρίνεται με τις τιμές στα σκιασμένα τετραγωνίδια στα αριστερά του μέσω της συνθήκης ελέγχου στη γραμμή 5. Τα γκρίζα βέλη υποδεικνύουν τα στοιχεία της συστοιχίας που μετακινούνται κατά μια θέση προς τα δεξιά μέσω της εντολής στη γραμμή 6, ενώ τα μαύρα δείχνουν σε ποια θέση μετακινείται το εξεταζόμενο κλειδί μέσω της εντολής στη γραμμή 8. **(στ)** Η τελική ταξινομημένη συστοιχία.

Αλγόριθμος ταξινόμησης (ενθετική)

Διαδικασία insertionsort$(a_1, a_2, \ldots, a_n)$
Για $i = 1$ έως n
    $v \leftarrow a_i$
    $j \leftarrow i - 1$
    Όσω $j \geq 1$ και $v < a_j$
        $a_{j+1} \leftarrow a_j$ % ολίσθηση
        $j \leftarrow j - 1$
    $a_{j+1} \leftarrow v$

Πολυπλοκότητα Αλγορίθμων

- Μέθοδος Φυσαλίδας: $\Theta(n^2)$ (χείριστη πολυπλοκότητα)

- Μέθοδος Εισαγωγής (Ενθετική): $\Theta(n^2)$ (χείριστη πολυπλοκότητα)

Συγκρίσεις ανά δύο

Σπάσε τα στοιχεία σε ζευγάρια και σύγκρινέ τα ανά δύο.

Στη συνέχεια άρχισε να συγχωνεύεις ανά δύο τα ταξινομημένα ζεύγη.

Divide et impera!

Διαίρει και Κυρίευε

Η τεχνική διαίρει και κυρίευε βασίζεται στα εξής βήματα:

- **Διαιρούμε** τα δεδομένα του προβλήματος μας σε μικρότερα υποπροβλήματα **που δεν έχουν κοινά στοιχεία**.
- **Κυριεύουμε** τα μικρότερα υποπροβλήματα ξεχωριστά (και αναδρομικά).
- **Συνδυάζουμε** τις λύσεις που βρήκαμε για να λύσουμε το αρχικό πρόβλημα.

Διαίρει και Κυρίευε

Συνήθως εφαρμόζεται ως εξής:

- Διαιρούμε τα δεδομένα μεγέθους n σε δύο μικρότερα στιγμιότυπα μεγέθους $\lceil \frac{n}{2} \rceil$ και $\lfloor \frac{n}{2} \rfloor$ (τα οποία δεν έχουν κοινά στοιχεία).
- Επιλύουμε το πρόβλημα με αναδρομική κλήση του αλγόριθμου στα υποπροβλήματα. (Θεωρούμε ότι ο αλγόριθμος επιστρέφει τη λύση σα μαύρο κουτί.)
- Συνδυάζουμε τις επιμέρους λύσεις σε λύση ολόκληρου του προβλήματος.

Επαγωγή

Η τεχνική Διαίρει και Κυρίευε βασίζεται σε *επαγωγική* κατασκευή της λύσης.

Αρκεί και χρειάζεται να γνωρίζουμε πώς:

Επαγωγή

Η τεχνική Διαίρει και Κυρίευε βασίζεται σε *επαγωγική* κατασκευή της λύσης.

Αρκεί και χρειάζεται να γνωρίζουμε πώς:

- Να επιλύσουμε το πρόβλημα για μικρή (σταθερή) διάσταση δεδομένων π.χ. για δεδομένα διάστασης 1 ή 2.

Επαγωγή

Η τεχνική Διαίρει και Κυρίευε βασίζεται σε *επαγωγική* κατασκευή της λύσης.

Αρκεί και χρειάζεται να γνωρίζουμε πώς:

- Να επιλύσουμε το πρόβλημα για μικρή (σταθερή) διάσταση δεδομένων π.χ. για δεδομένα διάστασης 1 ή 2.

- Να συνδυάσουμε τις ορθές λύσεις των ξένων υποπροβλημάτων μας σε μία καθολική λύση για όλα τα δεδομένα.

Πώς ταξινομούμε σε αύξουσα σειρά;

Αν ο πίνακας έχει ένα στοιχείο τότε είναι ήδη ταξινομημένος και ο αλγόριθμός μας επιστρέφει τον ίδιο πίνακα.

Αλλιώς:

- Διαιρούμε (σπάμε) τον πίνακα στη μέση και θεωρούμε τα δύο μισά.
- Ταξινομούμε κάθε ένα από τα δύο κάνοντας μία αναδρομική κλήση του ίδιου αλγόριθμου για το κάθε πρόβλημα.
- Συνδυάζουμε (συγχωνεύουμε) τις λύσεις που βρήκαμε.

Παράδειγμα

Η είσοδος μας αποτελείται από τον γκρί πίνακα.

10	5	8	6	3	4	7	1	9	2	11

Καλούμε αναδρομικά τον αλγόριθμο στους δύο υποπίνακες.

10	5	8	6	3	4

7	1	9	2	11

Παράδειγμα

Η είσοδος μας αποτελείται από τον γκρί πίνακα.

10	5	8	6	3	4	7	1	9	2	11

Καλούμε αναδρομικά τον αλγόριθμο στους δύο υποπίνακες.

10	5	8	6	3	4

7	1	9	2	11

Συγχωνεύουμε τους δύο υποπίνακες που ο αλγόριθμος επιστρέφει ταξινομημένους.

3	4	5	6	8	10

1	2	7	9	11

Παράδειγμα

Η είσοδος μας αποτελείται από τον γκρί πίνακα.

10	5	8	6	3	4	7	1	9	2	11

Καλούμε αναδρομικά τον αλγόριθμο στους δύο υποπίνακες.

10	5	8	6	3	4

7	1	9	2	11

Συγχωνεύουμε τους δύο υποπίνακες που ο αλγόριθμος επιστρέφει ταξινομημένους.

3	4	5	6	8	10

1	2	7	9	11

1

Παράδειγμα

Η είσοδος μας αποτελείται από τον γκρί πίνακα.

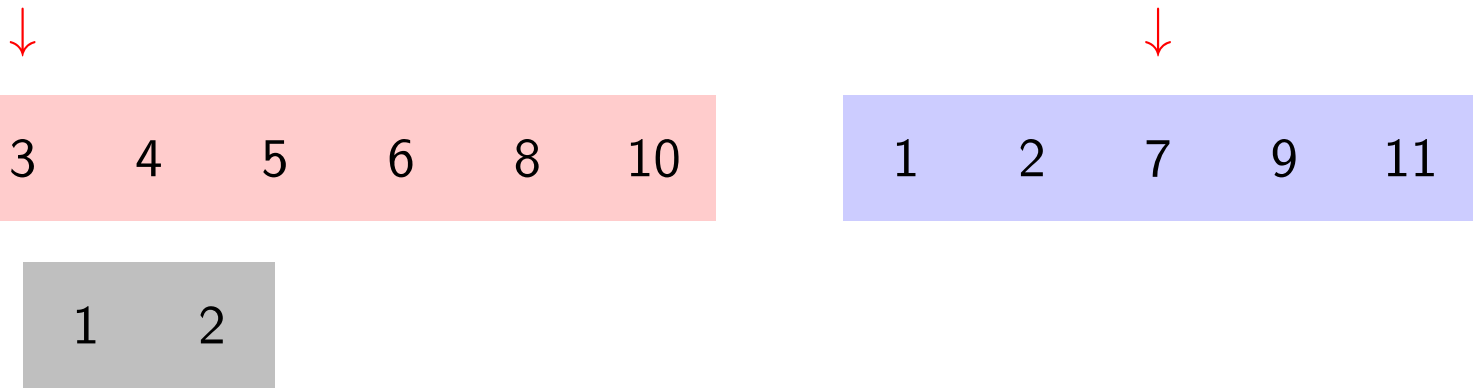
10	5	8	6	3	4	7	1	9	2	11

Καλούμε αναδρομικά τον αλγόριθμο στους δύο υποπίνακες.

10	5	8	6	3	4

7	1	9	2	11

Συγχωνεύουμε τους δύο υποπίνακες που ο αλγόριθμος επιστρέφει ταξινομημένους.



Παράδειγμα

Η είσοδος μας αποτελείται από τον γκρί πίνακα.

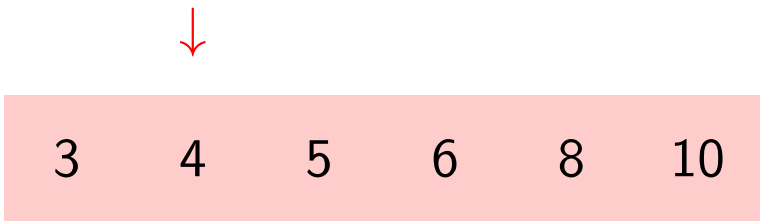
10	5	8	6	3	4	7	1	9	2	11

Καλούμε αναδρομικά τον αλγόριθμο στους δύο υποπίνακες.

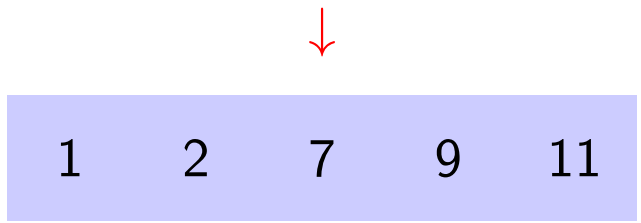
10	5	8	6	3	4

7	1	9	2	11

Συγχωνεύουμε τους δύο υποπίνακες που ο αλγόριθμος επιστρέφει ταξινομημένους.



3	4	5	6	8	10



1	2	7	9	11

1	2	3

Παράδειγμα

Η είσοδος μας αποτελείται από τον γκρί πίνακα.

10	5	8	6	3	4	7	1	9	2	11

Καλούμε αναδρομικά τον αλγόριθμο στους δύο υποπίνακες.

10	5	8	6	3	4

7	1	9	2	11

Συγχωνεύουμε τους δύο υποπίνακες που ο αλγόριθμος επιστρέφει ταξινομημένους.

3	4	5	6	8	10

1	2	7	9	11

1	2	3	4

Παράδειγμα

Η είσοδος μας αποτελείται από τον γκρί πίνακα.

10	5	8	6	3	4	7	1	9	2	11

Καλούμε αναδρομικά τον αλγόριθμο στους δύο υποπίνακες.

10	5	8	6	3	4

7	1	9	2	11

Συγχωνεύουμε τους δύο υποπίνακες που ο αλγόριθμος επιστρέφει ταξινομημένους.

3	4	5	6	8	10

1	2	7	9	11

1	2	3	4	5

Παράδειγμα

Η είσοδος μας αποτελείται από τον γκρί πίνακα.

10	5	8	6	3	4	7	1	9	2	11

Καλούμε αναδρομικά τον αλγόριθμο στους δύο υποπίνακες.

10	5	8	6	3	4

7	1	9	2	11

Συγχωνεύουμε τους δύο υποπίνακες που ο αλγόριθμος επιστρέφει ταξινομημένους.

3	4	5	6	8	10

1	2	7	9	11

1	2	3	4	5	6

Παράδειγμα

Η είσοδος μας αποτελείται από τον γκρί πίνακα.

10	5	8	6	3	4	7	1	9	2	11

Καλούμε αναδρομικά τον αλγόριθμο στους δύο υποπίνακες.

10	5	8	6	3	4

7	1	9	2	11

Συγχωνεύουμε τους δύο υποπίνακες που ο αλγόριθμος επιστρέφει ταξινομημένους.

3	4	5	6	8	10

1	2	7	9	11

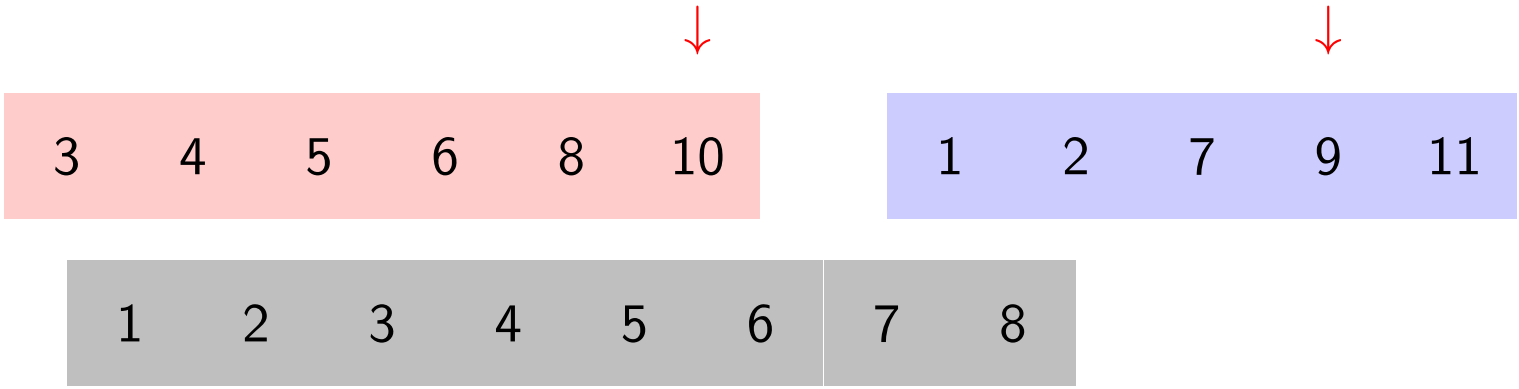
1	2	3	4	5	6	7

Παράδειγμα

Η είσοδος μας αποτελείται από τον γκρί πίνακα.

Καλούμε αναδρομικά τον αλγόριθμο στους δύο υποπίνακες.

Συγχωνεύουμε τους δύο υποπίνακες που ο αλγόριθμος επιστρέφει ταξινομημένους.



Παράδειγμα

Η είσοδος μας αποτελείται από τον γκρί πίνακα.

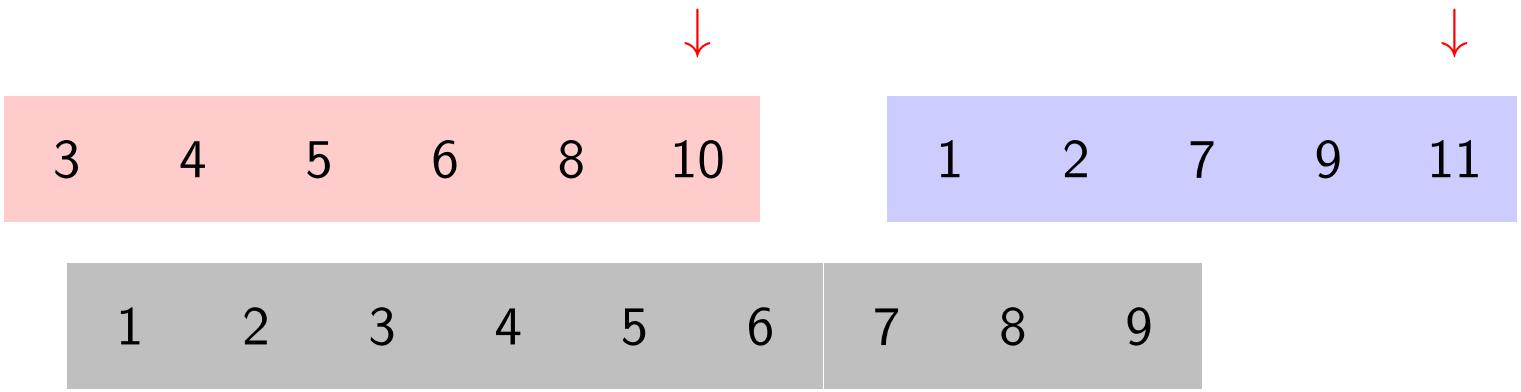
10	5	8	6	3	4	7	1	9	2	11

Καλούμε αναδρομικά τον αλγόριθμο στους δύο υποπίνακες.

10	5	8	6	3	4

7	1	9	2	11

Συγχωνεύουμε τους δύο υποπίνακες που ο αλγόριθμος επιστρέφει ταξινομημένους.



Παράδειγμα

Η είσοδος μας αποτελείται από τον γκρί πίνακα.

10	5	8	6	3	4	7	1	9	2	11

Καλούμε αναδρομικά τον αλγόριθμο στους δύο υποπίνακες.

10	5	8	6	3	4

7	1	9	2	11

Συγχωνεύουμε τους δύο υποπίνακες που ο αλγόριθμος επιστρέφει ταξινομημένους.

3	4	5	6	8	10

1	2	7	9	11

1	2	3	4	5	6	7	8	9	10

Παράδειγμα

Η είσοδος μας αποτελείται από τον γκρί πίνακα.

10	5	8	6	3	4	7	1	9	2	11

Καλούμε αναδρομικά τον αλγόριθμο στους δύο υποπίνακες.

10	5	8	6	3	4

7	1	9	2	11

Συγχωνεύουμε τους δύο υποπίνακες που ο αλγόριθμος επιστρέφει ταξινομημένους.

3	4	5	6	8	10

1	2	7	9	11

1	2	3	4	5	6	7	8	9	10	11

Συγχώνευση ταξινομημένων πινάκων

Έστω a και b δύο ταξινομημένοι πίνακες με m και n στοιχεία αντίστοιχα.

Διαδικασία συγχώνευση(a, b)
$i \leftarrow 1$
$j \leftarrow 1$
Για $k = 1$ ως $m + n$
    Εάν $a_i < b_j$
        τότε $c_k \leftarrow a_i$
            $i \leftarrow i + 1$
        αλλιώς $c_k \leftarrow b_j$
            $j \leftarrow j + 1$

Συγχώνευση ταξινομημένων πινάκων

Έστω a και b δύο ταξινομημένοι πίνακες με m και n στοιχεία αντίστοιχα.

Διαδικασία συγχώνευση(a, b)
$i \leftarrow 1$
$j \leftarrow 1$
Για $k = 1$ ως $m + n$
    Εάν $a_i < b_j$
        τότε $c_k \leftarrow a_i$
            $i \leftarrow i + 1$
        αλλιώς $c_k \leftarrow b_j$
            $j \leftarrow j + 1$

Πολυπλοκότητα: $\mathcal{O}(m + n)$

Συγχωνευτική ταξινόμηση (mergesort)

Διαδικασία mergesort(A, p, r)
Εάν $p = r$
    Επίστρεψε(A, p, r)
Εάν $p < r$
    $q = \lfloor \frac{p+r}{2} \rfloor$
    mergesort(A, p, q)
    mergesort$(A, q + 1, r)$
    Επίστρεψε συγχώνευση$(A[p, \ldots, q], A[q + 1, \ldots, r])$

Συγχωνευτική ταξινόμηση (mergesort)

Διαδικασία mergesort(A, p, r)
Εάν $p = r$
    Επίστρεψε(A, p, r)
Εάν $p < r$
    $q = \lfloor \frac{p+r}{2} \rfloor$
    mergesort(A, p, q)
    mergesort$(A, q + 1, r)$
    Επίστρεψε συγχώνευση$(A[p, \ldots, q], A[q + 1, \ldots, r])$

Γιατί διαιρούμε τον πίνακά μας σε δύο ίσα μέρη (που έχουν μέγεθος το μισό του αρχικού) αντί για έναν πίνακα μεγέθους $n - 2$ και ένα πίνακα μεγέθους 2;

Πολυπλοκότητα συγχωνευτικής ταξινόμησης

Ορίζουμε ως $T(n)$ την πολυπλοκότητα του αλγόριθμου για είσοδο μεγέθους n.

Πολυπλοκότητα συγχωνευτικής ταξινόμησης

Ορίζουμε ως $T(n)$ την πολυπλοκότητα του αλγόριθμου για είσοδο μεγέθους n.

$$T(n) = \begin{cases} 0 & \text{εάν } n = 1 \\ T(\lfloor \frac{n}{2} \rfloor) + T(\lceil \frac{n}{2} \rceil) + n & \text{διαφορετικά} \end{cases}$$

Πολυπλοκότητα συγχωνευτικής ταξινόμησης

Ορίζουμε ως $T(n)$ την πολυπλοκότητα του αλγόριθμου για είσοδο μεγέθους n.

$$T(n) = \begin{cases} 0 & \text{εάν } n = 1 \\ T(\lfloor \frac{n}{2} \rfloor) + T(\lceil \frac{n}{2} \rceil) + n & \text{διαφορετικά} \end{cases}$$

Θα δείξουμε ότι $T(n) = \mathcal{O}(n \log n)$.

Απόδειξη (πρώτη)

Αν $T(n) = 2T(\frac{n}{2}) + n$ τότε $T(n) = n \log n$.

Έστω $n > 1$.

$$\frac{T(n)}{n} = \frac{2T(\frac{n}{2})}{n} + 1$$

Απόδειξη (πρώτη)

Αν $T(n) = 2T(\frac{n}{2}) + n$ τότε $T(n) = n \log n$.

Έστω $n > 1$.

$$\begin{aligned} \frac{T(n)}{n} &= \frac{2T(\frac{n}{2})}{n} + 1 \\ &= \frac{T(\frac{n}{2})}{\frac{n}{2}} + 1 \end{aligned}$$

Απόδειξη (πρώτη)

Αν $T(n) = 2T(\frac{n}{2}) + n$ τότε $T(n) = n \log n$.

Έστω $n > 1$.

$$\begin{aligned} \frac{T(n)}{n} &= \frac{2T(\frac{n}{2})}{n} + 1 \\ &= \frac{T(\frac{n}{2})}{\frac{n}{2}} + 1 \\ &= \frac{T(\frac{n}{4})}{\frac{n}{4}} + 1 + 1 \end{aligned}$$

Απόδειξη (πρώτη)

Αν $T(n) = 2T(\frac{n}{2}) + n$ τότε $T(n) = n \log n$.

Έστω $n > 1$.

$$\begin{aligned}
\frac{T(n)}{n} &= \frac{2T(\frac{n}{2})}{n} + 1 \\
&= \frac{T(\frac{n}{2})}{\frac{n}{2}} + 1 \\
&= \frac{T(\frac{n}{4})}{\frac{n}{4}} + 1 + 1 \\
&\vdots
\end{aligned}$$

Απόδειξη (πρώτη)

Αν $T(n) = 2T(\frac{n}{2}) + n$ τότε $T(n) = n \log n$.

Έστω $n > 1$.

$$\begin{aligned}
\frac{T(n)}{n} &= \frac{2T(\frac{n}{2})}{n} + 1 \\
&= \frac{T(\frac{n}{2})}{\frac{n}{2}} + 1 \\
&= \frac{T(\frac{n}{4})}{\frac{n}{4}} + 1 + 1 \\
&\vdots \\
&= \frac{T(\frac{n}{n})}{\frac{n}{n}} + \underbrace{1 + 1 + \cdots + 1}_{\log n}
\end{aligned}$$

Απόδειξη (πρώτη)

Αν $T(n) = 2T(\frac{n}{2}) + n$ τότε $T(n) = n \log n$.

Έστω $n > 1$.

$$
\begin{aligned}
\frac{T(n)}{n} &= \frac{2T(\frac{n}{2})}{n} + 1 \\
&= \frac{T(\frac{n}{2})}{\frac{n}{2}} + 1 \\
&= \frac{T(\frac{n}{4})}{\frac{n}{4}} + 1 + 1 \\
&\vdots \\
&= \frac{T(\frac{n}{n})}{\frac{n}{n}} + \underbrace{1 + 1 + \cdots + 1}_{\log n} \\
&= \log n.
\end{aligned}
$$

Απόδειξη (πρώτη)

Αν $T(n) = 2T(\frac{n}{2}) + n$ τότε $T(n) = n \log n$.

Έστω $n > 1$.

$$\begin{aligned}
\frac{T(n)}{n} &= \frac{2T(\frac{n}{2})}{n} + 1 \\
&= \frac{T(\frac{n}{2})}{\frac{n}{2}} + 1 \\
&= \frac{T(\frac{n}{4})}{\frac{n}{4}} + 1 + 1 \\
&\vdots \\
&= \frac{T(\frac{n}{n})}{\frac{n}{n}} + \underbrace{1 + 1 + \cdots + 1}_{\log n} \\
&= \log n.
\end{aligned}$$

Άρα $T(n) = n \log n$.

Απόδειξη (δεύτερη)

Με επαγωγή στο n.

- Η βάση ισχύει ($T(1) = 0 = 1 \cdot \log 1$).
- Έστω ότι $T(n) = n \log n$.
- Θα δείξουμε ότι $T(2n) = 2n \cdot \log(2n)$.

Απόδειξη (δεύτερη)

Πράγματι,

$$T(2n) = 2T(n) + 2n$$

Απόδειξη (δεύτερη)

Πράγματι,

$$\begin{aligned} T(2n) &= 2T(n) + 2n \\ &= 2n \log n + 2n \end{aligned}$$

Απόδειξη (δεύτερη)

Πράγματι,

$$\begin{aligned} T(2n) &= 2T(n) + 2n \\ &= 2n \log n + 2n \\ &= 2n(\log n + 1 - 1) + 2n \end{aligned}$$

Απόδειξη (δεύτερη)

Πράγματι,

$$\begin{aligned} T(2n) &= 2T(n) + 2n \\ &= 2n \log n + 2n \\ &= 2n(\log n + 1 - 1) + 2n \\ &= 2n(\log n + \log 2 - 1) + 2n \end{aligned}$$

Απόδειξη (δεύτερη)

Πράγματι,

$$\begin{aligned} T(2n) &= 2T(n) + 2n \\ &= 2n \log n + 2n \\ &= 2n(\log n + 1 - 1) + 2n \\ &= 2n(\log n + \log 2 - 1) + 2n \\ &= 2n(\log(2n) - 1) + 2n \end{aligned}$$

Απόδειξη (δεύτερη)

Πράγματι,

$$\begin{aligned} T(2n) &= 2T(n) + 2n \\ &= 2n \log n + 2n \\ &= 2n(\log n + 1 - 1) + 2n \\ &= 2n(\log n + \log 2 - 1) + 2n \\ &= 2n(\log(2n) - 1) + 2n \\ &= 2n \log(2n) \end{aligned}$$

Επιπλέον τρόπος μετρήματος

Εάν $T(n) \begin{cases} = 0 & \text{εάν } n = 1 \\ \leq T(\frac{n}{2}) + T(\frac{n}{2}) + c \cdot n & \text{διαφορετικά} \end{cases}$ τότε $T(n) \leq cn\lceil \log n \rceil$.

Παρατηρούμε ότι

$$\begin{aligned} T(n) &\leq 2T\left(\frac{n}{2}\right) + c \cdot n \\ T\left(\frac{n}{2}\right) &\leq 2T\left(\frac{n}{2^2}\right) + c \cdot \frac{n}{2} \\ T\left(\frac{n}{2^2}\right) &\leq 2T\left(\frac{n}{2^3}\right) + c \cdot \frac{n}{2^2} \\ &\vdots \\ T\left(\frac{n}{2^{k-1}}\right) &\leq 2T\left(\frac{n}{2^k}\right) + c \cdot \frac{n}{2^{k-1}} \end{aligned}$$

Για $n = 2^k$ ισχύει ότι $k = \log n$.

Επιπλέον τρόπος μετρήματος

Εάν $T(n) \begin{cases} = 0 & \text{εάν } n = 1 \\ \leq T(\frac{n}{2}) + T(\frac{n}{2}) + c \cdot n & \text{διαφορετικά} \end{cases}$ τότε $T(n) \leq cn\lceil \log n \rceil$.

Πολλαπλασιάζουμε κάθε γραμμή με κατάλληλη δύναμη του 2:

$$\begin{aligned} T(n) &\leq 2T\left(\frac{n}{2}\right) + c \cdot n \\ 2T\left(\frac{n}{2}\right) &\leq 2^2 T\left(\frac{n}{2^2}\right) + 2c \cdot \frac{n}{2} \\ 2^2 T\left(\frac{n}{2^2}\right) &\leq 2^3 T\left(\frac{n}{2^3}\right) + 2^2 c \cdot \frac{n}{2^2} \\ &\vdots \\ 2^{k-1} T\left(\frac{n}{2^{k-1}}\right) &\leq 2^k T\left(\frac{n}{2^k}\right) + 2^{k-1} c \cdot \frac{n}{2^{k-1}} \end{aligned}$$

Προσθέτοντας προκύπτει ότι

$$T(n) \leq 2^k T\left(\tfrac{n}{2^k}\right) + (c \cdot n)k = cn \log n.$$

Απόδειξη (γενική)

Εάν $T(n) = \begin{cases} 0 & \text{εάν } n = 1 \\ T(\lfloor \frac{n}{2} \rfloor) + T(\lceil \frac{n}{2} \rceil) + n & \text{διαφορετικά} \end{cases}$ τότε $T(n) \leq n\lceil \log n \rceil$.

Απόδειξη με (ισχυρή) επαγωγή.

Απόδειξη (γενική)

Εάν $T(n) = \begin{cases} 0 & \text{εάν } n = 1 \\ T(\lfloor \frac{n}{2} \rfloor) + T(\lceil \frac{n}{2} \rceil) + n & \text{διαφορετικά} \end{cases}$ τότε $T(n) \leq n\lceil \log n \rceil$.

Απόδειξη με (ισχυρή) επαγωγή.

- Βάση της επαγωγής όπως πριν.

Απόδειξη (γενική)

Εάν $T(n) = \begin{cases} 0 & \text{εάν } n = 1 \\ T(\lfloor \frac{n}{2} \rfloor) + T(\lceil \frac{n}{2} \rceil) + n & \text{διαφορετικά} \end{cases}$ τότε $T(n) \leq n\lceil \log n \rceil$.

Απόδειξη με (ισχυρή) επαγωγή.

- Βάση της επαγωγής όπως πριν.
- Για την επαγωγική υπόθεση θα υποθέσουμε ότι η σχέση ισχύει για $k = 1, 2, \ldots, n-1$.

Απόδειξη (γενική)

Εάν $T(n) = \begin{cases} 0 & \text{εάν } n = 1 \\ T(\lfloor \frac{n}{2} \rfloor) + T(\lceil \frac{n}{2} \rceil) + n & \text{διαφορετικά} \end{cases}$ τότε $T(n) \leq n\lceil \log n \rceil$.

Απόδειξη με (ισχυρή) επαγωγή.

- Βάση της επαγωγής όπως πριν.
- Για την επαγωγική υπόθεση θα υποθέσουμε ότι η σχέση ισχύει για $k = 1, 2, \ldots, n-1$.
- Θα δείξουμε ότι $T(n) \leq n\lceil \log n \rceil$.

Απόδειξη (γενική)

Για ευκολία, θέτουμε $n_1 = \lfloor \frac{n}{2} \rfloor$ και $n_2 = \lceil \frac{n}{2} \rceil$.

$$T(n) \quad = \quad T(n_1) + T(n_2) + n$$

Απόδειξη (γενική)

Για ευκολία, θέτουμε $n_1 = \lfloor \frac{n}{2} \rfloor$ και $n_2 = \lceil \frac{n}{2} \rceil$.

$$\begin{aligned} T(n) &= T(n_1) + T(n_2) + n \\ &\leq n_1 \lceil \log n_1 \rceil + n_2 \lceil \log n_2 \rceil + n \end{aligned}$$

Απόδειξη (γενική)

Για ευκολία, θέτουμε $n_1 = \lfloor \frac{n}{2} \rfloor$ και $n_2 = \lceil \frac{n}{2} \rceil$.

$$\begin{aligned} T(n) &= T(n_1) + T(n_2) + n \\ &\leq n_1 \lceil \log n_1 \rceil + n_2 \lceil \log n_2 \rceil + n \\ &\leq n_1 \lceil \log n_2 \rceil + n_2 \lceil \log n_2 \rceil + n \end{aligned}$$

Απόδειξη (γενική)

Για ευκολία, θέτουμε $n_1 = \lfloor \frac{n}{2} \rfloor$ και $n_2 = \lceil \frac{n}{2} \rceil$.

$$\begin{aligned} T(n) &= T(n_1) + T(n_2) + n \\ &\leq n_1 \lceil \log n_1 \rceil + n_2 \lceil \log n_2 \rceil + n \\ &\leq n_1 \lceil \log n_2 \rceil + n_2 \lceil \log n_2 \rceil + n \\ &= n \lceil \log n_2 \rceil + n \end{aligned}$$

Απόδειξη (γενική)

Για ευκολία, θέτουμε $n_1 = \lfloor \frac{n}{2} \rfloor$ και $n_2 = \lceil \frac{n}{2} \rceil$.

$$\begin{aligned} T(n) &= T(n_1) + T(n_2) + n \\ &\leq n_1 \lceil \log n_1 \rceil + n_2 \lceil \log n_2 \rceil + n \\ &\leq n_1 \lceil \log n_2 \rceil + n_2 \lceil \log n_2 \rceil + n \\ &= n \lceil \log n_2 \rceil + n \\ &\leq n(\lceil \log n \rceil - 1) + n \end{aligned}$$

Απόδειξη (γενική)

Για ευκολία, θέτουμε $n_1 = \lfloor \frac{n}{2} \rfloor$ και $n_2 = \lceil \frac{n}{2} \rceil$.

$$\begin{aligned} T(n) &= T(n_1) + T(n_2) + n \\ &\leq n_1 \lceil \log n_1 \rceil + n_2 \lceil \log n_2 \rceil + n \\ &\leq n_1 \lceil \log n_2 \rceil + n_2 \lceil \log n_2 \rceil + n \\ &= n \lceil \log n_2 \rceil + n \\ &\leq n(\lceil \log n \rceil - 1) + n \\ &= n \lceil \log n \rceil \end{aligned}$$

Απόδειξη (γενική)

Για ευκολία, θέτουμε $n_1 = \lfloor \frac{n}{2} \rfloor$ και $n_2 = \lceil \frac{n}{2} \rceil$.

$$\begin{aligned} T(n) &= T(n_1) + T(n_2) + n \\ &\leq n_1 \lceil \log n_1 \rceil + n_2 \lceil \log n_2 \rceil + n \\ &\leq n_1 \lceil \log n_2 \rceil + n_2 \lceil \log n_2 \rceil + n \\ &= n \lceil \log n_2 \rceil + n \\ &\leq n(\lceil \log n \rceil - 1) + n \\ &= n \lceil \log n \rceil \end{aligned}$$

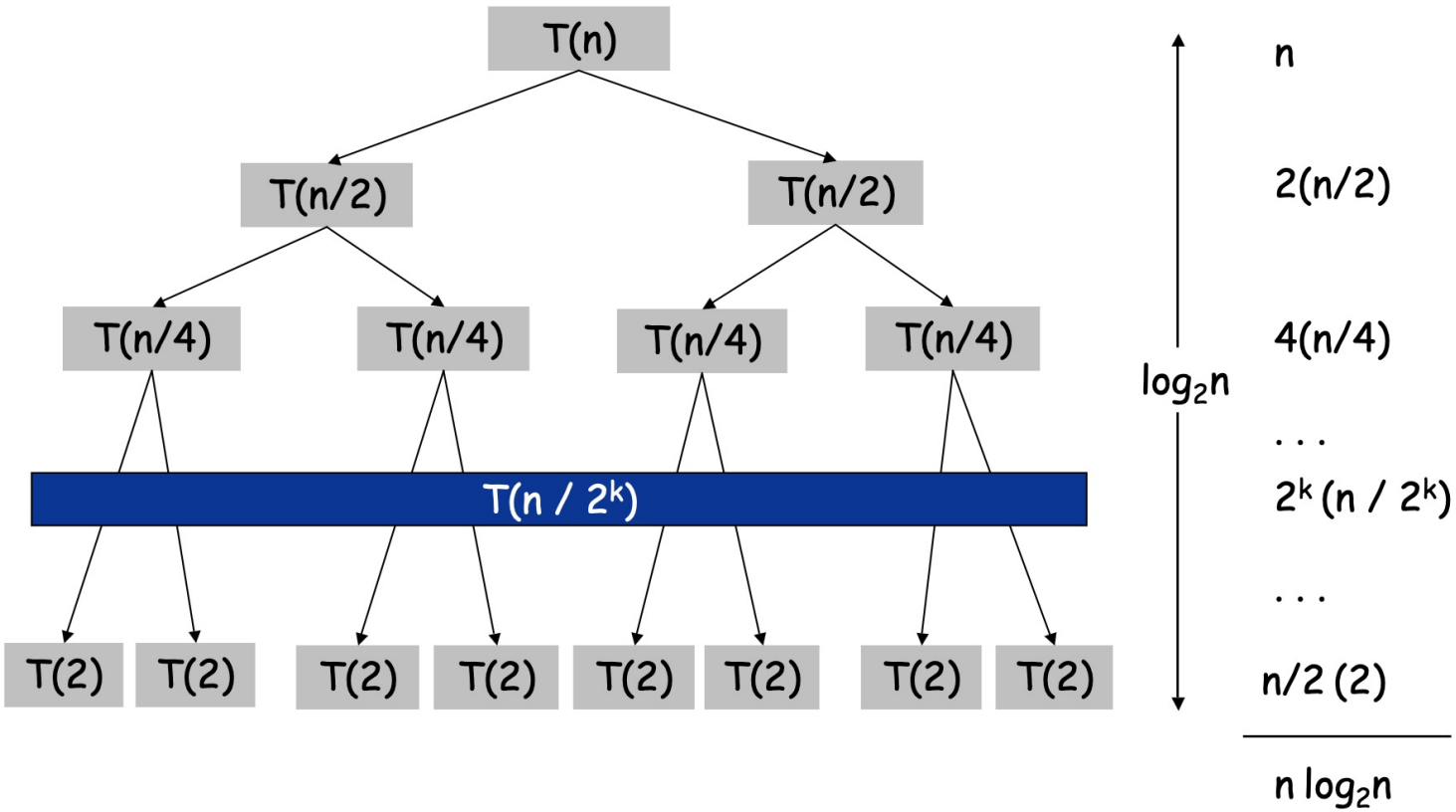
(Ισχύει ότι $n_2 = \left\lceil \frac{n}{2} \right\rceil \leq \left\lceil \frac{2^{\lceil \log n \rceil}}{2} \right\rceil = \frac{2^{\lceil \log n \rceil}}{2}$ και συνεπώς $\log n_2 \leq \log(\frac{2^{\lceil \log n \rceil}}{2}) = \lceil \log n \rceil - 1$.)

Ας το δούμε ακόμα πιο γενικά...

Ας υποθέσουμε αρχικά ότι το n είναι δύναμη του 2.

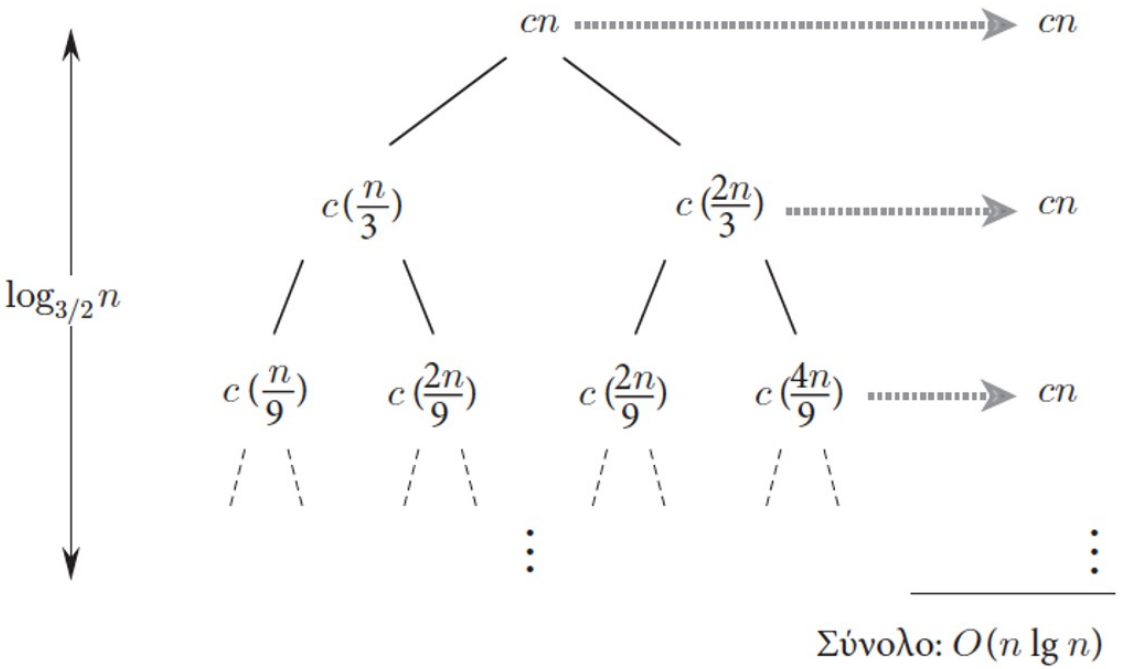
Ας το δούμε ακόμα πιο γενικά...

Ας υποθέσουμε αρχικά ότι το n είναι δύναμη του 2.



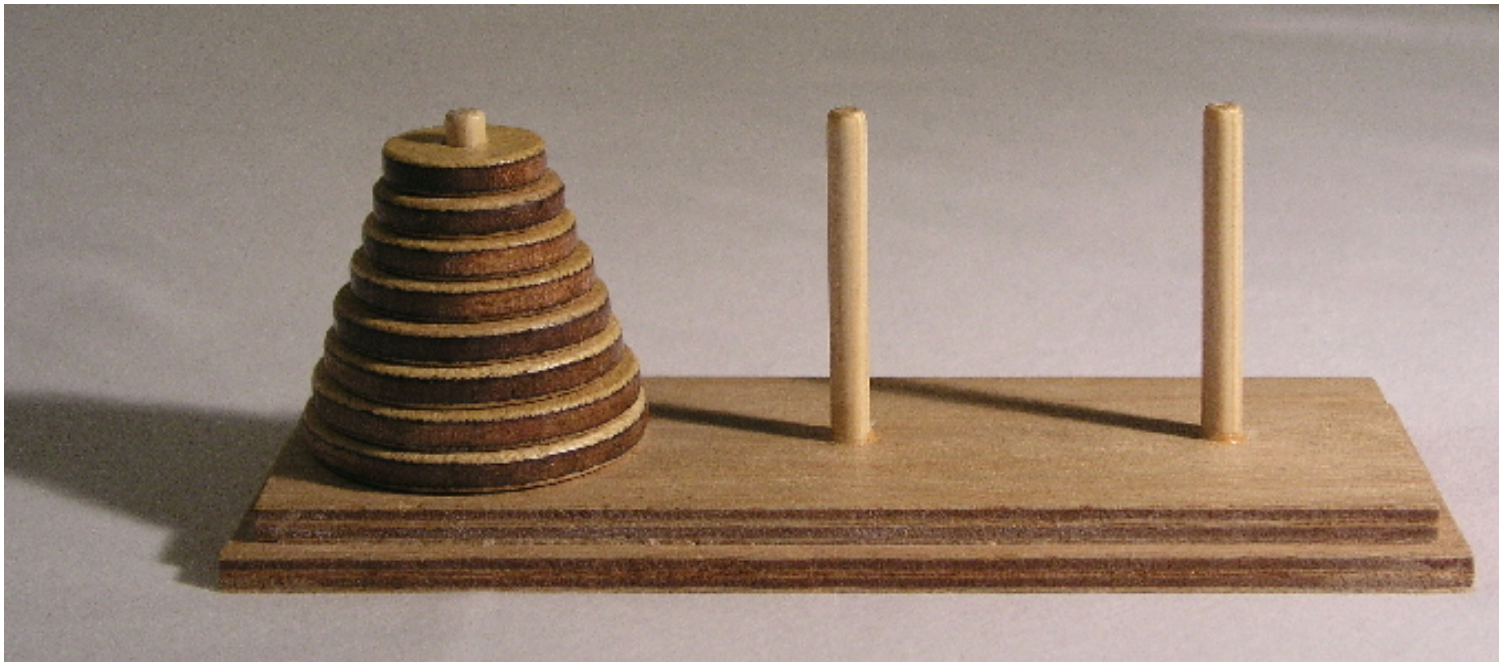
Παράδειγμα

$$T(n) = T(\tfrac{n}{3}) + T(\tfrac{2n}{3}) + cn$$



Σχήμα 4.6 Ένα δένδρο αναδρομής για την αναδρομική σχέση $T(n) = T(n/3) + T(2n/3) + cn$.

Ο πύργος του Hanoi



Για παράδειγμα...

Το πρόβλημα

Είσοδος: 3 στύλοι (1,2,3) και n δίσκοι τοποθετημένοι στον δίσκο 1 σε κωνική μορφή.
Ζητούμενο: Να μεταφερθούν όλοι στον στύλο 3 διατηρώντας την ίδια μορφή.

Το πρόβλημα

Είσοδος: 3 στύλοι (1,2,3) και n δίσκοι τοποθετημένοι στον δίσκο 1 σε κωνική μορφή.
Ζητούμενο: Να μεταφερθούν όλοι στον στύλο 3 διατηρώντας την ίδια μορφή.

Κανόνες:

Το πρόβλημα

Είσοδος: 3 στύλοι (1,2,3) και n δίσκοι τοποθετημένοι στον δίσκο 1 σε κωνική μορφή.
Ζητούμενο: Να μεταφερθούν όλοι στον στύλο 3 διατηρώντας την ίδια μορφή.

Κανόνες:

- Σε κάθε γύρο μετακινούμε μόνο έναν δίσκο.

Το πρόβλημα

Είσοδος: 3 στύλοι (1,2,3) και n δίσκοι τοποθετημένοι στον δίσκο 1 σε κωνική μορφή.
Ζητούμενο: Να μεταφερθούν όλοι στον στύλο 3 διατηρώντας την ίδια μορφή.

Κανόνες:

- Σε κάθε γύρο μετακινούμε μόνο έναν δίσκο.

- Σε κάθε κίνηση αφαιρούμε τον δίσκο που βρίσκεται στην κορυφή κάποιου στύλου και τον τοποθετούμε στην κορυφή κάποιου άλλου στύλου.

Το πρόβλημα

Είσοδος: 3 στύλοι (1,2,3) και n δίσκοι τοποθετημένοι στον δίσκο 1 σε κωνική μορφή.
Ζητούμενο: Να μεταφερθούν όλοι στον στύλο 3 διατηρώντας την ίδια μορφή.

Κανόνες:

- Σε κάθε γύρο μετακινούμε μόνο έναν δίσκο.
- Σε κάθε κίνηση αφαιρούμε τον δίσκο που βρίσκεται στην κορυφή κάποιου στύλου και τον τοποθετούμε στην κορυφή κάποιου άλλου στύλου.
- Δεν επιτρέπεται ένας δίσκος να τοποθετηθεί πάνω σε έναν άλλο δίσκο με μικρότερη διάμετρο από αυτόν.

Λύση του προβλήματος

- Αν $n = 1$ τότε το πρόβλημα μας λύνεται σε ένα βήμα (μεταφέροντας το δίσκο από το στύλο 1 στο στύλο 3).

Λύση του προβλήματος

- Αν $n = 1$ τότε το πρόβλημα μας λύνεται σε ένα βήμα (μεταφέροντας το δίσκο από το στύλο 1 στο στύλο 3).
- Έστω ότι ξέρουμε να λύνουμε το πρόβλημα για $n - 1$ δίσκους.

Λύση του προβλήματος

- Αν $n = 1$ τότε το πρόβλημα μας λύνεται σε ένα βήμα (μεταφέροντας το δίσκο από το στύλο 1 στο στύλο 3).
- Έστω ότι ξέρουμε να λύνουμε το πρόβλημα για $n - 1$ δίσκους.
- Μεταφέρουμε τους $n - 1$ δίσκους στο στύλο 2, μεταφέρουμε τον n-οστό δίσκο στο στύλο 3 και έπειτα μεταφέρουμε τους $n - 1$ δίσκους που βρίσκονται στο στύλο 2 στο στύλο 3 πάνω από τον n-οστό δίσκο.

Λύση του προβλήματος

- Αν $n = 1$ τότε το πρόβλημα μας λύνεται σε ένα βήμα (μεταφέροντας το δίσκο από το στύλο 1 στο στύλο 3).
- Έστω ότι ξέρουμε να λύνουμε το πρόβλημα για $n - 1$ δίσκους.
- Μεταφέρουμε τους $n - 1$ δίσκους στο στύλο 2, μεταφέρουμε τον n-οστό δίσκο στο στύλο 3 και έπειτα μεταφέρουμε τους $n - 1$ δίσκους που βρίσκονται στο στύλο 2 στο στύλο 3 πάνω από τον n-οστό δίσκο.

Πόσα βήματα παίρνει;

Λύση του προβλήματος

- Αν $n = 1$ τότε το πρόβλημα μας λύνεται σε ένα βήμα (μεταφέροντας το δίσκο από το στύλο 1 στο στύλο 3).
- Έστω ότι ξέρουμε να λύνουμε το πρόβλημα για $n - 1$ δίσκους.
- Μεταφέρουμε τους $n - 1$ δίσκους στο στύλο 2, μεταφέρουμε τον n-οστό δίσκο στο στύλο 3 και έπειτα μεταφέρουμε τους $n - 1$ δίσκους που βρίσκονται στο στύλο 2 στο στύλο 3 πάνω από τον n-οστό δίσκο.

Πόσα βήματα παίρνει;

$$T(n) = T(n-1) + 1 + T(n-1) = 2T(n-1) + 1$$

Λύση του προβλήματος

- Αν $n = 1$ τότε το πρόβλημα μας λύνεται σε ένα βήμα (μεταφέροντας το δίσκο από το στύλο 1 στο στύλο 3).
- Έστω ότι ξέρουμε να λύνουμε το πρόβλημα για $n - 1$ δίσκους.
- Μεταφέρουμε τους $n - 1$ δίσκους στο στύλο 2, μεταφέρουμε τον n-οστό δίσκο στο στύλο 3 και έπειτα μεταφέρουμε τους $n - 1$ δίσκους που βρίσκονται στο στύλο 2 στο στύλο 3 πάνω από τον n-οστό δίσκο.

Πόσα βήματα παίρνει;

$$T(n) = T(n-1) + 1 + T(n-1) = 2T(n-1) + 1$$

και άρα

$$T(n) = 2^n - 1.$$

Θεώρημα Κυρίαρχου Όρου (Master Theorem)

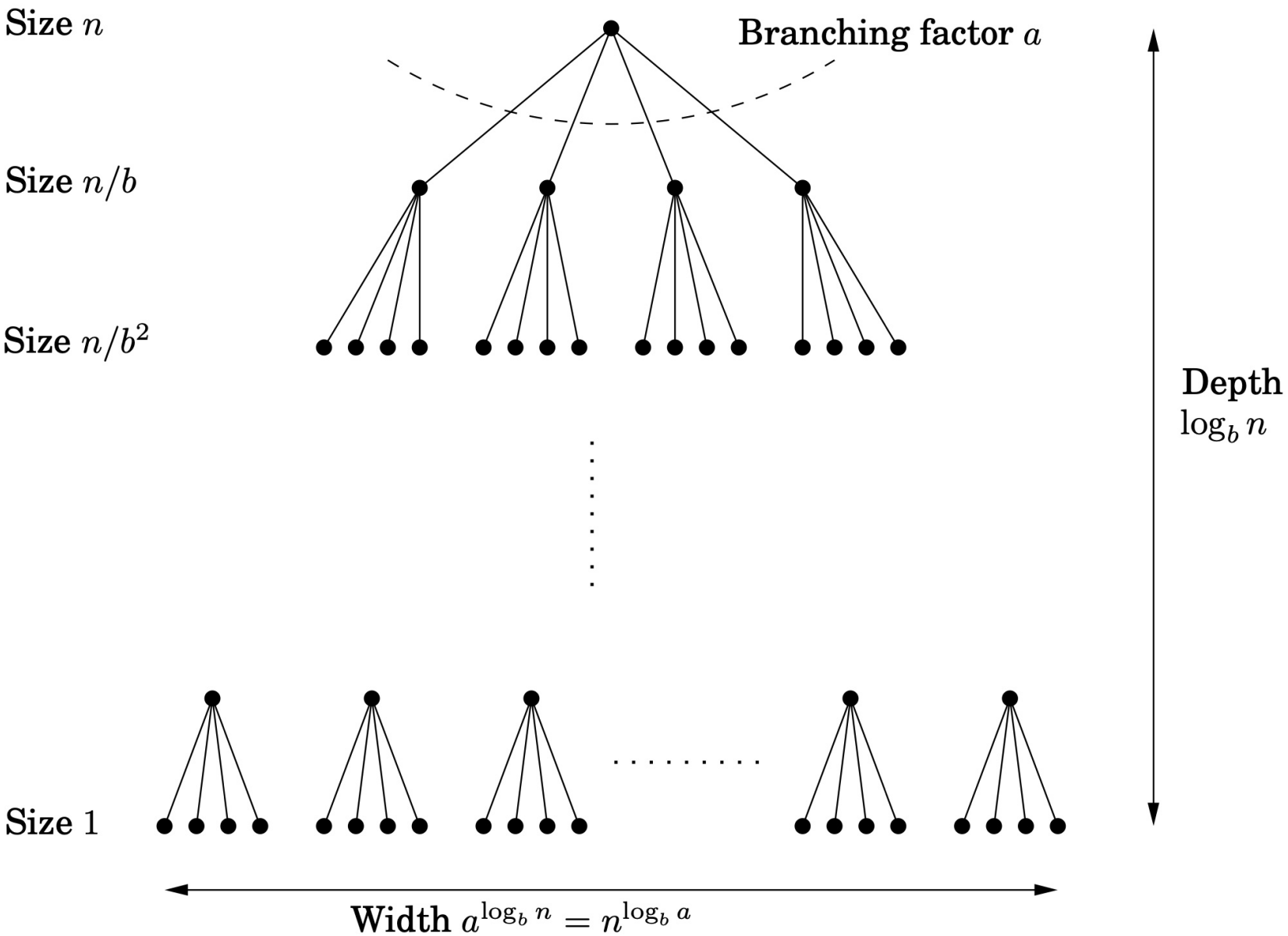
Έστω $a > 0$, $b \geq 1$, $d \geq 0$ σταθερές και $T(n)$ ορισμένη στους μη-αρνητικούς ακέραιους ως

$$T(n) = aT\left(\frac{n}{b}\right) + \mathcal{O}(n^d).$$

Τότε:

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

Ιδέα


Size n
Branching factor a
Size n/b
Size n/b^2
Depth
$\log_b n$
Size 1
Width $a^{\log_b n} = n^{\log_b a}$

Ιδέα

Στο k-οστό επίπεδο έχουμε:

- a^k υποπροβλήματα μεγέθους $\frac{n}{b^k}$ που το καθένα απαιτεί $\mathcal{O}\left(\left(\frac{n}{b^k}\right)^d\right)$ βήματα.

Ιδέα

Στο k-οστό επίπεδο έχουμε:

- a^k υποπροβλήματα μεγέθους $\frac{n}{b^k}$ που το καθένα απαιτεί $\mathcal{O}\left(\left(\frac{n}{b^k}\right)^d\right)$ βήματα.
- άρα ο χρόνος που απαιτείται είναι $a^k \times \mathcal{O}\left(\left(\frac{n}{b^k}\right)^d\right) = \mathcal{O}(n^d) \times \left(\frac{a}{b^d}\right)^k$.

Συνολικά

$$\sum_{k=0}^{\log_b n} \mathcal{O}(n^d) \times \left(\frac{a}{b^d}\right)^k = \mathcal{O}(n^d) \times \sum_{k=0}^{\log_b n} \left(\frac{a}{b^d}\right)^k.$$

Ιδέα

Στο k-οστό επίπεδο έχουμε:

- a^k υποπροβλήματα μεγέθους $\frac{n}{b^k}$ που το καθένα απαιτεί $\mathcal{O}\left(\left(\frac{n}{b^k}\right)^d\right)$ βήματα.
- άρα ο χρόνος που απαιτείται είναι $a^k \times \mathcal{O}\left(\left(\frac{n}{b^k}\right)^d\right) = \mathcal{O}(n^d) \times \left(\frac{a}{b^d}\right)^k$.

Συνολικά

$$\sum_{k=0}^{\log_b n} \mathcal{O}(n^d) \times \left(\frac{a}{b^d}\right)^k = \mathcal{O}(n^d) \times \sum_{k=0}^{\log_b n} \left(\frac{a}{b^d}\right)^k.$$

Το τελικό αποτέλεσμα εξαρτάται από τον λόγο $\frac{a}{b^d}$.

Ιδέα

Ισχύει ότι:

$$\sum_{k=0}^{\log_b n} \left(\frac{a}{b^d}\right)^k = \begin{cases} \mathcal{O}(1) & \frac{a}{b^d} < 1 \\ \log_b n + 1 & \frac{a}{b^d} = 1 \\ \mathcal{O}\left(\left(\frac{a}{b^d}\right)^{\log_b n}\right) = \mathcal{O}\left(\frac{n^{\log_b a}}{n^d}\right) & \frac{a}{b^d} > 1 \end{cases}$$

Και άρα

$$\mathcal{O}(n^d) \times \sum_{k=0}^{\log_b n} \left(\frac{a}{b^d}\right)^k = \begin{cases} \mathcal{O}(n^d) \cdot \mathcal{O}(1) = \mathcal{O}(n^d) & \frac{a}{b^d} < 1 \\ \mathcal{O}(n^d)(\log_b n + 1) = \mathcal{O}(n^d \log n) & \frac{a}{b^d} = 1 \\ \mathcal{O}(n^d) \cdot \mathcal{O}\left(\frac{n^{\log_b a}}{n^d}\right) = \mathcal{O}(n^{\log_b a}) & \frac{a}{b^d} > 1 \end{cases}$$