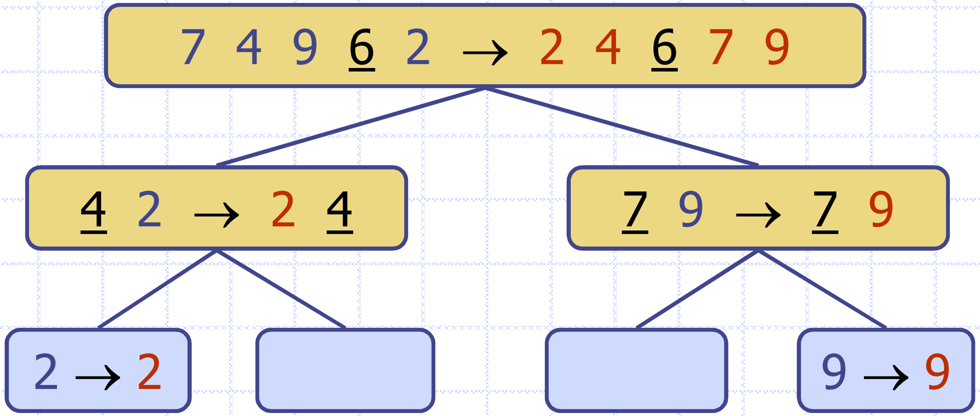
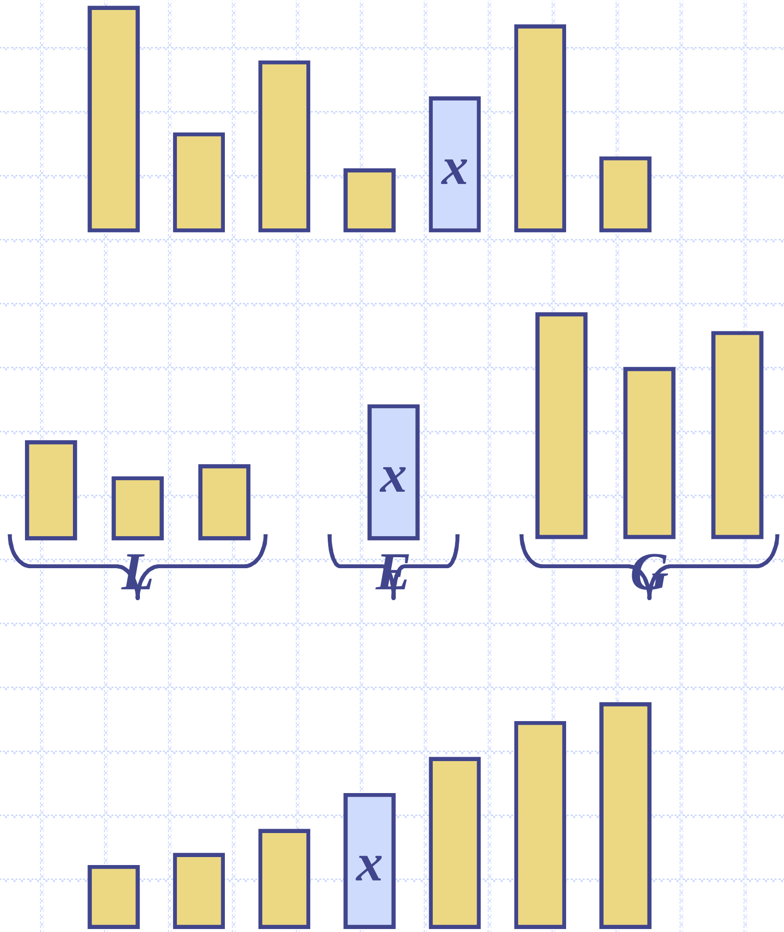
Γρήγορη Ταξινόμηση

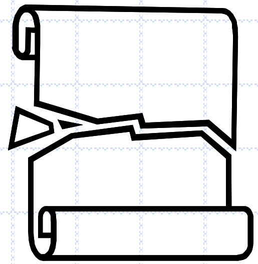


Γρήγορη Ταξινόμηση

- Η γρήγορη ταξινόμηση είναι ένας τυχαίος αλγόριθμος ταξινόμησης που βασίζεται στο παράδειγμα διαίρει και κυρίευε:
  - Διαίρει: επιλέγεται τυχαία ένα στοιχείο x (λέγεται pivot) και διαμερίζουμε την S σε
    - L τα μικρότερα του x στοιχεία
    - E τα ίσα με το x στοιχεία
    - G τα μεγαλύτερα του x στοιχεία
  - Αναδρομή: ταξινόμηση των L και G
  - Κυρίευε: συνένωση των L, E και G



Διαμέριση

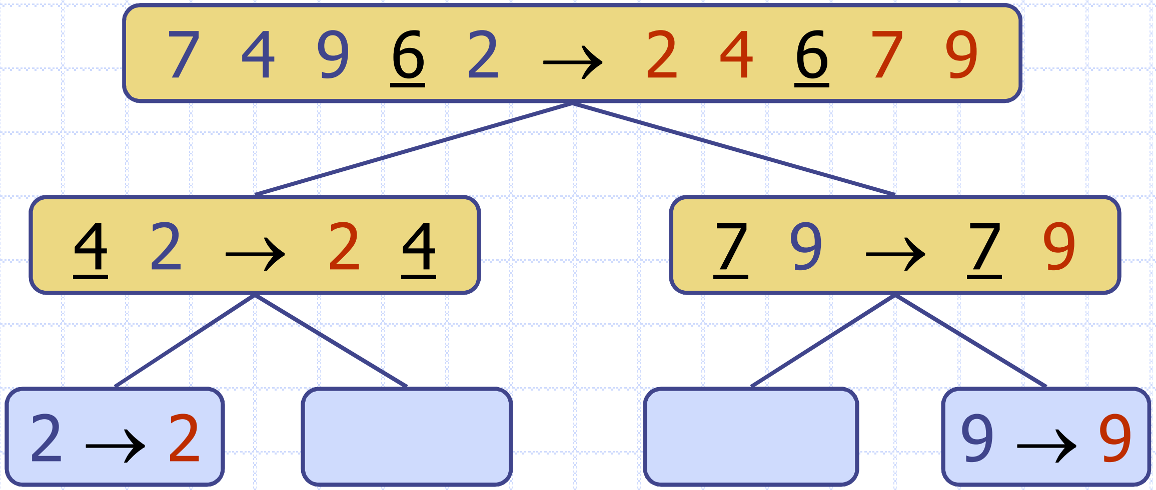


- Διαμερίζουμε μια ακολουθία εισόδου σε:
  - Διαγράφουμε, με την σειρά, κάθε στοιχείο y από την S και
  - Εισάγουμε το y στις L, E ή G, ανάλογα με το αποτέλεσμα ης σύγκρισης με το pivot x
- Κάθε εισαγωγή και διαγραφή γίνεται στην αρχή ή το τέλος μιας ακολουθίας, και επομένως απαιτεί χρόνο $O(1)$
- Επομένως, το βήμα της διαμέρισης απαιτεί χρόνο $O(n)$

```
Algorithm partition(S, p)
    Input sequence S, position p of pivot
    Output subsequences L, E, G of the
        elements of S less than, equal to,
        or greater than the pivot, resp.
    L, E, G ← empty sequences
    x ← S.remove(p)
    while ¬S.isEmpty()
        y ← S.remove(S.first())
        if y < x
            L.addLast(y)
        else if y = x
            E.addLast(y)
        else { y > x }
            G.addLast(y)
    return L, E, G
```

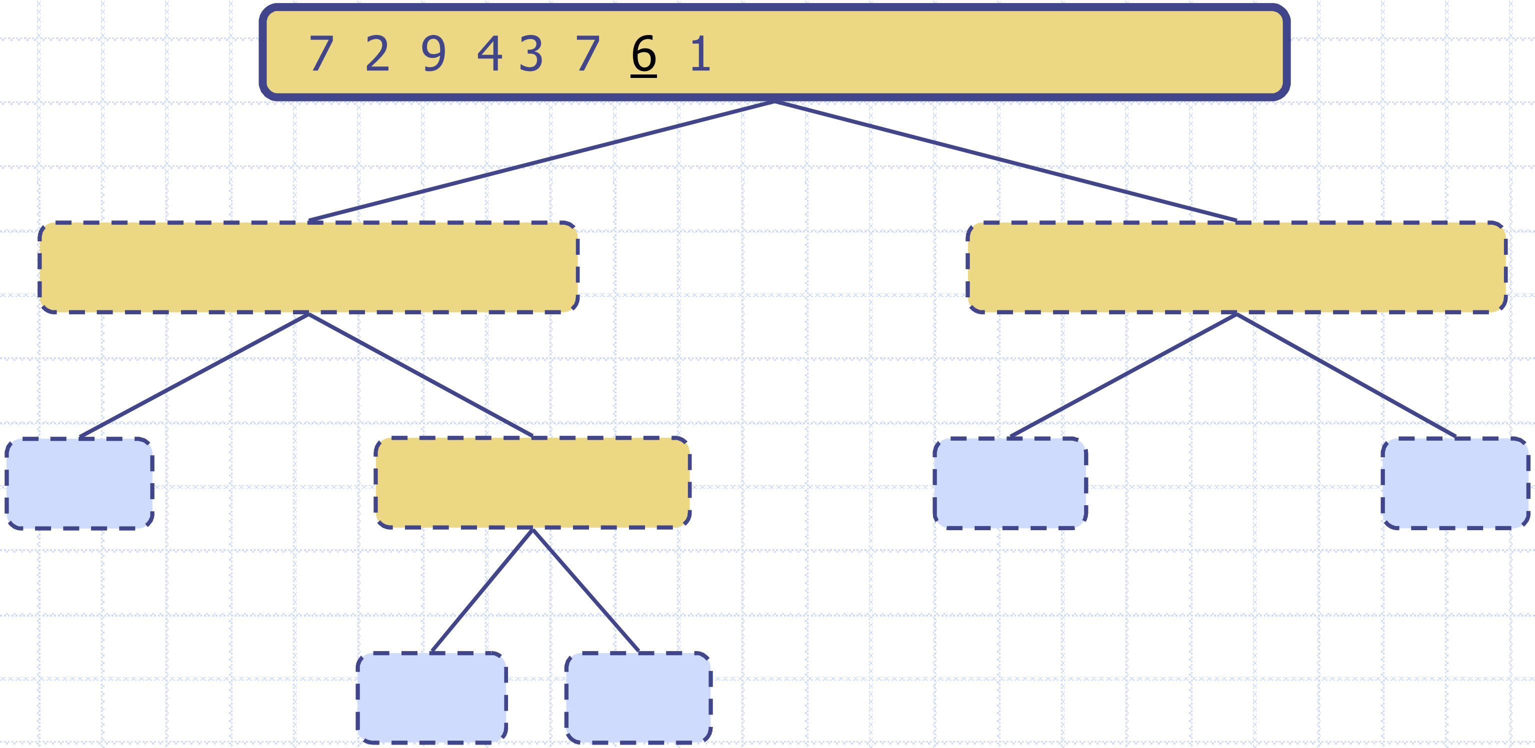
Δένδρο Γρήγορης Ταξινόμησης

- Μια εκτέλεση της γρήγορης ταξινόμησης απεικονίζεται με ένα δυαδικό δένδρο
  - Κάθε κόμβος παριστάνει μια αναδρομική κλήση της γρήγορης ταξινόμησης και αποθηκεύει
    - Μη αποθηκευμένη ακολουθία πριν την εκτέλεση και το pivot
    - Στο τέλος της εκτέλεσης μια ταξινομημένη ακολουθία
  - Η ρίζα είναι η αρχική κλήση
  - Τα φύλλα είναι κλήσεις σε υποακολουθίες μεγέθους 0 or 1



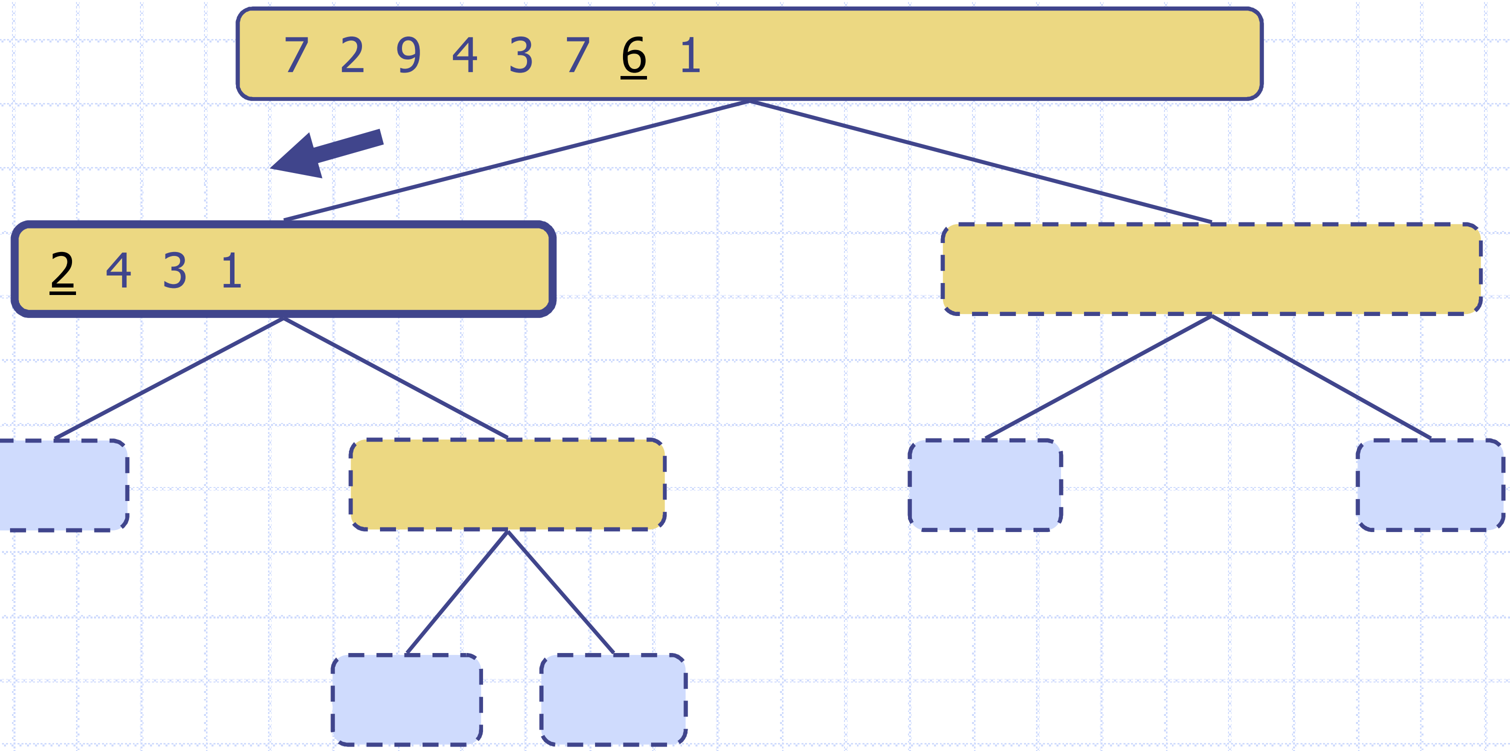
Παράδειγμα Εκτέλεσης

- Επιλογή Pivot



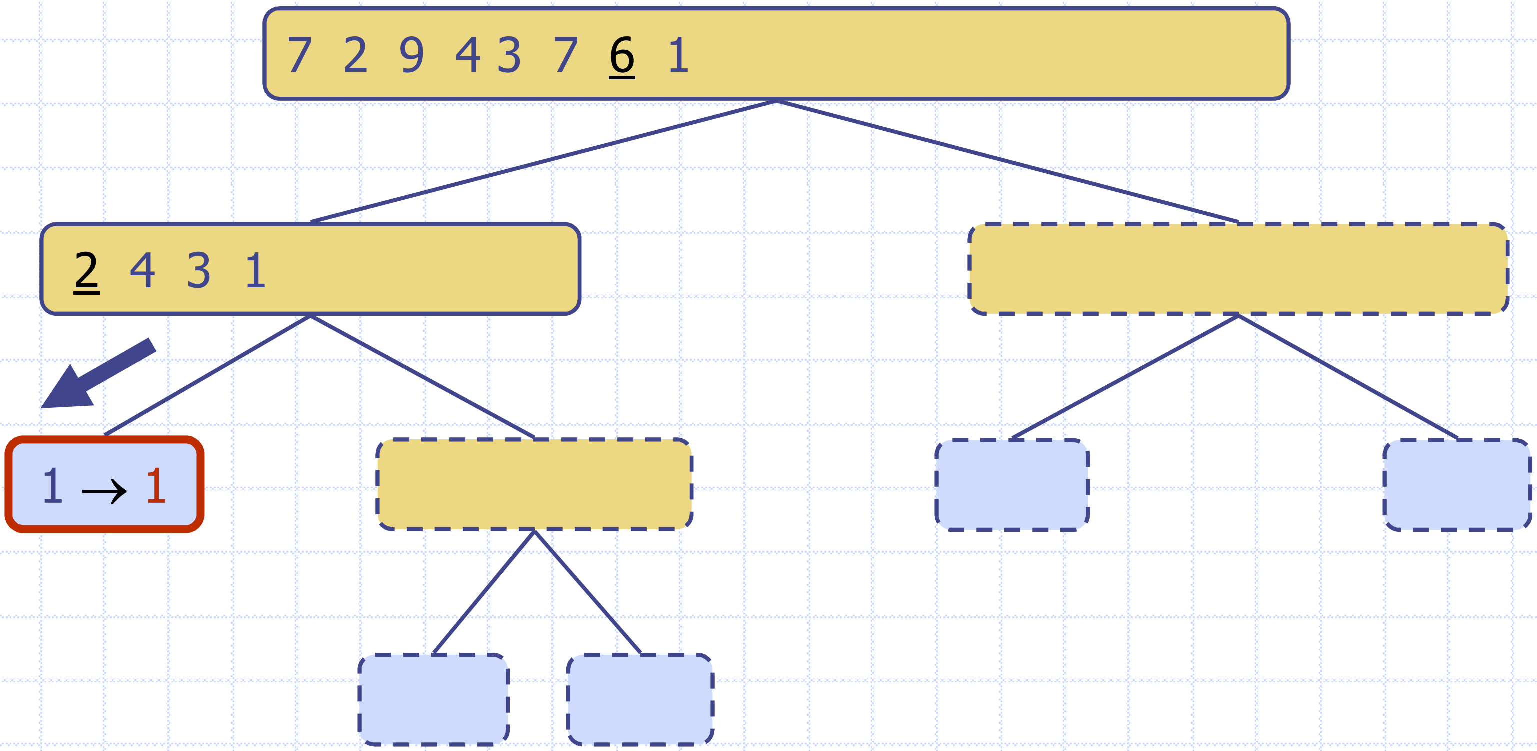
Παράδειγμα Εκτέλεσης (συν.)

- Διαμέριση, αναδρομική κλήση, επιλογή pivot



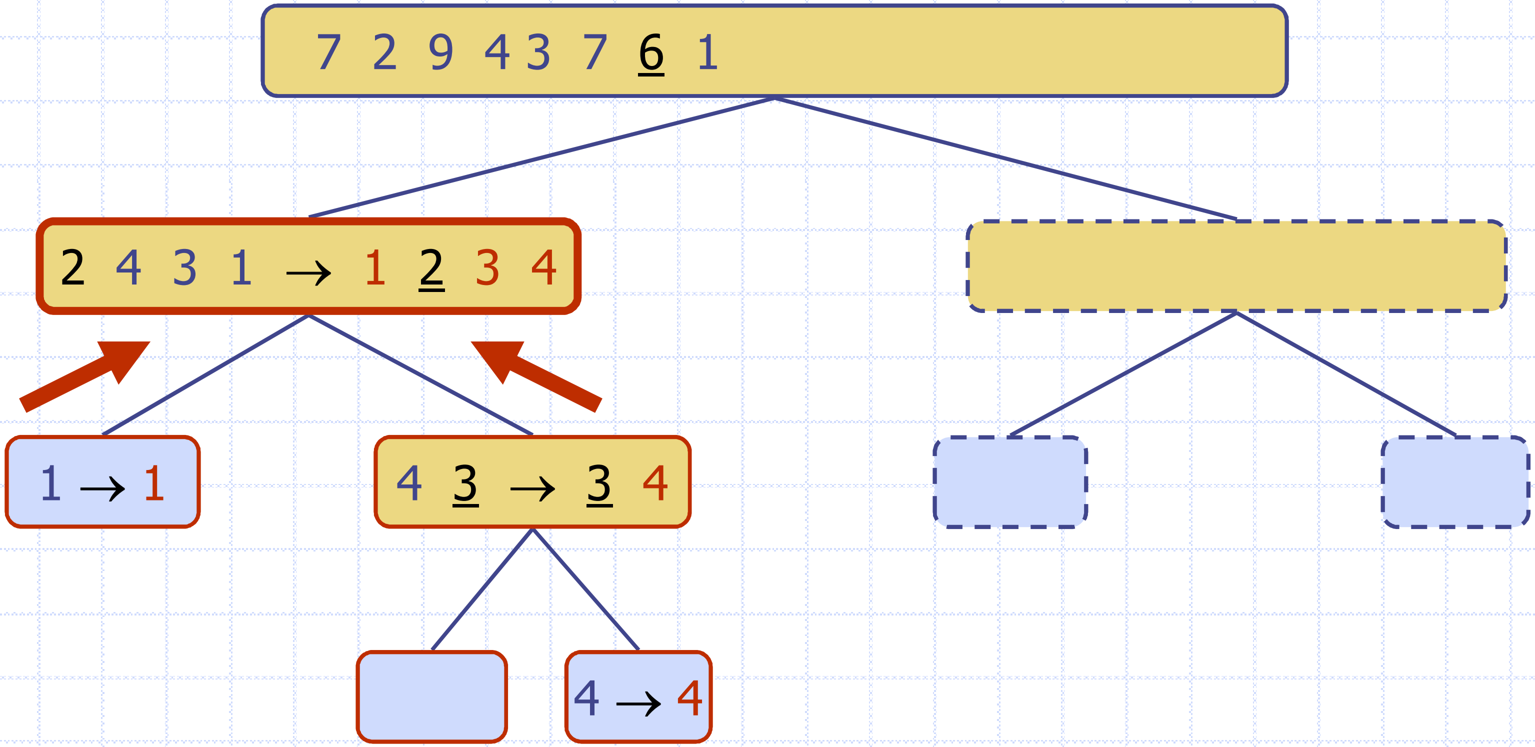
Παράδειγμα Εκτέλεσης (συν.)

- Διαμέριση, αναδρομική κλήση, περίπτωση βάσης



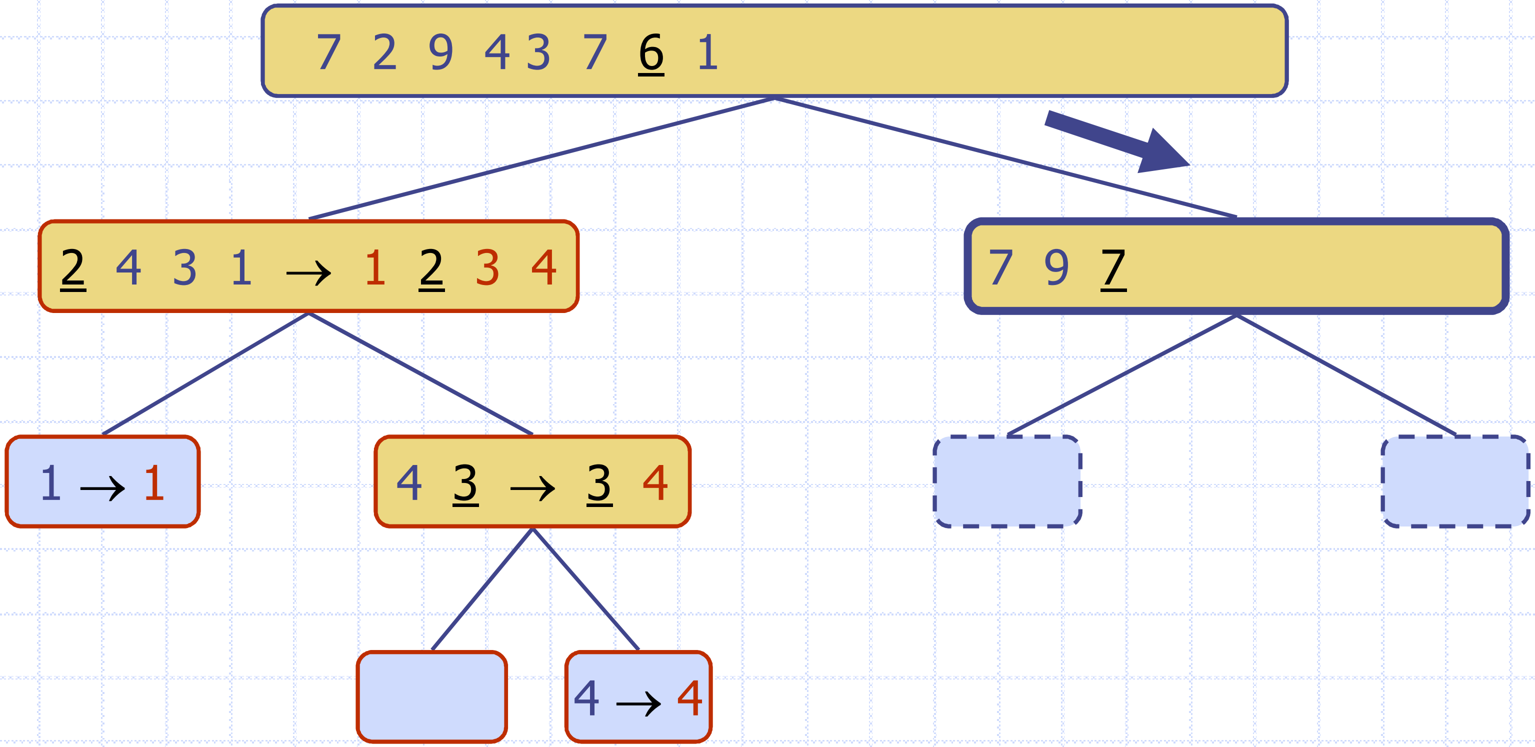
Παράδειγμα Εκτέλεσης (συν.)

- Αναδρομική κλήση, ..., περίπτωση βάσης, συνένωση



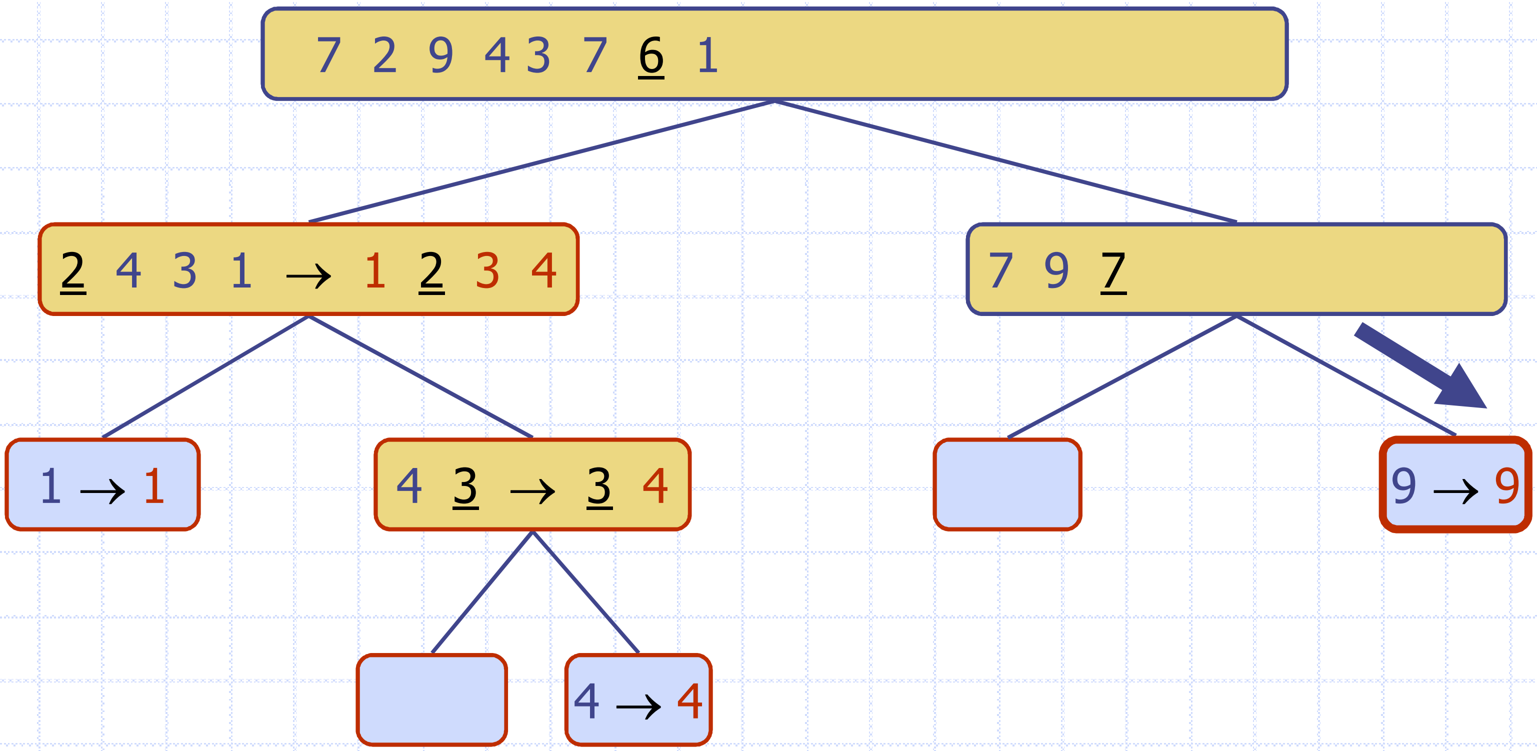
Παράδειγμα Εκτέλεσης (συν.)

- Αναδρομική κλήση, επιλογή pivot



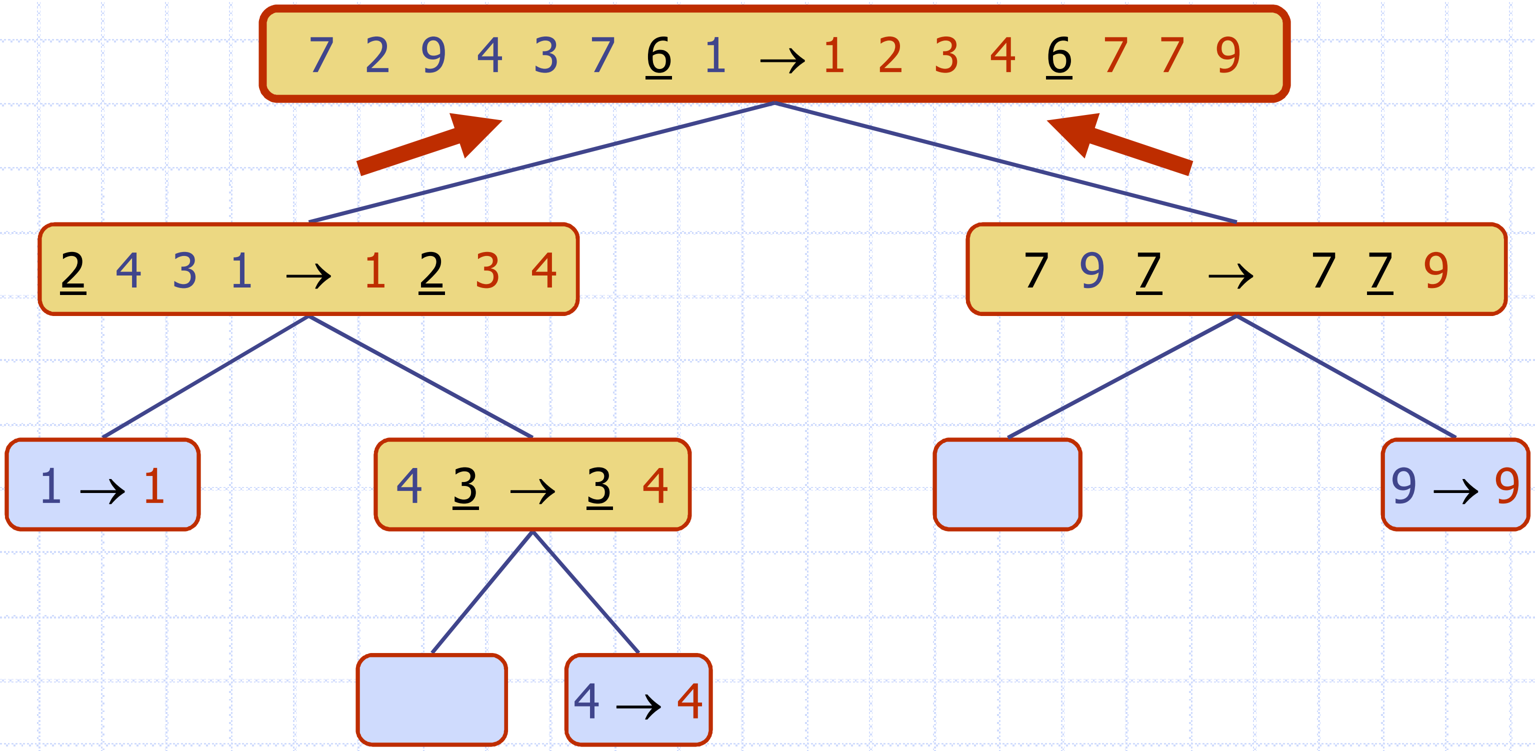
Παράδειγμα Εκτέλεσης (συν.)

- Διαμέριση, …, αναδρομική κλήση, περίπτωση βάσης



Παράδειγμα Εκτέλεσης (συν.)

- Συνένωση



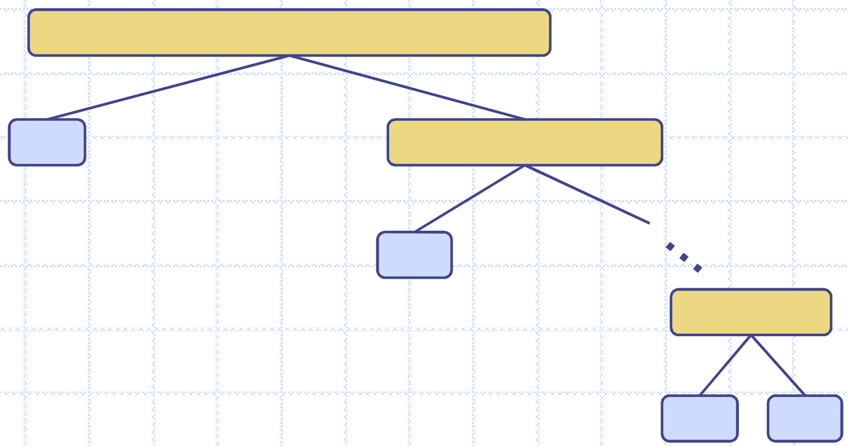
Χειρότερη Περίπτωση Χρόνου Τρεξίματος

- Η χειρότερη περίπτωση της γρήγορης ταξινόμησης συμβαίνει όταν το pivot είναι το μοναδικό ελάχιστο ή μέγιστο στοιχείο
- Μια από τις $\boldsymbol{L}$ και $\boldsymbol{G}$ έχει μέγεθος $\boldsymbol{n} - 1$ και η άλλη 0
- Ο χρόνος τρεξίματος είναι ανάλογος του αθροίσματος

$$\boldsymbol{n} + (\boldsymbol{n} - 1) + \ldots + 2 + 1$$

- Επομένως, ο χειρότερος χρόνος τρεξίματος είναι $\boldsymbol{O}(\boldsymbol{n}^2)$

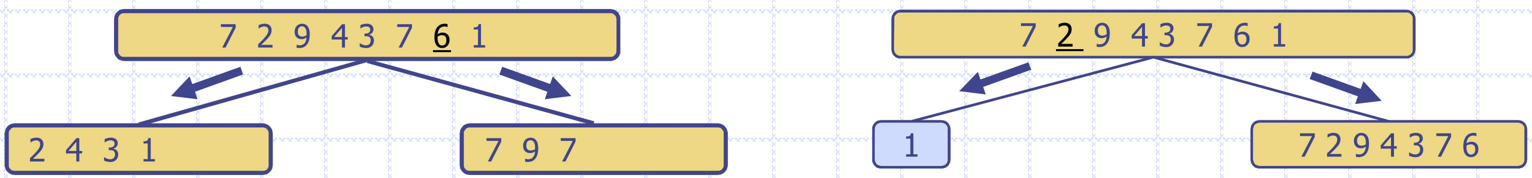
βάθος	χρόνος
0	$\boldsymbol{n}$
1	$\boldsymbol{n} - 1$
…	…
$\boldsymbol{n} - 1$	1



Αναμενόμενος Χρόνος Τρεξίματος

- Έστω μια αναδρομική κλήση της γρήγορης ταξινόμησης σε μια ακολουθία μεγέθους s
  - **Καλή κλήση:** τα μεγέθη των L και G είναι το καθένα μικρότερο από $3s/4$
  - **Κακή κλήση:** μια από τις L και G έχει μέγεθος μεγαλύτερο από $3s/4$

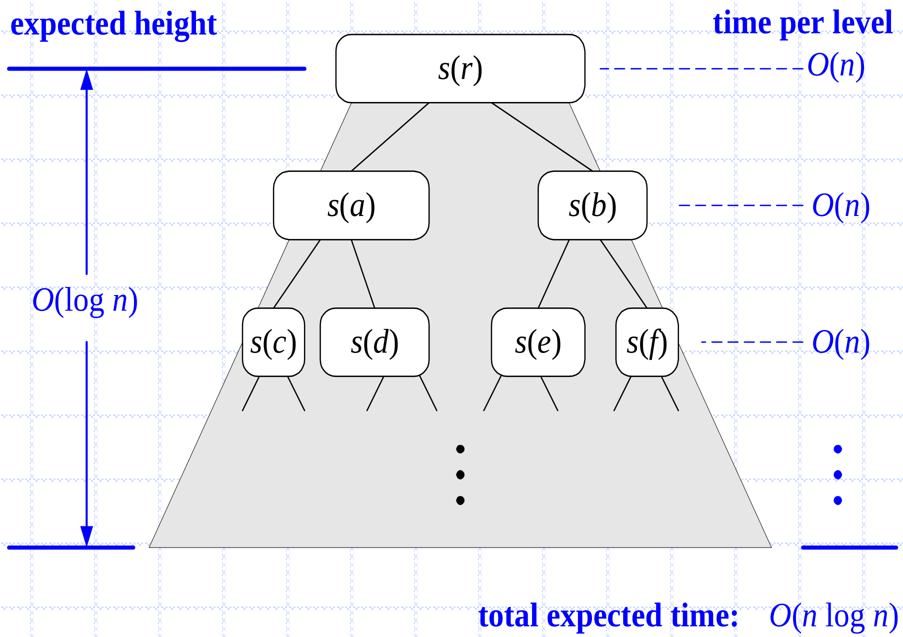
Καλή Κλήση **Κακή Κλήση**



- Μια κλήση είναι καλή με πιθανότητα $1/2$
  - 1/2 τα πιθανά pivots που προκαλούν καλές κλήσεις:

Αναμενόμενος Χρόνος Τρεξίματος,

- Πιθανοτικό γεγονός: Το αναμενόμενο πλήθος ρίψεων ενός νομίσματος για να πάρουμε k κορώνες είναι $2k$
- Για ένα κόμβο βάθους i, αναμένουμε
  - $i/2$ πρόγονοι είναι καλές κλήσεις
  - Το μέγεθος της ακολουθίας εισόδου για την τρέχουσα κλήση είναι το πολύ $(3/4)^{i/2}n$
- Επομένως, έχουμε
  - Για ένα κόμβο βάθους $2\log_{4/3}n$, το αναμενόμενο μέγεθος της εισόδου είναι ένα
  - Το αναμενόμενο ύψος του δένδρου γρήγορης ταξινόμησης είναι $O(\log n)$
- Η εργασία που γίνεται στους κόμβους ίδιου βάθους είναι $O(n)$
- Επομένως, ο αναμενόμενος χρόνος τρεξίματος είναι $O(n \log n)$



Ένα στοιχείο τίθεται στη θέση j. Οπότε το αναμενόμενο κόστος είναι το μέσο κόστος ταξινόμησης μιας ακολουθίας με j-1 στοιχεία και μιας με n-j

$$T = T_{avg}(j-1) + T_{avg}(n-j)$$

$$T_{avg}(n) \le cn + \frac{1}{n}\sum_{j=1}^{n}(T_{avg}(j-1) + T_{avg}(n-j)) \quad n \ge 2 \qquad = cn + \frac{2}{n}\sum_{j=0}^{n-1}(T_{avg}(j))$$

Υποθέτουμε $T_{avg}(0) \le b$ και $T_{avg}(1) \le b$ για κάποια σταθερά b

Για n=2: $T_{avg} \le 2c+2b \le kn\log_e 2$ για $1 \le n \le m$

$$\textit{Υπόθεση}: T_{avg}(m) \le kn \log_e n \;\; \textit{για}\; 1 \le n < m$$

$$T_{avg}(m) \le cm + \frac{4b}{m} + \frac{2}{m}\sum_{j=2}^{m-1}(T_{avg}(j))$$

Η συνάρτηση $j\log_e j$ είναι αύξουσα συνάρτηση

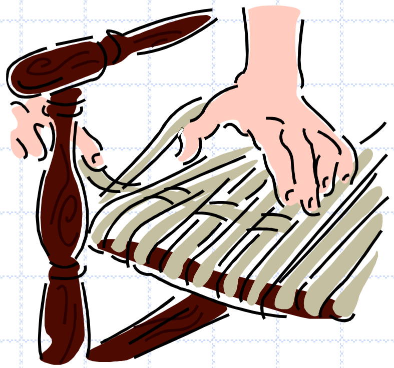
$$T_{avg}(m) \le cm + \frac{4b}{m} + \frac{2k}{m}\int_2^m x\log_e x\,dx$$

$$= cm + \frac{4b}{m} + \frac{2k}{m}\left[\frac{m^2\log_e m}{2} - \frac{m^2}{4}\right]$$

$$= cm + \frac{4b}{m} + km\log_e m - \frac{km}{2}$$

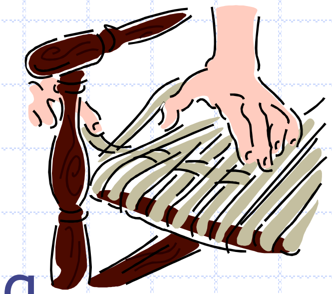
$$\le km\log_e m \quad \gamma\iota\alpha\; m \ge 2$$

Γρήγορη Ταξινόμηση Θέσης



- Η γρήγορη ταξινόμηση μπορεί να υλοποιηθεί στις ίδιες θέσεις θέσεις
- Στο βήμα της διαμέρισης, χρησιμοποιούμε πράξεις αντικατάστασης για αναδιάταξη των στοιχείων της ακολουθίας εισόδου έτσι ώστε
  - τα μικρότερα του pivot στοιχεία να έχουν βαθμό μικρότερο από ***h***
  - τα στοιχεία που είναι ίσα με το pivot έχουν βαθμό μεταξύ ***h*** και ***k***
  - τα μεγαλύτερα του pivot στοιχεία έχουν βαθμό μεγαλύτερο από ***k***
- Οι αναδρομικές κλήσεις εξετάζουν
  - στοιχεία με βαθμό μικρότερο από ***h***
  - Στοιχεία με βαθμό μεγαλύτερο από ***k***

Algorithm ***inPlaceQuickSort***(***S, l, r***)
  Input sequence ***S***, ranks ***l*** and ***r***
  Output sequence ***S*** with the elements of rank between ***l*** and ***r*** rearranged in increasing order
  if $l \geq r$
    return
  $i \leftarrow$ a random integer between ***l*** and ***r***
  $x \leftarrow$ ***S.elemAtRank***(***i***)
  (***h, k***) $\leftarrow$ ***inPlacePartition***(***x***)
  inPlaceQuickSort(***S, l, h*** − 1)
  inPlaceQuickSort(***S, k*** + 1, ***r***)



Διαμέριση στη θέση

- Εκτελείται η διαμέριση με χρήση δύο δεικτών για διαχωρισμό της S σε L και E U G (μια παρόμοια μέθοδος μπορεί να διαχωρίσει E U G σε E και G).

j k

3 2 5 1 0 7 3 5 9 2 7 9 8 9 7 **6** 9 (pivot = 6)

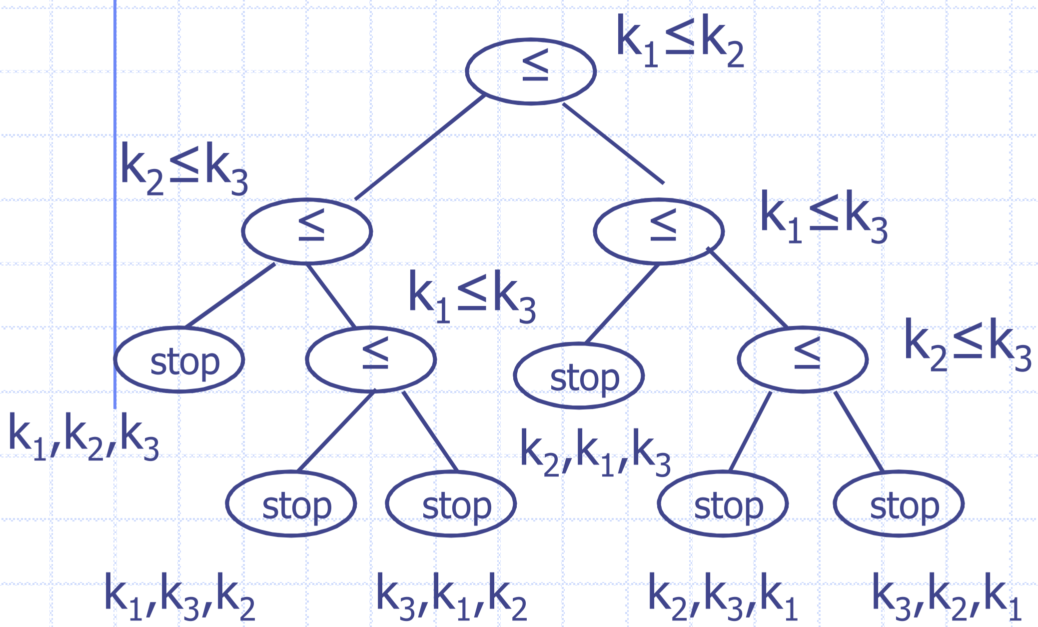
- Επανάληψη με να διασταυρωθούν τα j και k:
  - Σάρωση του j στα δεξιά μέχρι να βρεθεί ένα στοιχείο $\geq$ x.
  - Σάρωση του k στα αριστερά μέχρι να βρεθεί ένα στοιχείο< x.
  - Αντιμετάθεση των στοιχείων με δείκτες j και k

→ j k ←

3 2 5 1 0 $\boxed{7}$ 3 5 9 $\boxed{2}$ 7 9 8 9 7 **6** 9

Πόσο γρήγορα μπορούμε να ταξινομήσουμε χρησιμοποιώντας αλγόριθμους σύγκρισης-απόφασης

Παράδειγμα k_1,k_2,k_3



Επομένως ένα δένδρο αποφάσεων για την ταξινόμηση n στοιχείων θα πρέπει να καταλήγει σε n! πιθανά αποτελέσματα και άρα το ύψος του θα είναι τουλάχιστον $\log_2(n!)$

Όμως

$n!=n(n-1)(n-2)\ldots(2)(1)\geq(n/2)^{n/2}$

$\log_2 n!\geq(n/2)\log_2(n/2)=O(n\log_2 n)$

Σύνοψη

Αλγόριθμος	Χρόνος	Σημειώσεις
επιλογή	$O(n^2)$	▪ σε θέση ▪ αργή (καλή για μικρή είσοδο)
εισαγωγή	$O(n^2)$	▪ σε θέση ▪ αργή (μικρή είσοδος)
γρήγορη	$O(n \log n)$ expected	▪ σε θέση, τυχαίο ▪ πολύ γρήγορη
σωρού	$O(n \log n)$	▪ σε θέση ▪ γρήγορη
συγχώνευση	$O(n \log n)$	▪ γραμμική προσπέλαση ▪ γρήγορη