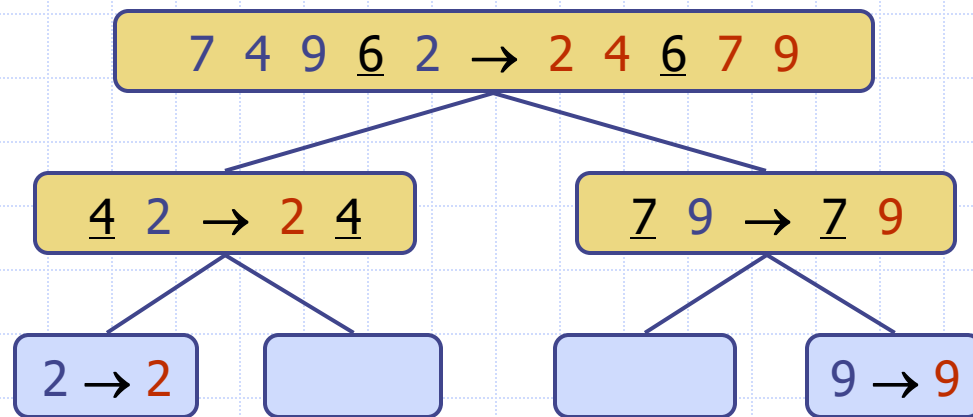


Παρουσίαση για χρήση με το σύγγραμμα, **Αλγόριθμοι Σχεδίαση και Εφαρμογές**, των Μ. Τ. Goodrich and R. Tamassia, Wiley, 2015 (στα ελληνικά από εκδόσεις Μ. Γκιούρδας)

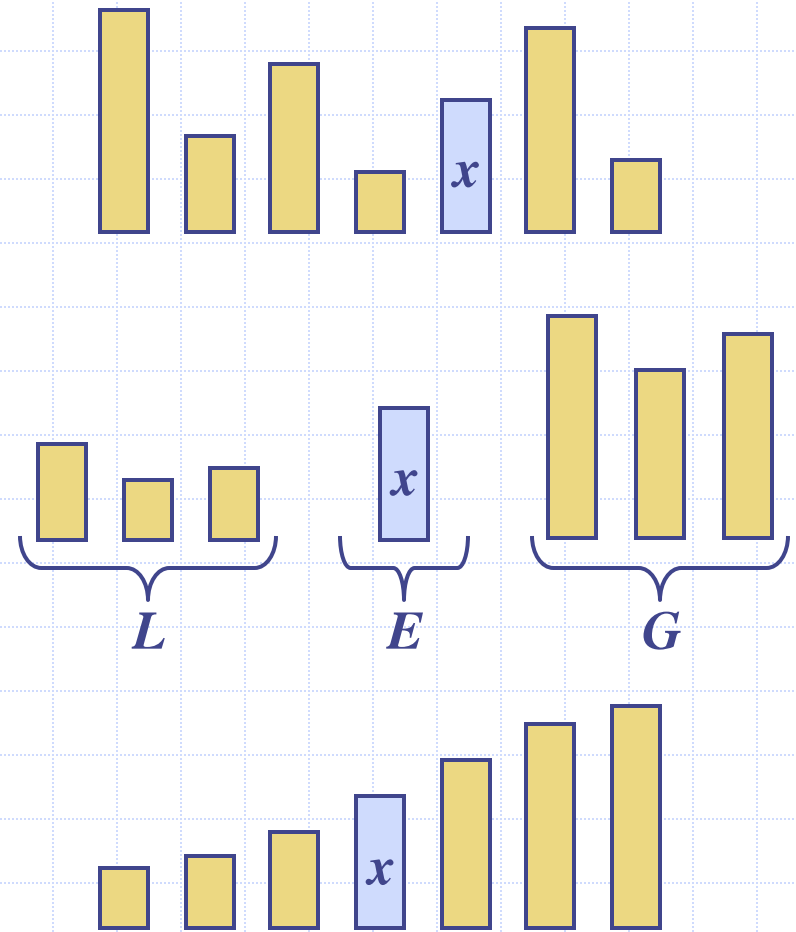
Γρήγορη ταξινόμηση (Quick-Sort)

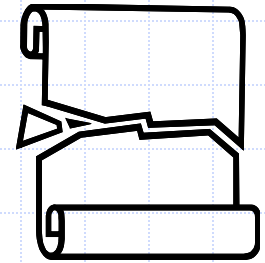


Quick-Sort

◆ Ο **quick-sort** είναι ένας τυχαιοποιημένος (randomized) αλγόριθμος ταξινόμησης που βασίζεται στο παράδειγμα διαίρει και βασίλευε:

- **Διαίρει**: επιλογή ενός τυχαίου στοιχείου x (ονομάζεται **σημείο περιστροφής - pivot**) και διαμερισμός του S σε
 - ◆ L στοιχεία μικρότερα του x
 - ◆ E στοιχεία ίσα του x
 - ◆ G στοιχεία μεγαλύτερα του x
- **Επανάλαβε (αναδρομικά)**: ταξινόμηση του L και G
- **Βασίλευε**: συνένωσε L , E και G





Διαμερισμός (partition)

- ◆ Διαμερίζουμε την ακολουθία εισόδου ως εξής:
 - Αφαιρούμε στη σειρά κάθε στοιχείο y από το S και
 - Εισάγουμε το y στο L , E ή G , βάσει του αποτελέσματος της σύγκρισης με το σημείο περιστροφής x
- ◆ Κάθε εισαγωγή ή διαγραφή γίνεται στην αρχή ή στο τέλος της ακολουθίας οπότε χρειάζεται χρόνο $O(1)$
- ◆ Έτσι, το βήμα διαμερισμού του quick-sort 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.

$L, E, G \leftarrow$ empty sequences

$x \leftarrow S.remove(p)$

while $\neg S.isEmpty()$

$y \leftarrow S.remove(S.first())$

if $y < x$

$L.addLast(y)$

else if $y = x$

$E.addLast(y)$

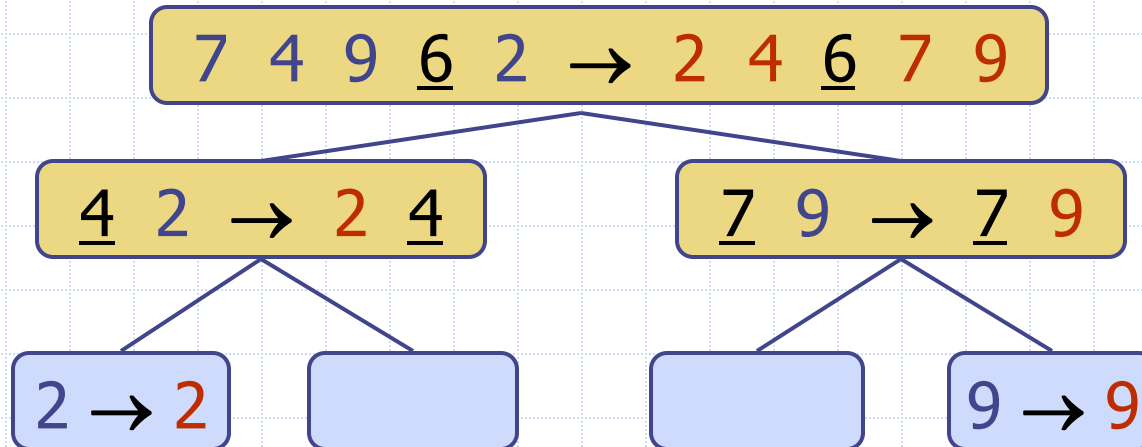
else $\{ y > x \}$

$G.addLast(y)$

return L, E, G

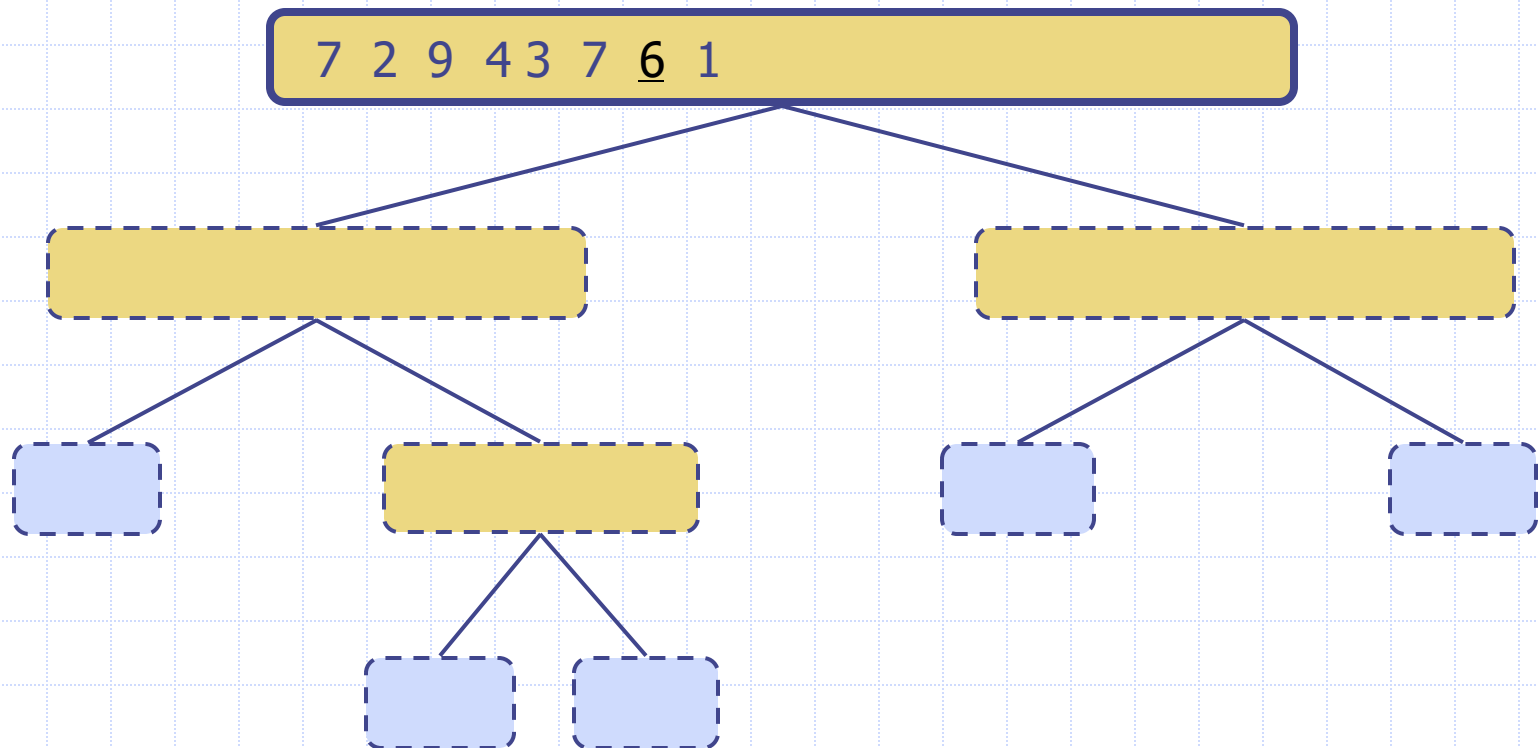
Δένδρο Quick-Sort

- ◆ Η εκτέλεση του quick-sort μπορεί να αναπαρασταθεί με ένα δυαδικό δένδρο
 - Κάθε κόμβος αναπαριστά μία αναδρομική κλήση του quick-sort και αποθηκεύει
 - ◆ Τη μη-ταξινομημένη ακολουθία πριν την εκτέλεση και το σημείο περιστροφής
 - ◆ Την ταξινομημένη ακολουθία στο τέλος της εκτέλεσης
 - Η ρίζα αναπαριστά την αρχική κλήση
 - Τα φύλλα αναπαριστούν κλήσεις υπό-ακολουθιών μεγέθους 0 ή 1



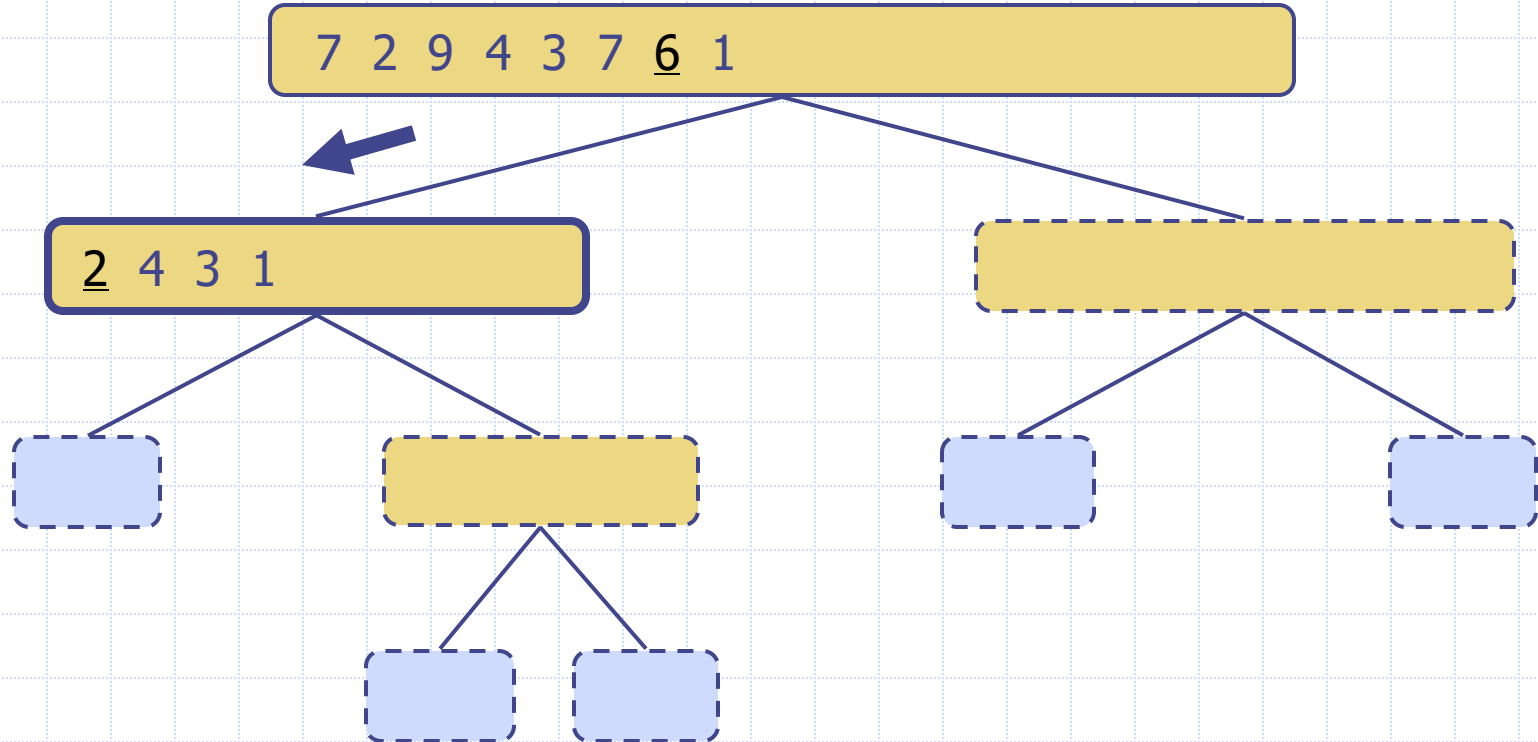
Παράδειγμα εκτέλεσης

◆ Επιλογή σημείου περιστροφής (pivot)



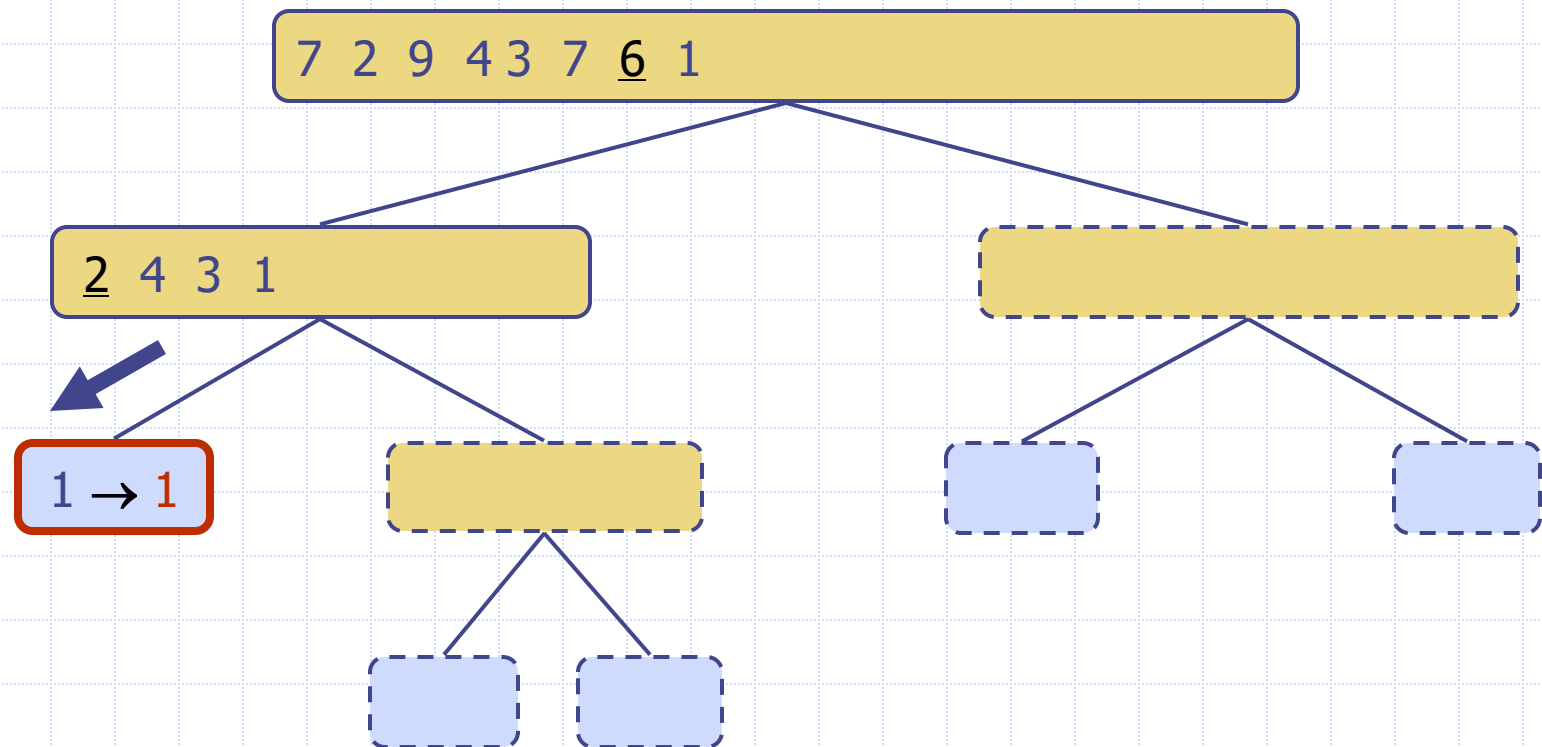
Παράδειγμα εκτέλεσης (συν.)

- ◆ Διαμερισμός, αναδρομική κλήση, επιλογή σημείου περιστροφής



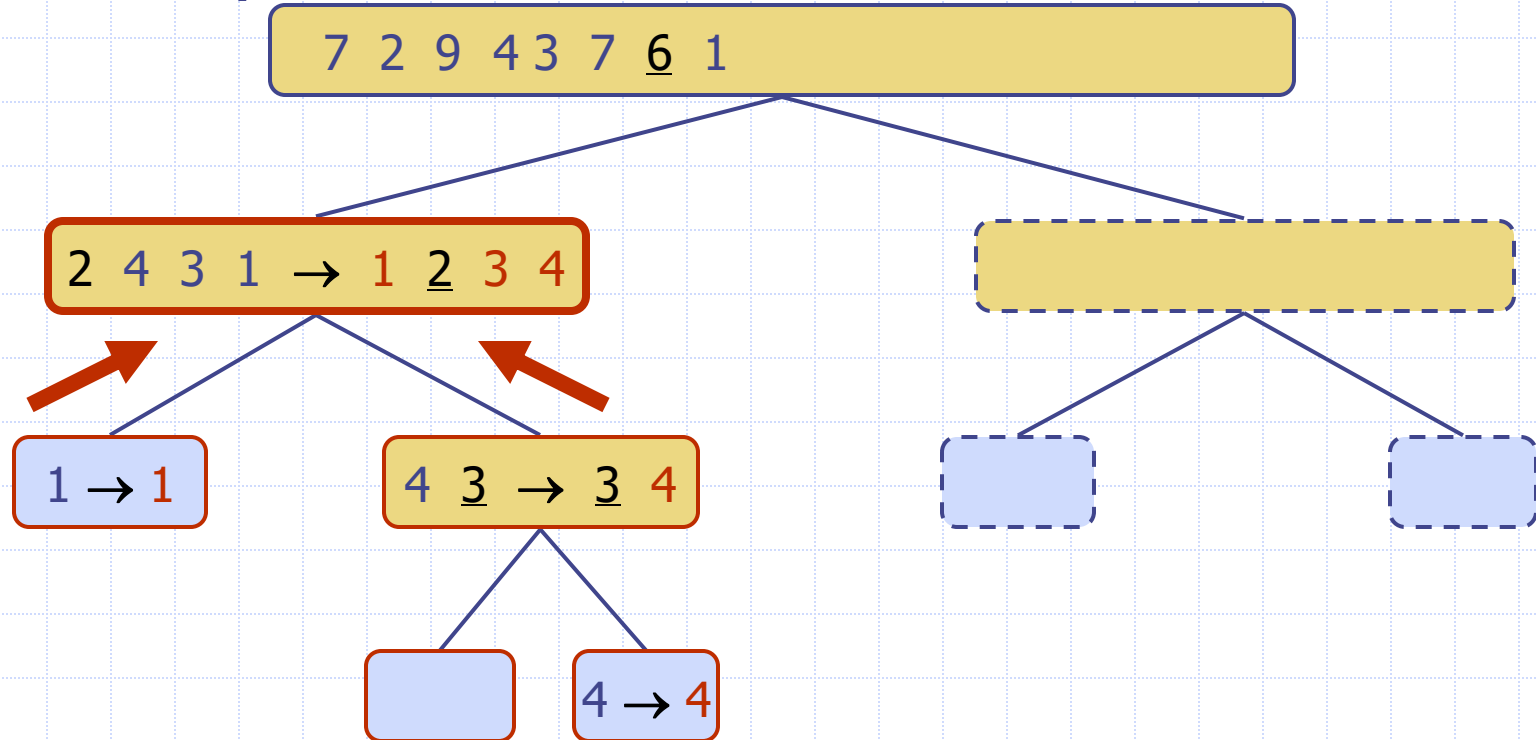
Παράδειγμα εκτέλεσης (συν.)

- ◆ Διαμερισμός, αναδρομική κλήση, βασική περίπτωση



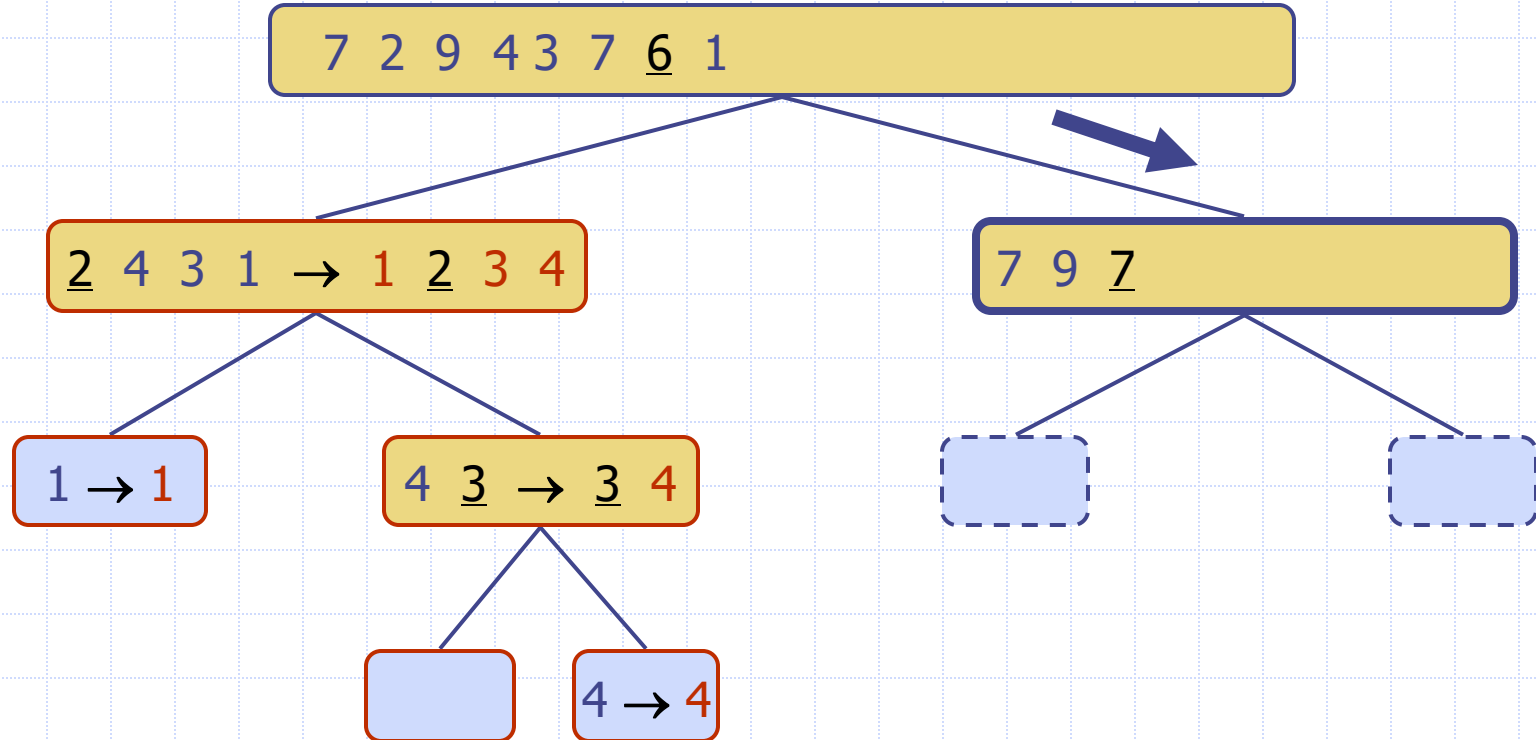
Παράδειγμα εκτέλεσης (συν.)

◆ Αναδρομική κλήση, ..., βασική περίπτωση, ένωση



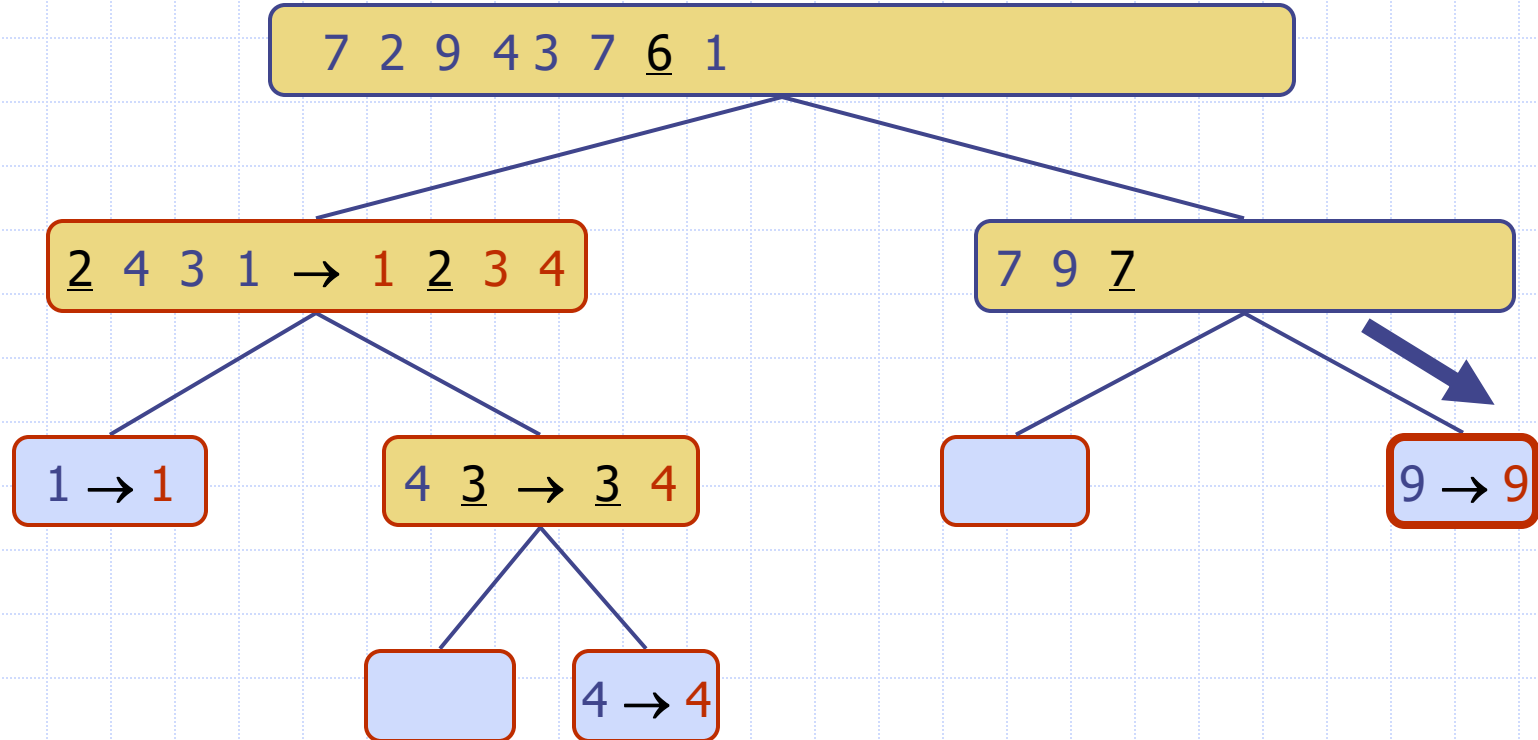
Παράδειγμα εκτέλεσης (συν.)

- ◆ Αναδρομική κλήση, επιλογή σημείου περιστροφής



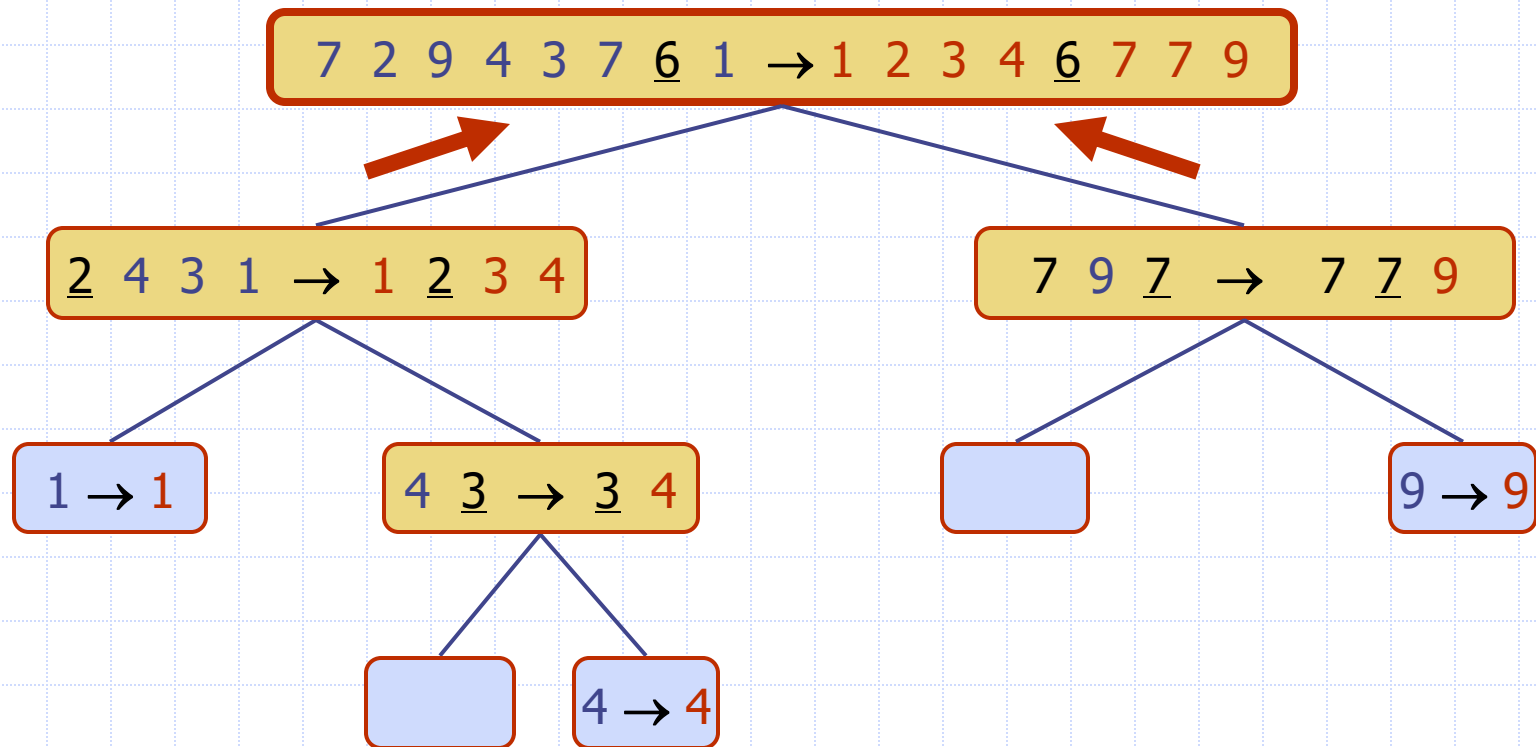
Παράδειγμα εκτέλεσης (συν.)

- ◆ Διαχωρισμός, ..., αναδρομική κλήση, βασική περίπτωση



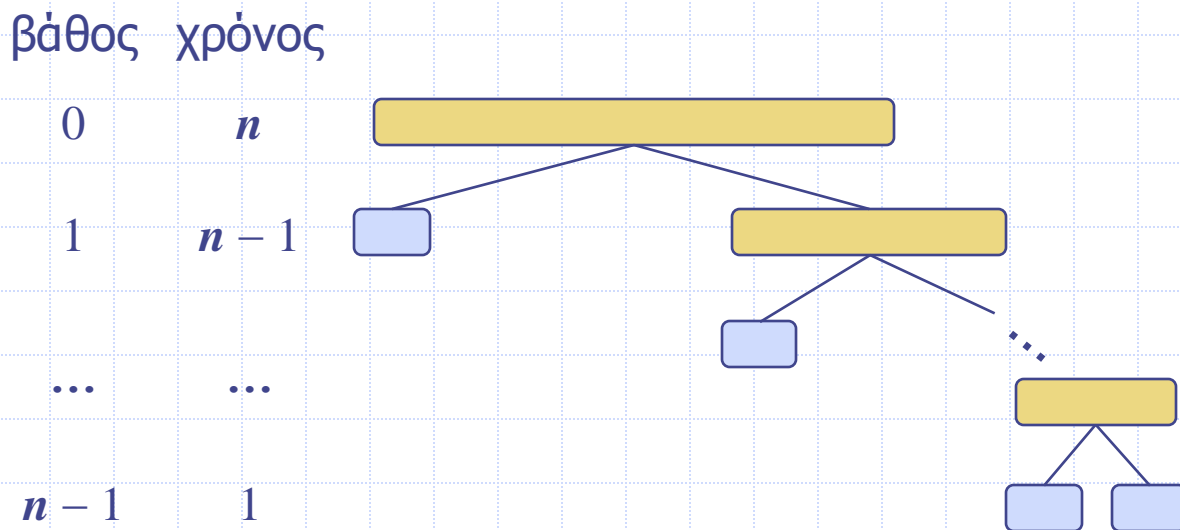
Παράδειγμα εκτέλεσης (συν.)

◆ Ένωση, ένωση



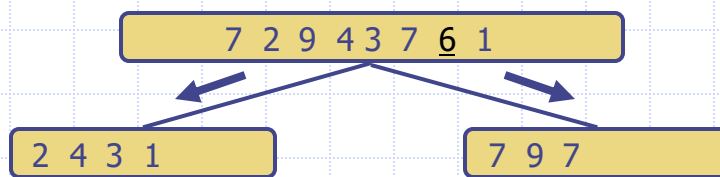
Χρόνος εκτέλεσης στη χειρότερη περίπτωση

- ◆ Η χειρότερη περίπτωση για τον quick-sort είναι όταν το σημείο περιστροφής είναι το ελάχιστο ή το μέγιστο στοιχείο της ακολουθίας
- ◆ Τότε, ένα από τα L και G έχει μέγεθος $n - 1$ και το άλλο έχει μέγεθος 0
- ◆ Ο χρόνος εκτέλεσης είναι ανάλογος του
$$n + (n - 1) + \dots + 2 + 1$$
- ◆ Έτσι, ο χρόνος εκτέλεσης του quick-sort στην χειρότερη περίπτωση είναι $O(n^2)$

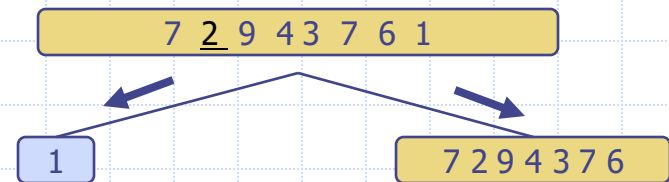


Αναμενόμενος χρόνος εκτέλεσης (1/2)

- ◆ Έστω μια αναδρομική κλήση του quick-sort σε μία ακολουθία μεγέθους s
 - **Καλή κλήση**: τα μεγέθη του L και του G είναι μικρότερα του $3s/4$
 - **Κακή κλήση**: το L ή το G είναι μεγαλύτερο του $3s/4$

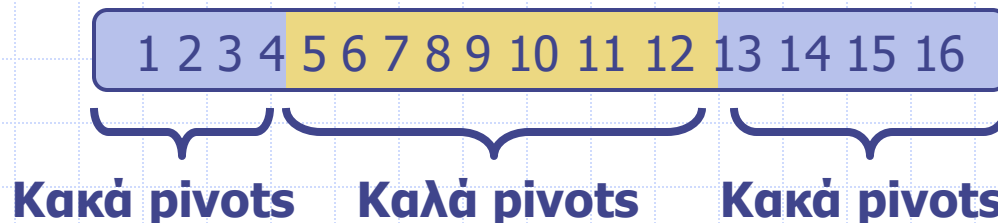


Καλή κλήση



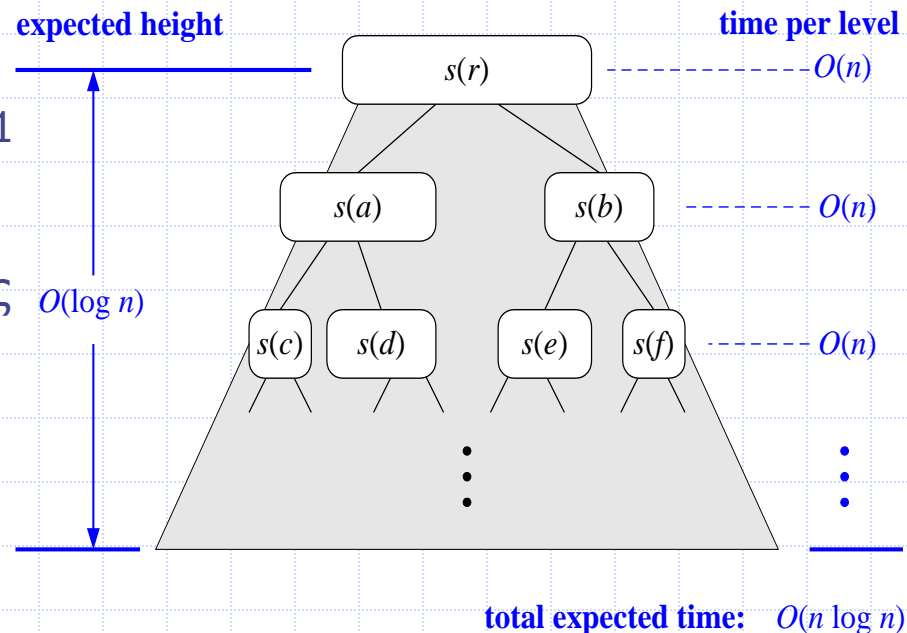
Κακή κλήση

- ◆ Μία κλήση είναι **καλή** με πιθανότητα $1/2$
 - $1/2$ από όλα τα πιθανά σημεία περιστροφής (σ.π.) προκαλούν καλές κλήσεις:

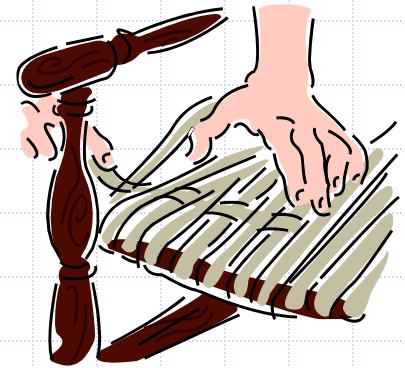


Αναμενόμενος χρόνος εκτέλεσης (2/2)

- ◆ Αποδεικνύεται από τη θεωρία πιθανοτήτων: Ο αναμενόμενος αριθμός ρίψεων κερμάτων για να έρθουν k κορώνες είναι 2^k
- ◆ Για έναν κόμβο σε βάθος i , περιμένουμε
 - $i/2$ των προγόνων να είναι καλές κλήσεις
 - Το μέγεθος της ακολουθίας εισόδου για την τρέχουσα κλήση να είναι το πολύ $(3/4)^{i/2}n$
- ◆ Έτσι, έχουμε
 - Για κόμβο βάθους $2\log_{4/3}n$, το αναμενόμενο μέγεθος εισόδου είναι 1
 - Το αναμενόμενο ύψος του δένδρου του quick-sort να είναι $O(\log n)$
- ◆ Ο χρόνος που δαπανάται σε κόμβους του ίδιου βάθους είναι $O(n)$
- ◆ Έτσι, ο αναμενόμενος χρόνος εκτέλεσης του quick-sort είναι $O(n \log n)$



Επί-τόπου Quick-Sort



- ◆ Ο Quick-sort μπορεί να υλοποιηθεί έτσι ώστε να εκτελείται επί-τόπου
- ◆ Στο βήμα του διαμερισμού, χρησιμοποιούμε λειτουργίες αντικατάστασης για να αναδιατάξουμε τα στοιχεία της ακολουθίας εισόδου έτσι ώστε
 - Τα στοιχεία που είναι μικρότερα του pivot έχουν θέση μικρότερη του h
 - Τα στοιχεία ίσα με το pivot έχουν θέση μεταξύ h και k
 - Τα στοιχεία μεγαλύτερα του pivot έχουν θέση μεγαλύτερη του k
- ◆ Οι αναδρομικές κλήσεις αφορούν
 - Στοιχεία με θέση μικρότερη του h
 - Στοιχεία με θέση μεγαλύτερη του 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

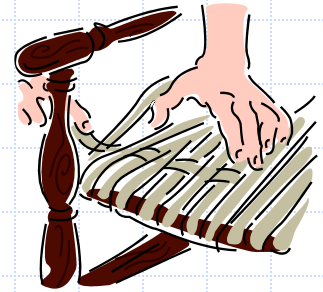
$x \leftarrow S.\text{elemAtRank}(i)$

$(h, k) \leftarrow \text{inPlacePartition}(x)$

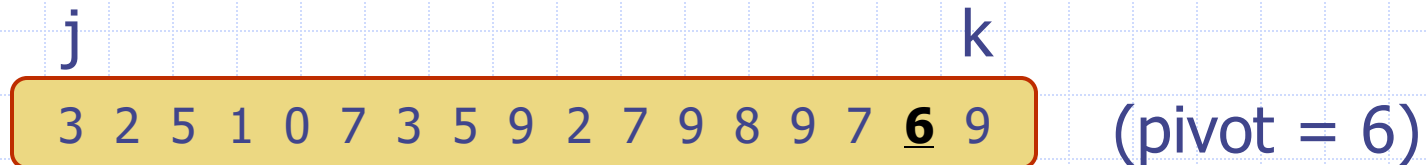
inPlaceQuickSort($S, l, h - 1$)

inPlaceQuickSort($S, k + 1, r$)

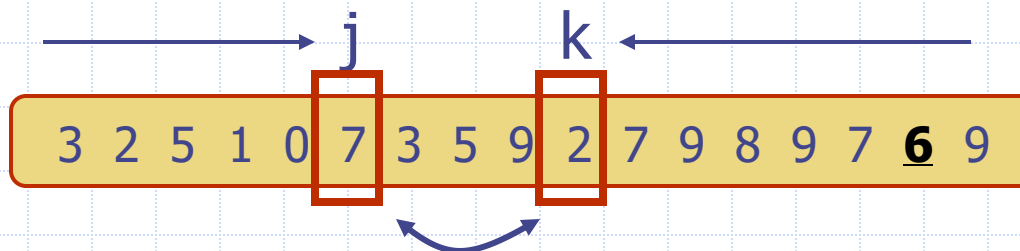
Επί-τόπου διαμερισμός



- Εκτέλεση του διαμερισμού χρησιμοποιώντας δύο δείκτες για διαίρεση της ακολουθίας **S** σε **L** (ακολουθία μικρότερων στοιχείων) και **E U G** (ακολουθία ίσων ή μεγαλύτερων στοιχείων) - μία παρόμοια μέθοδος μπορεί να χωρίσει τα **E U G** σε **E** και **G**.



- Επανάληψη μέχρι να συναντηθούν τα **j** και **k**:
 - Έλεγχος από το **j** προς τα δεξιά μέχρι να βρεθεί στοιχείο \geq **pivot**.
 - Έλεγχος από το **k** προς τα αριστερά μέχρι να βρεθεί στοιχείο $<$ **pivot**.
 - Ανταλλαγή των στοιχείων στους δείκτες **j** και **k**



Σύνοψη αλγορίθμων ταξινόμησης

Αλγόριθμος	Χρόνος	Σημειώσεις
selection-sort	$O(n^2)$	<ul style="list-style-type: none">■ επί-τόπου■ αργό (καλό για μικρά σετ δεδομένων)
insertion-sort	$O(n^2)$	<ul style="list-style-type: none">■ επί-τόπου■ αργό (καλό για μικρά σετ δεδομένων)
quick-sort	$O(n \log n)$ αναμενόμενο	<ul style="list-style-type: none">■ επί-τόπου, τυχαιοποιημένο■ το γρηγορότερο (καλό για μεγάλα δεδομένα)
heap-sort	$O(n \log n)$	<ul style="list-style-type: none">■ επί-τόπου■ γρήγορο (καλό για μεγάλα δεδομένα)
merge-sort	$O(n \log n)$	<ul style="list-style-type: none">■ διαδοχική προσπέλαση δεδομένων■ γρήγορο (καλό για πολύ μεγάλα δεδομένα)