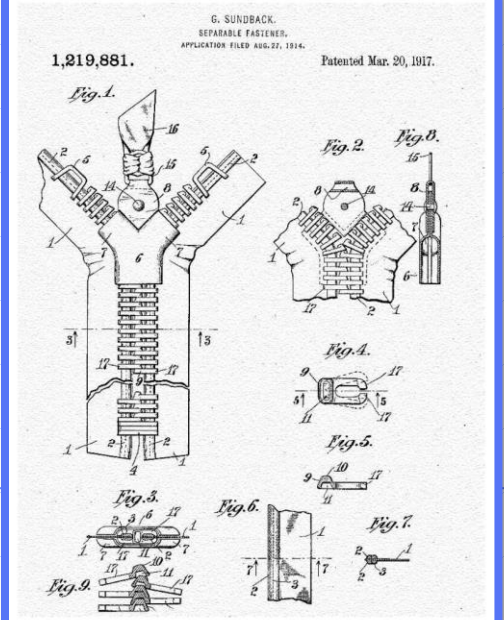
Παρουσίαση για χρήση με το σύγγραμμα, Αλγόριθμοι Σχεδίαση και Εφαρμογές, των M. T. Goodrich and R. Tamassia, Wiley, 2015 (στα ελληνικά από εκδόσεις Μ. Γκιούρδας)

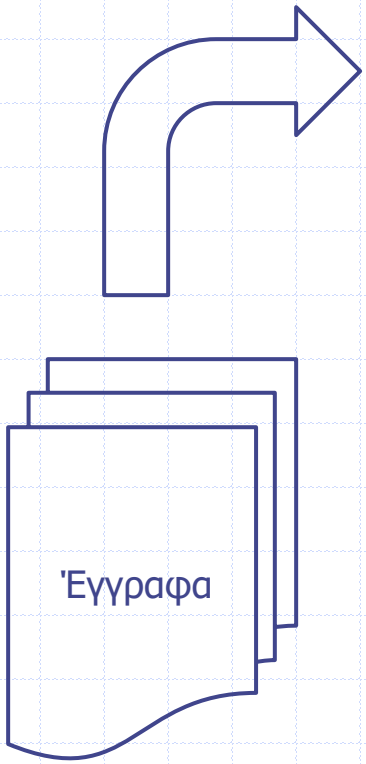
Merge Sort



Separable Fastener, U.S. Patent 1,219,881, 1917. Public domain image.

Εφαρμογή: Μηχανές αναζήτησης του Internet

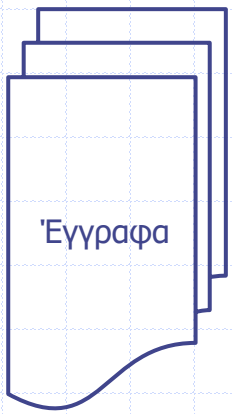
- Η ταξινόμηση έχει πολλές εφαρμογές, συμπεριλαμβανομένων των μηχανών αναζήτησης.
- Η ταξινόμηση απαιτείται στα βήματα που πρέπει να γίνουν ώστε να κατασκευαστεί μία δομή δεδομένων, που θα επιτρέπει σε μια μηχανή αναζήτησης να επιστρέφει γρήγορα μια λίστα με έγγραφα που περιέχουν μια συγκεκριμένη λέξη – κλειδί. Αυτή η δομή δεδομένων ονομάζεται **αντεστραμμένο αρχείο.**



Word	Document Number & word location
banana	1:3, 2:45
butterfly	2:15, 3:12
camel	4:40
dog	1:60, 1:70, 2:22, 3:20, 4:11
horse	4:21
pig	2:55
pizza	1:56, 3:33

Εφαρμογή: Πως η ταξινόμηση υποστηρίζει μία μηχανή αναζήτησης

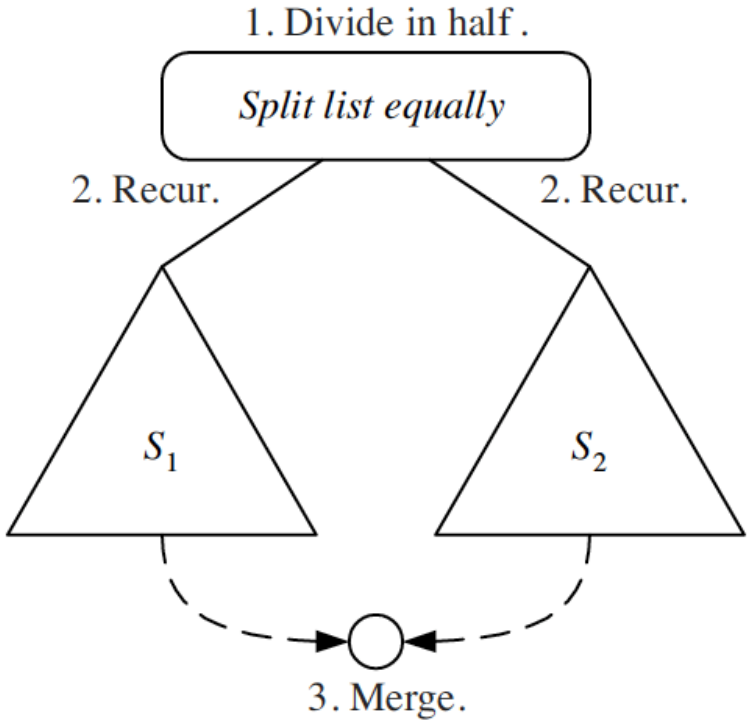
- Η κατασκευή μίας δομής δεδομένων αντεστραμμένου αρχείου, προϋποθέτει τη συγκέντρωση, για κάθε λέξη-κλειδί k, όλων των εγγράφων που περιέχουν το k.
- Η συγκέντρωση όλων αυτών των εγγράφων μπορεί να γίνει απλά με μια ταξινόμηση του συνόλου των ζευγών λέξεων-κλειδιών και εγγράφων κατά λέξεις - κλειδιά.
- Αυτή η ενέργεια τοποθετεί όλα τα ζεύγη (k, d) με την ίδια λέξη-κλειδί k, το ένα μετά το άλλο στη λίστα εξόδου.
- Απ' αυτήν την ταξινομημένη λίστα, ο έλεγχος της λίστας και η κατασκευή ενός πίνακα αναζήτησης εγγράφων για κάθε λέξη-κλειδί που εμφανίζεται σ' αυτήν την ταξινομημένη λίστα είναι θέμα ενός απλού υπολογισμού.



Word	Document Number & word location
banana	1:3, 2:45
butterfly	2:15, 3:12
camel	4:40
dog	1:60, 1:70, 2:22, 3:20, 4:11
horse	4:21
pig	2:55
pizza	1:56, 3:33

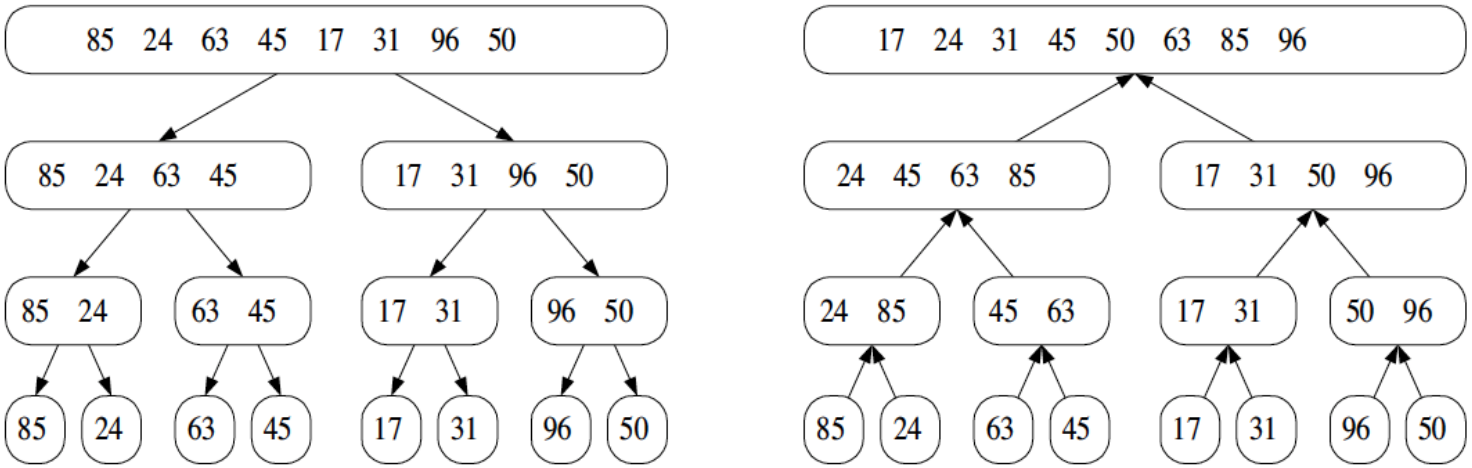
Διαίρει και Βασίλευε

- Το πρότυπο διαίρει και βασίλευε μπορεί να περιγραφεί ως μία διαδικασία που αποτελείται από τα τρία παρακάτω βήματα:
  - Διαίρεσε: διαιρούμε τα δεδομένα εισόδου S σε δύο ξεχωριστά υποσύνολα S_1 και S_2
  - Επανάλαβε: Λύνουμε αναδρομικά τα δευτερεύοντα προβλήματα, που συσχετίζονται με τα υποσύνολα S_1 και S_2
  - Κυρίευσε: «συγχωνεύουμε» τις λύσεις για τα δευτερεύοντα προβλήματα S_1 και S_2 σε μία λύση για το αρχικό πρόβλημα S
- Η βασική περίπτωση της αναδρομής είναι υπό-προβλήματα μεγέθους 0 ή 1



Ταξινόμηση με συγχώνευση (Merge-Sort)

- Merge-sort είναι ένας αλγόριθμος ταξινόμησης που βασίζεται στο πρότυπο Διαίρει και βασίλευε
- Όπως η ταξινόμηση σωρού
  - Έχει χρόνο εκτέλεσης $O(n \log n)$
- Αντίθετα με την ταξινόμηση σωρού
  - Δεν χρησιμοποιεί μια βοηθητική ουρά προτεραιότητας
  - Η προσπέλαση των δεδομένων γίνεται διαδοχικά (κατάλληλη για να ταξινομεί δεδομένα σε δίσκο)



Ο Αλγόριθμος Merge-Sort

- Merge-sort σε μία είσοδο S με n στοιχεία:
  - Διαίρει: Τοποθετούνται τα στοιχεία του S σε δύο ακολουθίες S_1 και S_2 με καθεμία να περιέχει περίπου $n/2$ στοιχεία
  - Επανάλαβε: Ταξινομούμε αναδρομικά τις ακολουθίες S_1 και S_2
  - Κυρίευσε: Συγχωνεύουμε τις ακολουθίες S_1 και S_2 σε μία ταξινομημένη ακολουθία

Algorithm *mergeSort*(S)
  Input sequence S with n elements
  Output sequence S sorted according to C
  if $S.size() > 1$
    $(S_1, S_2) \leftarrow partition(S, n/2)$
    $mergeSort(S_1)$
    $mergeSort(S_2)$
    $S \leftarrow merge(S_1, S_2)$

Συγχώνευση δύο ταξινομημένων ακολουθιών

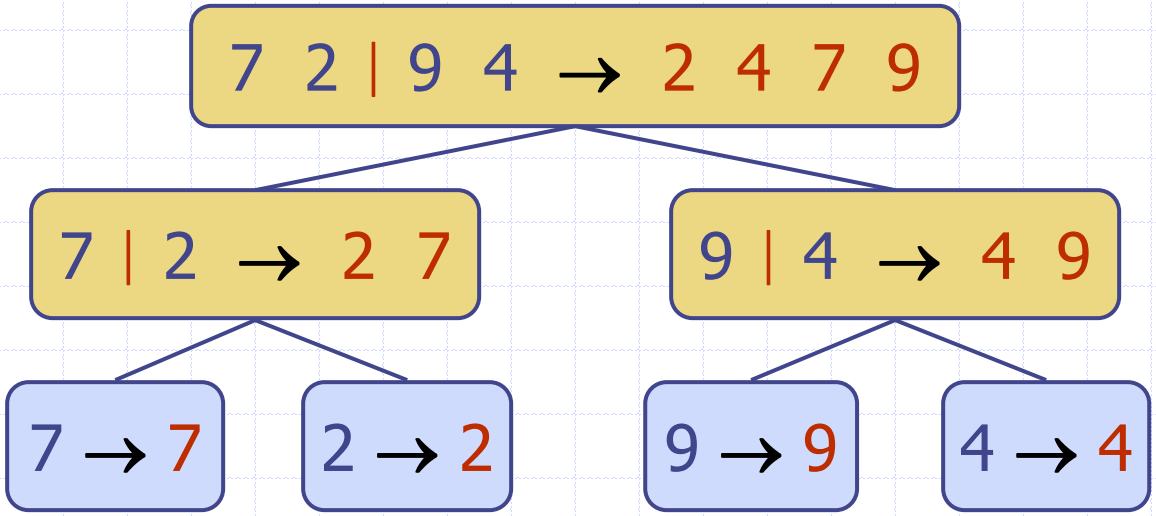
- Το τελευταίο βήμα (κυρίευσε) συγχωνεύει δύο ταξινομημένες ακολουθίες A και B σε μία ταξινομημένη ακολουθία S που περιέχει την ένωση των στοιχείων του A και του B
- Η συγχώνευση δύο ταξινομημένων ακολουθιών, καθεμία με $n/2$ στοιχεία υλοποιημένη με διπλά συνδεδεμένη λίστα είναι $O(n)$

Algorithm merge(S_1, S_2, S):
Input: Two arrays, S_1 and S_2, of size n_1 and n_2, respectively, sorted in non-decreasing order, and an empty array, S, of size at least $n_1 + n_2$
Output: S, containing the elements from S_1 and S_2 in sorted order

$i \leftarrow 1$
$j \leftarrow 1$
while $i \leq n$ **and** $j \leq n$ **do**
  if $S_1[i] \leq S_2[j]$ **then**
    $S[i + j - 1] \leftarrow S_1[i]$
    $i \leftarrow i + 1$
  else
    $S[i + j - 1] \leftarrow S_2[j]$
    $j \leftarrow j + 1$
while $i \leq n$ **do**
  $S[i + j - 1] \leftarrow S_1[i]$
  $i \leftarrow i + 1$
while $j \leq n$ **do**
  $S[i + j - 1] \leftarrow S_2[j]$
  $j \leftarrow j + 1$

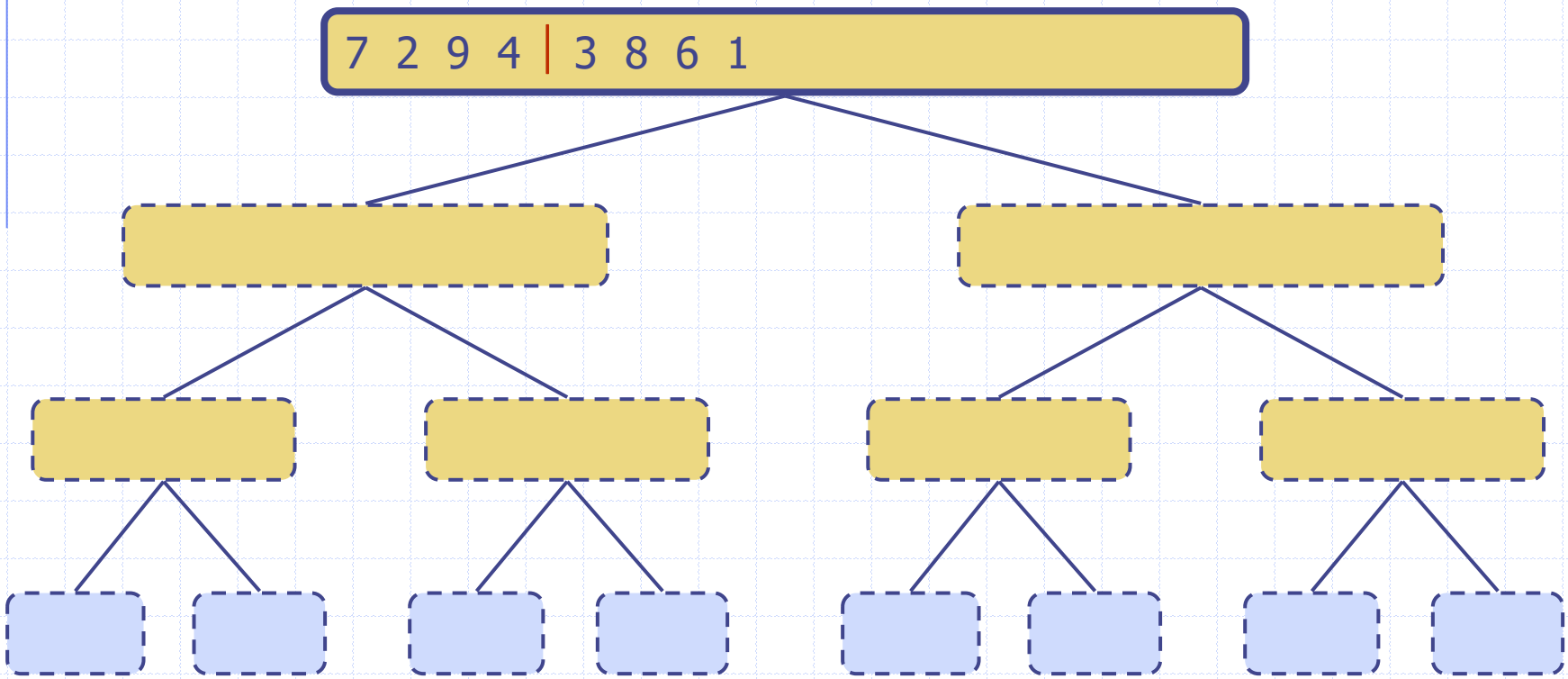
Δένδρο Merge-Sort

- Μία εκτέλεση merge-sort μπορεί να αναπαρασταθεί με ένα δυαδικό δένδρο
  - Κάθε κόμβος αναπαριστά μία αναδρομική κλήση του merge-sort και αποθηκεύει
    - μια μη ταξινομημένη ακολουθία πριν την εκτέλεση και τον διαχωρισμό της
    - μια ταξινομημένη ακολουθία στο τέλος της εκτέλεσης
  - η ρίζα είναι η αρχική κλήση
  - τα φύλλα είναι οι κλήσεις σε υπό-ακολουθίες μεγέθους 0 ή 1



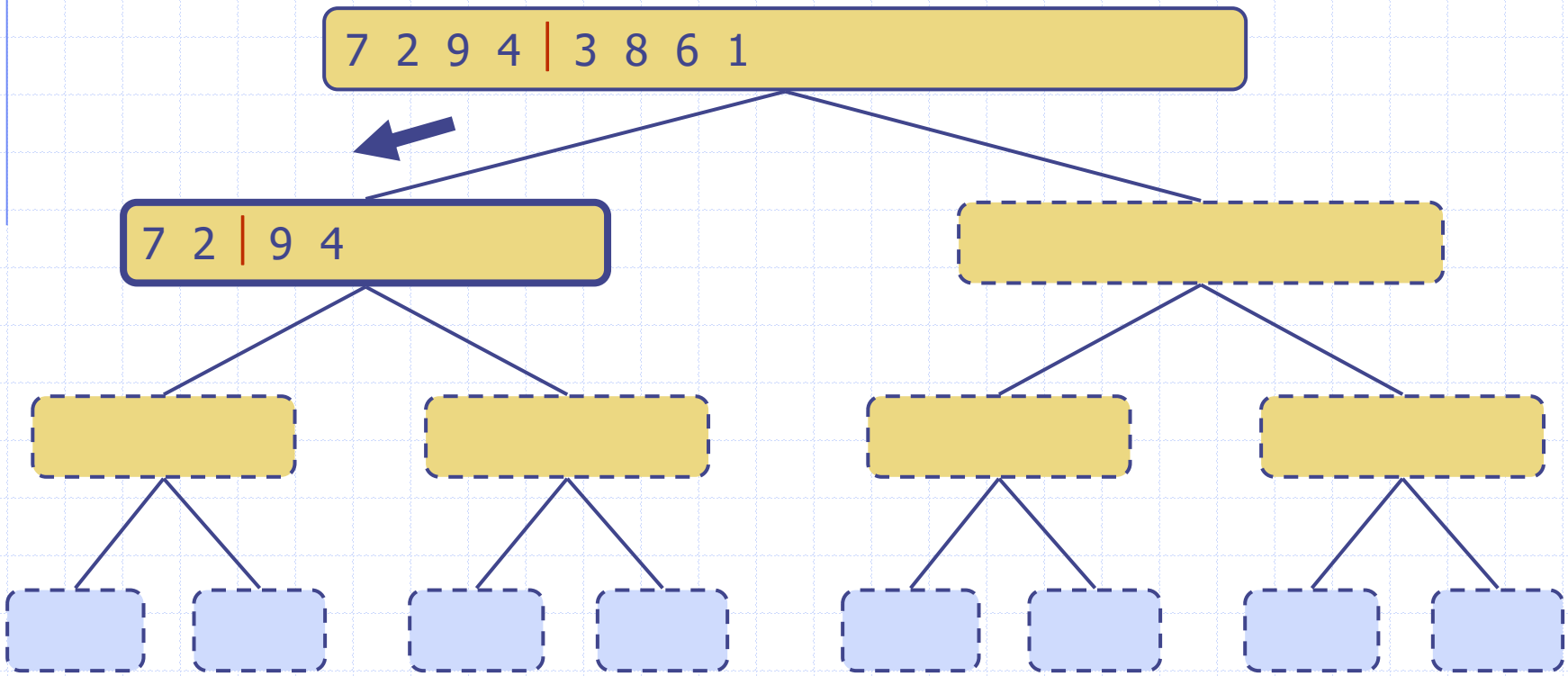
Παράδειγμα εκτέλεσης

- Διαίρει



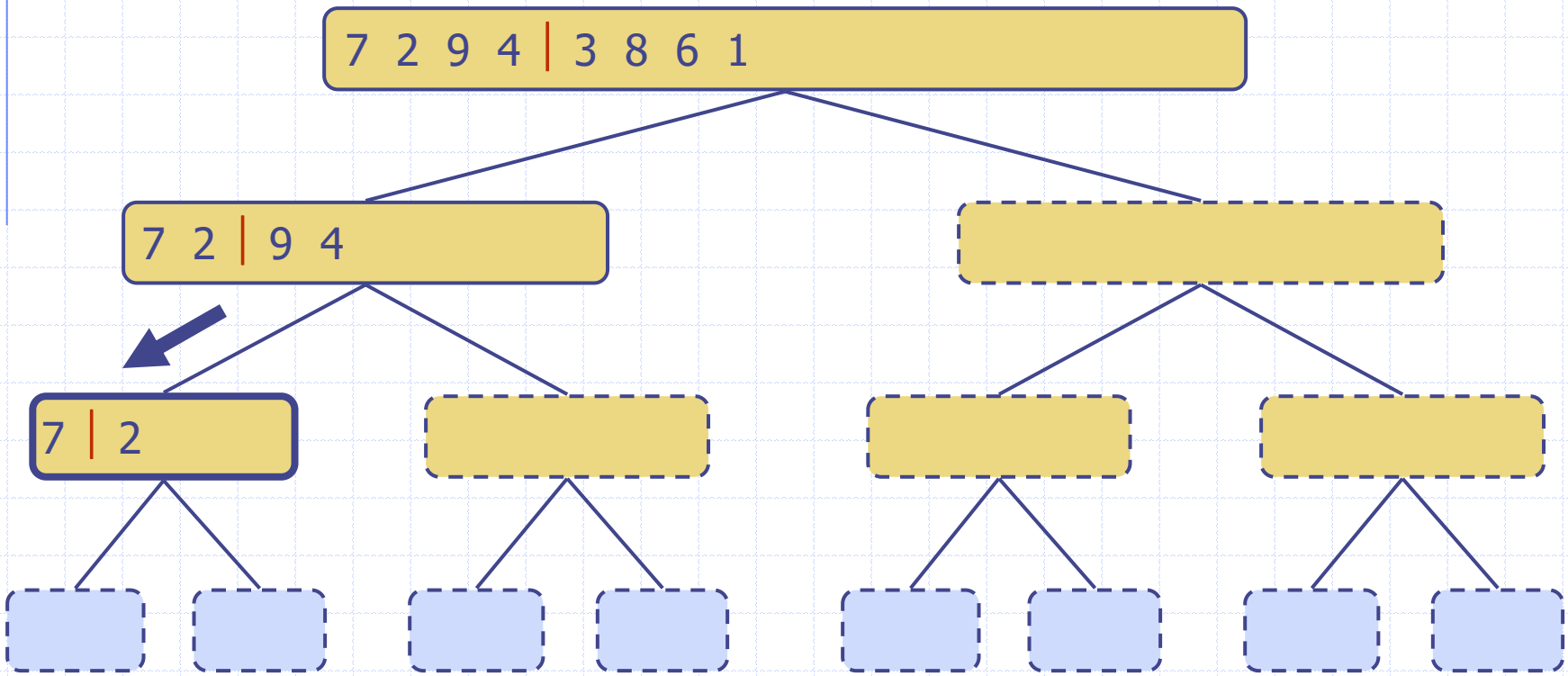
Execution Example (cont.)

- Αναδρομική κλήση, partition



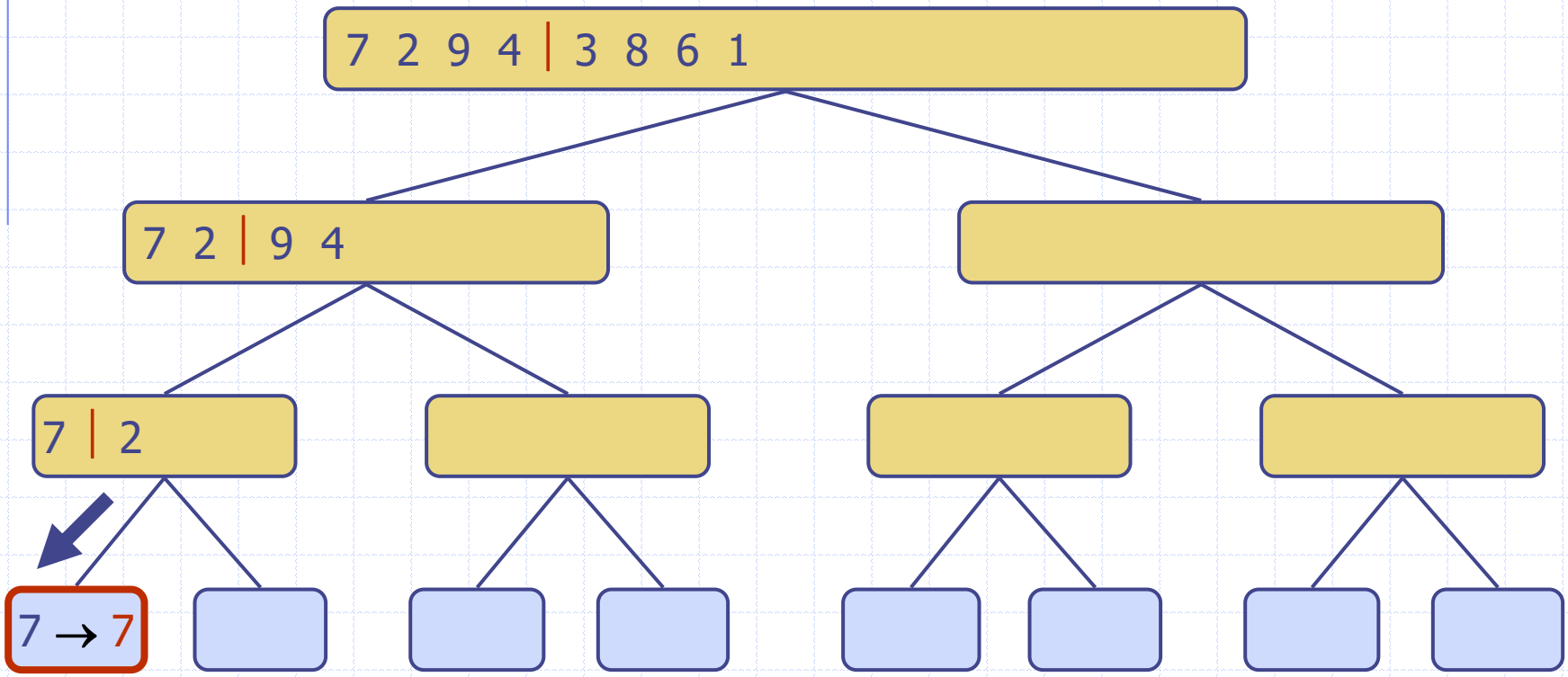
Παράδειγμα εκτέλεσης (συν.)

- Recursive call, διαχωρισμός



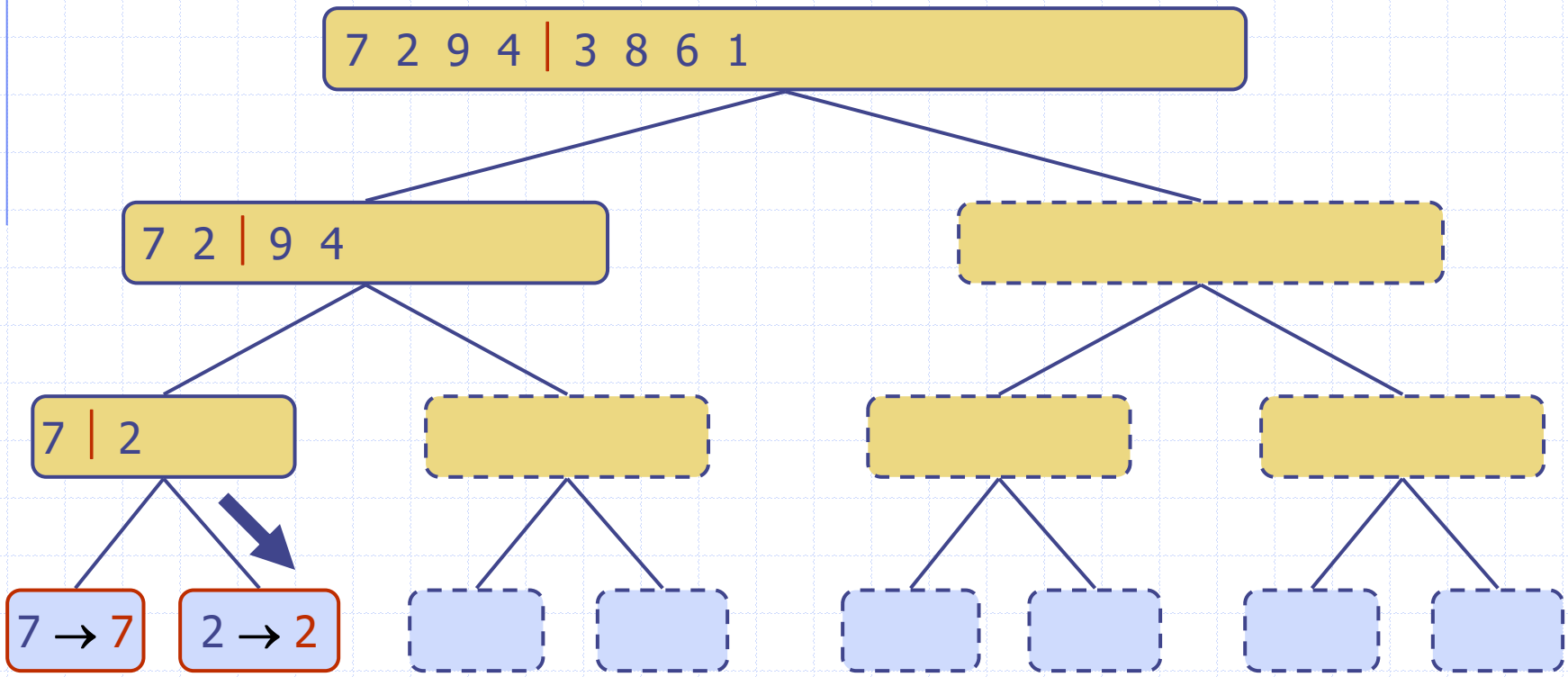
Παράδειγμα εκτέλεσης (συν.)

- Αναδρομική κλήση, βασική περίπτωση



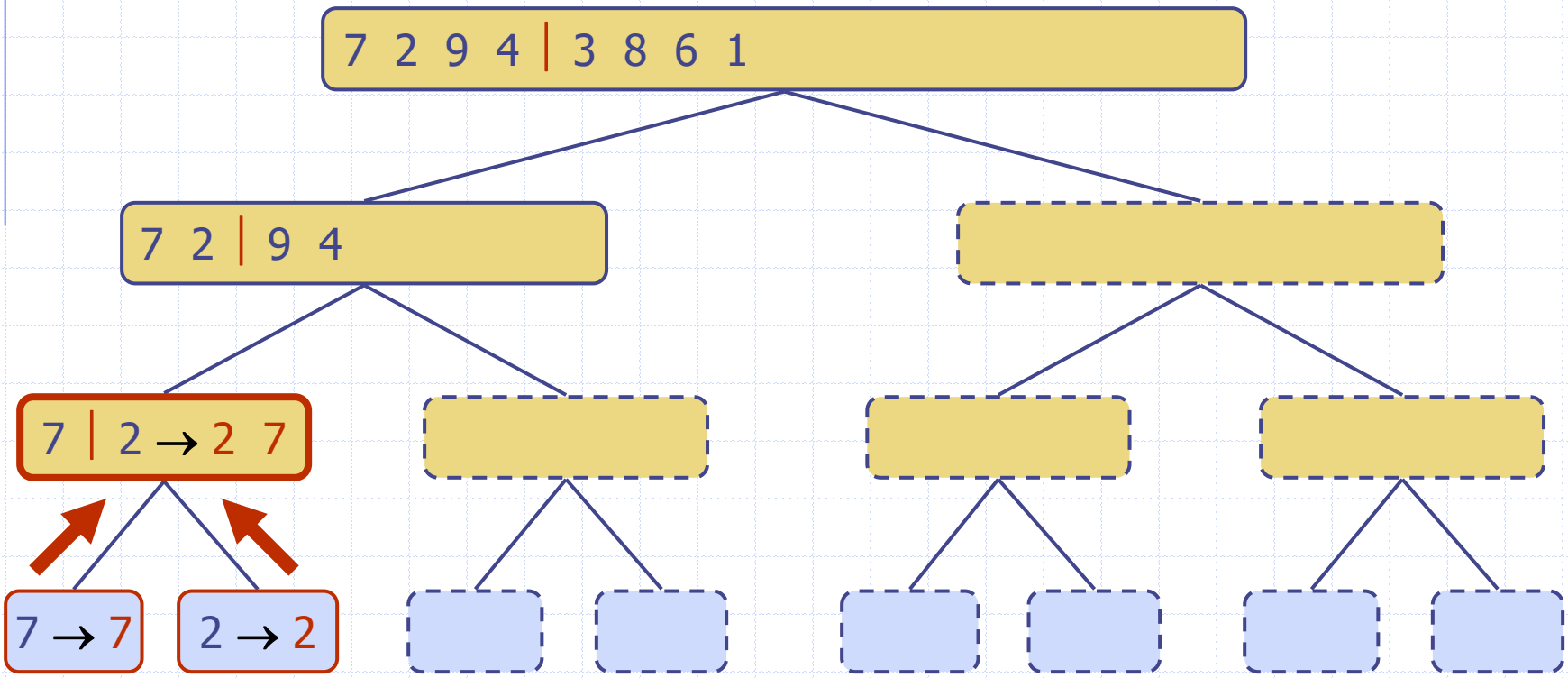
Παράδειγμα εκτέλεσης (συν.)

- Αναδρομική κλήση, βασική περίπτωση



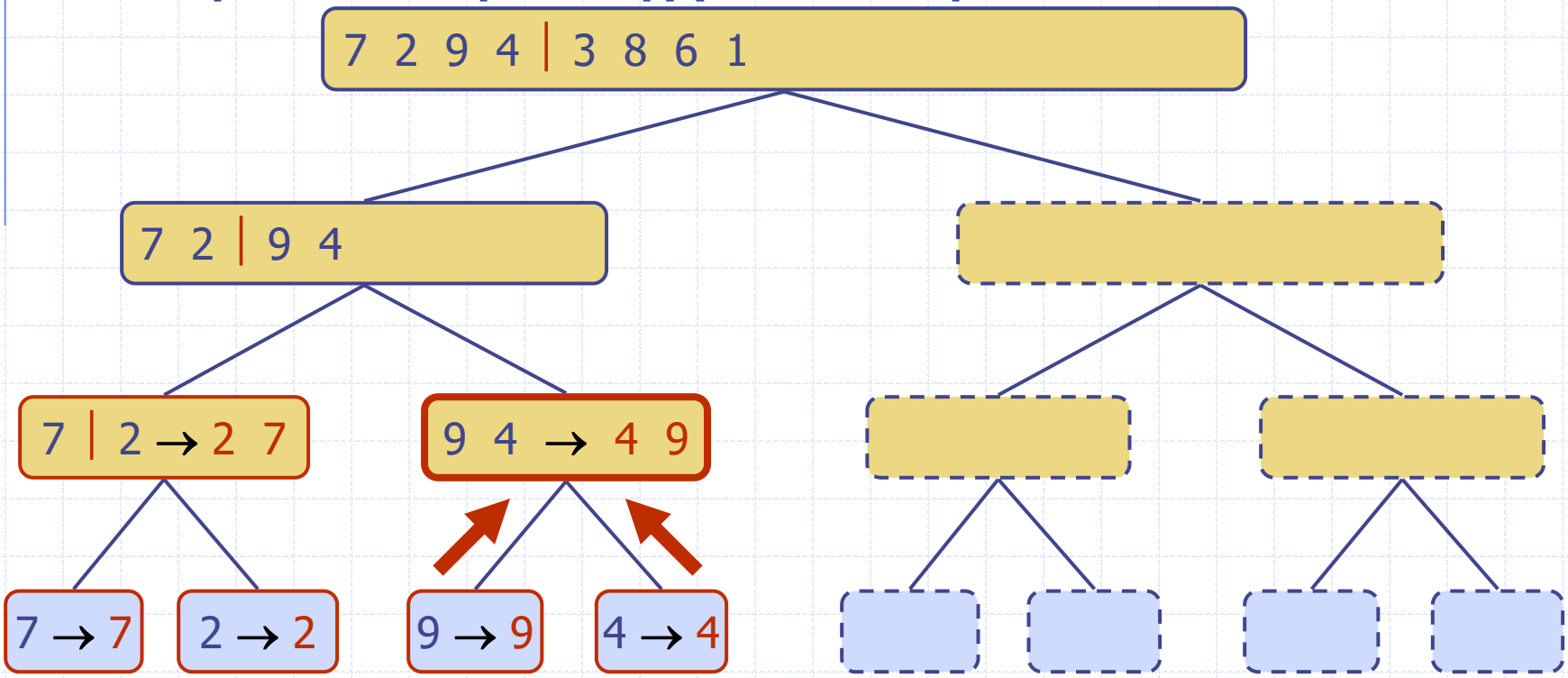
Παράδειγμα εκτέλεσης (συν.)

- Συγχώνευση



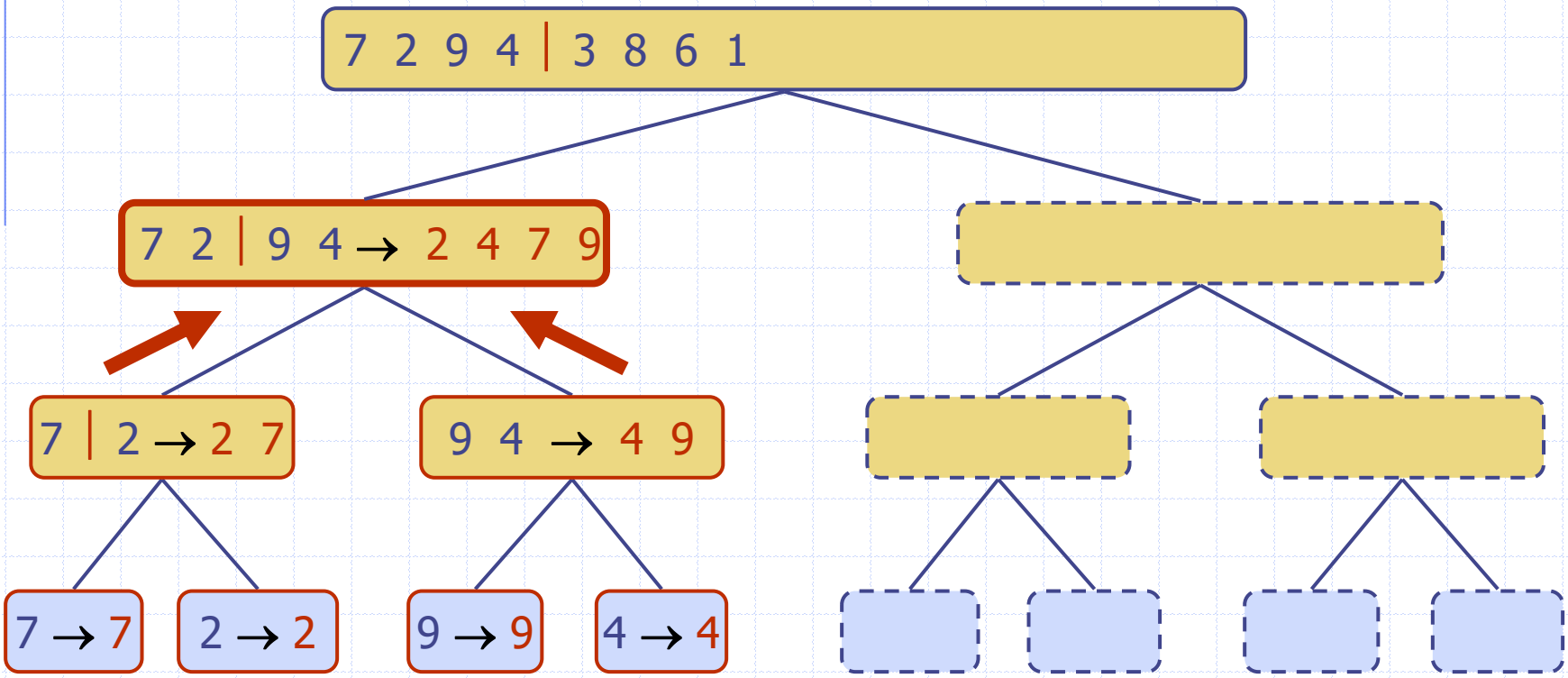
Παράδειγμα εκτέλεσης (συν.)

- Αναδρομική κλήση, …, βασική περίπτωση, συγχώνευση



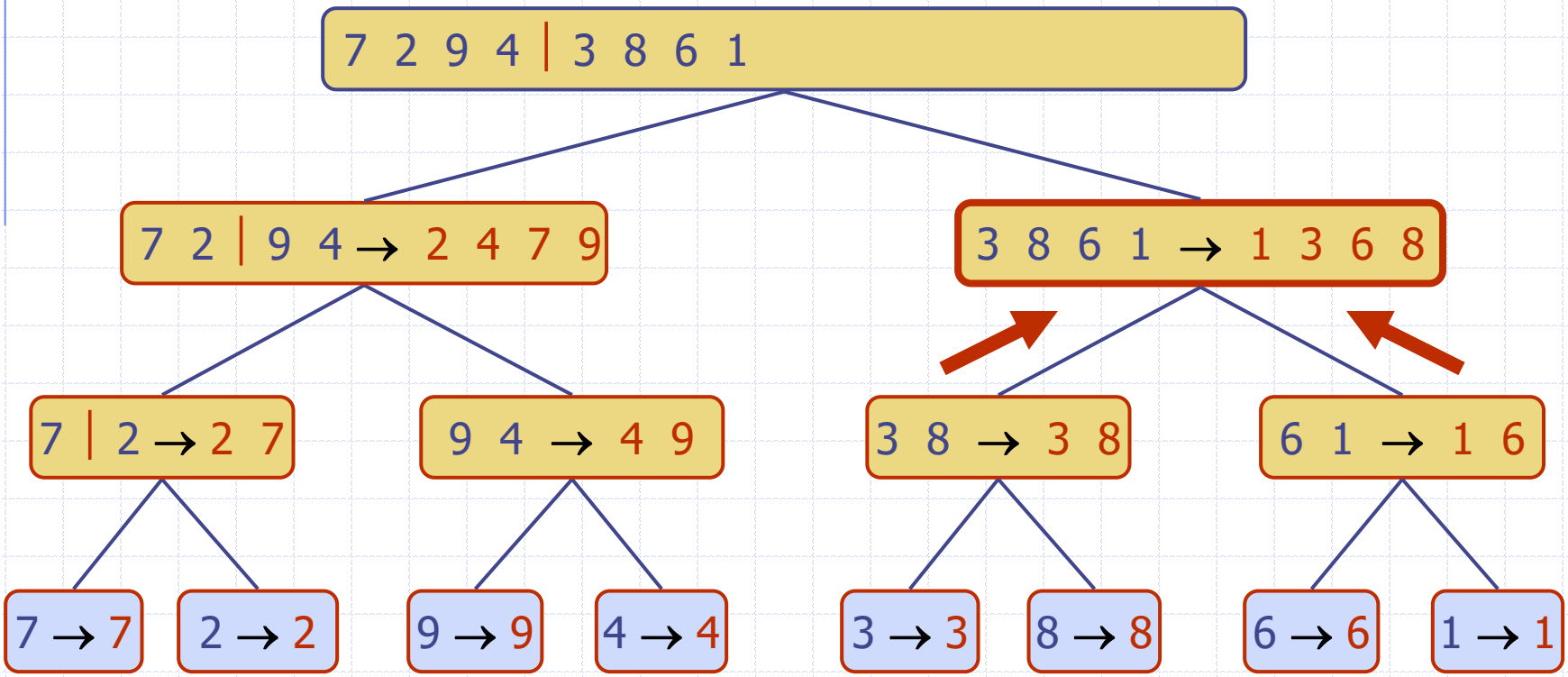
Παράδειγμα εκτέλεσης (συν.)

- Συγχώνευση



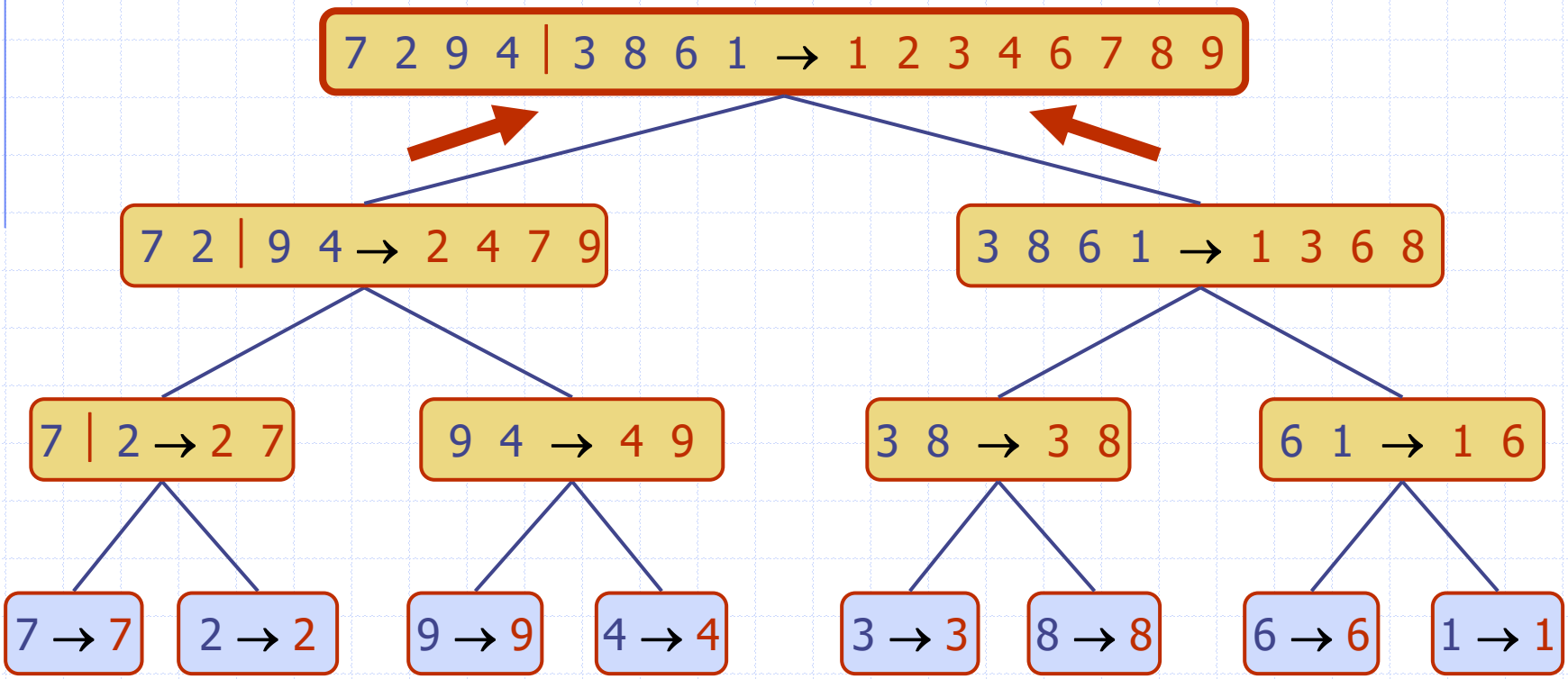
Παράδειγμα εκτέλεσης (συν.)

- Αναδρομική κλήση, …, συγχώνευση, συγχώνευση



Παράδειγμα εκτέλεσης (συν.)

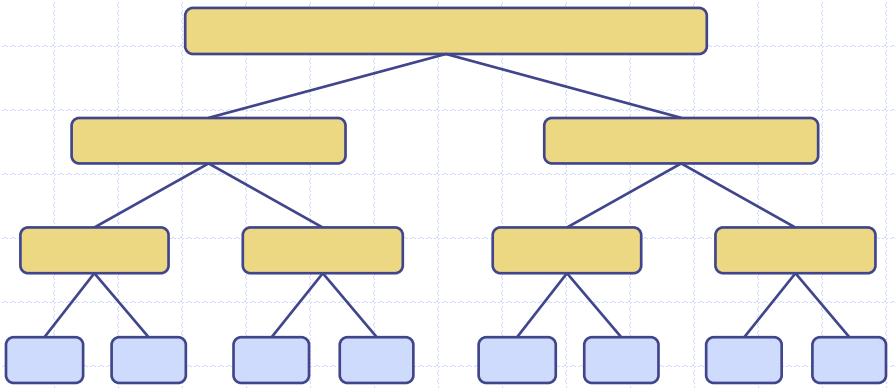
- Συγχώνευση



Ανάλυση Merge-Sort

- Το ύψος h του δένδρου merge-sort είναι $O(\log n)$
  - σε κάθε αναδρομική κλήση διαιρούμε την ακολουθία στην μέση,
- Ο συνολικός χρόνος για τα φύλλα στο βάθος i είναι $O(n)$
  - χωρίζουμε και συγχωνεύουμε 2^i ακολουθίες μεγέθους $n/2^i$
  - κάνουμε 2^{i+1} αναδρομικές κλήσεις
- Έτσι, ο συνολικός χρόνος του merge-sort είναι $O(n \log n)$

depth	#seqs	size
0	1	n
1	2	$n/2$
i	2^i	$n/2^i$
…	…	…



Σύνοψη αλγορίθμων ταξινόμησης

Αλγόριθμος	Χρόνος	Σημειώσεις
selection-sort	$O(n^2)$	▪ αργή ▪ επί-τόπου ▪ για μικρά σετ (< 1K)
insertion-sort	$O(n^2)$	▪ αργή ▪ επί-τόπου ▪ για μικρά σετ (< 1K)
heap-sort	$O(n \log n)$	▪ γρήγορη ▪ επί-τόπου ▪ για μεγάλα σετ (1K — 1M)
merge-sort	$O(n \log n)$	▪ γρήγορη ▪ διαδοχική προσπέλαση δεδομένων ▪ για τεράστια σετ (> 1M)