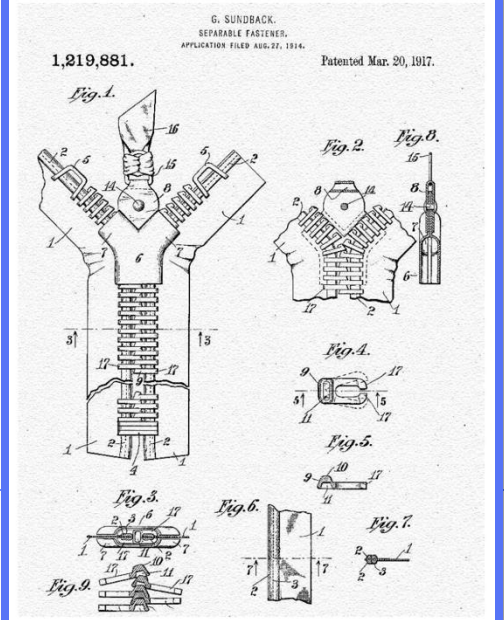
Παρουσίαση για χρήση με το σύγγραμμα, Αλγόριθμοι Σχεδίαση και Εφαρμογές, των M. T. Goodrich and R. Tamassia, Wiley, 2015 (στα ελληνικά από εκδόσεις Μ. Γκιούρδας)

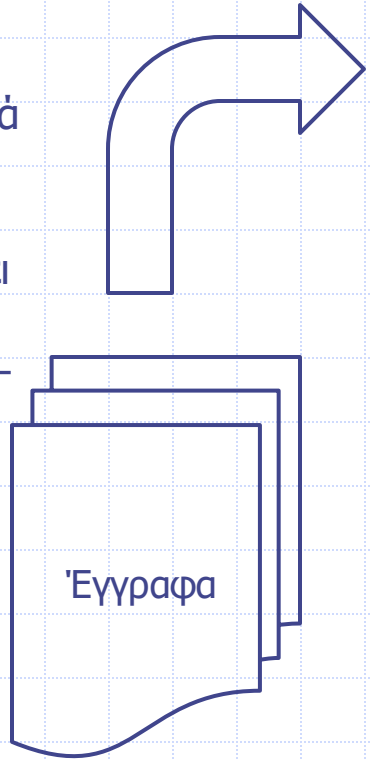
Ταξινόμηση με συγχώνευση (Merge Sort)



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

Εφαρμογή: Μηχανές αναζήτησης στο Internet

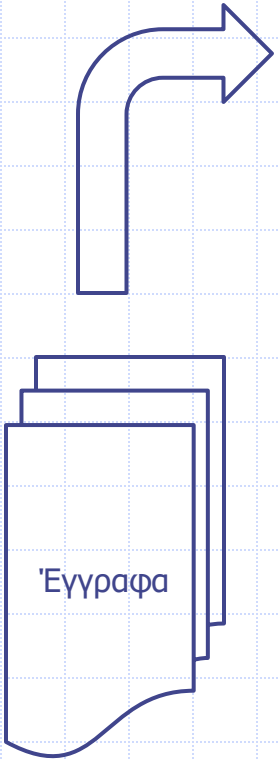
- Η ταξινόμηση έχει πολλές εφαρμογές, συμπεριλαμβανομένων των μηχανών αναζήτησης.
- Η ταξινόμηση ανακύπτει στα βήματα που πρέπει να γίνουν κατά την κατασκευή μιας δομής δεδομένων, που επιτρέπει σε μια μηχανή αναζήτησης να επιστρέφει γρήγορα μια λίστα έγγραφων που περιέχουν μια συγκεκριμένη λέξη–κλειδί. Αυτή η δομή δεδομένων ονομάζεται **ανεστραμμένο αρχείο (inverted file) ή ανεστραμμένο ευρετήριο (inverted index)**.



Λέξη	Αριθμός εγγράφου & θέση λέξης
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: Πώς η ταξινόμηση κατασκευάζει μία μηχανή αναζήτησης;

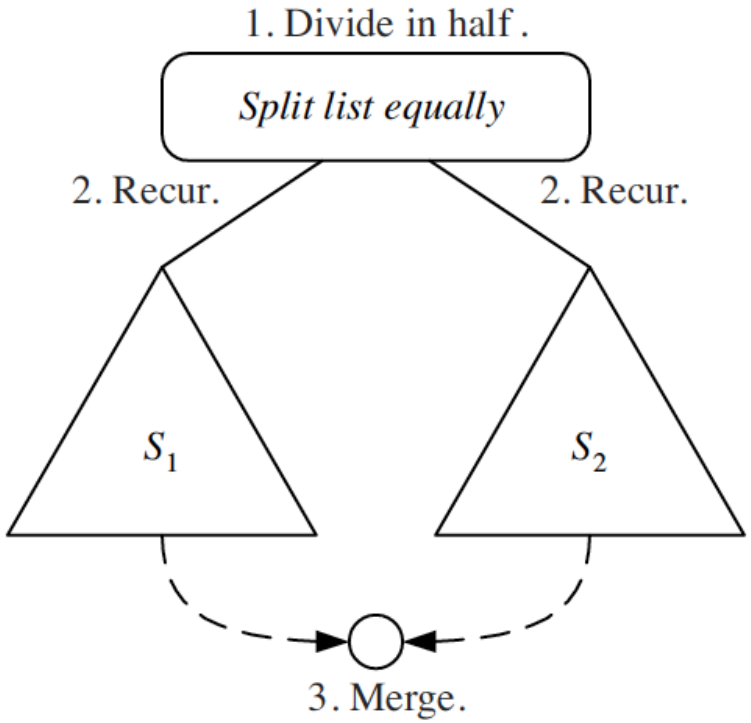
- Η κατασκευή ενός ανεστραμμένου αρχείου, προϋποθέτει την αναγνώριση, για κάθε λέξη-κλειδί k, όλων των εγγράφων που περιέχουν το k.
- Η συγκέντρωση όλων των σχετικών εγγράφων μαζί μπορεί να γίνει απλά με ταξινόμηση του συνόλου των ζευγών (λέξεων-κλειδιών, εγγράφων) κατά λέξεις-κλειδιά.
- Αυτή η ενέργεια τοποθετεί όλα τα ζεύγη (k, d) με την ίδια λέξη-κλειδί k, το ένα μετά το άλλο.
- Διασχίζοντας την ταξινομημένη λίστα, μπορούμε να κατασκευάσουμε έναν πίνακα αναζήτησης εγγράφων για κάθε λέξη-κλειδί.



Λέξη	Αριθμός εγγράφου & θέση λέξης
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

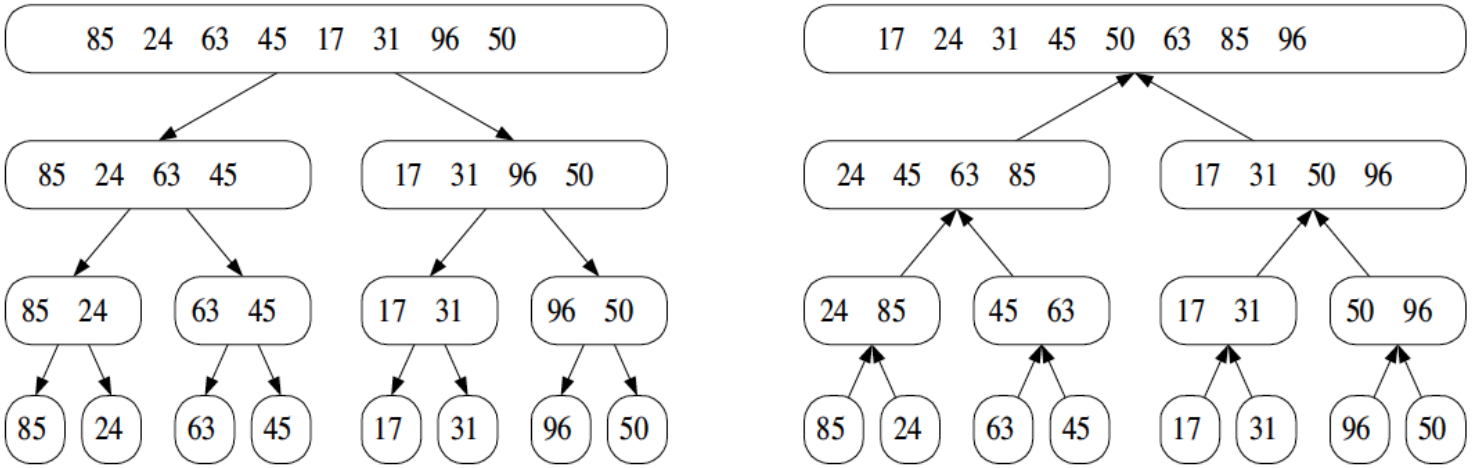
Διαίρει και Βασίλευε (Divide & Conquer)

- Η αλγοριθμική προσέγγιση «Διαίρει και βασίλευε» είναι ένα γενικό παράδειγμα σχεδίασης αλγορίθμων που περιγράφεται από τρία βήματα:
  - Διαίρει: διαιρεί τα δεδομένα εισόδου S σε δύο ξεχωριστά υποσύνολα S_1 και S_2
  - Επανάλαβε (αναδρομικά): Λύσε τα υποπροβλήματα, που συσχετίζονται με τα υποσύνολα S_1 και S_2
  - Βασίλευε: συγχώνευσε 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)

- Η ταξινόμηση με συγχώνευση είναι ένας αλγόριθμος ταξινόμησης που βασίζεται στο «παράδειγμα» διαίρει και βασίλευε
- Όπως η ταξινόμηση σωρού (heap-sort)
  - Έχει χρόνο εκτέλεσης $O(n \log n)$
- Αντίθετα με την ταξινόμηση σωρού
  - Δεν χρησιμοποιεί βοηθητική ουρά προτεραιότητας
  - Η προσπέλαση των δεδομένων γίνεται διαδοχικά (είναι κατάλληλη μέθοδος για ταξινόμηση δεδομένων που βρίσκονται σε σκληρό δίσκο)



Ο Αλγόριθμος Merge-Sort

- Merge-sort σε μία είσοδο S με n στοιχεία:
  - Διαίρει: Τα στοιχεία του S τοποθετούνται σε δύο ακολουθίες S_1 και S_2 με καθεμία να περιέχει περίπου $n/2$ στοιχεία
  - Επανάλαβε: Οι ακολουθίες S_1 και S_2 ταξινομούνται αναδρομικά
  - Βασίλευε: Οι ακολουθίες S_1 και S_2 συγχωνεύονται σε μία ενιαία 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

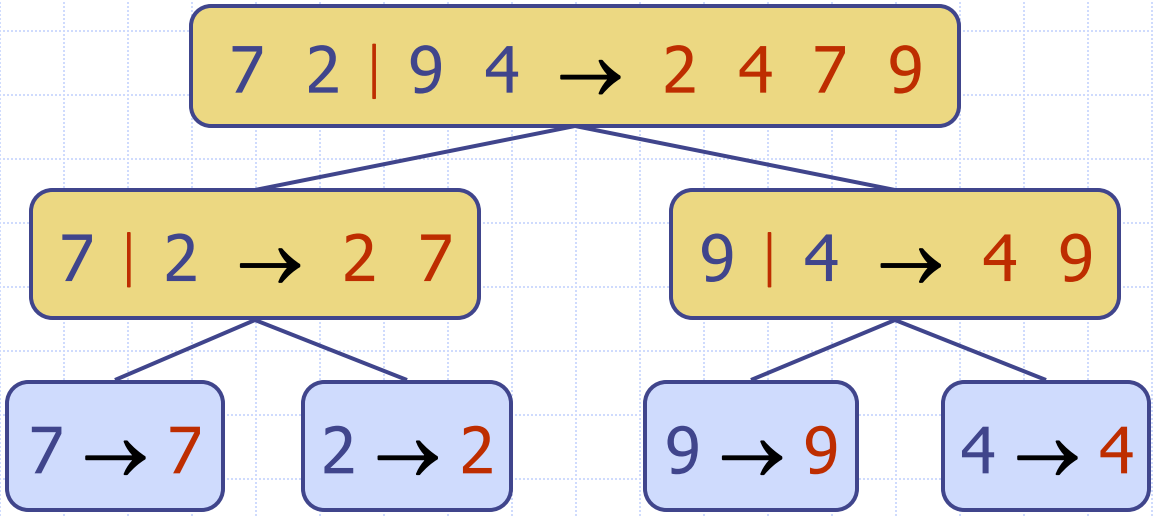
- Το τελευταίο βήμα (βασίλευε - κυρίευσε) συγχωνεύει δύο ταξινομημένες ακολουθίες A και B σε μία ταξινομημένη ακολουθία S που περιέχει την ένωση των στοιχείων του A και του B
- Η συγχώνευση δύο ταξινομημένων ακολουθιών, με καθεμία ακολουθία να έχει $n/2$ στοιχεία απαιτεί χρόνο $O(n)$

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**
  $S[i + j - 1] \leftarrow S_2[j]$
  $j \leftarrow j + 1$

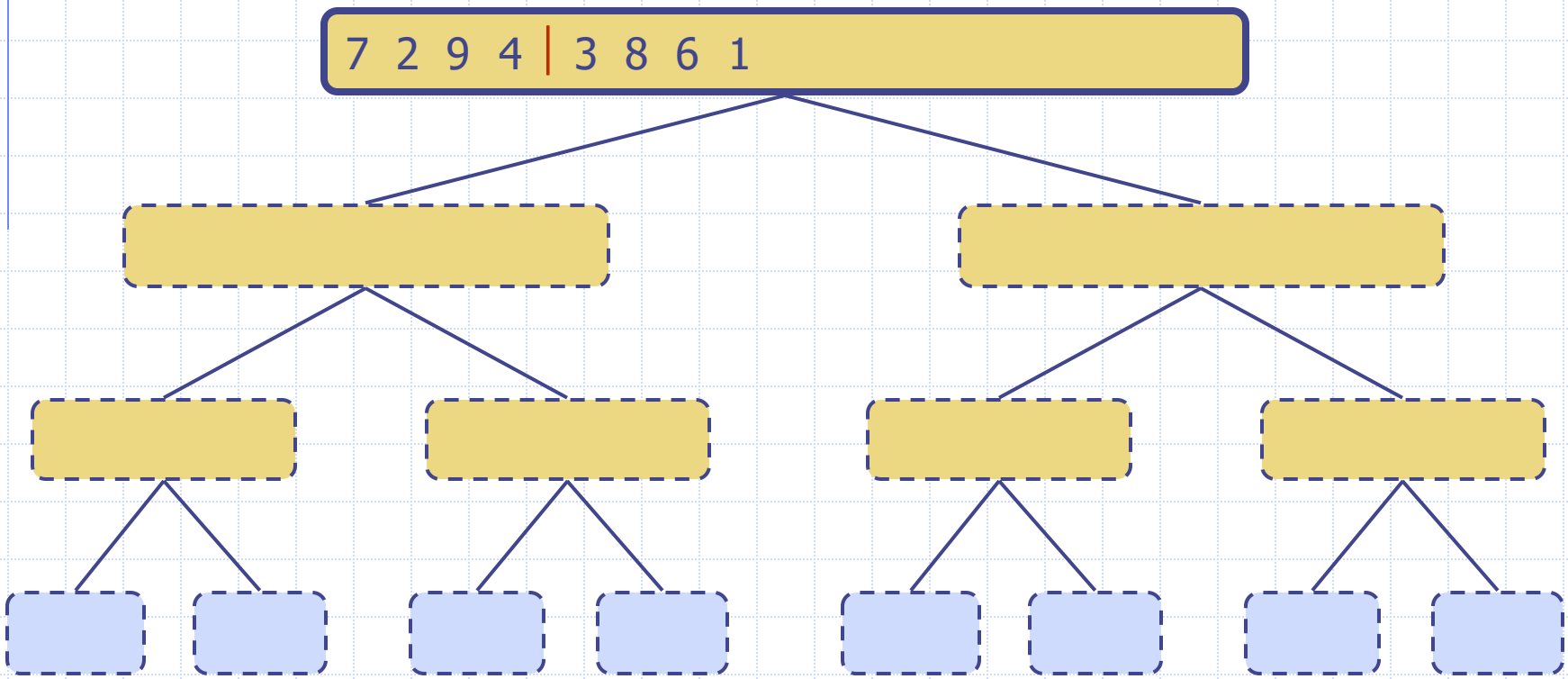
Δένδρο Merge-Sort

- Η εκτέλεση της merge-sort αναπαρίσταται με ένα δυαδικό δένδρο
  - Κάθε κόμβος αφορά μία αναδρομική κλήση του merge-sort και αποθηκεύει
    - μια μη ταξινομημένη ακολουθία πριν την εκτέλεση και το διαμερισμό της
    - μια ταξινομημένη ακολουθία στο τέλος της εκτέλεσης
  - η ρίζα του δένδρου είναι η αρχική κλήση
  - τα φύλλα είναι οι κλήσεις σε υπό-ακολουθίες μεγέθους 0 ή 1



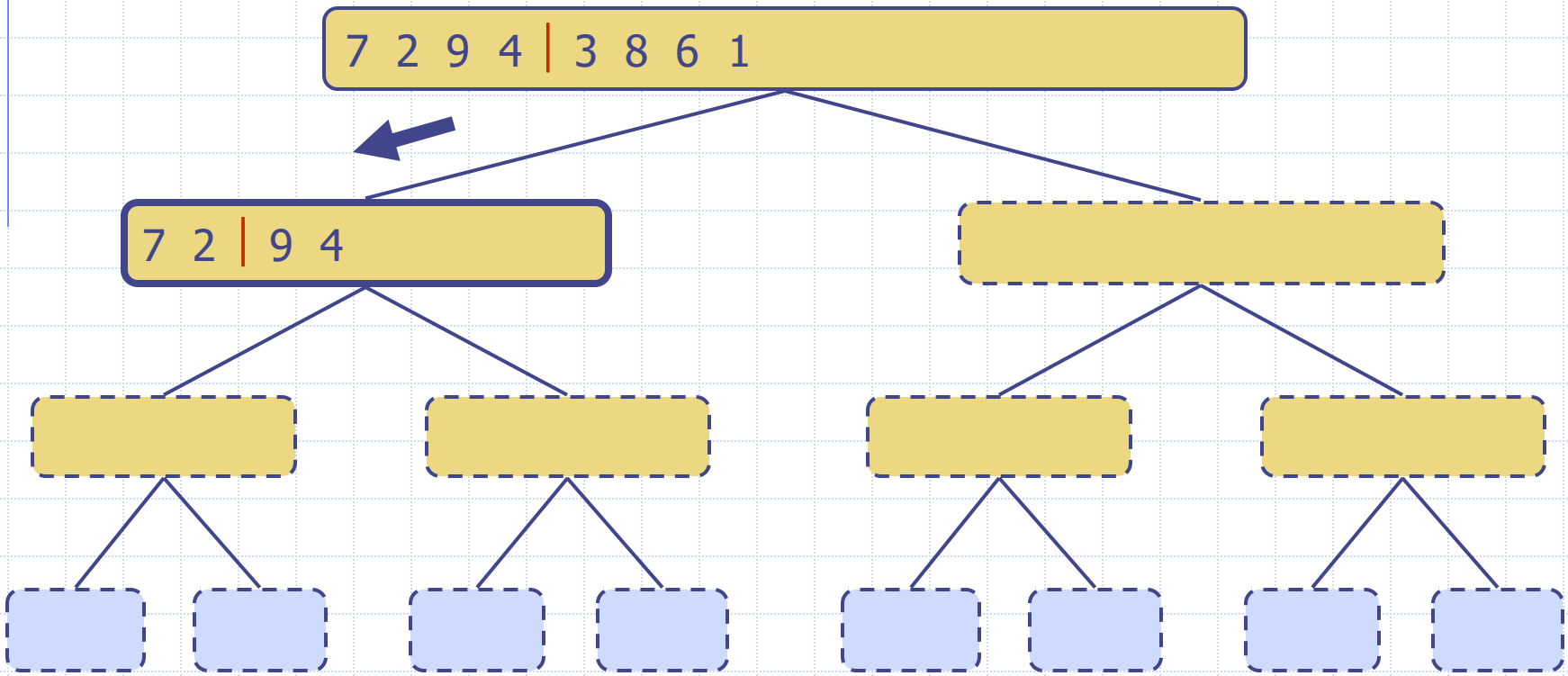
Παράδειγμα εκτέλεσης

- Διαίρει



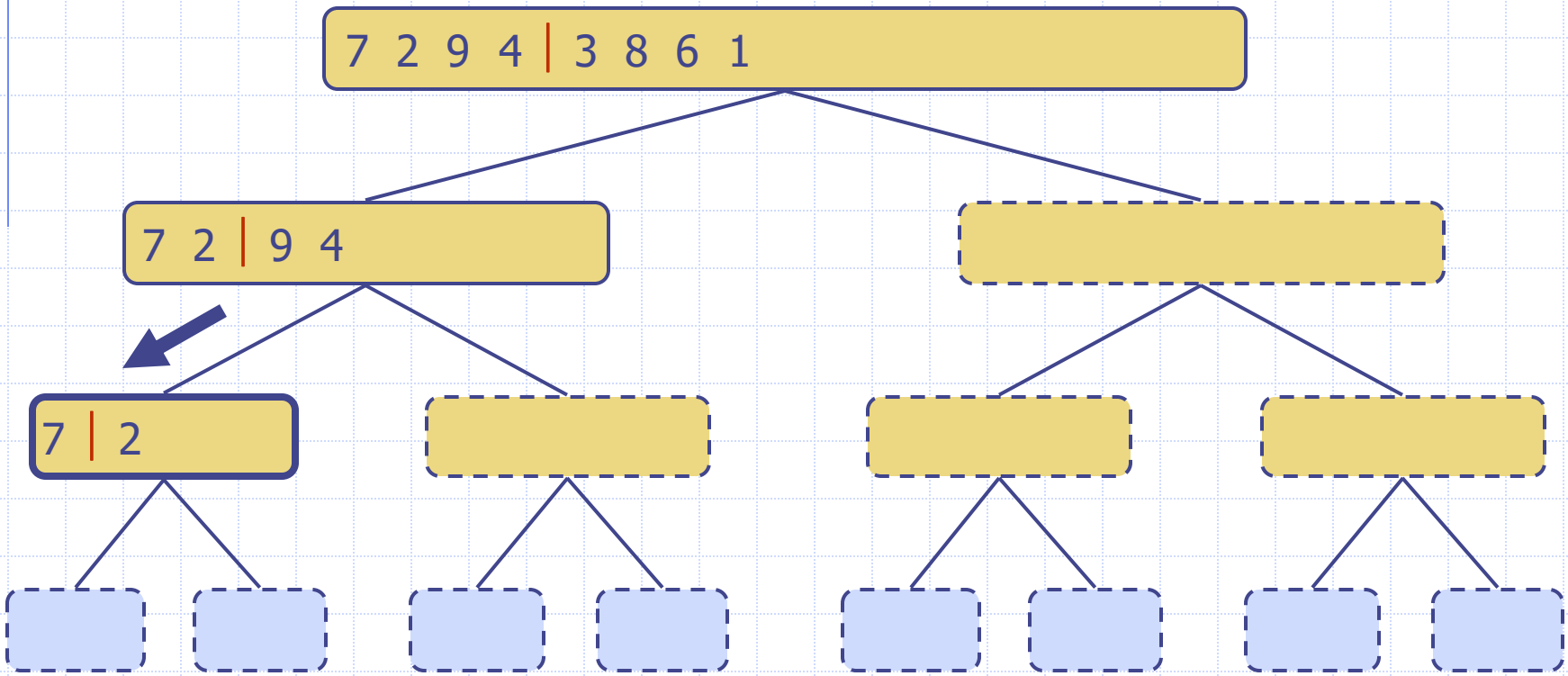
Παράδειγμα εκτέλεσης (συν.)

- Αναδρομική κλήση, διαμερισμός



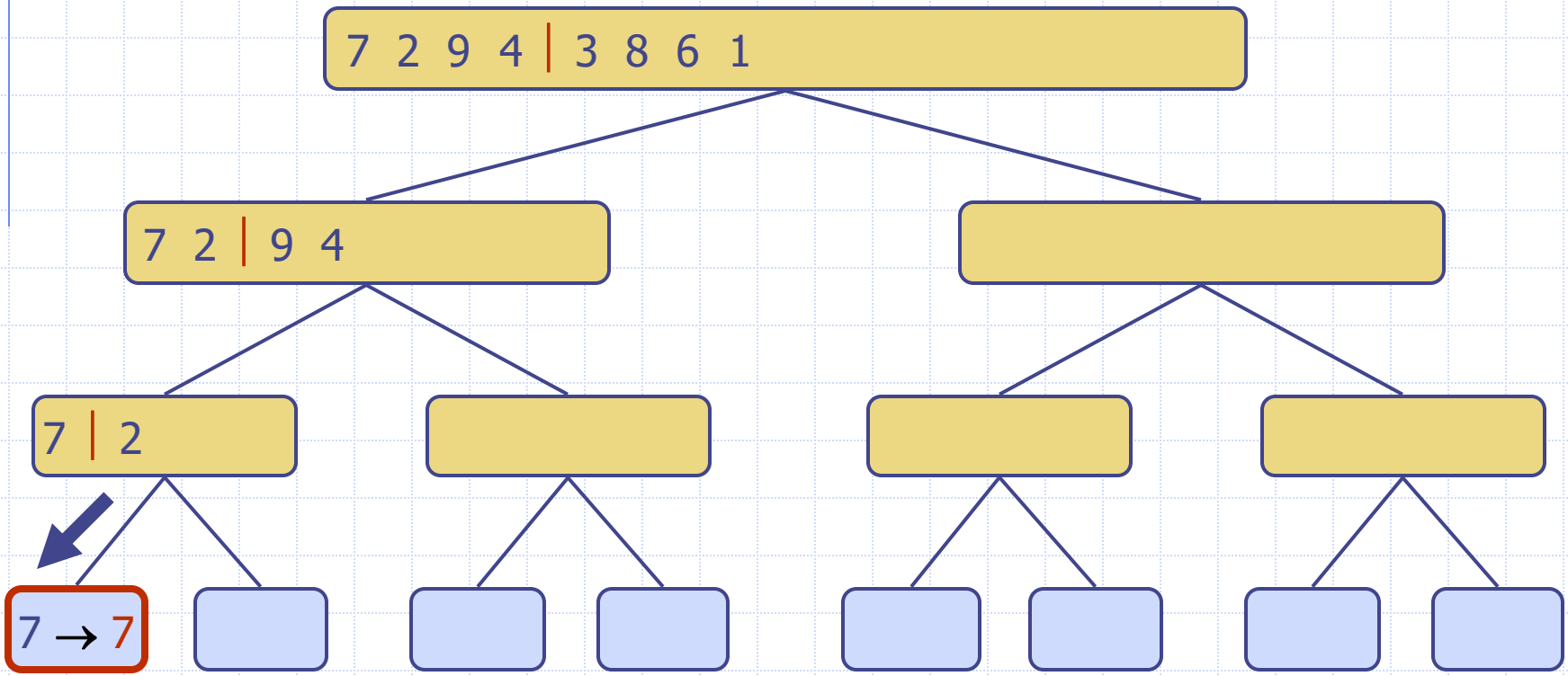
Παράδειγμα εκτέλεσης (συν.)

- Αναδρομική κλήση, διαμερισμός



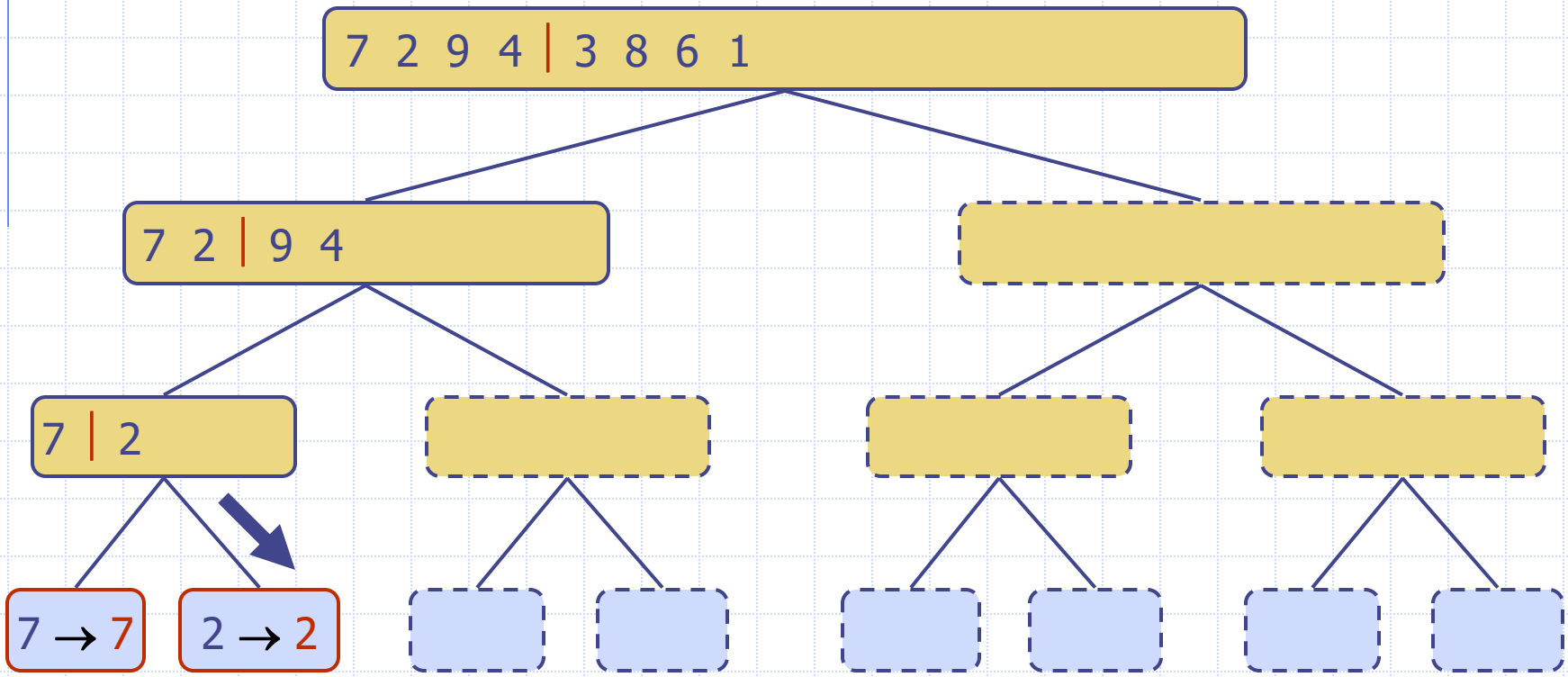
Παράδειγμα εκτέλεσης (συν.)

- Αναδρομική κλήση, βασική περίπτωση



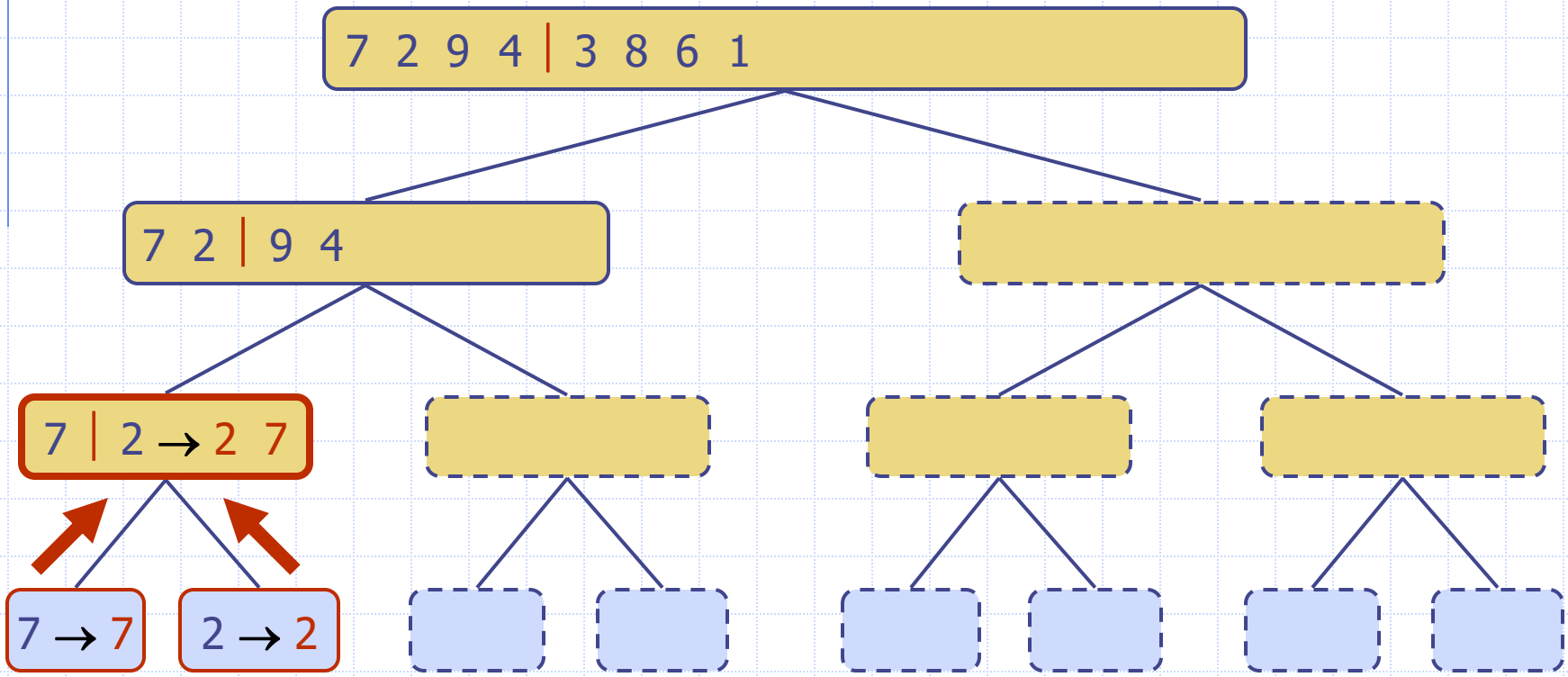
Παράδειγμα εκτέλεσης (συν.)

- Αναδρομική κλήση, βασική περίπτωση



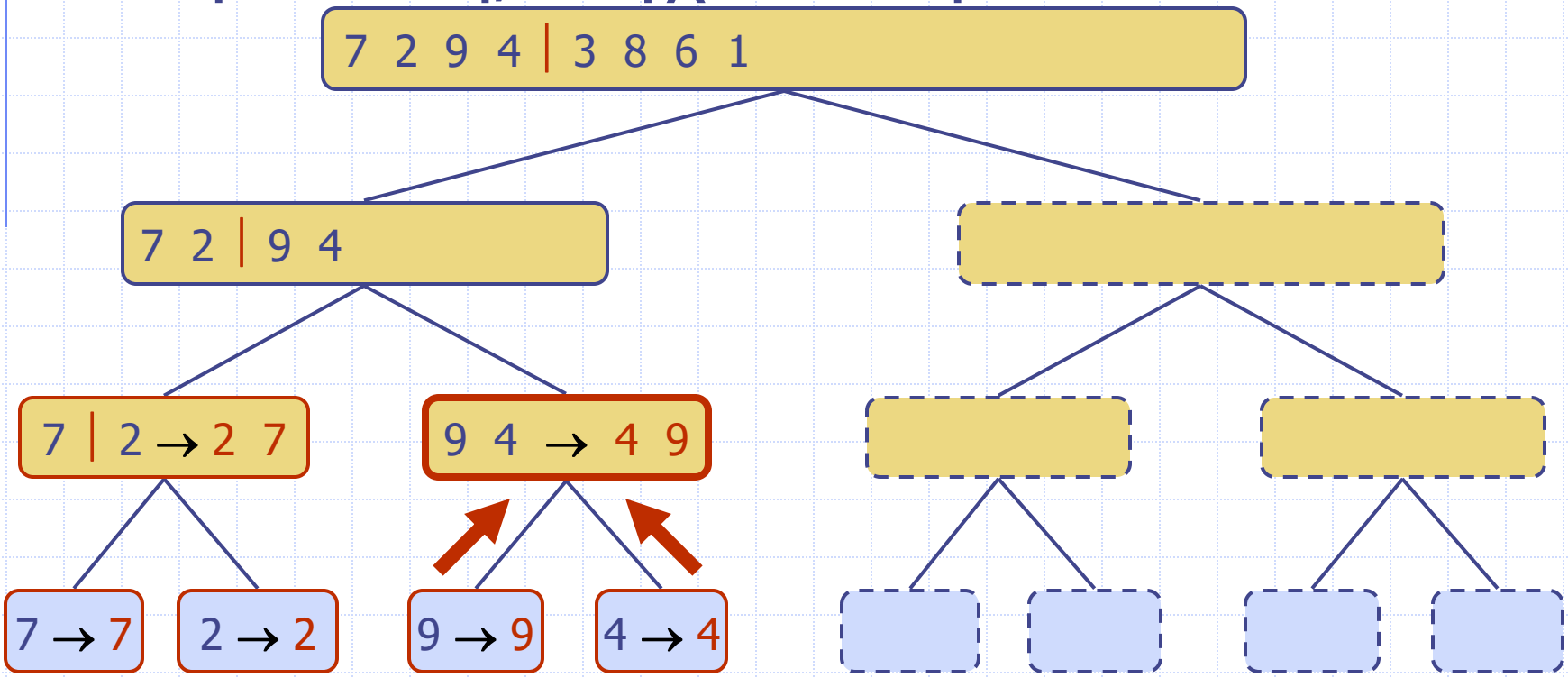
Παράδειγμα εκτέλεσης (συν.)

- Συγχώνευση



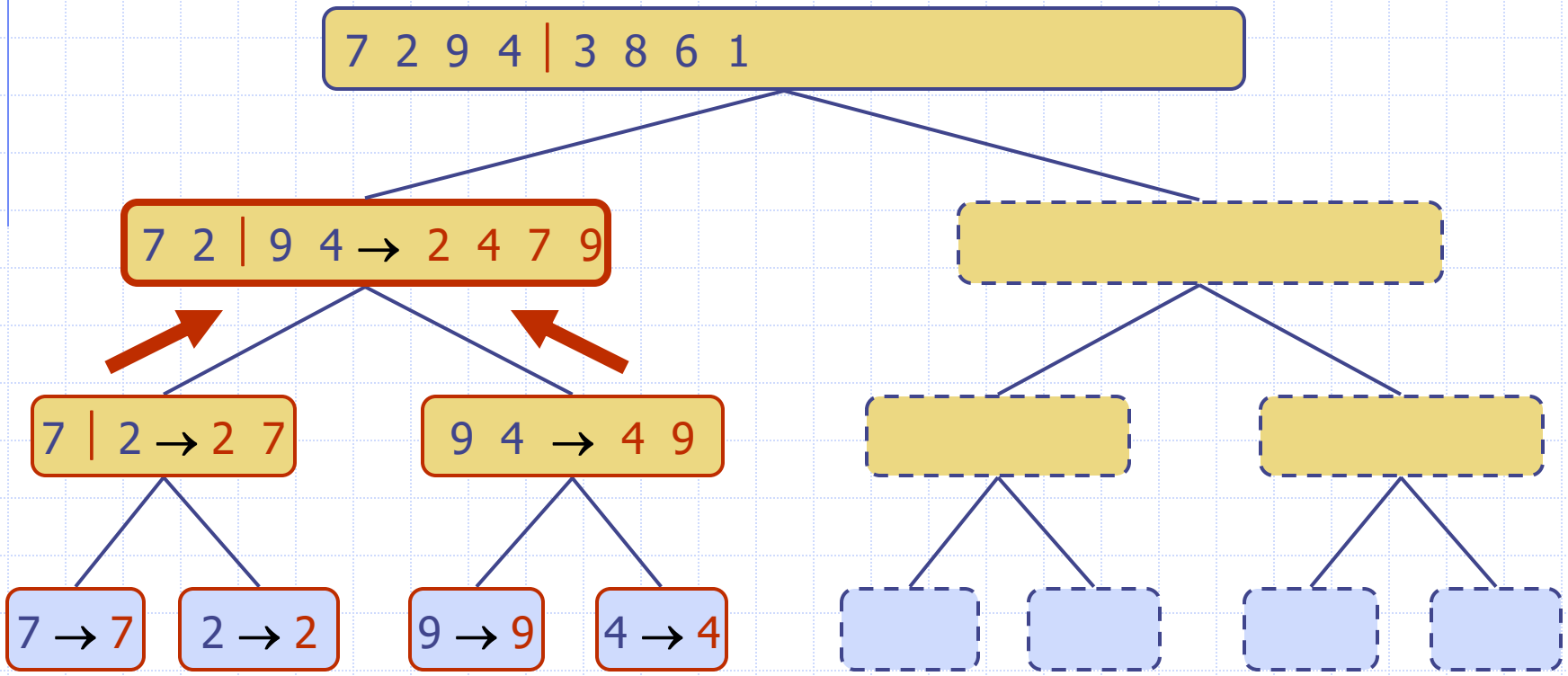
Παράδειγμα εκτέλεσης (συν.)

- Αναδρομική κλήση, …, βασική περίπτωση, συγχώνευση



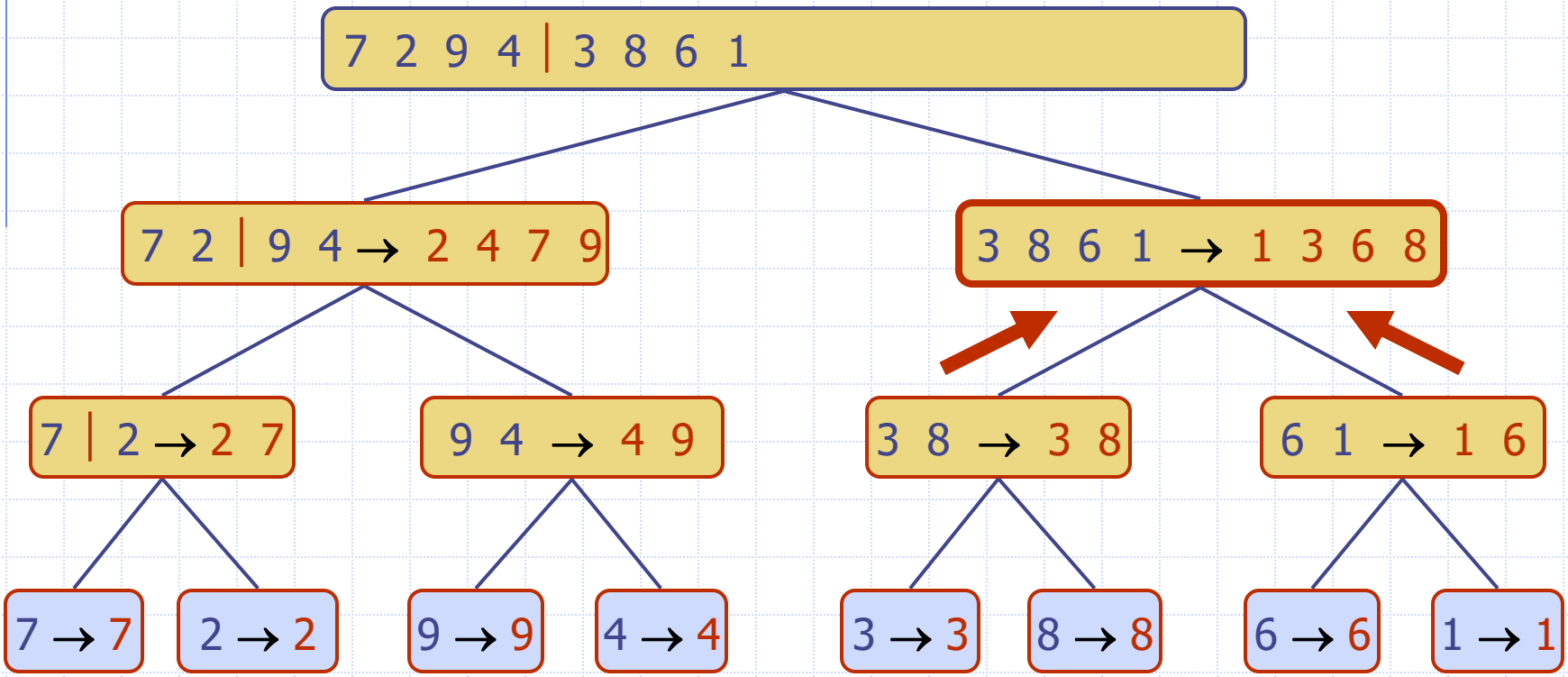
Παράδειγμα εκτέλεσης (συν.)

- Συγχώνευση



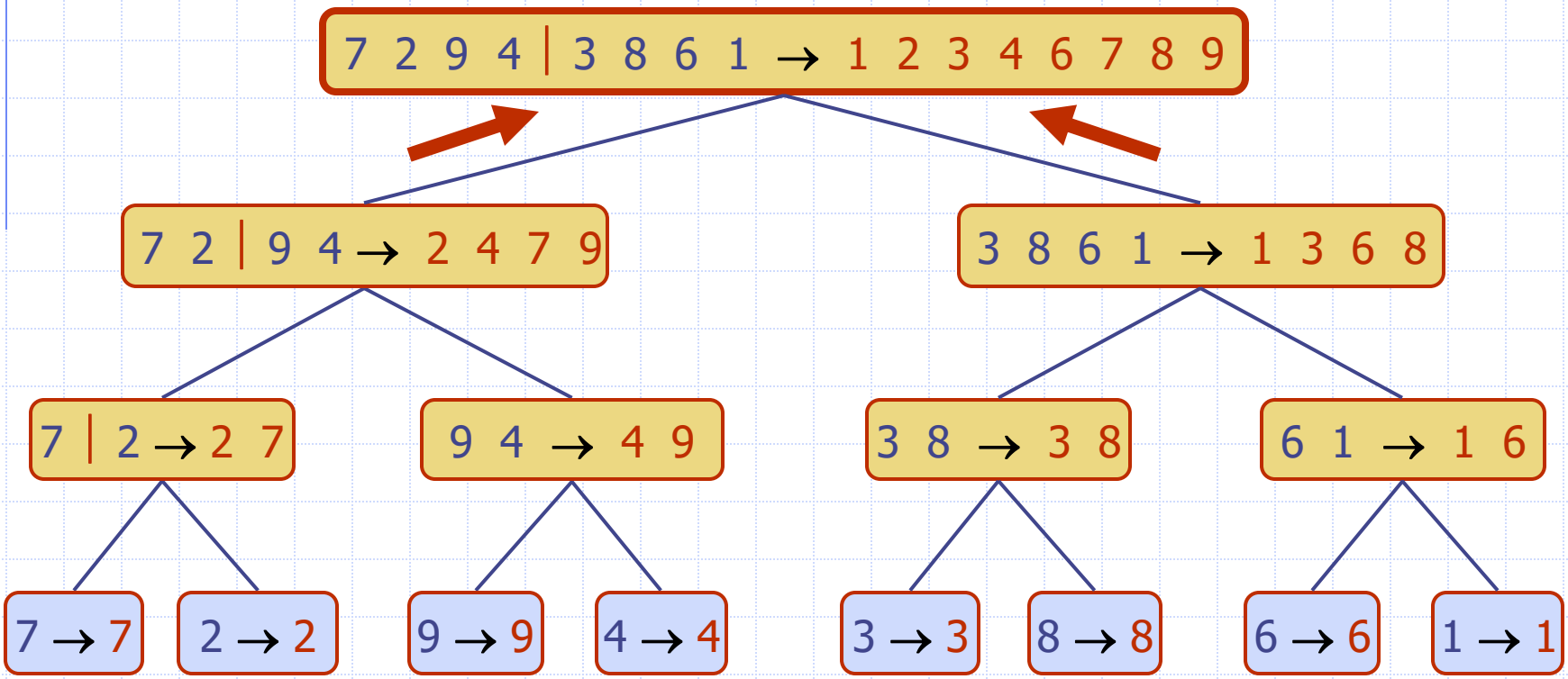
Παράδειγμα εκτέλεσης (συν.)

- Αναδρομική κλήση, …, συγχώνευση, συγχώνευση



Παράδειγμα εκτέλεσης (συν.)

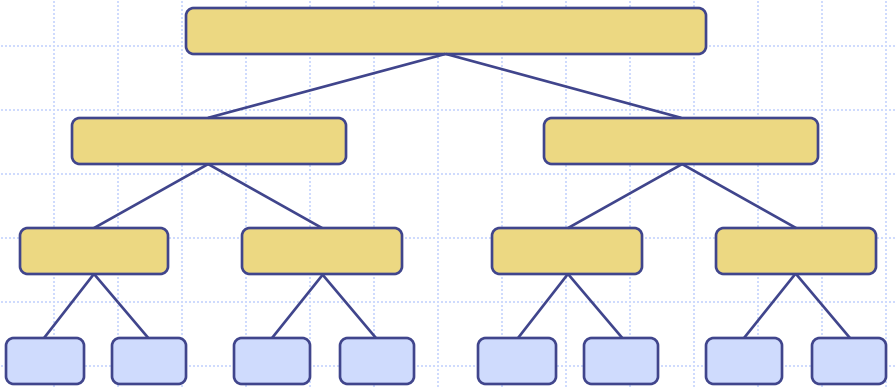
- Συγχώνευση



Ανάλυση Merge-Sort

- Το ύψος h του δένδρου merge-sort είναι $O(\log n)$
  - σε κάθε αναδρομική κλήση διαιρούμε την ακολουθία στη μέση
- Η συνολική ποσότητα εργασίας που επιτελείται στα φύλλα βάθους i είναι $O(n)$
  - διαμερίζουμε και συγχωνεύουμε 2^i ακολουθίες μεγέθους $n/2^i$
  - κάνουμε 2^{i+1} αναδρομικές κλήσεις
- Έτσι, ο συνολικός χρόνος εκτέλεσης του merge-sort είναι $O(n \log n)$

βάθος	#ακολουθιών	μέγεθος
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 (in-place) ▪ για μικρά σετ δεδομένων (< 1K)
insertion-sort	$O(n^2)$	▪ αργή ▪ επί-τόπου ▪ για μικρά σετ δεδομένων (< 1K)
heap-sort	$O(n \log n)$	▪ γρήγορη ▪ επί-τόπου ▪ για μεγάλα σετ δεδομένων (1K — 1M)
merge-sort	$O(n \log n)$	▪ γρήγορη ▪ σειριακή προσπέλαση δεδομένων ▪ για τεράστια σετ δεδομένων (> 1M)