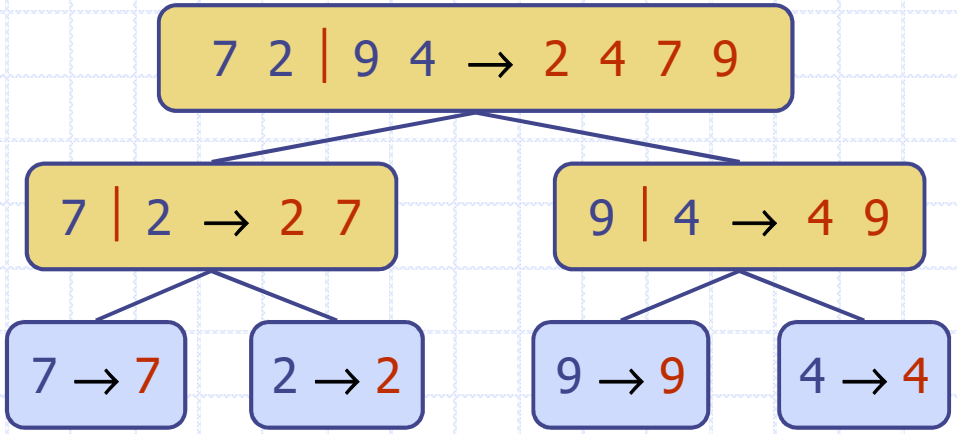
Ταξινόμηση με Συγχώνευση



Διαίρει-και-Βασίλευε

- Το Διαίρει-και βασίλευε είναι ένα παράδειγμα γενικού αλγοριθμικού σχεδιασμού:
  - Διαίρει: διαιρεί την είσοδο των δεδομένων S σε δύο ξένα υποσύνολα S_1 και S_2
  - Αναδρομή: επίλυση των υποπροβλημάτων που σχετίζονται με τα S_1 και S_2
  - Βασίλευε: συνδυασμός των λύσεων για τα S_1 και S_2 σε μια λύση για το S
- Η βασική περίπτωση για την αναδρομή είναι υποπροβλήματα μεγέθους 0 ή 1

- Ταξινόμηση με συγχώνευση είναι ένας αλγόριθμος ταξινόμησης που βασίζεται στο παράδειγμα διαίρει και βασίλευε
- Όπως η ταξινόμηση σωρού
  - Χρησιμοποιεί έναν συγκριτή
  - Έχει $O(n \log n)$ χρόνο τρεξίματος
- Σε αντίθεση με την ταξινόμηση σωρού
  - Δεν χρησιμοποιεί βοηθητική ουρά προτεραιότητας
  - Η προσπέλαση στα δεδομένα γίνεται κατά γραμμικό τρόπο (κατάλληλη για ταξινόμηση σε δίσκο)

Ταξινόμηση με Συγχώνευση

- Η ταξινόμηση με συγχώνευση σε μια είσοδο S με n στοιχεία αποτελείται από τρία βήματα:
  - Διαίρει: διαμερίζουμε την S σε δύο ακολουθίες S_1 και S_2 με περίπου $n/2$ στοιχεία η κάθε μια
  - Αναδρομή: αναδρομική ταξινόμηση των S_1 και S_2
  - Βασίλευε: συγχώνευση των S_1 και S_2 σε μια μοναδική ταξινομημένη ακολουθία

Algorithm *mergeSort*(S, C)

Input sequence S with n elements, comparator C

Output sequence S sorted according to C

if $S.size() > 1$

$(S_1, S_2) \leftarrow partition(S, n/2)$

$mergeSort(S_1, C)$

$mergeSort(S_2, C)$

$S \leftarrow merge(S_1, S_2)$

Συγχώνευση δύο ταξινομημένων ακολουθιών

- Το βήμα βασίλευε της ταξινόμησης με συγχώνευση αποτελείται με την συγχώνευση των ταξινομημένων ακολουθιών A και B σε μια ταξινομημένη ακολουθία S που περιέχει την ένωση των στοιχείων των A και B
- Η συγχώνευση δυο ταξινομημένων ακολουθιών με $n/2$ στοιχεία που υλοποιείται με μια διπλά συνδεδεμένη λίστα απαιτεί χρόνο $O(n)$

Algorithm *merge*(*A*, *B*)

Input sequences *A* and *B* with *n*/2 elements each

Output sorted sequence of $A \cup B$

S ← empty sequence

while $\neg$*A.isEmpty*() $\wedge$ $\neg$*B.isEmpty*()

if *A.first*().*element*() < *B.first*().*element*()

S.addLast(*A.remove*(*A.first*()))

else

S.addLast(*B.remove*(*B.first*()))

while $\neg$*A.isEmpty*()

S.addLast(*A.remove*(*A.first*()))

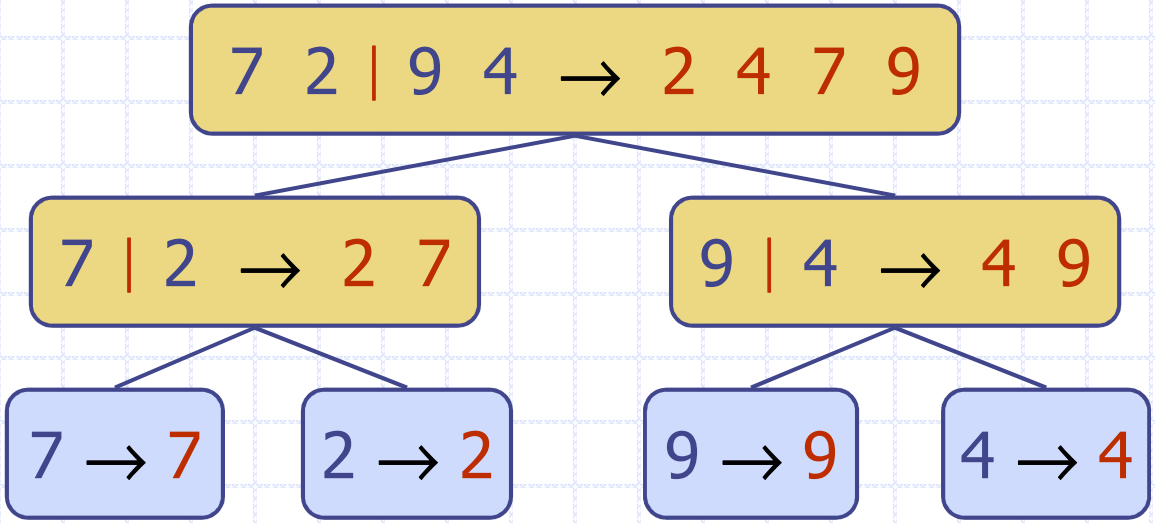
while $\neg$*B.isEmpty*()

S.addLast(*B.remove*(*B.first*()))

return *S*

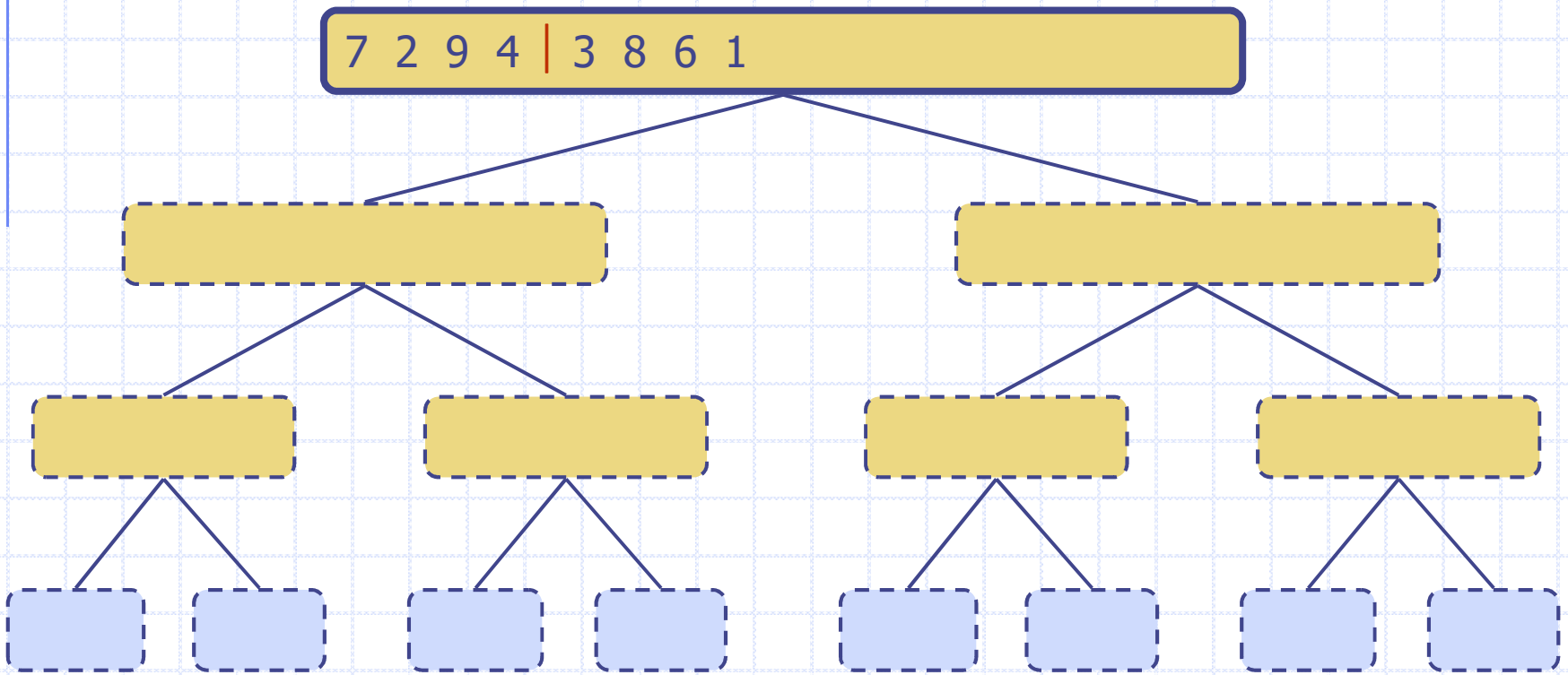
Δένδρο Ταξινόμησης Συγχώνευσης

- Μια εκτέλεση της ταξινόμησης με συγχώνευση απεικονίζεται με ένα δυαδικό δένδρο
  - κάθε κόμβος παριστάνει μια αναδρομική κλήση της ταξινόμησης με συγχώνευση και αποθηκεύει
    - αταξινόμητη ακολουθία πριν την εκτέλεση και την διαμέριση
    - ταξινομημένη λίστα στο τέλος της εκτέλεσης
  - η ρίζα είναι η αρχική κλήση
  - τα φύλλα είναι κλήσεις ακολουθιών μεγέθους 0 ή 1



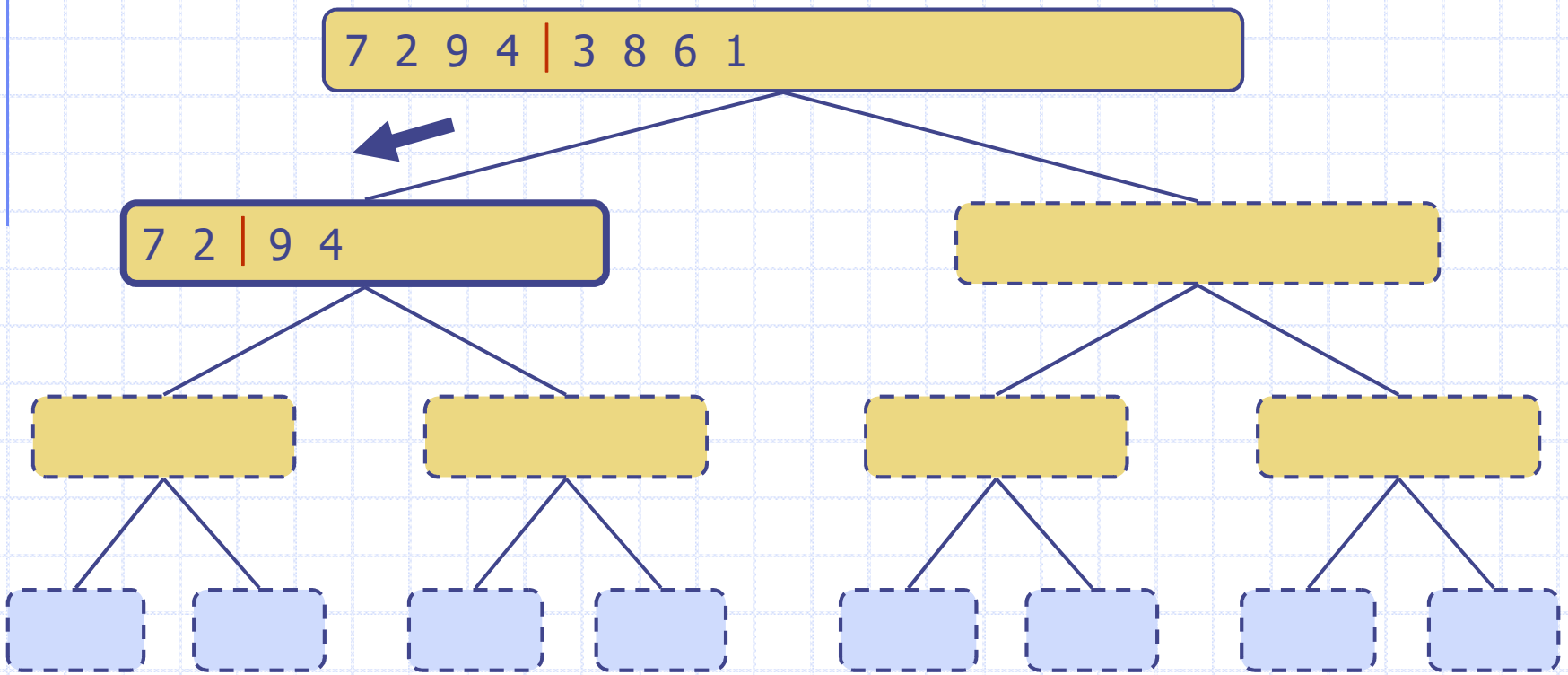
Παράδειγμα Εκτέλεσης

- Διαμέριση



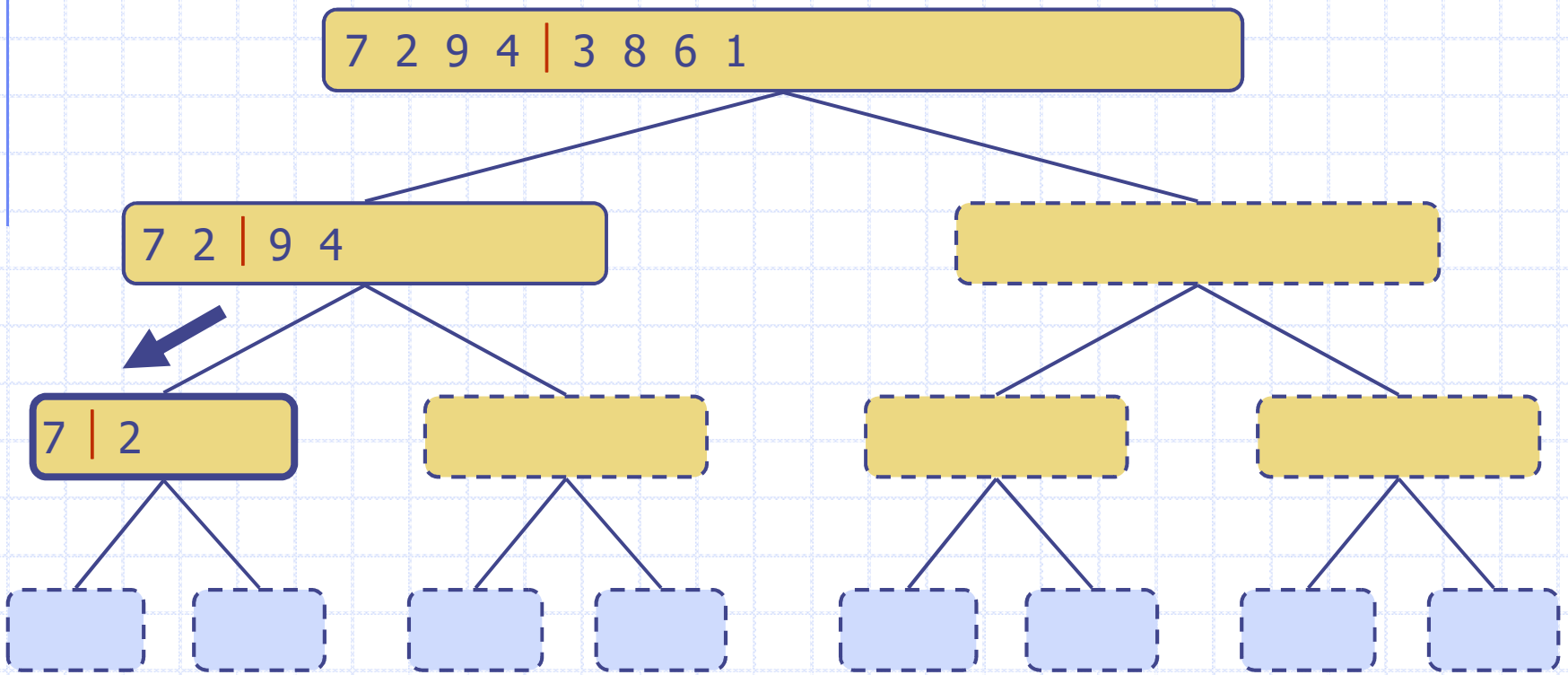
Παράδειγμα Εκτέλεσης (συν.)

- Αναδρομική κλήση, διαμέριση



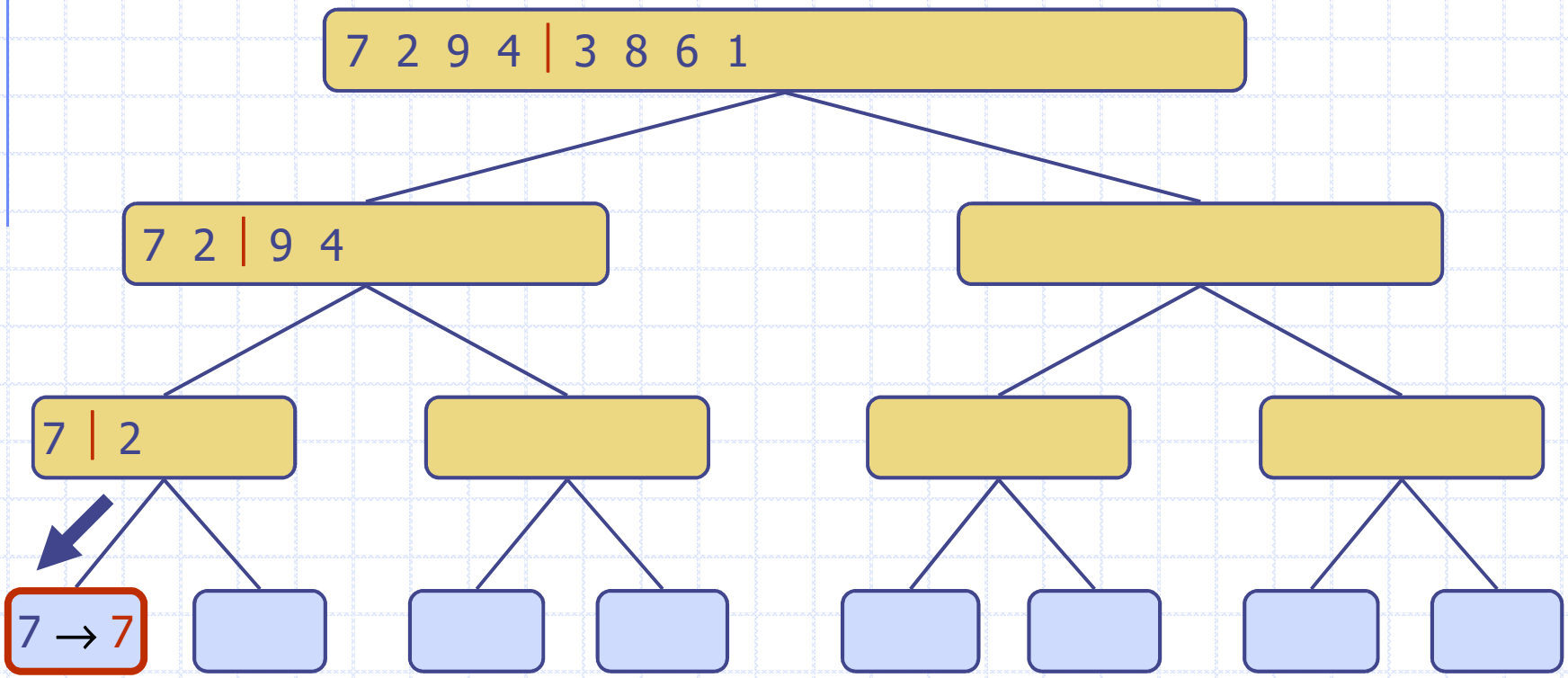
Παράδειγμα Εκτέλεσης (συν.)

- Αναδρομική κλήση, διαμέριση



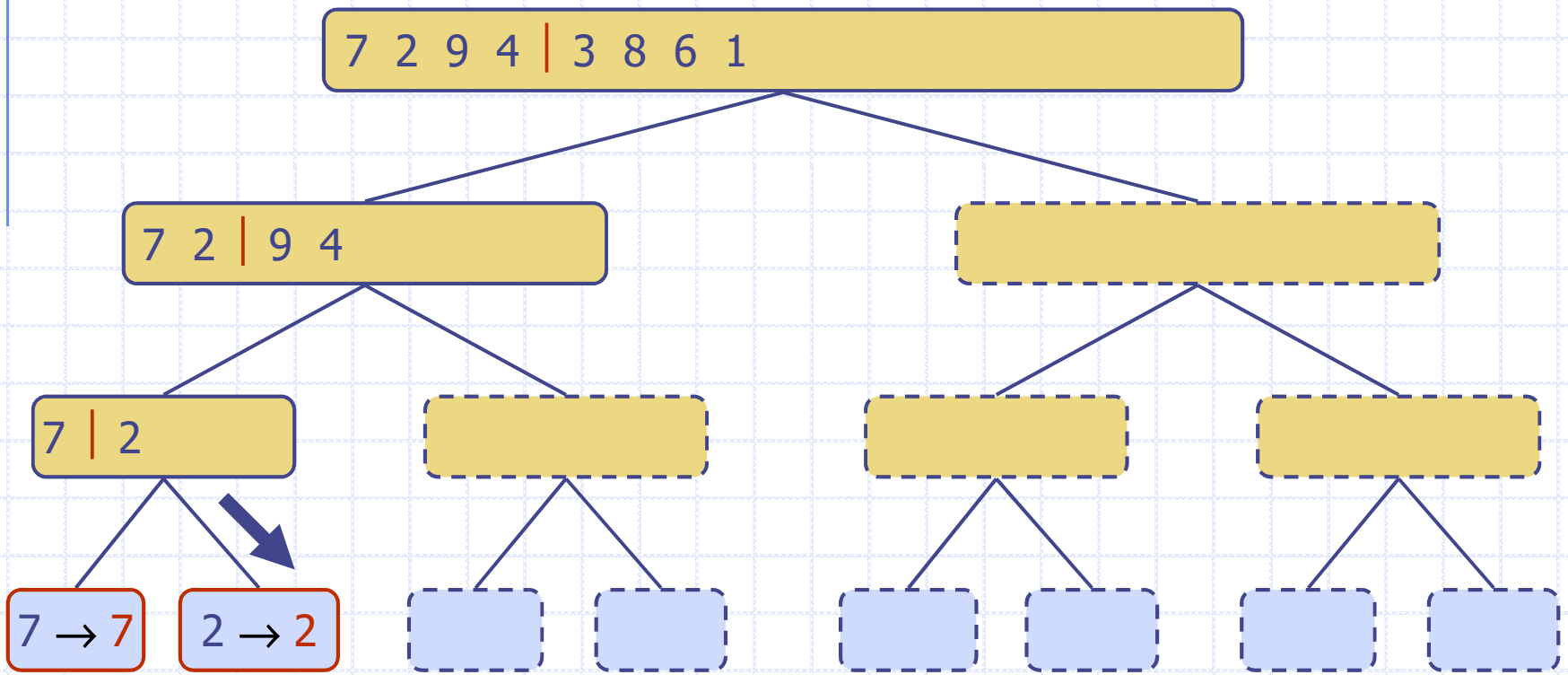
Παράδειγμα Εκτέλεσης (συν.)

- Αναδρομική κλήση, περίπτωση βάσης



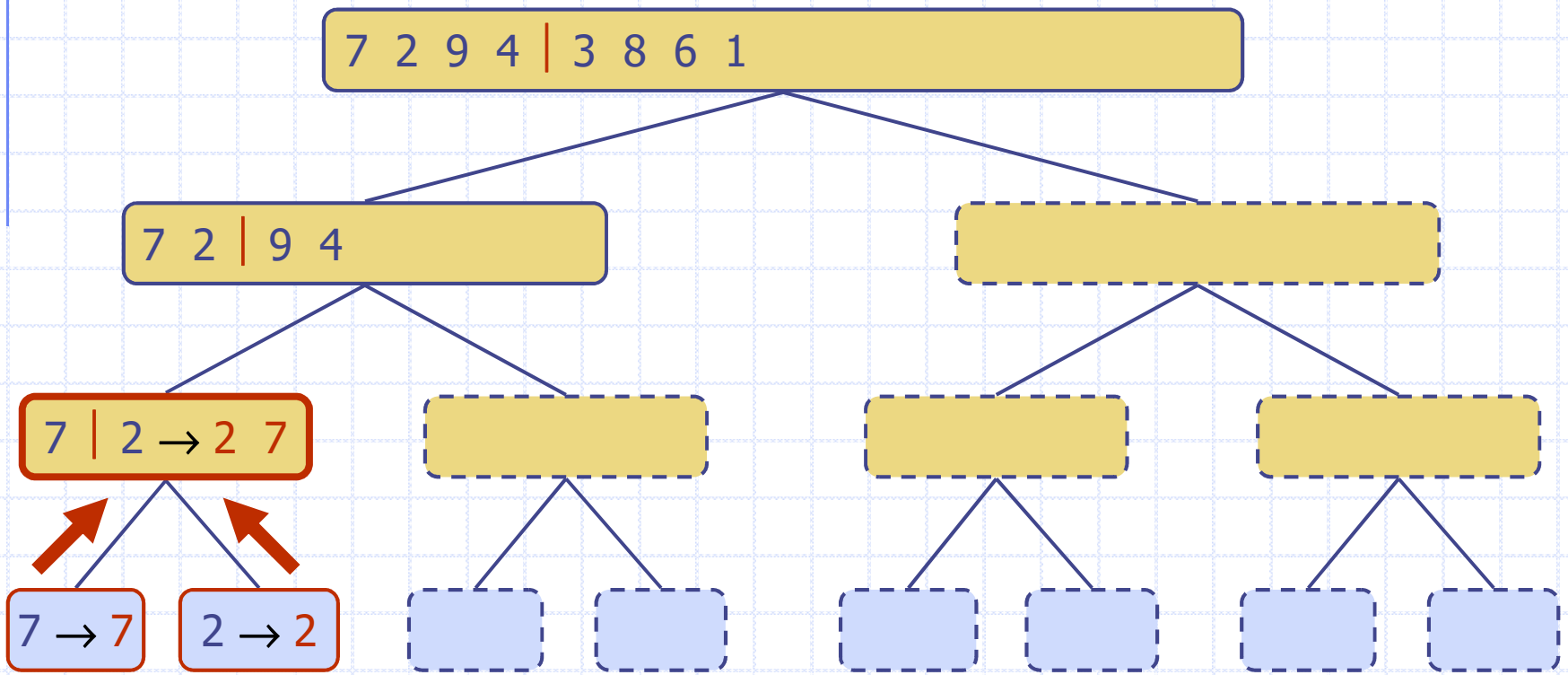
Παράδειγμα Εκτέλεσης (συν.)

- Αναδρομική κλήση, περίπτωση βάσης



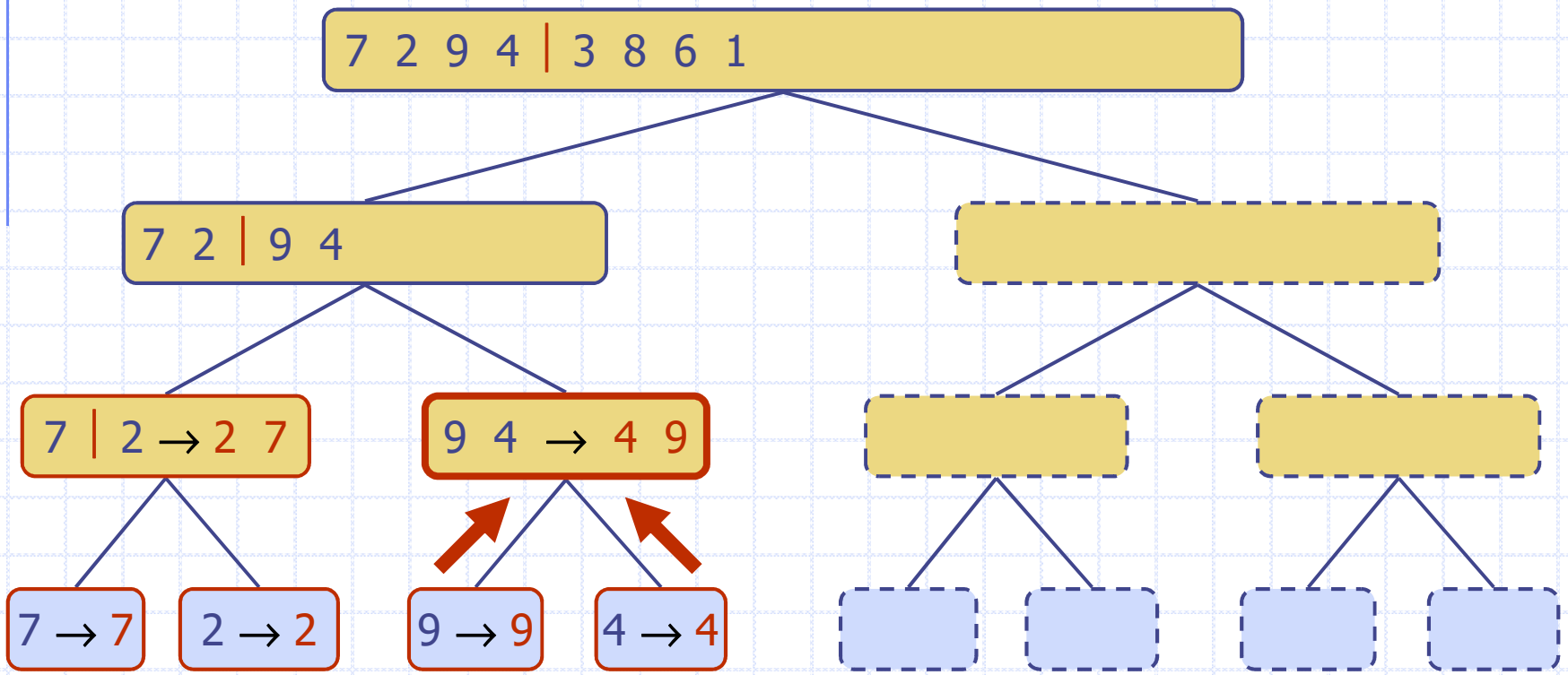
Παράδειγμα Εκτέλεσης (συν.)

- Συγχώνευση



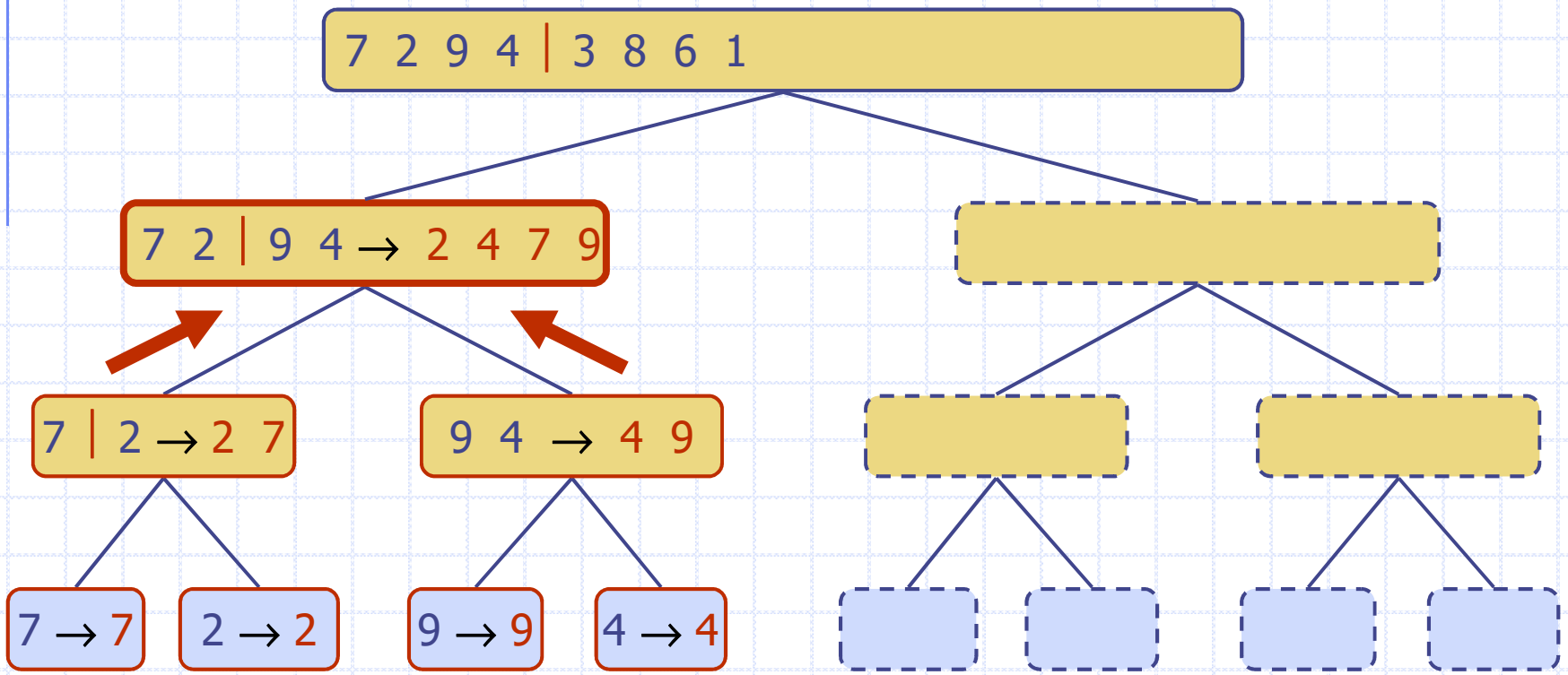
Παράδειγμα Εκτέλεσης (συν.)

- Αναδρομική κλήση, ..., περίπτωση βάσης, συγχώνευση



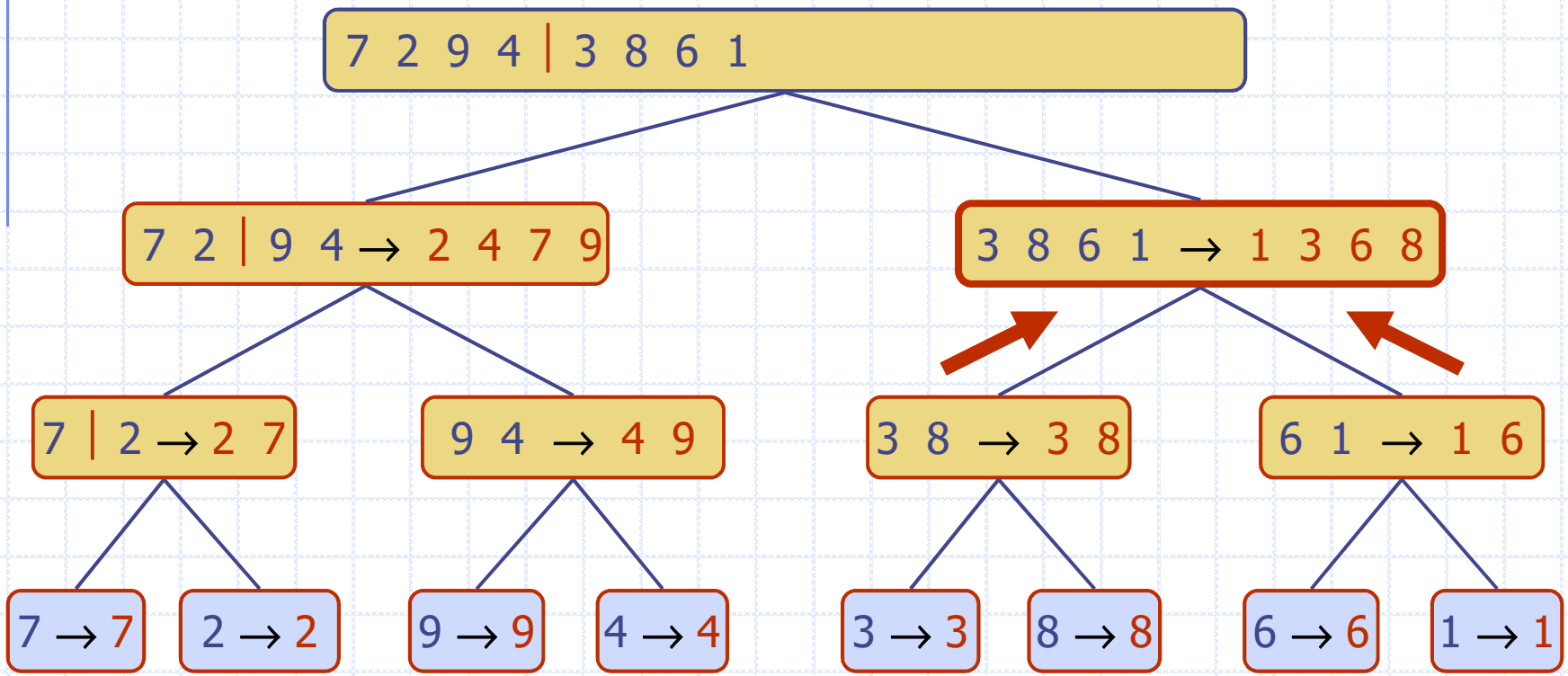
Παράδειγμα Εκτέλεσης (συν.)

- Συγχώνευση



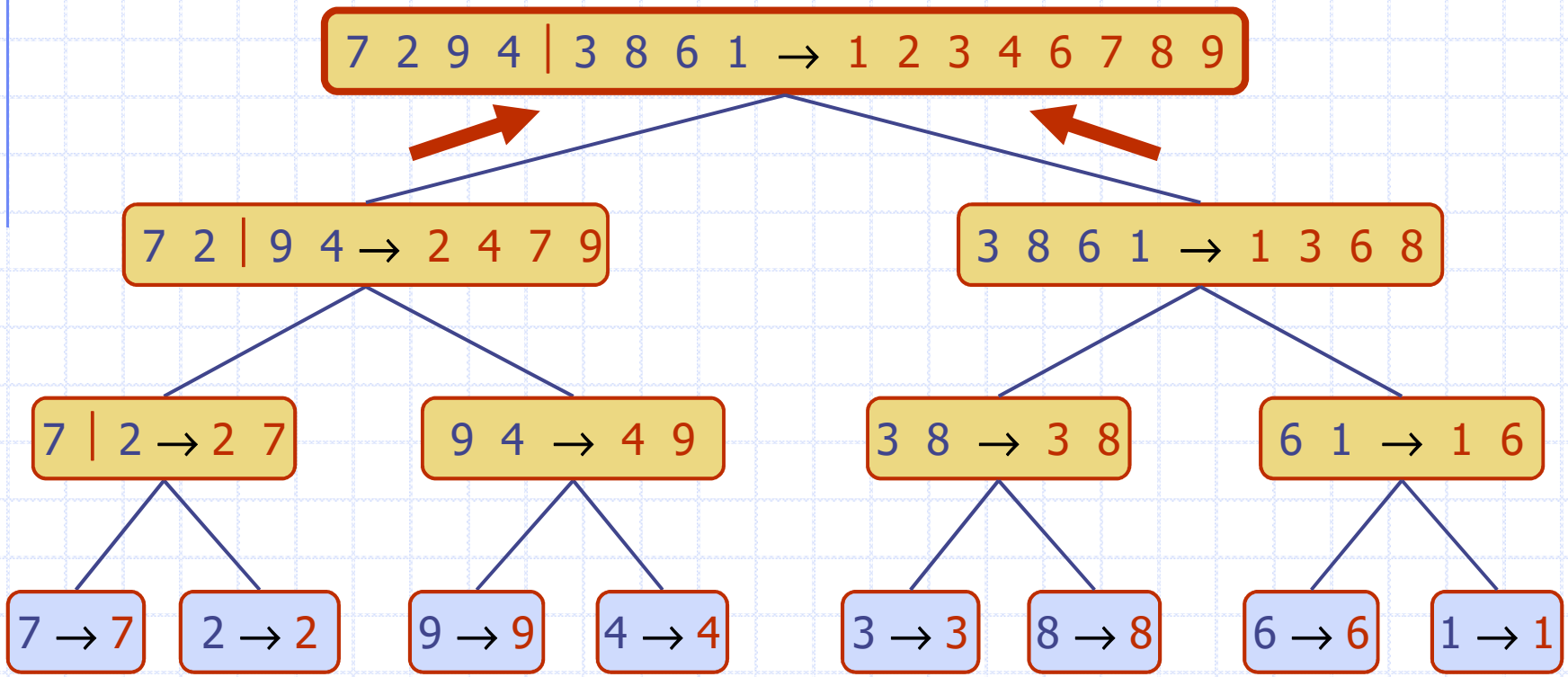
Παράδειγμα Εκτέλεσης (συν.)

- Αναδρομική κλήση, …, συγχώνευση, συγχώνευση



Παράδειγμα Εκτέλεσης (συν.)

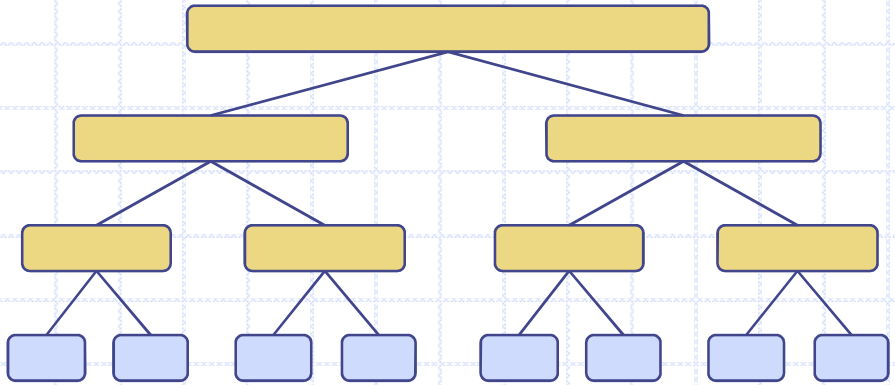
- Συγχώνευση



Ανάλυση της Ταξινόμησης με Συγχώνευση

- Το ύψος h οτου δένδρου ταξινόμησης συγχώνευσης είναι $O(\log n)$
  - σε κάθε αναδρομική κλήση χωρίζουμε την ακολουθία στα δύο,
- Η συνολική εργασία στους κόμβους βάθους i είναι $O(n)$
  - διαμερίζουμε και συγχωνεύουμε 2^i ακολουθίες μεγέθους $n/2^i$
  - εκτελούμε 2^{i+1} ταναδρομικές κλήσεις
- Επομένως, ο συνολικός χρόνος τρεξίματος της συγχώνευσης με ταξινόμηση είναι of $O(n \log n)$

βάθος	#ακολου	μέγεθος
0	1	n
1	2	$n/2$
i	2^i	$n/2^i$
…	…	…



Σύνοψη των Αλγορίθμων Συγχώνευσης

Αλγόριθμος	Χρόνος	Σημειώσεις
Ταξινόμηση με επιλογή	$O(n^2)$	▪ αργός ▪ στη θέση ▪ για μικρά σύνολα δεδομένων (< 1K)
Ταξινόμηση με εισαγωγή	$O(n^2)$	▪ αργός ▪ στη θέση ▪ για μικρά σύνολα δεδομένων(< 1K)
Ταξινόμηση σωρού	$O(n \log n)$	▪ γρήγορος ▪ στη θέση ▪ για μεγάλο όγκο δεδομένων(1K — 1M)
συγχώνευση	$O(n \log n)$	▪ γρήγορος ▪ σειριακή προσπέλαση δεδομένων ▪ για τεράστιο όγκο δεδομένων (> 1M)