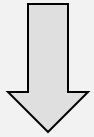
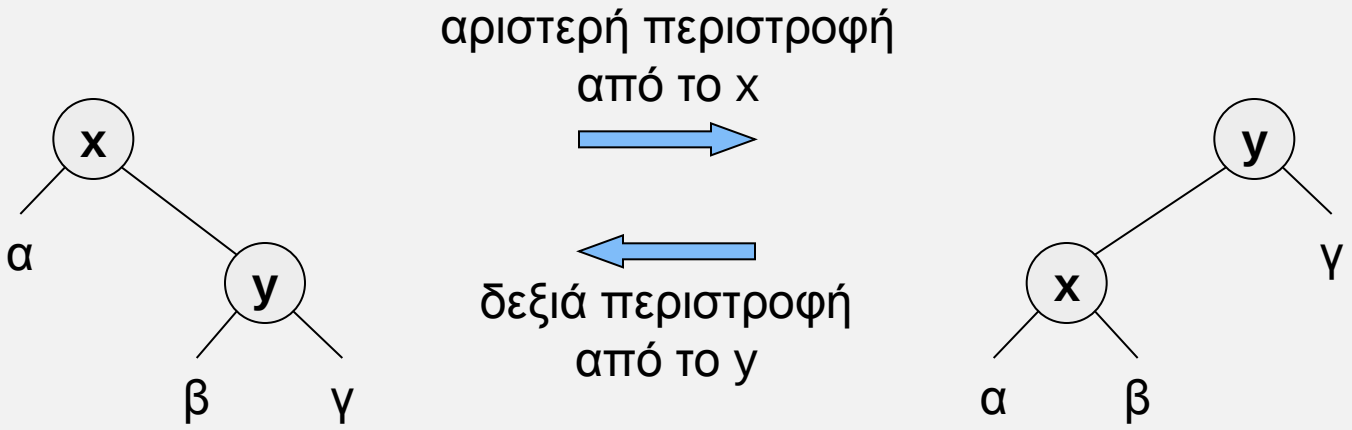
Ισορροπημένα Δένδρα

Μπορούμε να επιτύχουμε $O(\log N)$ χρόνο εκτέλεσης για κάθε λειτουργία;



Ισορροπημένο δένδρο : Διατηρεί ύψος $O(\log N)$ μετά από κάθε εισαγωγή ή διαγραφή

Περιστροφές




```
Node rotateLeft(Node x)
{
     Node y = x.right;
     x.right = y.left;
     y.left = x;
     return y;
}
```

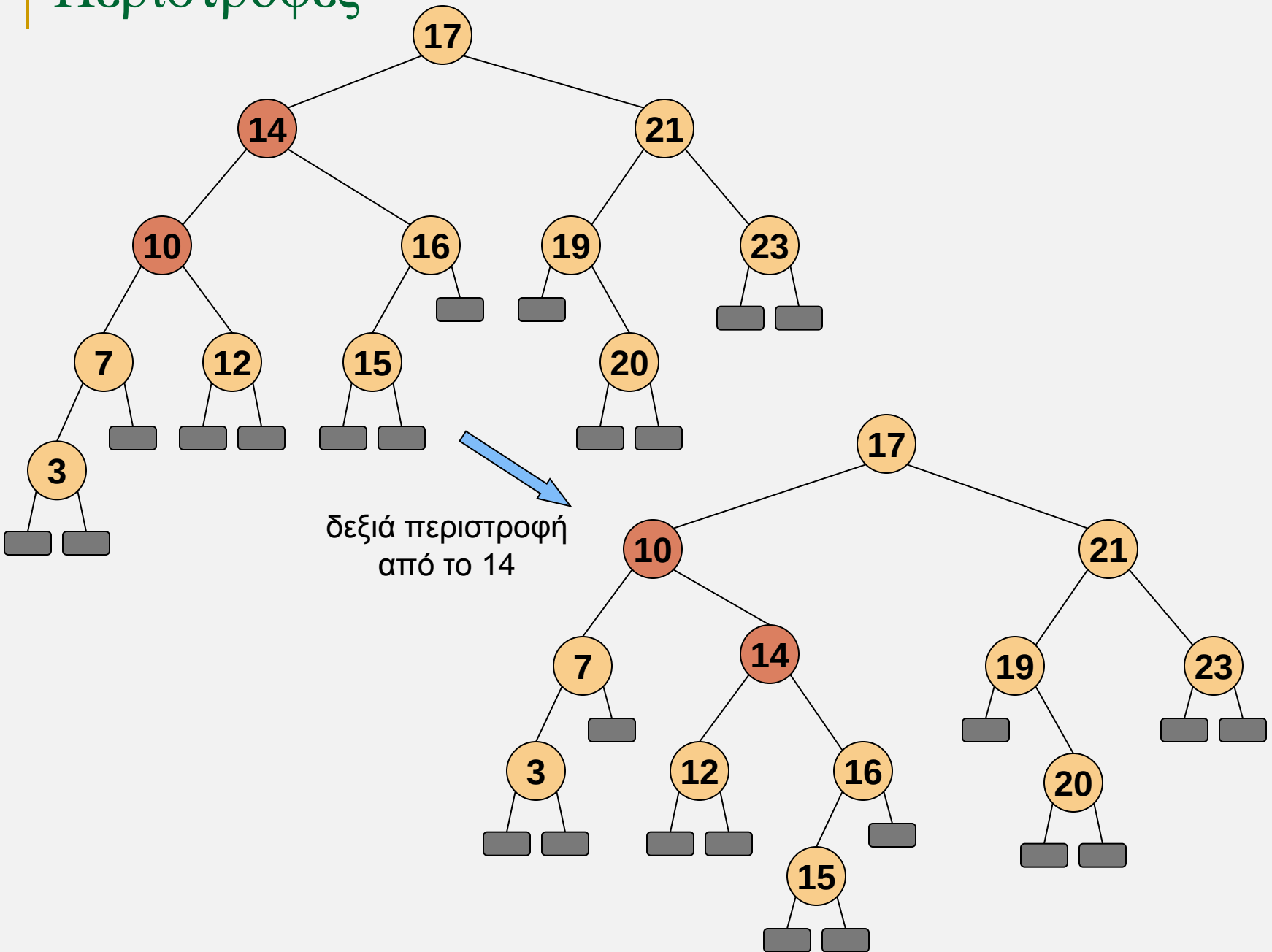
αριστερή περιστροφή

Η περιστροφή παίρνει χρόνο O(1)

```
Node rotateRight(Node y)
{
     Node x = y.left;
     y.left = x.right;
     x.right = y;
     return x;
}
```

δεξιά περιστροφή

Περιστροφές



Ισορροπημένα Δένδρα

Μερικοί τύποι ισορροπημένων δένδρων

- Τυχαιοποιημένα δένδρα
- Αρθρωτά δένδρα (splay trees) (*)
- Δένδρα AVL
- Κοκκινόμαυρα δένδρα
- (a,b) δένδρα

Όλα τα παραπάνω χρησιμοποιούν περιστροφές για να παραμείνουν ισορροπημένα

(*) Τα αρθρωτά δένδρα είναι ισορροπημένα κατά «αντισταθμιστική» έννοια

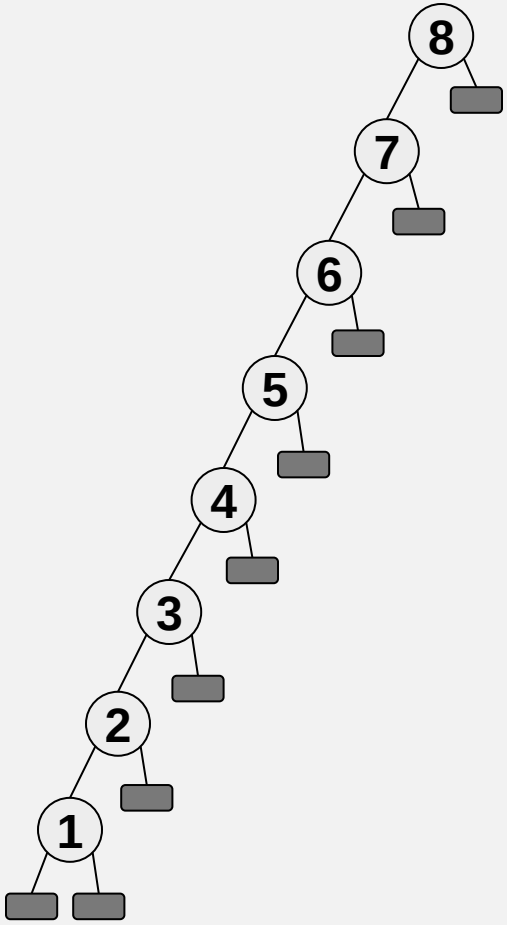
Αρθρωτά δένδρα (splay trees)

- Εκτελεί όλες τις πράξεις στη ρίζα του δένδρου.
- Χρησιμοποιεί ζεύγη περιστροφών καθώς μετακινεί ένα κλειδί στη ρίζα (λειτουργία **splay**) για να φέρει το δένδρο σε μεγαλύτερη ισορροπία.

DEMO :
http://aleph0.clarku.edu/~achou/cs102/examples/bst_animation/SplayTree-Example.html

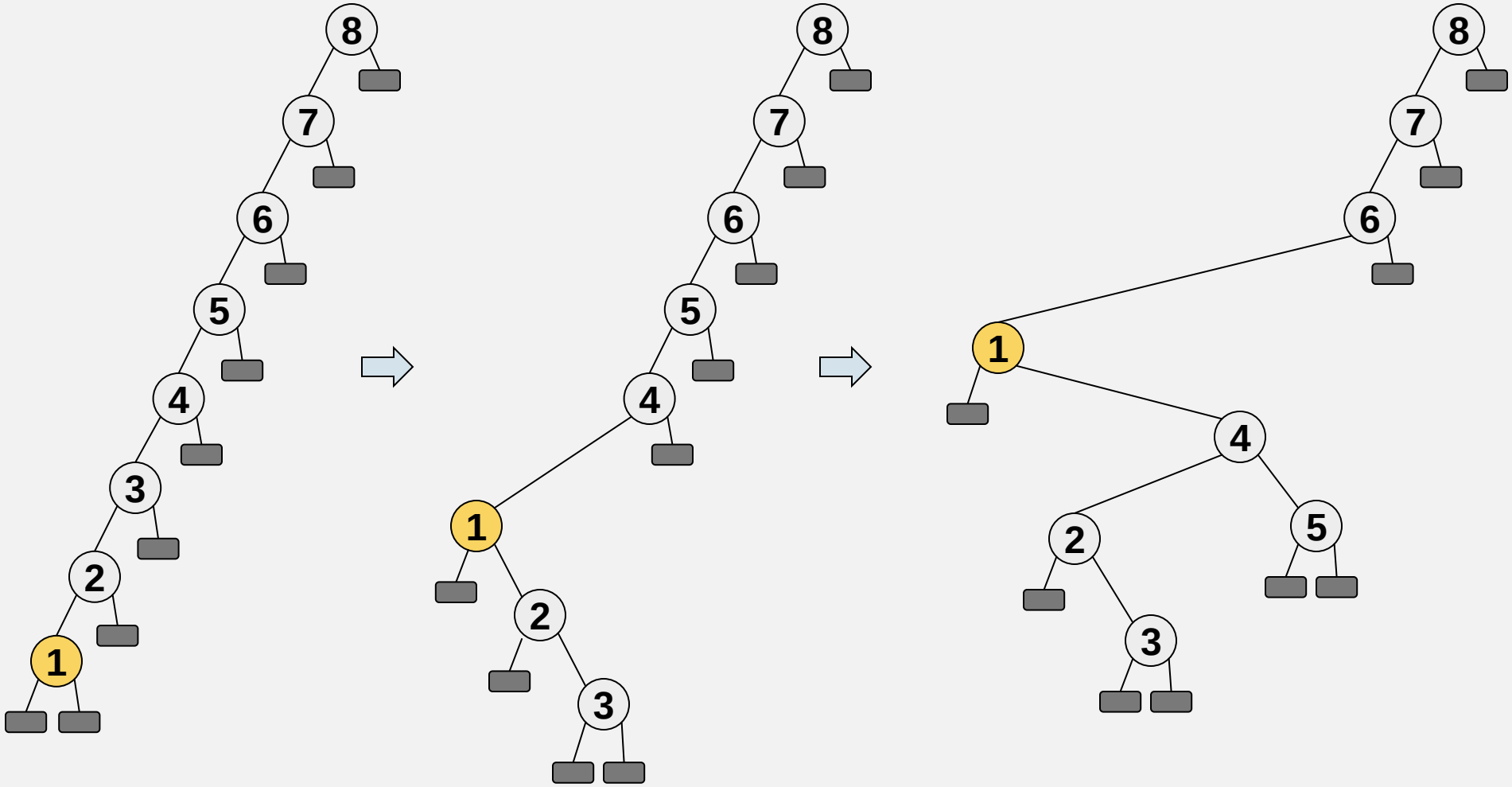
Αρθρωτά δένδρα (splay trees)

Εισαγωγή 1,2,3,4,5,6,7,8



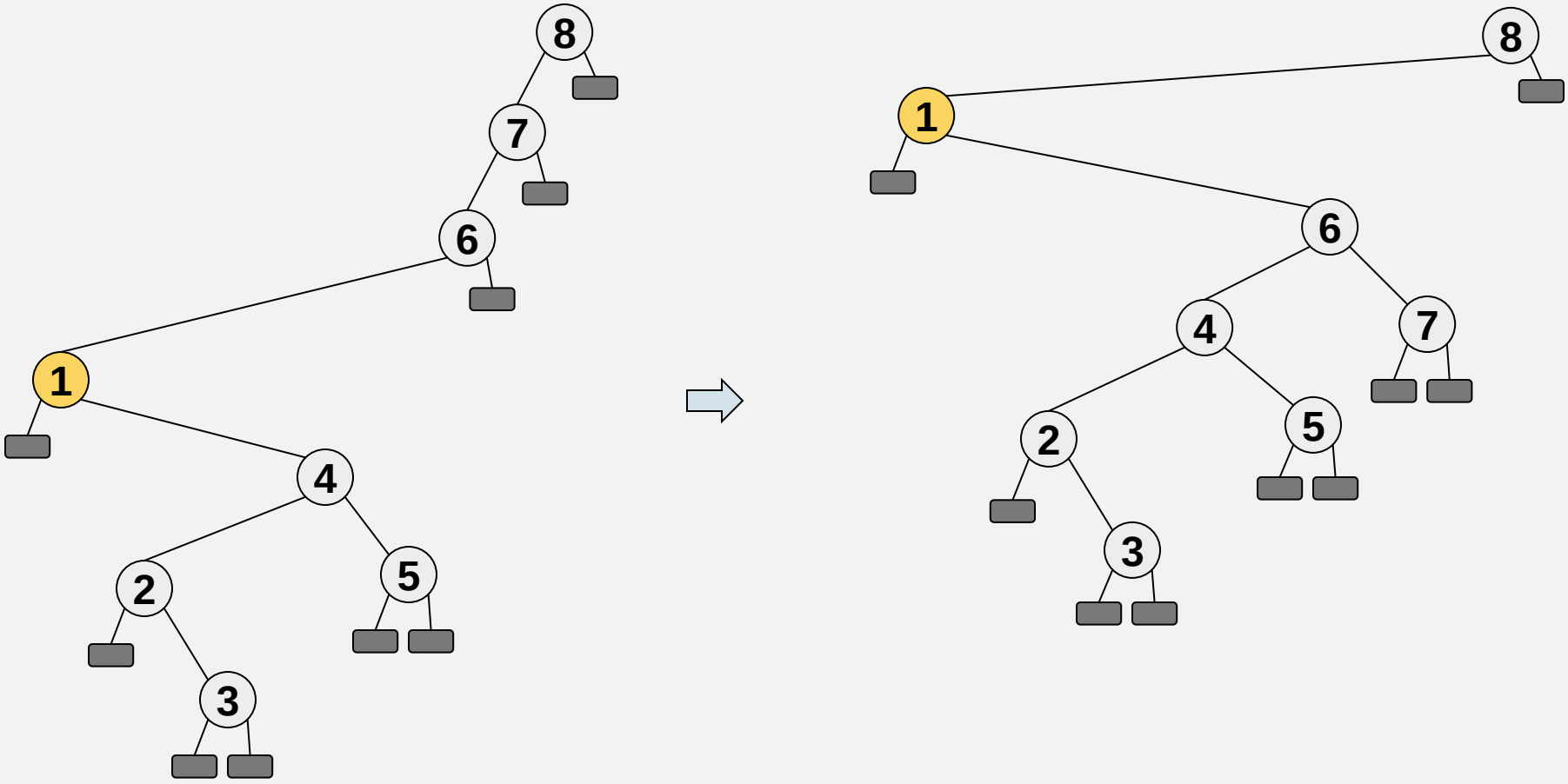
Αρθρωτά δένδρα (splay trees)

splay(1**)**



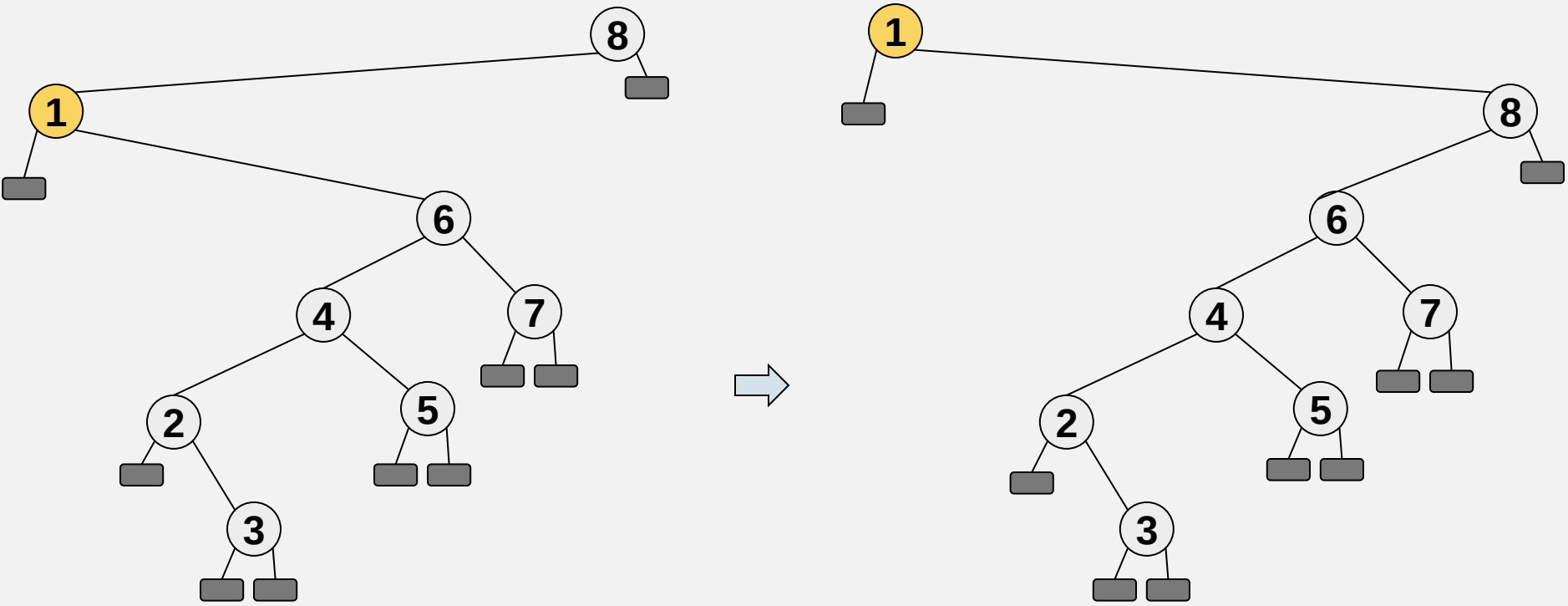
Αρθρωτά δένδρα (splay trees)

splay(1**)** (συνέχεια)



Αρθρωτά δένδρα (splay trees)

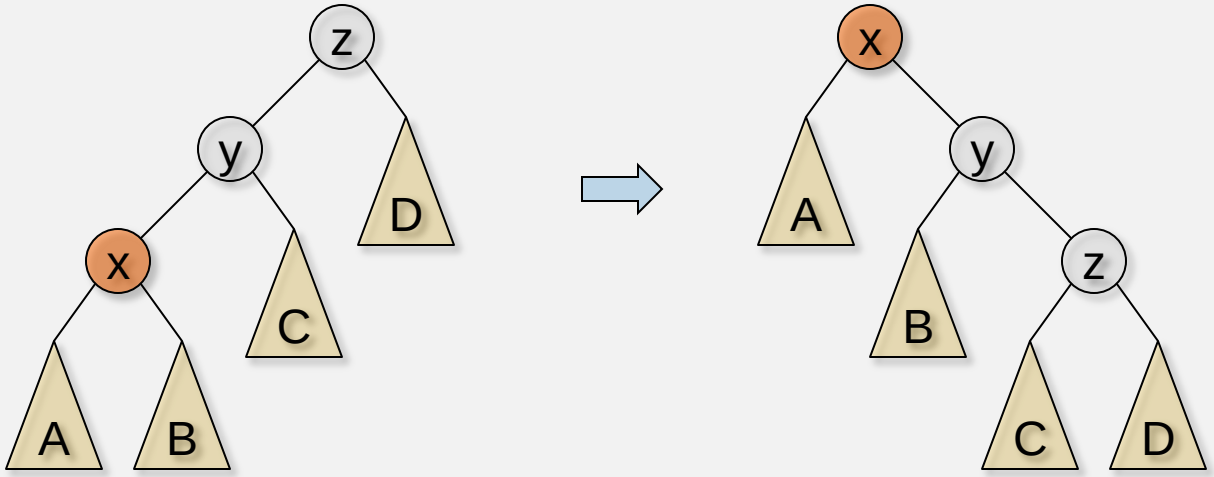
splay(1**)** (συνέχεια)



Αρθρωτά δένδρα (splay trees)

- Εκτελεί όλες τις πράξεις στη ρίζα του δένδρου.
- Χρησιμοποιεί ζεύγη περιστροφών καθώς μετακινεί ένα κλειδί στη ρίζα (λειτουργία **splay**) για να φέρει το δένδρο σε μεγαλύτερη ισορροπία.

splay(x) Περίπτωση zig-zig: από τον παππού του **x** ακολουθούμε δύο αριστερούς ή δύο δεξιούς συνδέσμους για να πάμε στο **x**

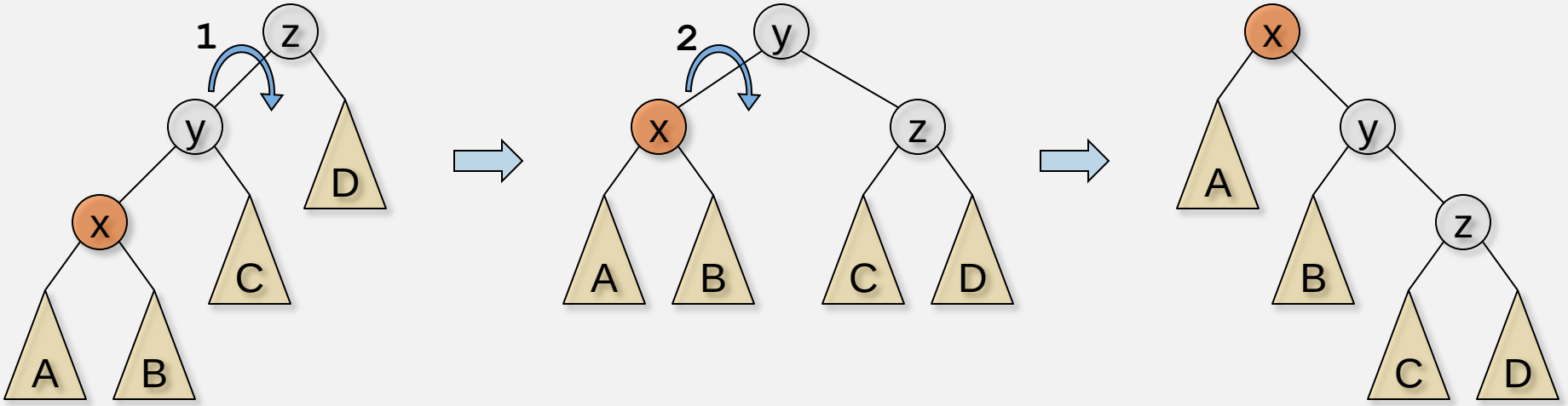


Αρθρωτά δένδρα (splay trees)

- Εκτελεί όλες τις πράξεις στη ρίζα του δένδρου.
- Χρησιμοποιεί ζεύγη περιστροφών καθώς μετακινεί ένα κλειδί στη ρίζα (λειτουργία **splay**) για να φέρει το δένδρο σε μεγαλύτερη ισορροπία.

splay(x) Περίπτωση zig-zig: από τον παππού του **x** ακολουθούμε δύο αριστερούς ή δύο δεξιούς συνδέσμους για να πάμε στο **x**.

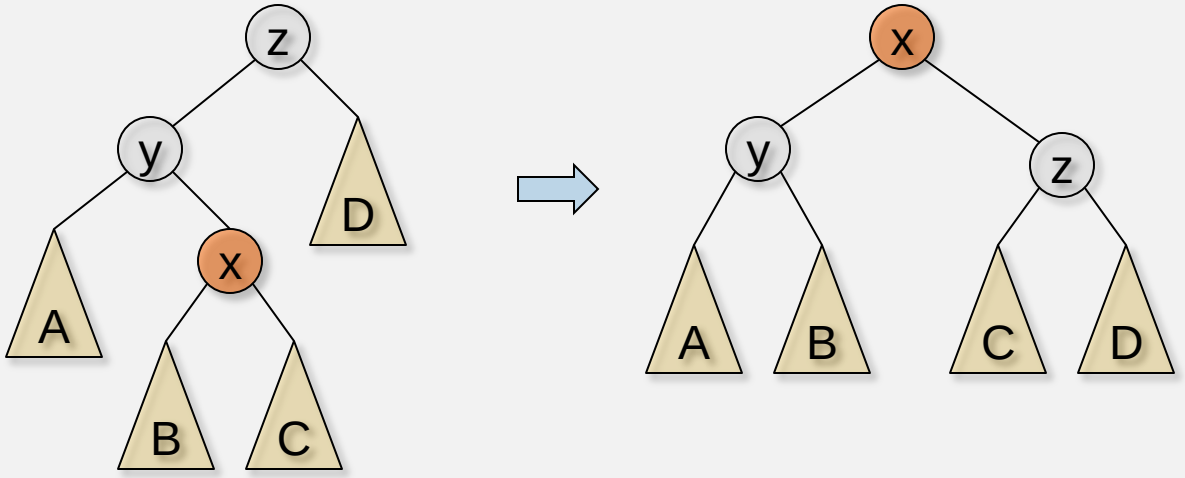
Προσοχή στη σειρά των περιστροφών!



Αρθρωτά δένδρα (splay trees)

- Εκτελεί όλες τις πράξεις στη ρίζα του δένδρου.
- Χρησιμοποιεί ζεύγη περιστροφών καθώς μετακινεί ένα κλειδί στη ρίζα (λειτουργία **splay**) για να φέρει το δένδρο σε μεγαλύτερη ισορροπία.

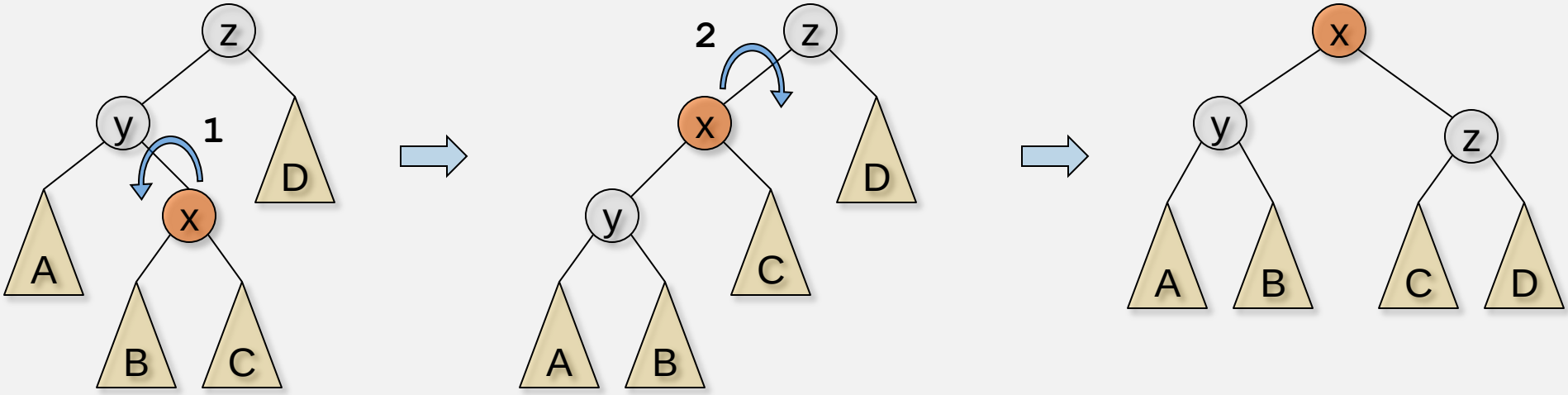
splay(x) Περίπτωση zig-zag: από τον παππού του **x** ακολουθούμε πρώτα έναν αριστερό και μετά ένα δεξιό σύνδεσμο ή πρώτα έναν δεξιό και μετά ένα αριστερό σύνδεσμο για να πάμε στο **x**



Αρθρωτά δένδρα (splay trees)

- Εκτελεί όλες τις πράξεις στη ρίζα του δένδρου.
- Χρησιμοποιεί ζεύγη περιστροφών καθώς μετακινεί ένα κλειδί στη ρίζα (λειτουργία **splay**) για να φέρει το δένδρο σε μεγαλύτερη ισορροπία.

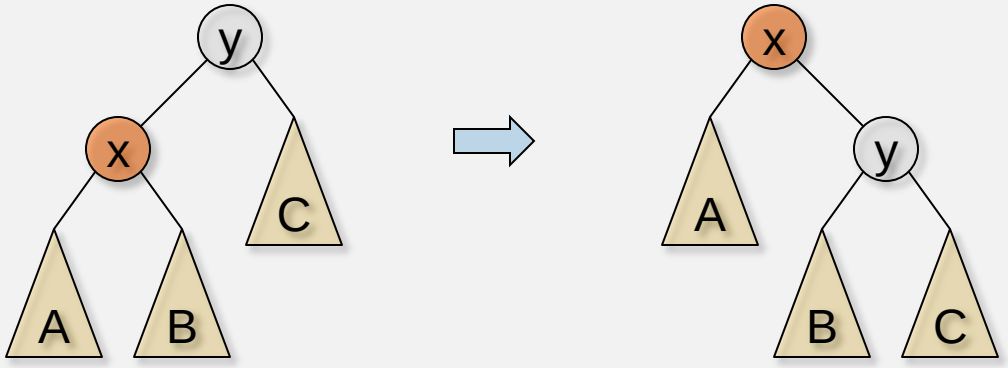
splay(x) Περίπτωση zig-zag: από τον παππού του **x** ακολουθούμε πρώτα έναν αριστερό και μετά ένα δεξιό σύνδεσμο ή πρώτα έναν δεξιό και μετά ένα αριστερό σύνδεσμο για να πάμε στο **x**



Αρθρωτά δένδρα (splay trees)

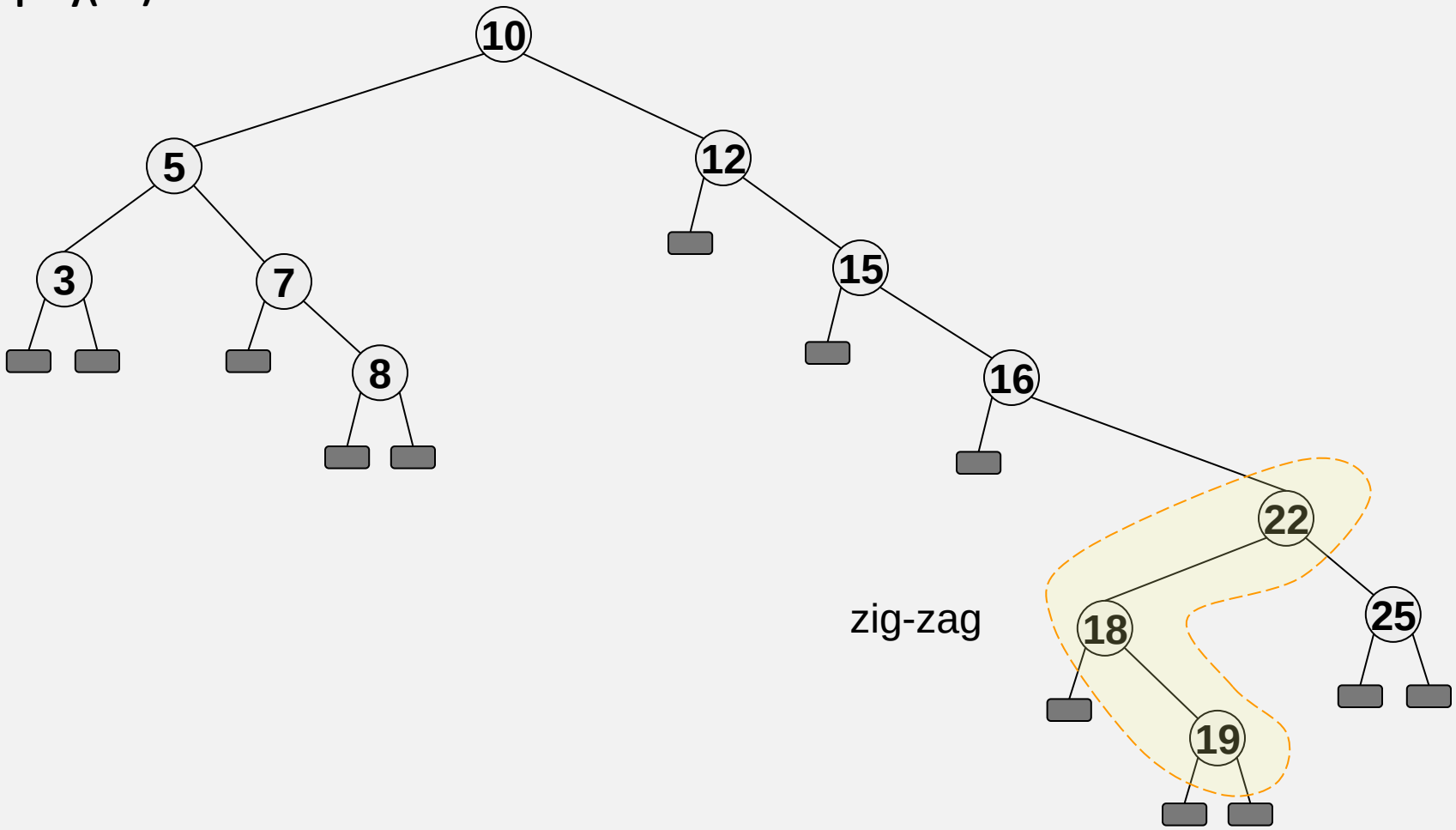
- Εκτελεί όλες τις πράξεις στη ρίζα του δένδρου.
- Χρησιμοποιεί ζεύγη περιστροφών καθώς μετακινεί ένα κλειδί στη ρίζα (λειτουργία **splay**) για να φέρει το δένδρο σε μεγαλύτερη ισορροπία.

splay(x) Αν ο πατέρας του **x** είναι η ρίζα τότε εκτελούμε μια απλή περιστροφή



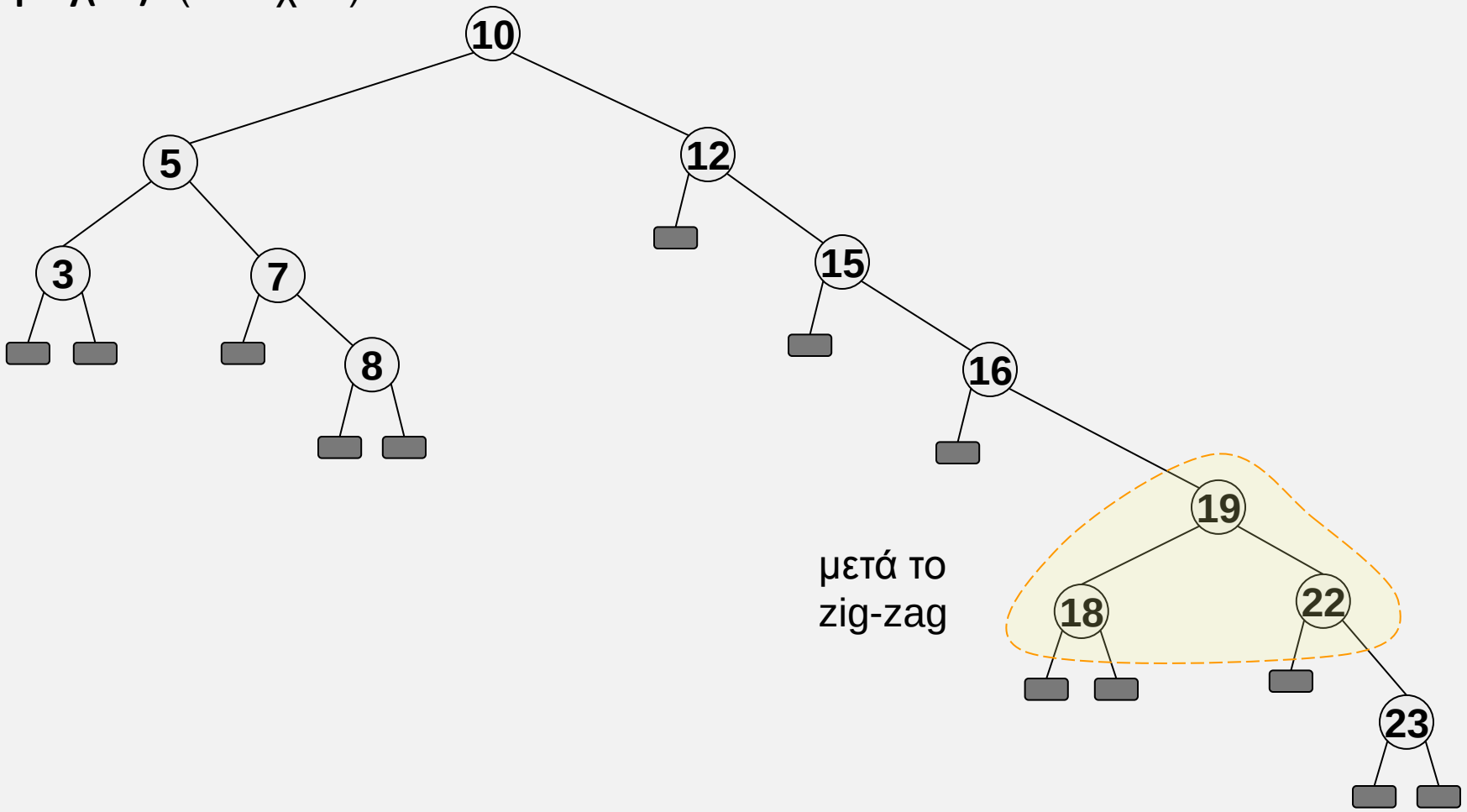
Αρθρωτά δένδρα (splay trees)

splay(19**)**



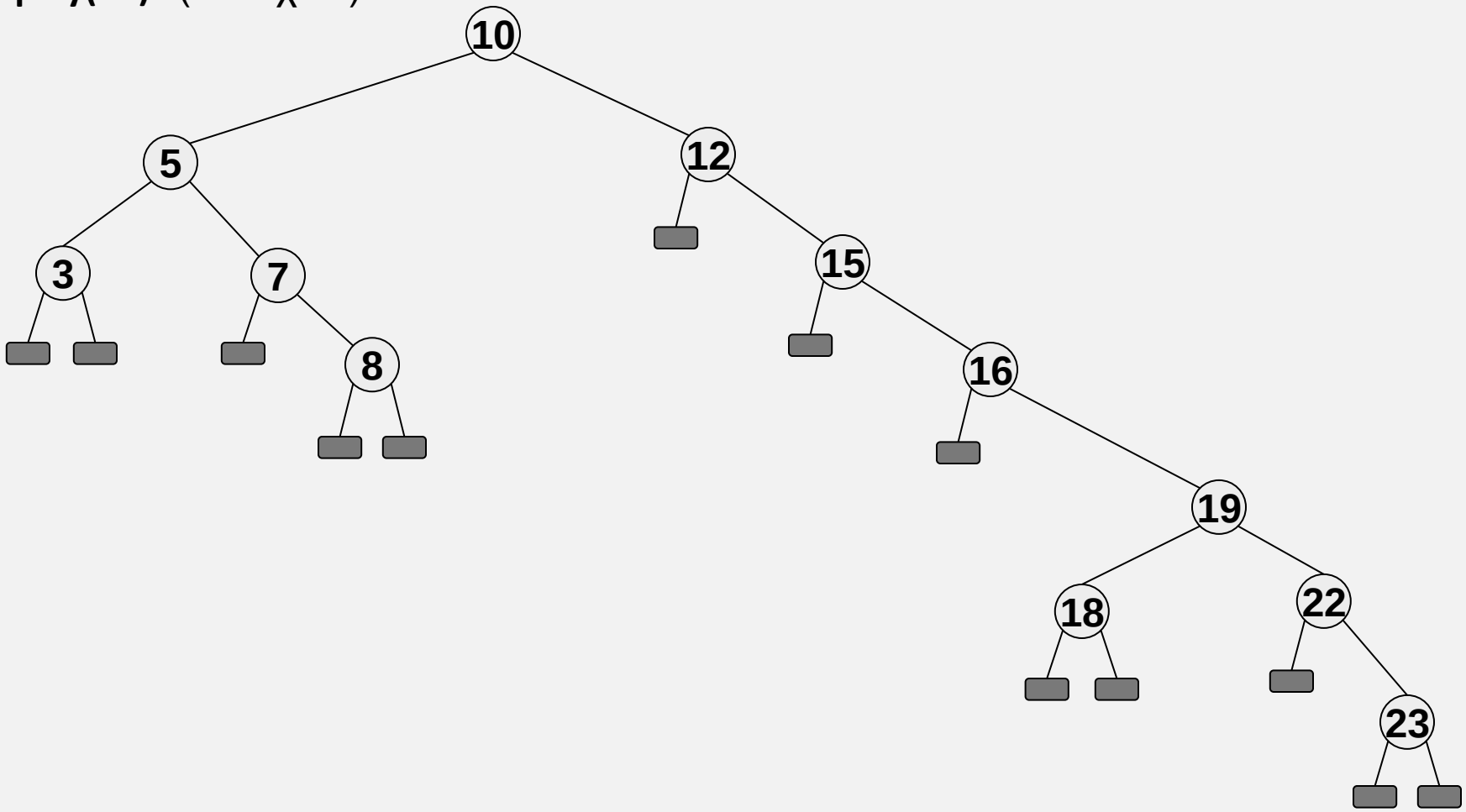
Αρθρωτά δένδρα (splay trees)

splay(19**)** (συνέχεια)



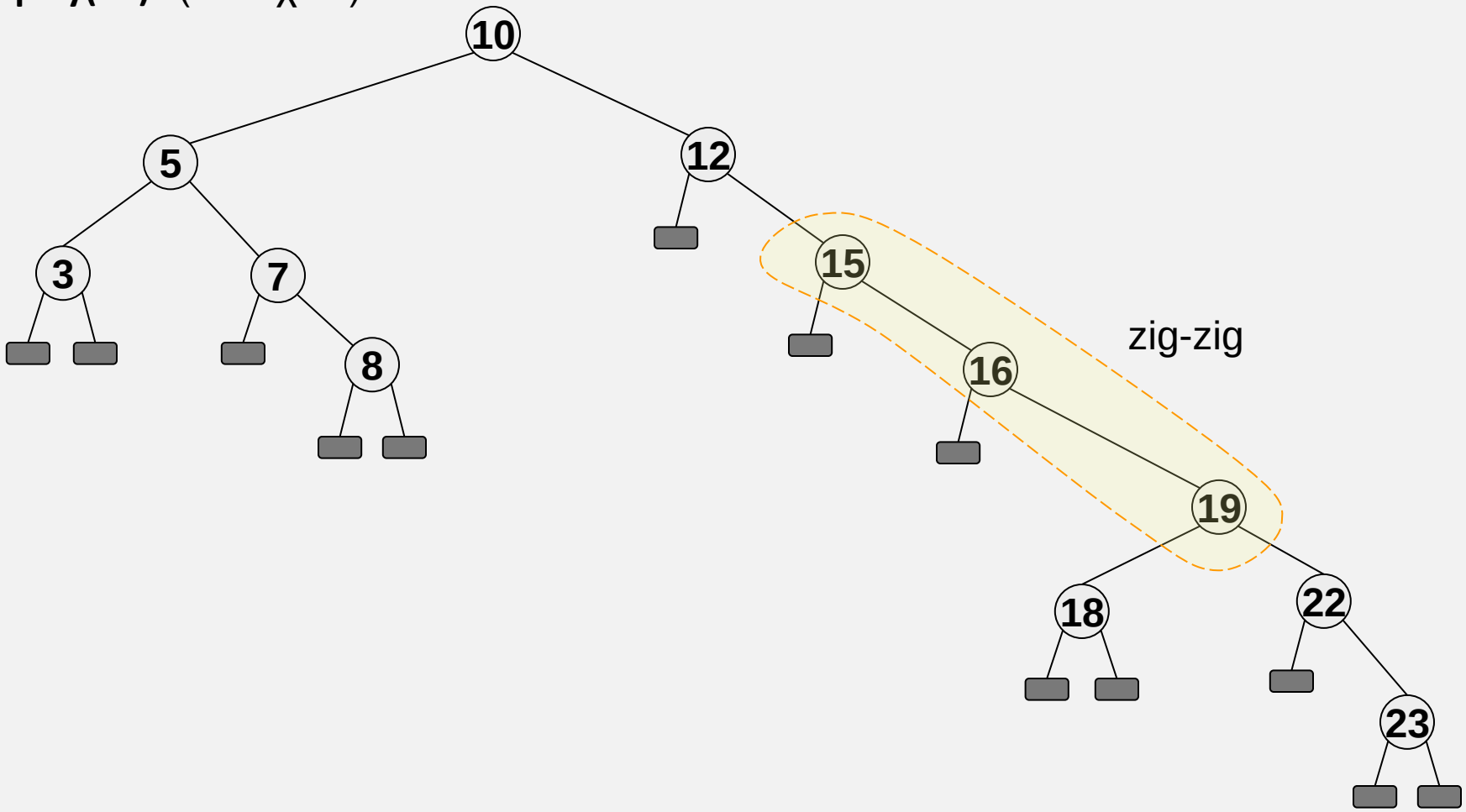
Αρθρωτά δένδρα (splay trees)

splay(19**)** (συνέχεια)



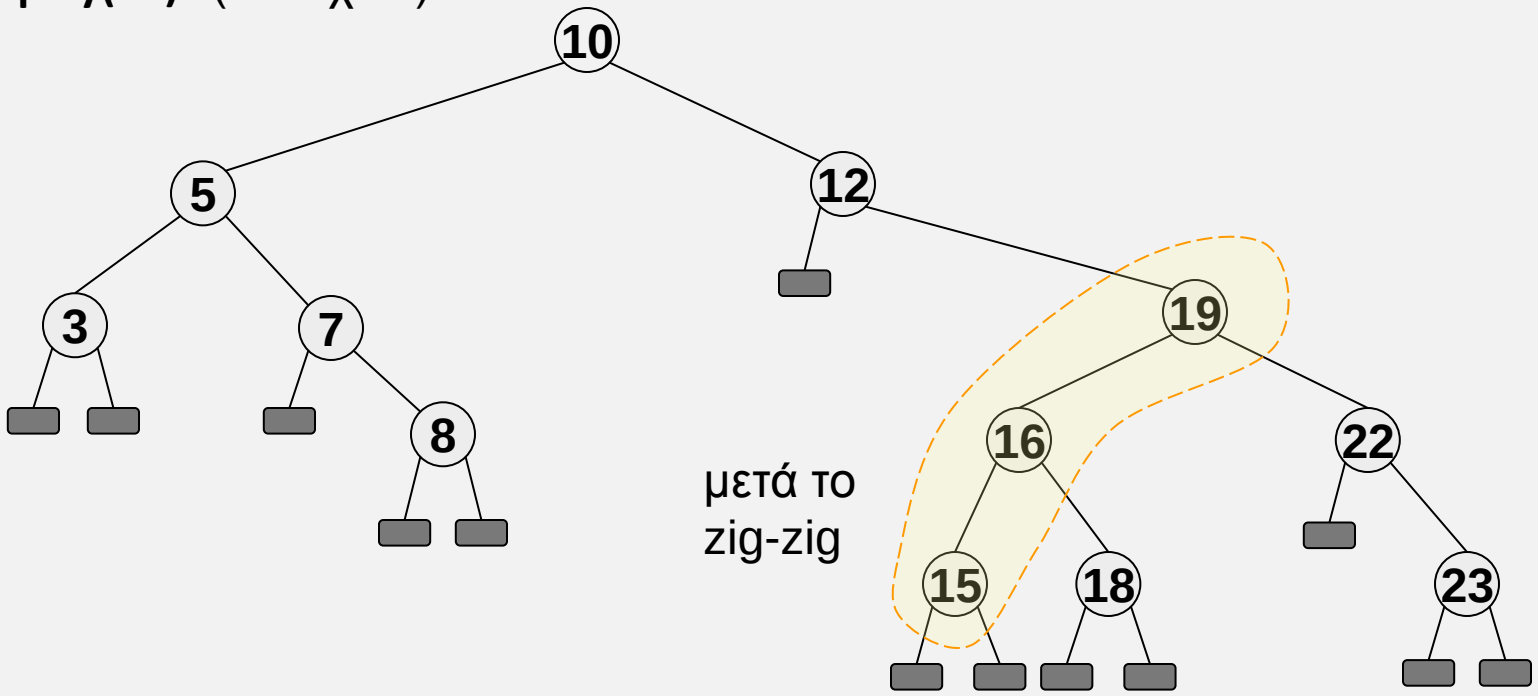
Αρθρωτά δένδρα (splay trees)

splay(19**)** (συνέχεια)



Αρθρωτά δένδρα (splay trees)

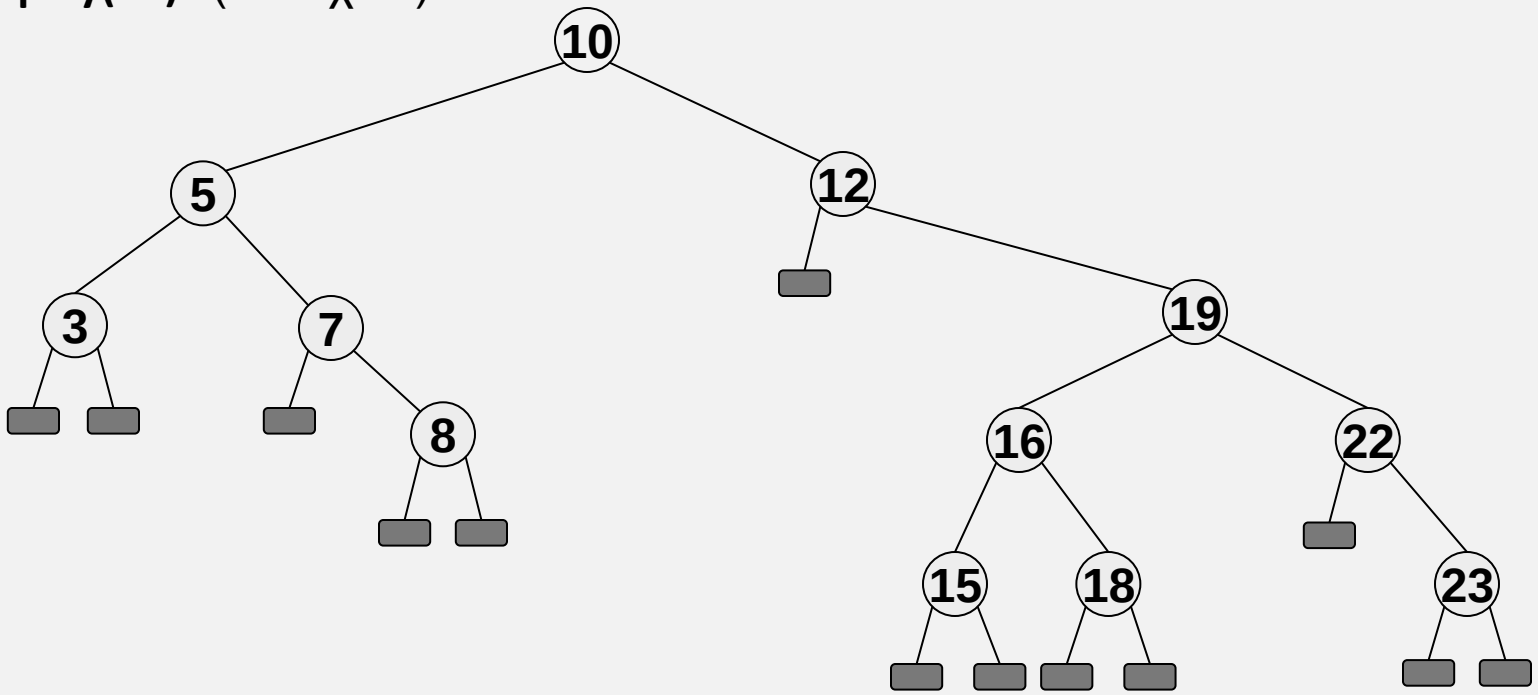
splay(19**)** (συνέχεια)



μετά το
zig-zig

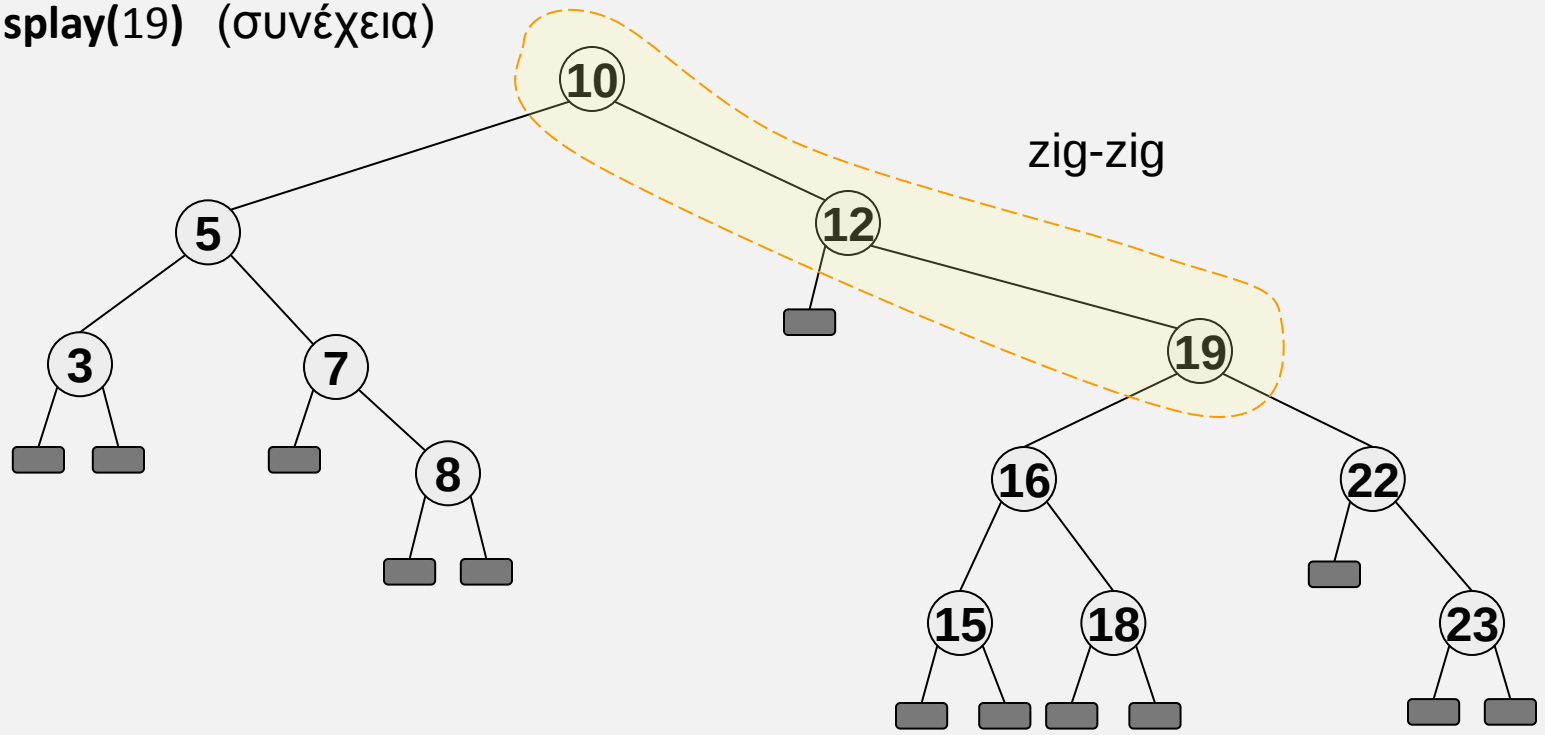
Αρθρωτά δένδρα (splay trees)

splay(19**)** (συνέχεια)



Αρθρωτά δένδρα (splay trees)

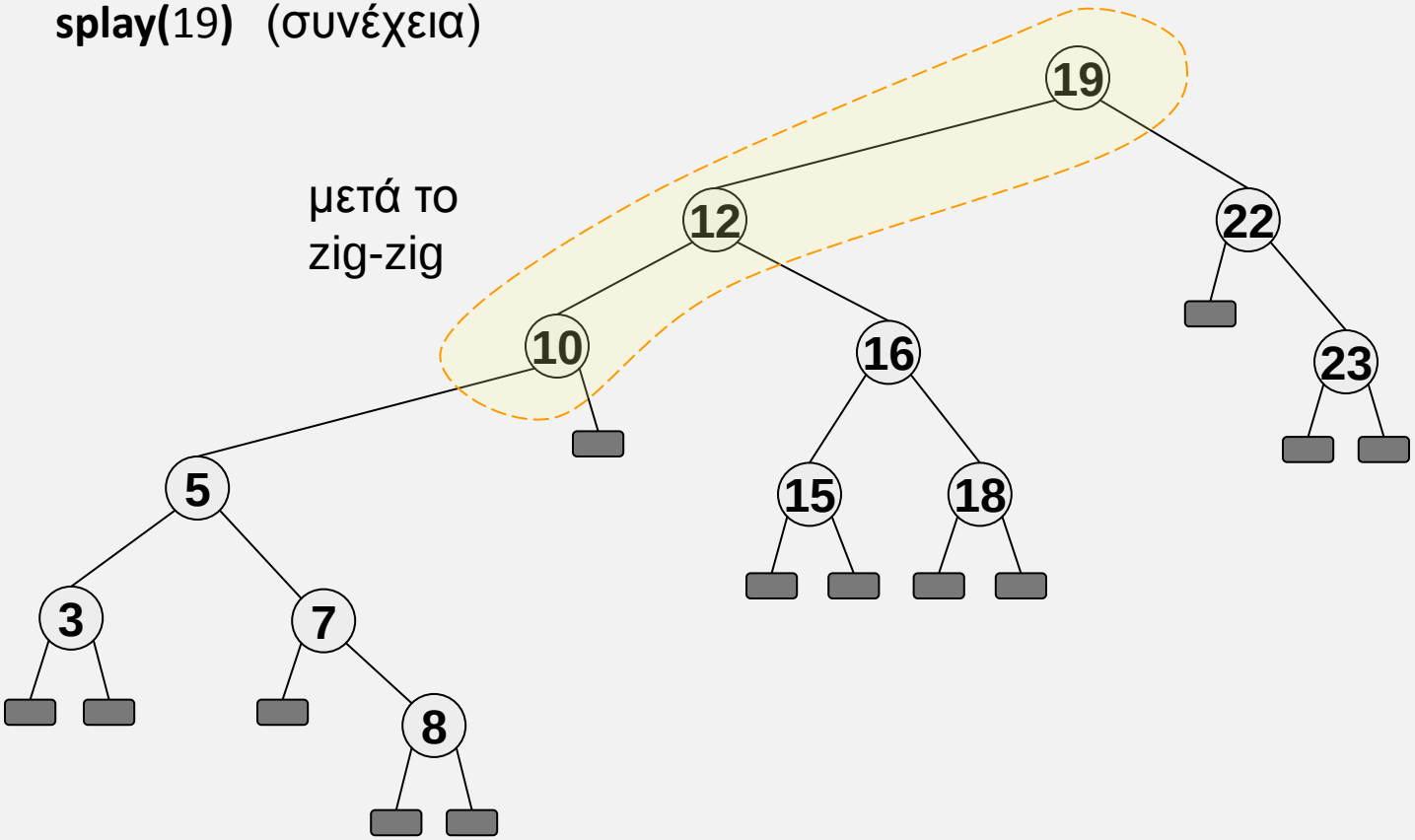
splay(19**)** (συνέχεια)



Αρθρωτά δένδρα (splay trees)

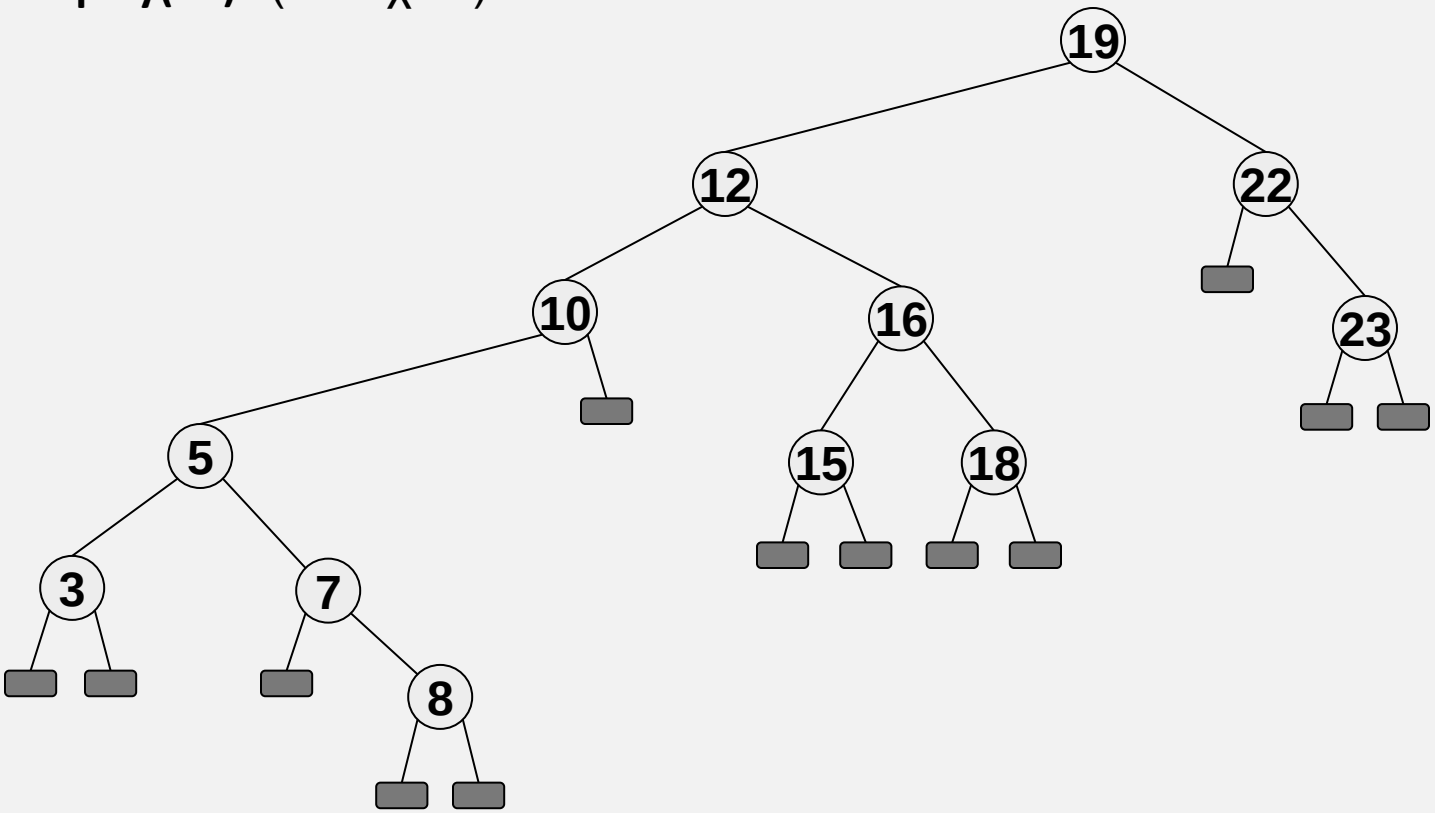
splay(19**)** (συνέχεια)

μετά το
zig-zig



Αρθρωτά δένδρα (splay trees)

splay(19**)** (συνέχεια)



Αρθρωτά δένδρα (splay trees)

Όλες οι λειτουργίες (αναζήτηση, εισαγωγή, διαγραφή) βασίζονται στη **splay**

Αναζήτηση στοιχείου με κλειδί k

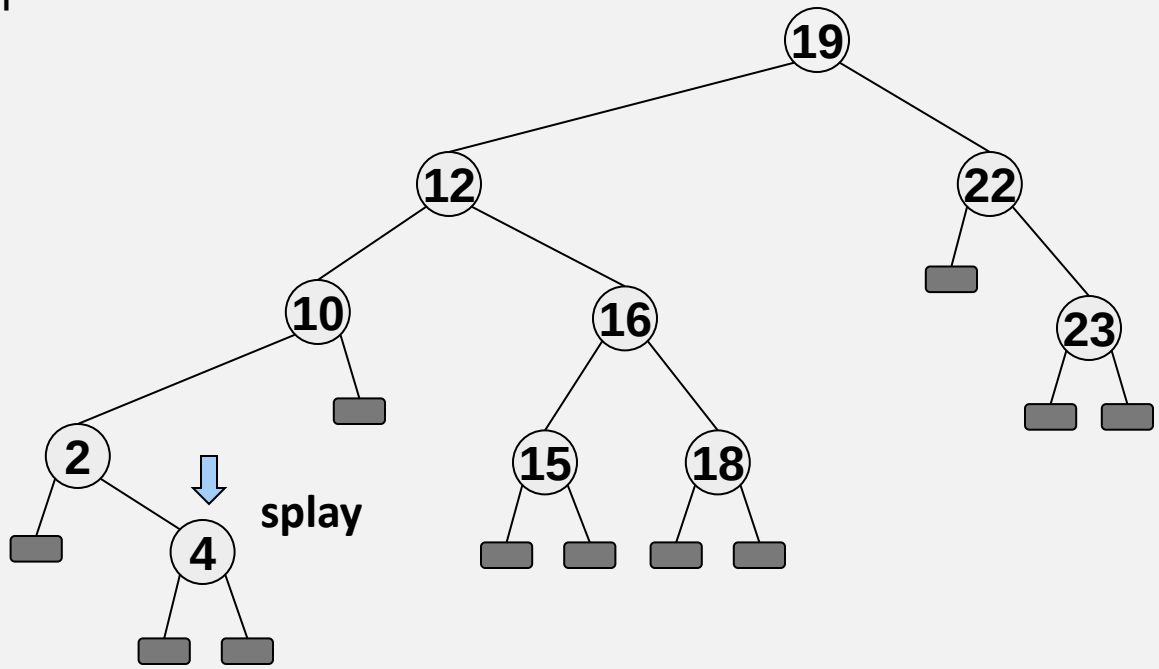
Βρίσκουμε τον τελευταίο μη κενό κόμβο **x** στο μονοπάτι αναζήτησης του k από τη ρίζα όπως στην αναζήτηση σε απλό δυαδικό δένδρο. (Αν το κλειδί είναι αποθηκευμένο στο δένδρο τότε ο **x** έχει κλειδί k .)

Εκτελούμε **splay(**x**)**

Επιστρέφουμε το στοιχείο της ρίζας

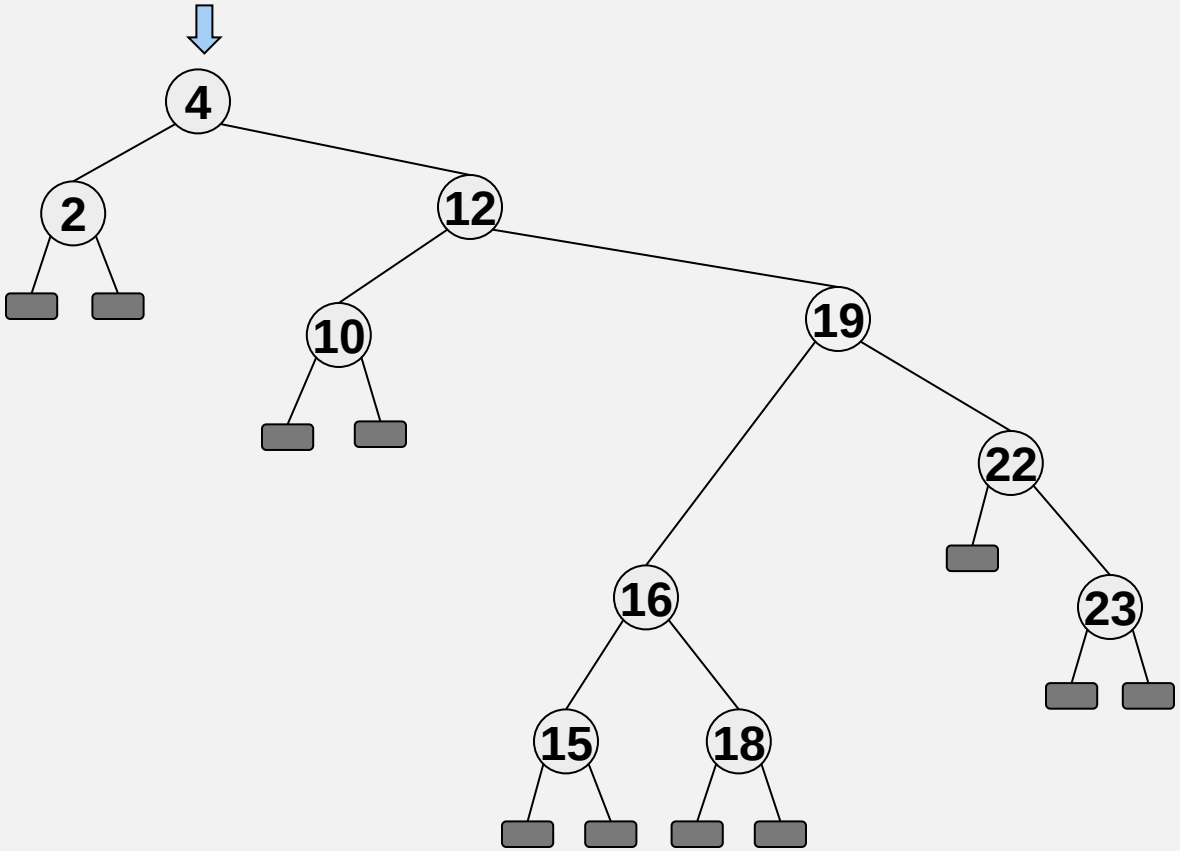
Αρθρωτά δένδρα (splay trees)

Αναζήτηση 4



Αρθρωτά δένδρα (splay trees)

Αναζήτηση 4



Αρθρωτά δένδρα (splay trees)

Όλες οι λειτουργίες (αναζήτηση, εισαγωγή, διαγραφή) βασίζονται στη **splay**

Εισαγωγή στοιχείου με κλειδί k

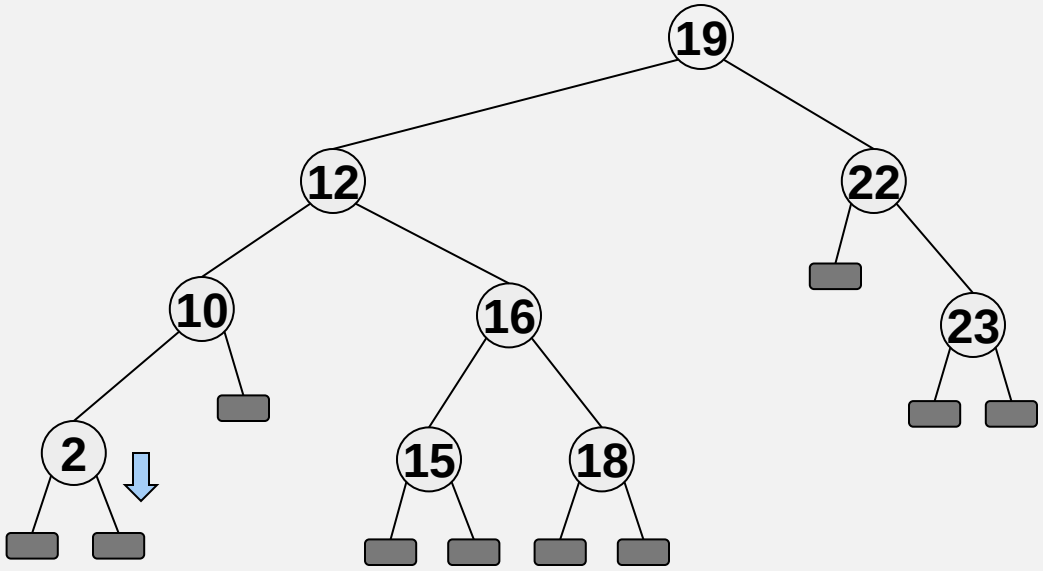
Εκτελούμε τα βήματα της εισαγωγής όπως στο απλό δυαδικό δένδρο.

Έστω **x** ο νέος κόμβος που αποθηκεύει το στοιχείο με κλειδί k

Εκτελούμε **splay(x)**

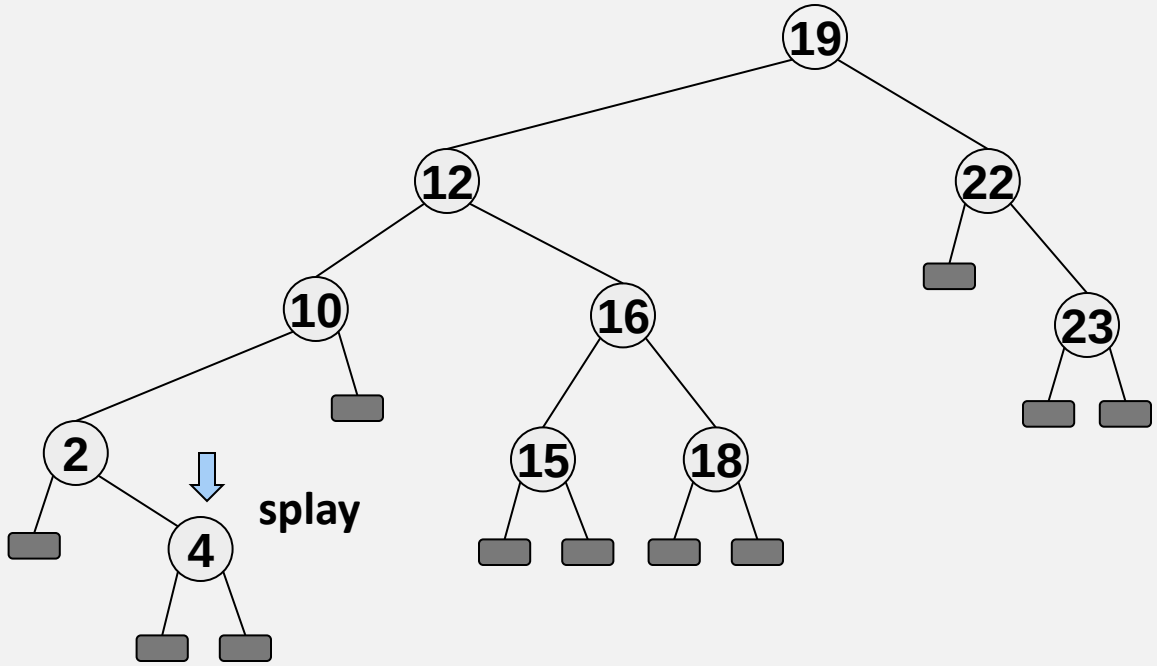
Αρθρωτά δένδρα (splay trees)

Εισαγωγή 4



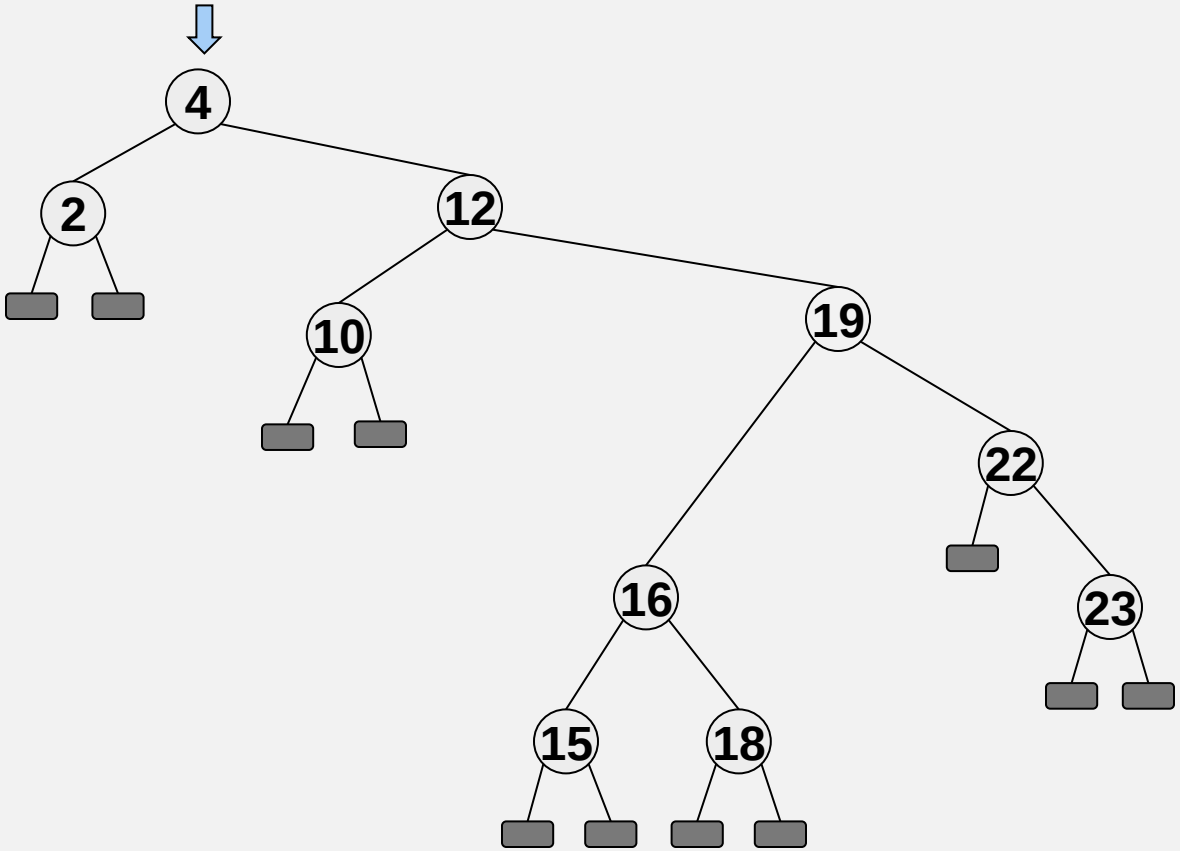
Αρθρωτά δένδρα (splay trees)

Εισαγωγή 4



Αρθρωτά δένδρα (splay trees)

Εισαγωγή 4



Αρθρωτά δένδρα (splay trees)

Όλες οι λειτουργίες (αναζήτηση, εισαγωγή, διαγραφή) βασίζονται στη **splay**

Διαγραφή κόμβου **x**

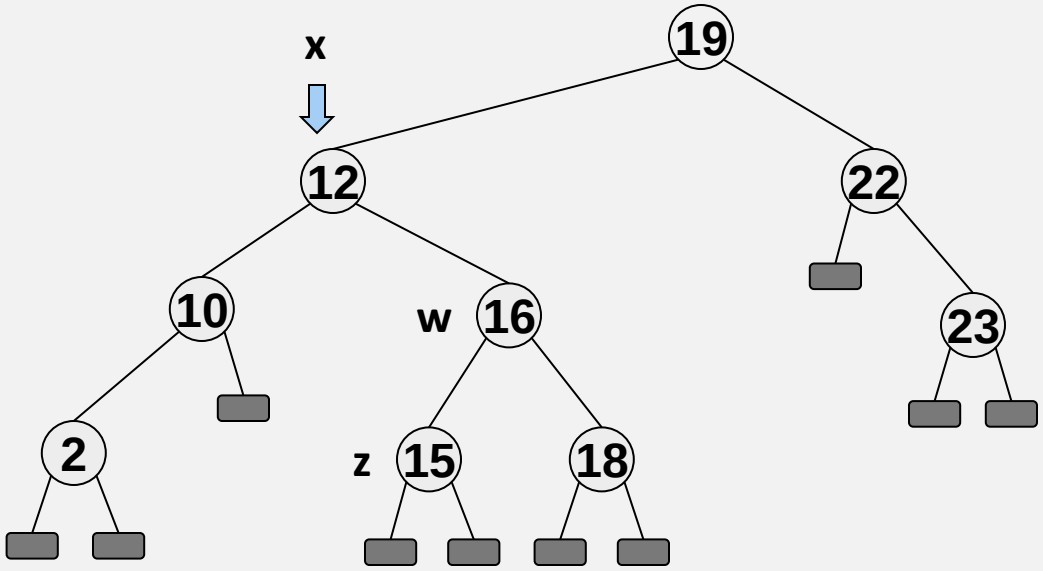
Εκτελούμε τα βήματα της εισαγωγής όπως στο απλό δυαδικό δένδρο.

Έστω **z** ο κόμβος που διαγράφεται.
(Αν ο **x** έχει κενό παιδί τότε **z=x**, διαφορετικά ο **z** είναι ο διάδοχος του **x**.)

Έστω **w** ο γονέας του **z**. Εκτελούμε **splay(w)**

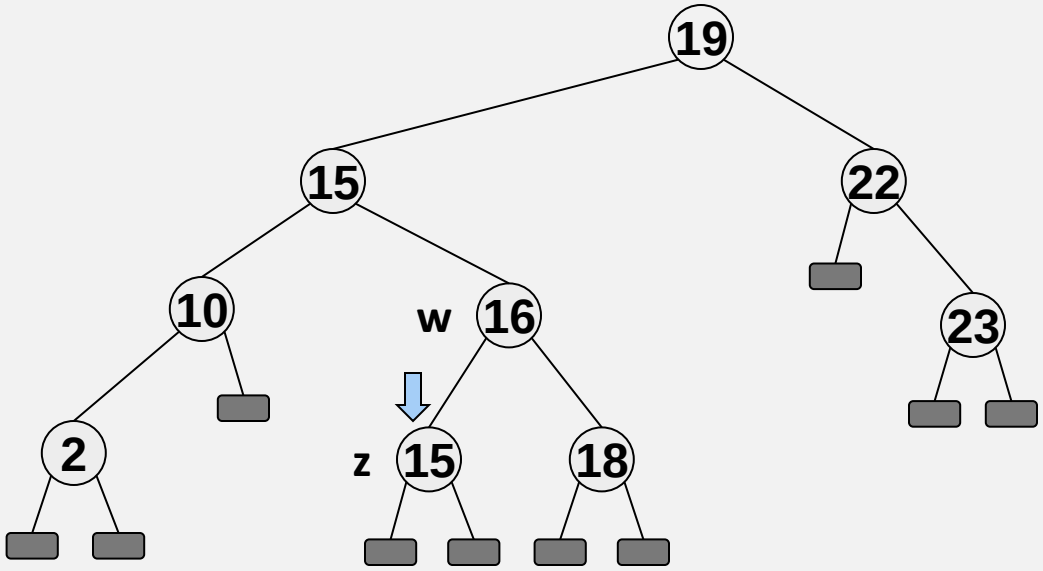
Αρθρωτά δένδρα (splay trees)

Διαγραφή **x**



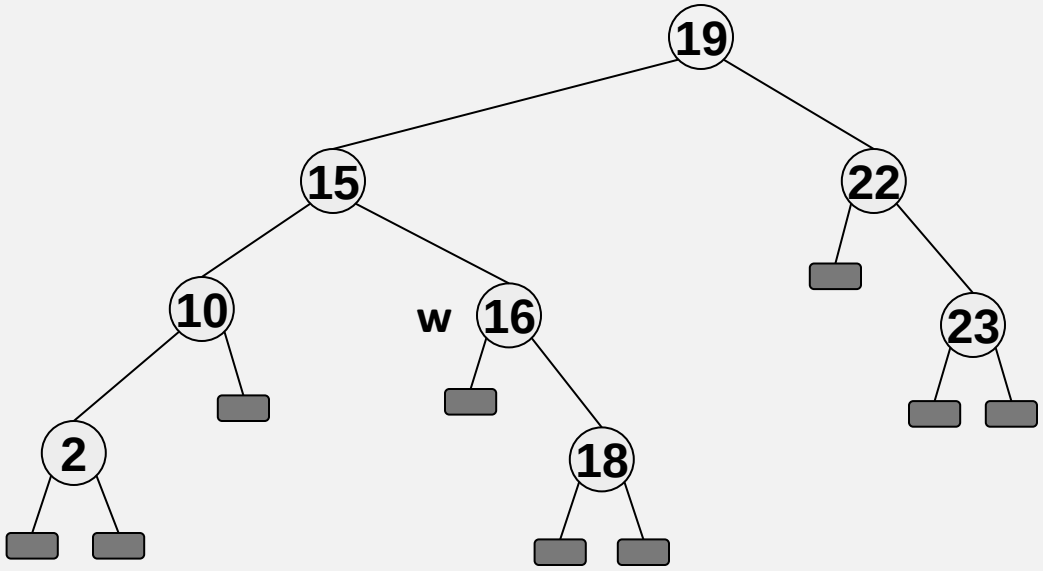
Αρθρωτά δένδρα (splay trees)

Διαγραφή **x**



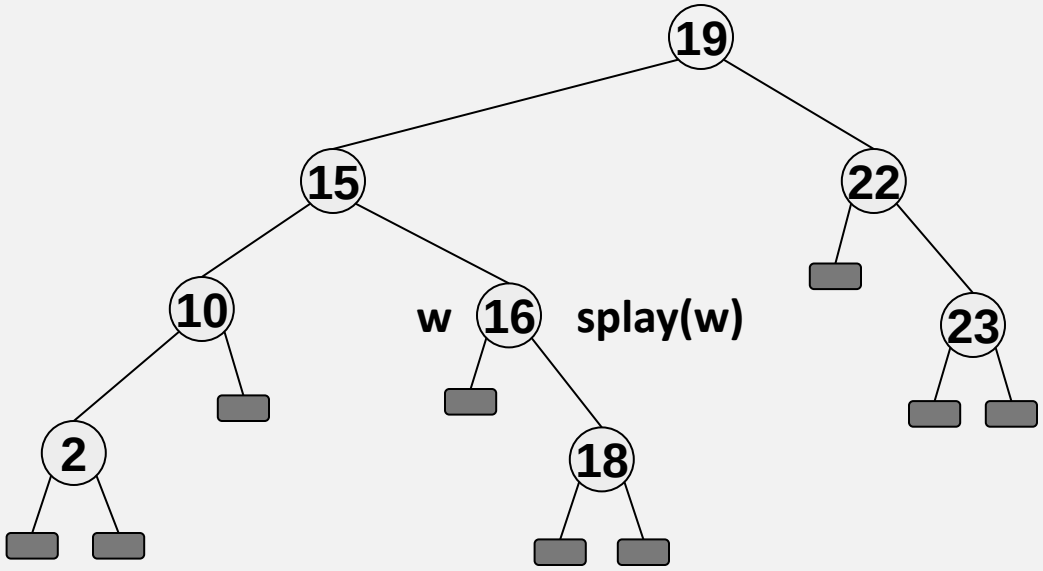
Αρθρωτά δένδρα (splay trees)

Διαγραφή **x**



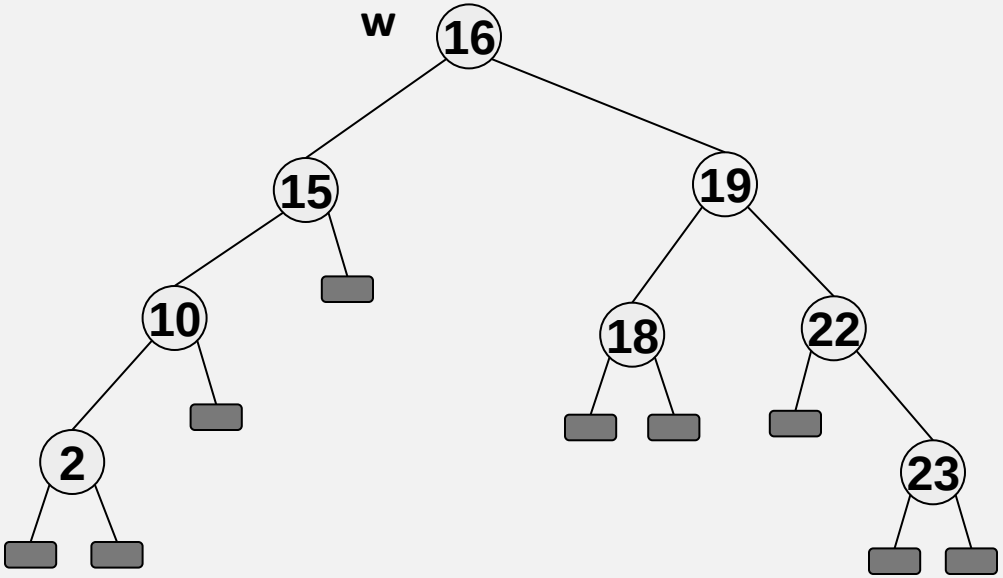
Αρθρωτά δένδρα (splay trees)

Διαγραφή **x**



Αρθρωτά δένδρα (splay trees)

Διαγραφή **x**



Αρθρωτά δένδρα (splay trees)

Ιδιότητες

Έστω ότι εκτελούμε m λειτουργίες σε αρχικά κενό αρθρωτό δένδρο, όπου κάθε λειτουργία είναι αναζήτηση, εισαγωγή ή διαγραφή. Έστω n ο συνολικός αριθμός εισαγωγών. Τότε ισχύουν οι παρακάτω ιδιότητες :

Ο συνολικός χρόνος εκτέλεσης όλων των m λειτουργιών είναι $O(m \log n)$

$\Rightarrow$ $O(\log n)$ αντισταθμιστικός χρόνος ανά λειτουργία

Έστω $f(i)$ ο αριθμός των λειτουργιών που γίνονται στο κλειδί i (δηλαδή ο αριθμός προσπελάσεων του κλειδιού i). Τότε ο συνολικός χρόνος εκτέλεσης όλων των m λειτουργιών είναι

$$O\left(m + \sum_{i=1}^{n} f(i) \log\left(m/f(i)\right)\right)$$

Αντισταθμιστική ανάλυση

Θεωρήστε έναν αλγόριθμο Α που χρησιμοποιεί μια δομή δεδομένων Δ :

Κατά τη διάρκεια εκτέλεσης του Α η Δ πραγματοποιεί μία ακολουθία από πράξεις.

Έστω $f(n)$ το κόστος μίας πράξης στη χειρότερη περίπτωση για n αντικείμενα.

Τότε ο συνολικός χρόνος για m πράξεις είναι $O(mf(n))$

Αντισταθμιστική ανάλυση

Θεωρήστε έναν αλγόριθμο Α που χρησιμοποιεί μια δομή δεδομένων Δ :

Κατά τη διάρκεια εκτέλεσης του Α η Δ πραγματοποιεί μία ακολουθία από πράξεις.

Έστω $f(n)$ το κόστος μίας πράξης στη χειρότερη περίπτωση για n αντικείμενα.

Τότε ο συνολικός χρόνος για m πράξεις είναι $O(mf(n))$

Ωστόσο σε κάποιες περιπτώσεις:

→ Κάθε πράξη μπορεί να έχει διαφορετικό κόστος ανάλογα με την στιγμή που εκτελείται.

→ Το κόστος μίας πράξης στη **χειρότερη περίπτωση** μπορεί να είναι πολύ μεγάλο.

→ Αλλά το **μέσο κόστος** ανά πράξη σε **οποιαδήποτε** ακολουθία πράξεων μπορεί να είναι αρκετά μικρότερο.

Αντισταθμιστική ανάλυση

Θεωρήστε έναν αλγόριθμο Α που χρησιμοποιεί μια δομή δεδομένων Δ :

Κατά τη διάρκεια εκτέλεσης του Α η Δ πραγματοποιεί μία ακολουθία από πράξεις.

Έστω $f(n)$ το κόστος μίας πράξης στη χειρότερη περίπτωση για n αντικείμενα.

Τότε ο συνολικός χρόνος για m πράξεις είναι $O(mf(n))$

Ωστόσο σε κάποιες περιπτώσεις:

→ Κάθε πράξη μπορεί να έχει διαφορετικό κόστος ανάλογα με την στιγμή που εκτελείται.

→ Το κόστος μίας πράξης στη **χειρότερη περίπτωση** μπορεί να είναι πολύ μεγάλο.

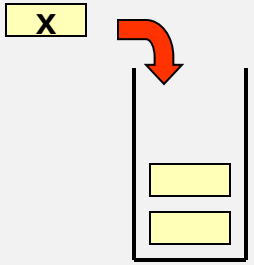
→ Αλλά το **μέσο κόστος** ανά πράξη σε **οποιαδήποτε** ακολουθία πράξεων μπορεί να είναι αρκετά μικρότερο.

Αντισταθμιστική ανάλυση: λαμβάνουμε το μέσο κόστος εκτέλεσης μίας πράξης όταν εκτελούμε μία ακολουθία πράξεων **χειρότερης περίπτωσης**

Αντισταθμιστική ανάλυση

Διαχείριση στοίβας

push(S,x) : τοποθετεί το x στην κορυφή της στοίβας S

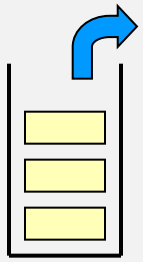


Αντισταθμιστική ανάλυση

Διαχείριση στοίβας

push(S,x) : τοποθετεί το x στην κορυφή της στοίβας S

pop(S) : αφαιρεί από τη στοίβα S το στοιχείο στην κορυφή της



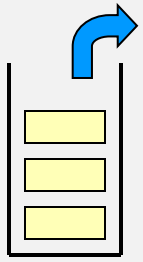
Αντισταθμιστική ανάλυση

Διαχείριση στοίβας

push(S,x) : τοποθετεί το x στην κορυφή της στοίβας S

pop(S) : αφαιρεί από τη στοίβα S το στοιχείο στην κορυφή της

multiPop(S,k) : αφαιρεί από τη στοίβα S τα k πρώτα στοιχεία στην κορυφή της



k=2

Αντισταθμιστική ανάλυση

Διαχείριση στοίβας

	χρόνος
push(S,x) : τοποθετεί το x στην κορυφή της στοίβας S	$O(1)$
pop(S) : αφαιρεί από τη στοίβα S το στοιχείο στην κορυφή της	$O(1)$
multiPop(S,k) : αφαιρεί από τη στοίβα S τα k πρώτα στοιχεία στην κορυφή της	$O(k)$

Αντισταθμιστική ανάλυση

Διαχείριση στοίβας

	χρόνος
push(S,x) : τοποθετεί το x στην κορυφή της στοίβας S	$O(1)$
pop(S) : αφαιρεί από τη στοίβα S το στοιχείο στην κορυφή της	$O(1)$
multiPop(S,k) : αφαιρεί από τη στοίβα S τα k πρώτα στοιχεία στην κορυφή της	$O(k)$

Ποιος είναι ο συνολικός χρόνος για μία ακολουθία από N πράξεις; $O(kN)$;

Αντισταθμιστική ανάλυση

Διαχείριση στοίβας

	χρόνος
push(S,x) : τοποθετεί το x στην κορυφή της στοίβας S	$O(1)$
pop(S) : αφαιρεί από τη στοίβα S το στοιχείο στην κορυφή της	$O(1)$
multiPop(S,k) : αφαιρεί από τη στοίβα S τα k πρώτα στοιχεία στην κορυφή της	$O(k)$

Ποιος είναι ο συνολικός χρόνος για μία ακολουθία από N πράξεις; $O(kN)$;

Πιο προσεκτική ανάλυση : Συνολικός χρόνος $O(N) \Rightarrow O(1)$ ανά πράξη

Αντισταθμιστική ανάλυση

Επαύξηση δυαδικού μετρητή

Έστω ένας μετρητής C με k bits : μια πράξη επαύξησης θέτει

$$C \leftarrow C + 1 \pmod{2^k}$$

Κόστος επαύξησης = αριθμός των bits που αλλάζουν

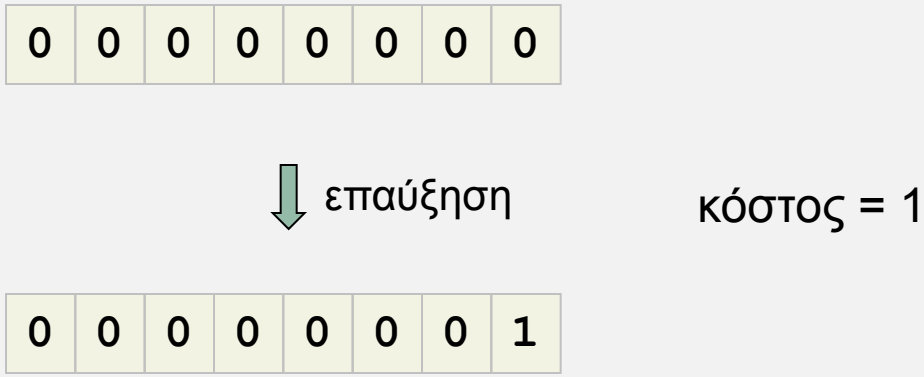
Αντισταθμιστική ανάλυση

Επαύξηση δυαδικού μετρητή

Έστω ένας μετρητής C με k bits : μια πράξη επαύξησης θέτει

$$C \leftarrow C + 1 \pmod{2^k}$$

Κόστος επαύξησης = αριθμός των bits που αλλάζουν



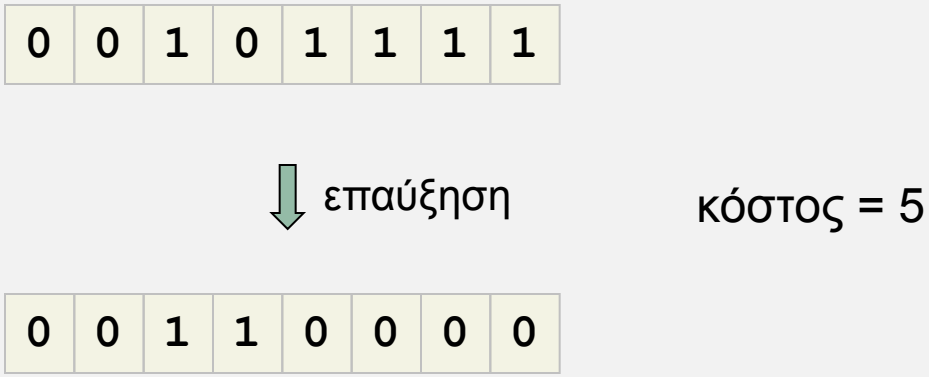
Αντισταθμιστική ανάλυση

Επαύξηση δυαδικού μετρητή

Έστω ένας μετρητής C με k bits : μια πράξη επαύξησης θέτει

$$C \leftarrow C + 1 \pmod{2^k}$$

Κόστος επαύξησης = αριθμός των bits που αλλάζουν



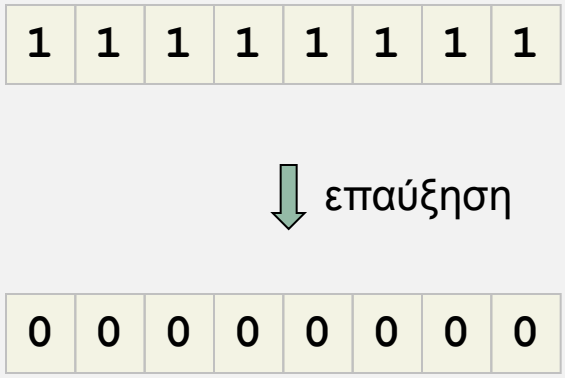
Αντισταθμιστική ανάλυση

Επαύξηση δυαδικού μετρητή

Έστω ένας μετρητής C με k bits : μια πράξη επαύξησης θέτει

$$C \leftarrow C + 1 \pmod{2^k}$$

Κόστος επαύξησης = αριθμός των bits που αλλάζουν



κόστος = 8

χειρότερη περίπτωση!

Αντισταθμιστική ανάλυση

Επαύξηση δυαδικού μετρητή

Έστω ένας μετρητής C με k bits : μια πράξη επαύξησης θέτει

$$C \leftarrow C + 1 \pmod{2^k}$$

Κόστος επαύξησης = αριθμός των bits που αλλάζουν

= $O(k)$ στη χειρότερη περίπτωση

$\Rightarrow$ Συνολικό κόστος για N επαυξήσεις = $O(kN)$

Αντισταθμιστική ανάλυση

Επαύξηση δυαδικού μετρητή

Έστω ένας μετρητής C με k bits : μια πράξη επαύξησης θέτει

$$C \leftarrow C + 1 \pmod{2^k}$$

Κόστος επαύξησης = αριθμός των bits που αλλάζουν

= $O(k)$ στη χειρότερη περίπτωση

$\Rightarrow$ Συνολικό κόστος για N επαυξήσεις = $O(kN)$

Βελτιωμένη ανάλυση : συνολικό κόστος = $O(N)$

Δηλαδή αντισταθμιστικό κόστος ανά πράξη = $O(1)$
(κατά μέσο όρο σταθερό κόστος ανά επαύξηση)

Αντισταθμιστική ανάλυση

Επαύξηση δυαδικού μετρητή

Έστω ένας μετρητής C με k bits : μια πράξη επαύξησης θέτει

$$C \leftarrow C + 1 \pmod{2^k}$$

Κόστος επαύξησης = αριθμός των bits που αλλάζουν

= $O(k)$ στη χειρότερη περίπτωση

$\Rightarrow$ Συνολικό κόστος για N επαυξήσεις = $O(kN)$

Βελτιωμένη ανάλυση : συνολικό κόστος = $O(N)$ *

Δηλαδή αντισταθμιστικό κόστος ανά πράξη = $O(1)$
(κατά μέσο όρο σταθερό κόστος ανά επαύξηση)

* Ξεκινώντας με μηδενισμένο μετρητή

Αντισταθμιστική ανάλυση

Επαύξηση δυαδικού μετρητή

Έστω ένας μετρητής C με k bits : μια πράξη επαύξησης θέτει

$$C \leftarrow C + 1 \pmod{2^k}$$

0	0	0	0
0	0	0	1
0	0	1	0
0	0	1	1
0	1	0	0
0	1	0	1
0	1	1	0
0	1	1	1

1	0	0	0
1	0	0	1
1	0	1	0
1	0	1	1
1	1	0	0
1	1	0	1
1	1	1	0
1	1	1	1

Αντισταθμιστική ανάλυση

Επαύξηση δυαδικού μετρητή

Έστω ένας μετρητής C με k bits : μια πράξη επαύξησης θέτει

$$C \leftarrow C + 1 \pmod{2^k}$$

0	0	0	0
0	0	0	1
0	0	1	0
0	0	1	1
0	1	0	0
0	1	0	1
0	1	1	0
0	1	1	1

1	0	0	0
1	0	0	1
1	0	1	0
1	0	1	1
1	1	0	0
1	1	0	1
1	1	1	0
1	1	1	1

1ο ψηφίο από το τέλος:

αλλάζει με κάθε επαύξηση

Αντισταθμιστική ανάλυση

Επαύξηση δυαδικού μετρητή

Έστω ένας μετρητής C με k bits : μια πράξη επαύξησης θέτει

$$C \leftarrow C + 1 \pmod{2^k}$$

0	0	0	0
0	0	0	1
0	0	1	0
0	0	1	1
0	1	0	0
0	1	0	1
0	1	1	0
0	1	1	1

1	0	0	0
1	0	0	1
1	0	1	0
1	0	1	1
1	1	0	0
1	1	0	1
1	1	1	0
1	1	1	1

2ο ψηφίο από το τέλος:

αλλάζει με κάθε δεύτερη επαύξηση

Αντισταθμιστική ανάλυση

Επαύξηση δυαδικού μετρητή

Έστω ένας μετρητής C με k bits : μια πράξη επαύξησης θέτει

$$C \leftarrow C + 1 \pmod{2^k}$$

0	0	0	0
0	0	0	1
0	0	1	0
0	0	1	1
0	1	0	0
0	1	0	1
0	1	1	0
0	1	1	1

1	0	0	0
1	0	0	1
1	0	1	0
1	0	1	1
1	1	0	0
1	1	0	1
1	1	1	0
1	1	1	1

3ο ψηφίο από το τέλος:

αλλάζει με κάθε τέταρτη επαύξηση

Αντισταθμιστική ανάλυση

Επαύξηση δυαδικού μετρητή

Έστω ένας μετρητής C με k bits : μια πράξη επαύξησης θέτει

$$C \leftarrow C + 1 \pmod{2^k}$$

0	0	0	0
0	0	0	1
0	0	1	0
0	0	1	1
0	1	0	0
0	1	0	1
0	1	1	0
0	1	1	1

1	0	0	0
1	0	0	1
1	0	1	0
1	0	1	1
1	1	0	0
1	1	0	1
1	1	1	0
1	1	1	1

4ο ψηφίο από το τέλος:

αλλάζει με κάθε όγδοη επαύξηση

Αντισταθμιστική ανάλυση

Επαύξηση δυαδικού μετρητή

Έστω ένας μετρητής C με k bits : μια πράξη επαύξησης θέτει

$$C \leftarrow C + 1 \pmod{2^k}$$

Γενικά το i-οστό ψηφίο από το τέλος αλλάζει μετά από 2^{i-1} επαυξήσεις.

$\Rightarrow$ Σε ακολουθία N πράξεων το i-οστό ψηφίο από το τέλος αλλάζει συνολικά $\left\lfloor \frac{N}{2^{i-1}} \right\rfloor$ φορές

$\Rightarrow$ Σύνολο αλλαγών για όλα τα ψηφία = $\sum_{i=1}^{k} \left\lfloor \frac{N}{2^{i-1}} \right\rfloor$

Αντισταθμιστική ανάλυση

Επαύξηση δυαδικού μετρητή

Έστω ένας μετρητής C με k bits : μια πράξη επαύξησης θέτει

$$C \leftarrow C + 1 \pmod{2^k}$$

Γενικά το i-οστό ψηφίο από το τέλος αλλάζει μετά από 2^{i-1} επαυξήσεις.

$\Rightarrow$ Σε ακολουθία N πράξεων το i-οστό ψηφίο από το τέλος αλλάζει συνολικά $\left\lfloor \frac{N}{2^{i-1}} \right\rfloor$ φορές

$\Rightarrow$ Σύνολο αλλαγών για όλα τα ψηφία = $\sum_{i=1}^{k} \left\lfloor \frac{N}{2^{i-1}} \right\rfloor < \sum_{i=1}^{k} \frac{N}{2^{i-1}} = N \sum_{j=0}^{k-1} 2^{-j}$

$$< N \sum_{j=0}^{\infty} 2^{-j} = 2N$$

Αντισταθμιστική ανάλυση

Επαύξηση δυαδικού μετρητή

Έστω ένας μετρητής C με k bits : μια πράξη επαύξησης θέτει

$$C \leftarrow C + 1 \pmod{2^k}$$

Γενικά το i-οστό ψηφίο από το τέλος αλλάζει μετά από 2^{i-1} επαυξήσεις.

$\Rightarrow$ Σε ακολουθία N πράξεων το i-οστό ψηφίο από το τέλος αλλάζει συνολικά $\left\lfloor \frac{N}{2^{i-1}} \right\rfloor$ φορές

$\Rightarrow$ Σύνολο αλλαγών για όλα τα ψηφία = $\sum_{i=1}^{k} \left\lfloor \frac{N}{2^{i-1}} \right\rfloor < \sum_{i=1}^{k} \frac{N}{2^{i-1}} = N \sum_{j=0}^{k-1} 2^{-j}$

$$< N \sum_{j=0}^{\infty} 2^{-j} = 2N$$

Αντισταθμιστικό κόστος ανά πράξη = $2\mathrm{N}/\mathrm{N} = 2$

Αντισταθμιστική ανάλυση

Ενεργειακή μέθοδος

→ Αποδίδουμε στη δομή δεδομένων ένα **δυναμικό** («δυναμική ενέργεια»).

$$\Phi(D) = \text{δυναμικό δομής δεδομένων D}$$

όπου $\Phi(D) \in \mathbf{R}$

→ Έστω D_0 η αρχική δομή δεδομένων και D_i η δομή μετά την i-οστη πράξη.
Επίσης, έστω c_i το κόστος της i-οστής πράξης.

Αντισταθμιστικό κόστος i-οστής πράξης : $\hat{c}_i = c_i + \Phi(D_i) - \Phi(D_{i-1})$

Συνολικό αντισταθμιστικό κόστος για N πράξεις :

$$\sum_{i=1}^{N} \hat{c}_i = \sum_{i=1}^{N} \Big(c_i + \Phi(D_i) - \Phi(D_{i-1}) \Big) = \sum_{i=1}^{N} c_i + \Phi(D_N) - \Phi(D_0)$$

→ Θέλουμε $\Phi(D_N) \geq \Phi(D_0)$ έτσι ώστε $\sum_{i=1}^{N} c_i \leq \sum_{i=1}^{N} \hat{c}_i$

Αντισταθμιστική ανάλυση

Ενεργειακή μέθοδος

Αντισταθμιστικό κόστος i-οστής πράξης : $\hat{c}_i = c_i + \Phi(D_i) - \Phi(D_{i-1})$

Ερμηνεία :

- αν $\Phi(D_i) - \Phi(D_{i-1}) > 0$ η δομή συγκεντρώνει δυναμικό και $\hat{c}_i > c_i$
- αν $\Phi(D_i) - \Phi(D_{i-1}) < 0$ τότε η δομή χάνει δυναμικό και $\hat{c}_i < c_i$

Δηλαδή το κόστος μιας ακριβής πράξης αποπληρώνεται από τη διαφορά δυναμικού

Αντισταθμιστική ανάλυση

Ενεργειακή μέθοδος

Συνολικό αντισταθμιστικό κόστος για N πράξεις :

$$\sum_{i=1}^{N} \hat{c}_i = \sum_{i=1}^{N} \Big(c_i + \Phi(D_i) - \Phi(D_{i-1}) \Big) = \sum_{i=1}^{N} c_i + \Phi(D_N) - \Phi(D_0)$$

Θέλουμε $\Phi(D_N) \geq \Phi(D_0)$ έτσι ώστε $\sum_{i=1}^{N} c_i \leq \sum_{i=1}^{N} \hat{c}_i$

Προσοχή: Επειδή μπορεί να μη γνωρίζουμε το πλήθος των πράξεων N απαιτούμε να ισχύει $\Phi(D_i) \geq \Phi(D_0)$ για κάθε i=1,2,…,N

Αντισταθμιστική ανάλυση

Ενεργειακή μέθοδος

Παράδειγμα: **Διαχείριση στοίβας**

Επιλέγουμε $\Phi(S) =$ αριθμός αντικειμένων στη στοίβα S

Για αρχικά κενή στοίβα έχουμε $\Phi(S_0) = 0$

$$\Rightarrow \Phi(S_i) \geq \Phi(S_0) = 0 \quad \text{για κάθε i=1,2,…,N}$$

$$\Rightarrow \sum_{j=1}^{i} c_j \leq \sum_{j=1}^{i} \hat{c}_j \quad \text{για κάθε i=1,2,…,N}$$

Επομένως το συνολικό αντισταθμιστικό κόστος αποτελεί άνω φράγμα του συνολικού πραγματικού κόστους.

Αντισταθμιστική ανάλυση

Ενεργειακή μέθοδος

Παράδειγμα: **Διαχείριση στοίβας**

Επιλέγουμε $\Phi(S) =$ αριθμός αντικειμένων στη στοίβα S

Μένει να υπολογίσουμε το αντισταθμιστικό κόστος $\hat{c}_i$ για κάθε τύπο πράξης

Έστω ότι η i-οστή πράξη είναι τύπου push :

push(S,x) : τοποθετεί το x στην κορυφή της στοίβας S

Έχουμε $c_i = 1$ και $\Phi(S_i) = \Phi(S_{i-1}) + 1$, επομένως

$$\hat{c}_i = c_i + \Phi(S_i) - \Phi(S_{i-1}) = 2$$

Αντισταθμιστική ανάλυση

Ενεργειακή μέθοδος

Παράδειγμα: **Διαχείριση στοίβας**

Επιλέγουμε $\Phi(S) =$ αριθμός αντικειμένων στη στοίβα S

Μένει να υπολογίσουμε το αντισταθμιστικό κόστος $\hat{c}_i$ για κάθε τύπο πράξης

Έστω ότι η i-οστή πράξη είναι τύπου pop :

pop(S) : αφαιρεί από τη στοίβα S το στοιχείο στην κορυφή της

Έχουμε $c_i = 1$ και $\Phi(S_i) = \Phi(S_{i-1}) - 1$, επομένως

$$\hat{c}_i = c_i + \Phi(S_i) - \Phi(S_{i-1}) = 0$$

Αντισταθμιστική ανάλυση

Ενεργειακή μέθοδος

Παράδειγμα: **Διαχείριση στοίβας**

Επιλέγουμε $\Phi(S) =$ αριθμός αντικειμένων στη στοίβα S

Μένει να υπολογίσουμε το αντισταθμιστικό κόστος $\hat{c}_i$ για κάθε τύπο πράξης

Έστω ότι η i-οστή πράξη είναι τύπου multiPop :

multiPop(S,k) : αφαιρεί από τη στοίβα S τα πρώτα k στοιχεία στην κορυφή της

Έχουμε $c_i = \min\{k, |S|\}$ και $\Phi(S_i) = \Phi(S_{i-1}) - \min\{k, |S|\}$, επομένως

$$\hat{c}_i = c_i + \Phi(S_i) - \Phi(S_{i-1}) = 0$$

Αρθρωτά δένδρα (splay trees)

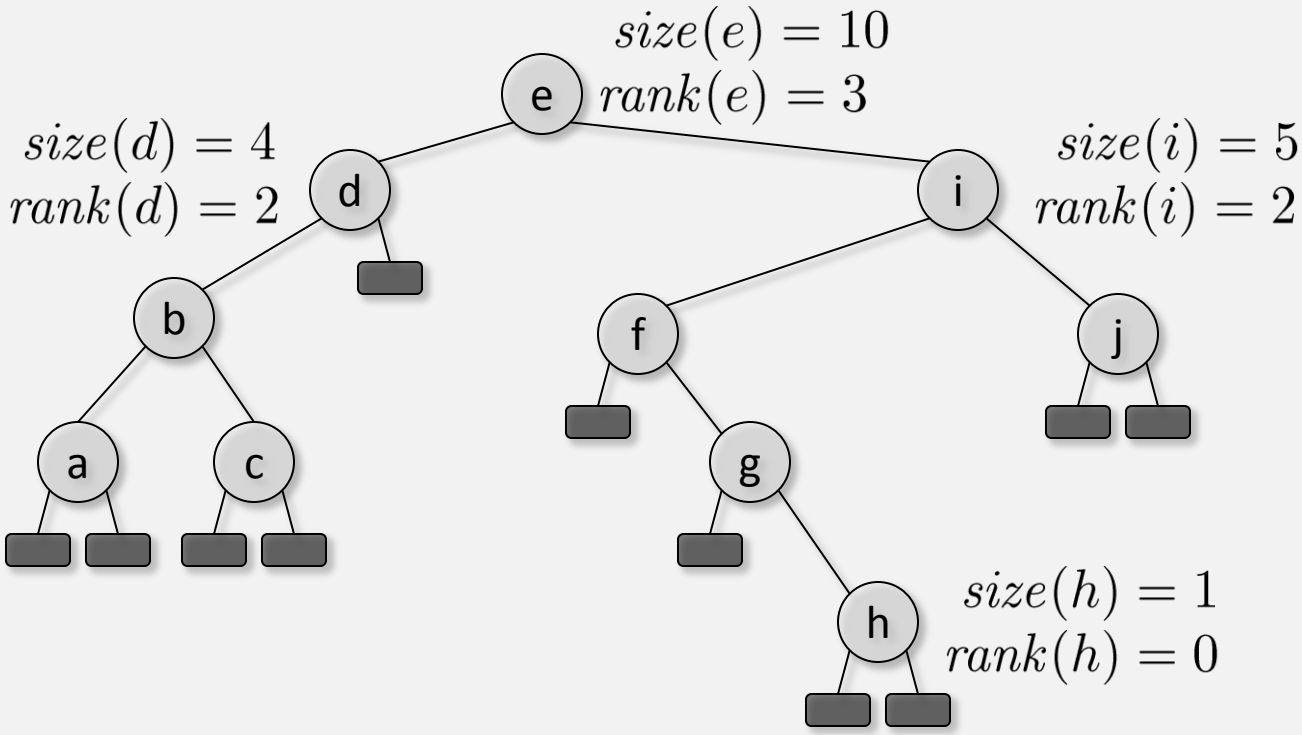
Ιδιότητα: Σε ένα αρθρωτό δένδρο με n κόμβους η λειτουργία **splay** εκτελείται σε $O(\log n)$ αντισταθμιστικό χρόνο

Αρθρωτά δένδρα (splay trees)

Ιδιότητα: Σε ένα αρθρωτό δένδρο με n κόμβους η λειτουργία **splay** εκτελείται σε $O(\log n)$ αντισταθμιστικό χρόνο

Θα χρησιμοποιήσουμε την ενεργειακή μέθοδο. Έστω $size(x)$ το πλήθος των απογόνων του κόμβου x στο αρθρωτό δένδρο S.

Ορίζουμε την «τάξη» του x ως $rank(x) = \lfloor \lg size(x) \rfloor$

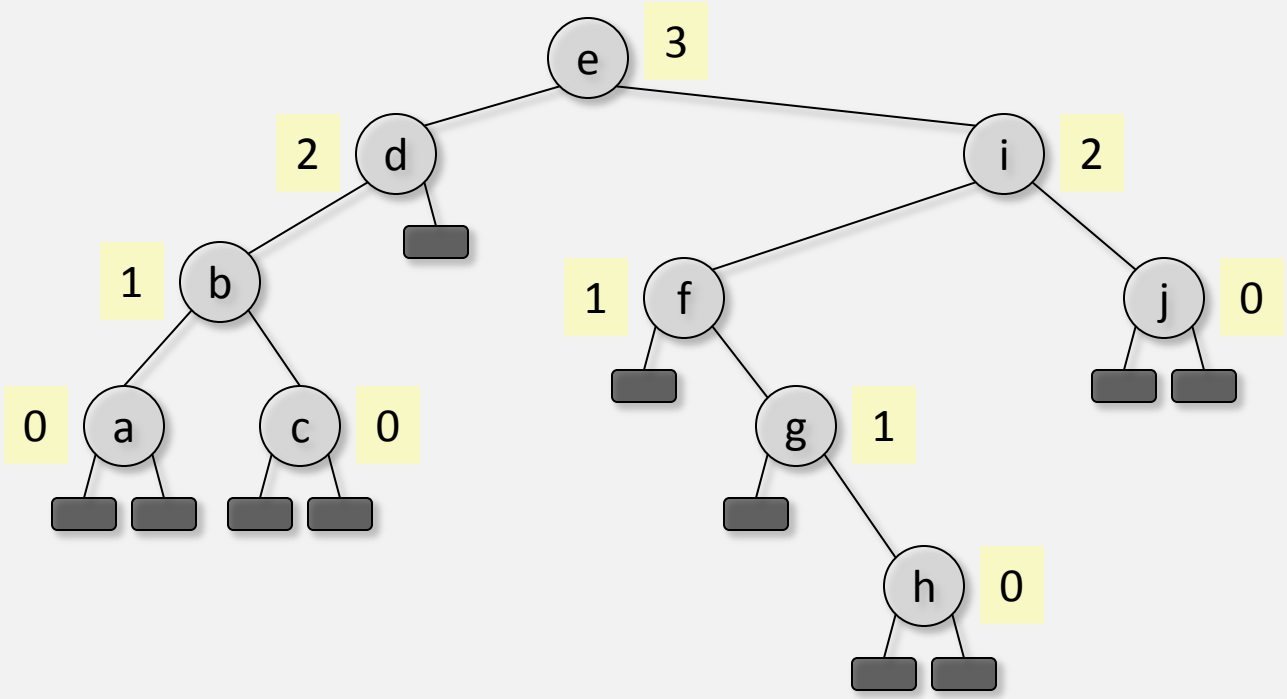


Αρθρωτά δένδρα (splay trees)

Ιδιότητα: Σε ένα αρθρωτό δένδρο με n κόμβους η λειτουργία **splay** εκτελείται σε $O(\log n)$ αντισταθμιστικό χρόνο

Θα χρησιμοποιήσουμε την ενεργειακή μέθοδο. Έστω $size(x)$ το πλήθος των απογόνων του κόμβου x στο αρθρωτό δένδρο S.

Ορίζουμε την «τάξη» του x ως $rank(x) = \lfloor \lg size(x) \rfloor$



Αρθρωτά δένδρα (splay trees)

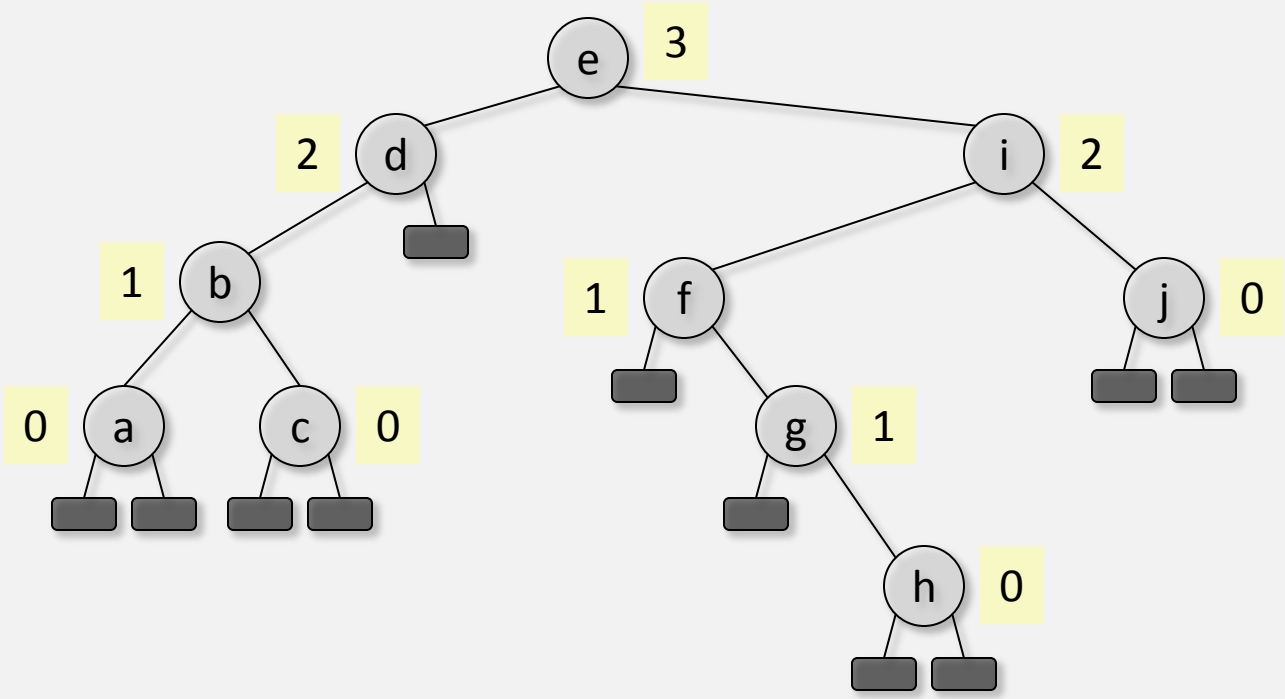
Ιδιότητα: Σε ένα αρθρωτό δένδρο με n κόμβους η λειτουργία **splay** εκτελείται σε $O(\log n)$ αντισταθμιστικό χρόνο

Θα χρησιμοποιήσουμε την ενεργειακή μέθοδο. Έστω $size(x)$ το πλήθος των απογόνων του κόμβου x στο αρθρωτό δένδρο S.

Ορίζουμε την «τάξη» του x ως $rank(x) = \lfloor \lg size(x) \rfloor$

Δυναμικό δένδρου S

$$\Phi(S) = \sum_{x \in S} rank(x)$$



Αρθρωτά δένδρα (splay trees)

Ιδιότητα: Σε ένα αρθρωτό δένδρο με n κόμβους η λειτουργία **splay** εκτελείται σε $O(\log n)$ αντισταθμιστικό χρόνο

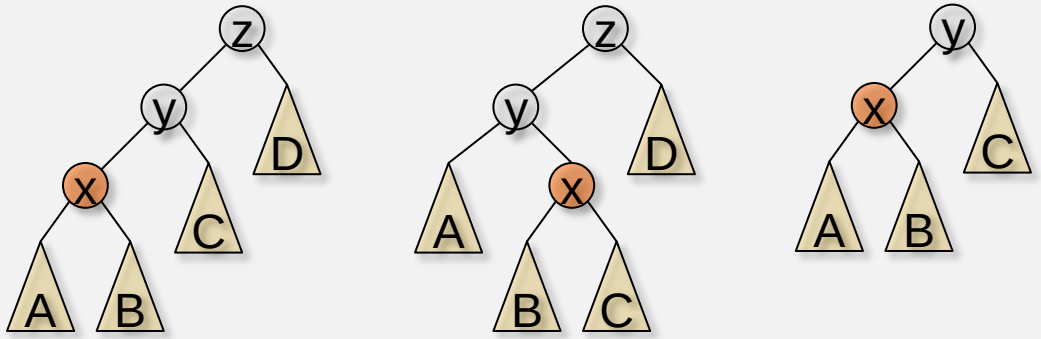
Θα χρησιμοποιήσουμε την ενεργειακή μέθοδο. Έστω $size(x)$ το πλήθος των απογόνων του κόμβου x στο αρθρωτό δένδρο S.

Ορίζουμε την «τάξη» του x ως $rank(x) = \lfloor \lg size(x) \rfloor$

Δυναμικό δένδρου S : $\Phi(S) = \sum_{x \in S} rank(x)$

Αναλύουμε την επίδραση του κάθε βήματος περιστροφών στο δυναμικό του δένδρου

Παρατηρούμε ότι σε κάθε περίπτωση αλλάζει η τάξη δύο ή τριών κόμβων (x, y και z)



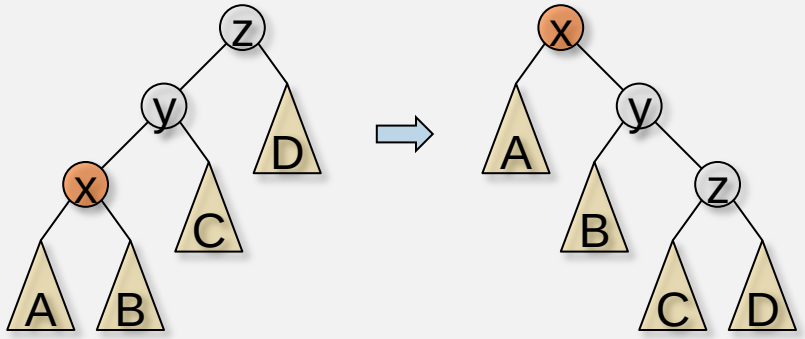
Αρθρωτά δένδρα (splay trees)

Ιδιότητα: Σε ένα αρθρωτό δένδρο με n κόμβους η λειτουργία **splay** εκτελείται σε $O(\log n)$ αντισταθμιστικό χρόνο

Θα χρησιμοποιήσουμε την ενεργειακή μέθοδο. Έστω $size(x)$ το πλήθος των απογόνων του κόμβου x στο αρθρωτό δένδρο S.

Ορίζουμε την «τάξη» του x ως $rank(x) = \lfloor \lg size(x) \rfloor$

Δυναμικό δένδρου S : $\Phi(S) = \sum_{x \in S} rank(x)$



Ας αναλύσουμε την περίπτωση zig-zig

Έστω $rank'(x), size'(x)$ και $\Phi'(S)$ οι τιμές μετά τη διπλή περιστροφή

Ισχύουν $size'(x) = size(z), size(y) \geq size(x), size'(y) \leq size'(x) \Rightarrow$

$rank'(x) = rank(z), rank(y) \geq rank(x), rank'(y) \leq rank'(x)$

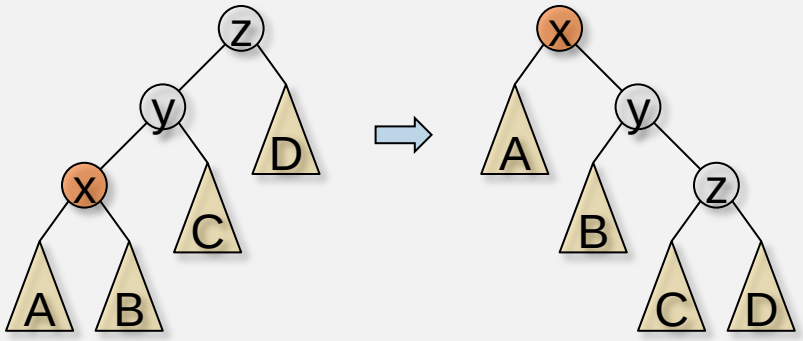
και

$$\Phi'(S) - \Phi(S) = \big(rank'(x) - rank(x)\big) + \big(rank'(y) - rank(y)\big) + \big(rank'(z) - rank(z)\big)$$

Αρθρωτά δένδρα (splay trees)

Ιδιότητα: Σε ένα αρθρωτό δένδρο με n κόμβους η λειτουργία **splay** εκτελείται σε $O(\log n)$ αντισταθμιστικό χρόνο

Ας αναλύσουμε την περίπτωση zig-zig



Ισχύουν $size'(x) = size(z), size(y) \geq size(x), size'(y) \leq size'(x) \Rightarrow$

$rank'(x) = rank(z), rank(y) \geq rank(x), rank'(y) \leq rank'(x)$

και

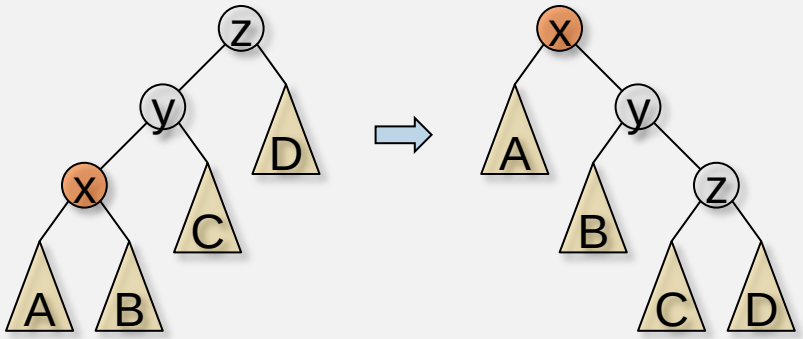
$$\Phi'(S) - \Phi(S) = \big(rank'(x) - rank(x)\big) + \big(rank'(y) - rank(y)\big) + \big(rank'(z) - rank(z)\big)$$

$$= \big(rank(z) - rank(x)\big) + \big(rank'(y) - rank(y)\big) + \big(rank'(z) - rank(z)\big)$$

Αρθρωτά δένδρα (splay trees)

Ιδιότητα: Σε ένα αρθρωτό δένδρο με n κόμβους η λειτουργία **splay** εκτελείται σε $O(\log n)$ αντισταθμιστικό χρόνο

Ας αναλύσουμε την περίπτωση zig-zig



Ισχύουν $size'(x) = size(z), size(y) \geq size(x), size'(y) \leq size'(x) \Rightarrow$
$rank'(x) = rank(z), rank(y) \geq rank(x), rank'(y) \leq rank'(x)$

και

$$\Phi'(S) - \Phi(S) = \big(rank'(x) - rank(x)\big) + \big(rank'(y) - rank(y)\big) + \big(rank'(z) - rank(z)\big)$$
$$= \big(rank(z) - rank(x)\big) + \big(rank'(y) - rank(y)\big) + \big(rank'(z) - rank(z)\big)$$
$$= rank'(x) + rank'(z) - 2rank(x)$$

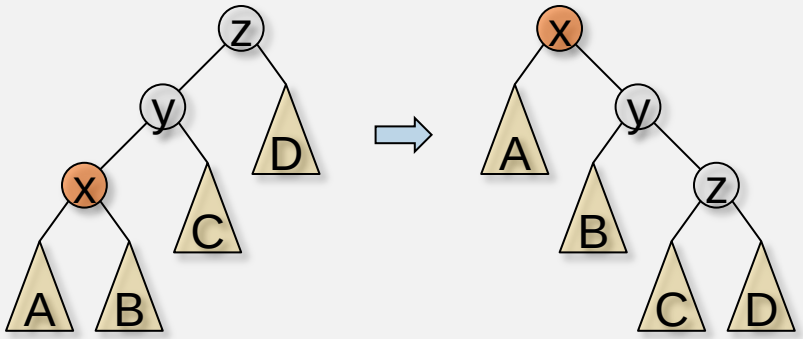
Επιπλέον $size(x) + size'(z) \leq size'(x) \Rightarrow \dfrac{size(x) + size'(z)}{2} \leq \dfrac{size'(x)}{2}$

$$\Rightarrow \frac{\lg size(x) + \lg size'(z)}{2} \leq \lg \frac{size'(x)}{2} \quad \left(\frac{\lg a + \lg b}{2} \leq \lg \frac{a+b}{2}\right)$$

Αρθρωτά δένδρα (splay trees)

Ιδιότητα: Σε ένα αρθρωτό δένδρο με n κόμβους η λειτουργία **splay** εκτελείται σε $O(\log n)$ αντισταθμιστικό χρόνο

Ας αναλύσουμε την περίπτωση zig-zig



Ισχύουν $size'(x) = size(z), size(y) \geq size(x), size'(y) \leq size'(x) \Rightarrow$
$rank'(x) = rank(z), rank(y) \geq rank(x), rank'(y) \leq rank'(x)$

και

$$\Phi'(S)-\Phi(S) = \big(rank'(x)-rank(x)\big)+\big(rank'(y)-rank(y)\big)+\big(rank'(z)-rank(z)\big)$$
$$= \big(rank(z) - rank(x)\big) + \big(rank'(y) - rank(y)\big) + \big(rank'(z) - rank(z)\big)$$
$$\leq rank'(x) + rank'(z) - 2rank(x)$$

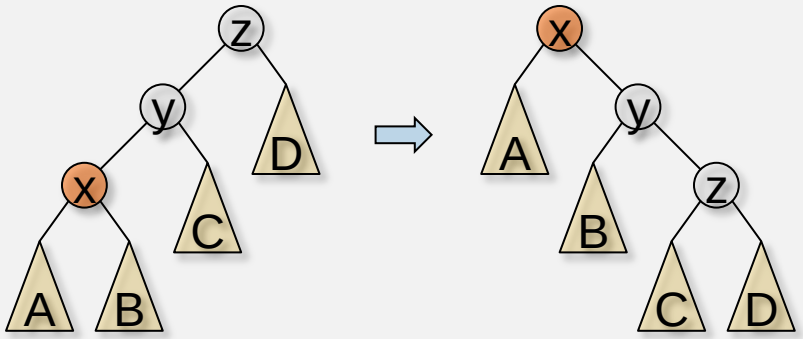
Επιπλέον $size(x) + size'(z) \leq size'(x) \Rightarrow \dfrac{size(x) + size'(z)}{2} \leq \dfrac{size'(x)}{2}$

$$\Rightarrow \frac{\lg size(x) + \lg size'(z)}{2} \leq \lg \frac{size'(x)}{2} \Rightarrow rank'(z) \leq 2rank'(x) - rank(x) - 2$$

Αρθρωτά δένδρα (splay trees)

Ιδιότητα: Σε ένα αρθρωτό δένδρο με n κόμβους η λειτουργία **splay** εκτελείται σε $O(\log n)$ αντισταθμιστικό χρόνο

Ας αναλύσουμε την περίπτωση zig-zig



Ισχύουν $size'(x) = size(z), size(y) \geq size(x), size'(y) \leq size(x) \Rightarrow$
$rank'(x) = rank(z), rank(y) \geq rank(x), rank'(y) \leq rank'(x)$

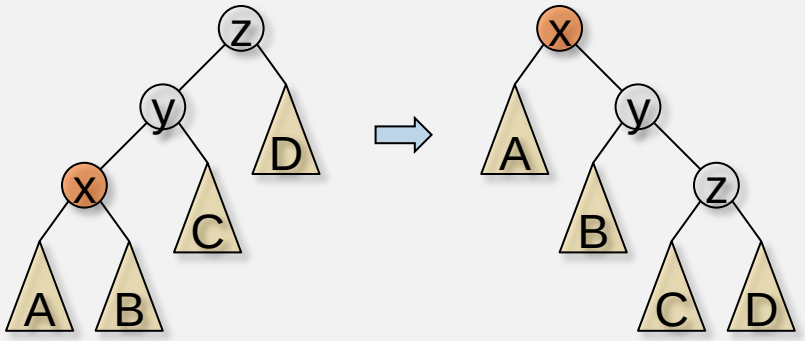
και

$$\Phi'(S)-\Phi(S) = \big(rank'(x)-rank(x)\big)+\big(rank'(y)-rank(y)\big)+\big(rank'(z)-rank(z)\big)$$
$$= \big(rank(z) - rank(x)\big) + \big(rank'(y) - rank(y)\big) + \big(rank'(z) - rank(z)\big)$$
$$\leq rank'(x) + rank'(z) - 2rank(x)$$
$$\leq rank'(x) + 2rank'(x) - rank(x) - 2 - 2rank(x)$$
$$= 3\big(rank'(x) - rank(x)\big) - 2$$

Αρθρωτά δένδρα (splay trees)

Ιδιότητα: Σε ένα αρθρωτό δένδρο με n κόμβους η λειτουργία **splay** εκτελείται σε $O(\log n)$ αντισταθμιστικό χρόνο

Ας αναλύσουμε την περίπτωση zig-zig



Άρα $\Phi'(S) - \Phi(S) \leq 3\big(rank'(x) - rank(x)\big) - 2$

Πραγματικό κόστος περιστροφών $c = 2$

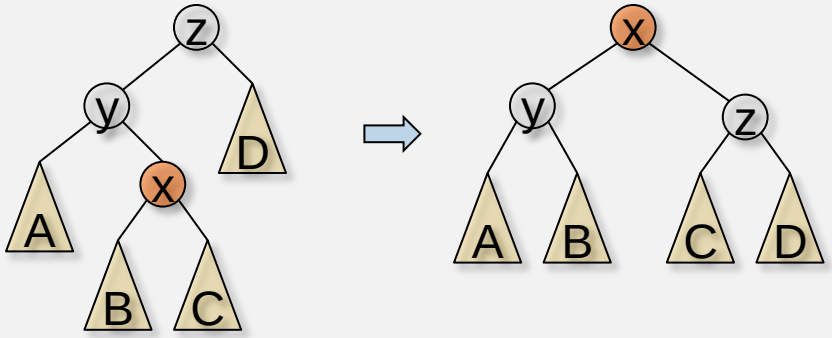
Αντισταθμιστικό κόστος περιστροφών $\hat{c} = c + \Phi'(S) - \Phi(S)$

$\leq 3\big(rank'(x) - rank(x)\big)$

Αρθρωτά δένδρα (splay trees)

Ιδιότητα: Σε ένα αρθρωτό δένδρο με n κόμβους η λειτουργία **splay** εκτελείται σε $O(\log n)$ αντισταθμιστικό χρόνο

Ομοίως αναλύεται η περίπτωση zig-zag



Άρα $\Phi'(S) - \Phi(S) \leq 3\big(rank'(x) - rank(x)\big) - 2$

Πραγματικό κόστος περιστροφών $c = 2$

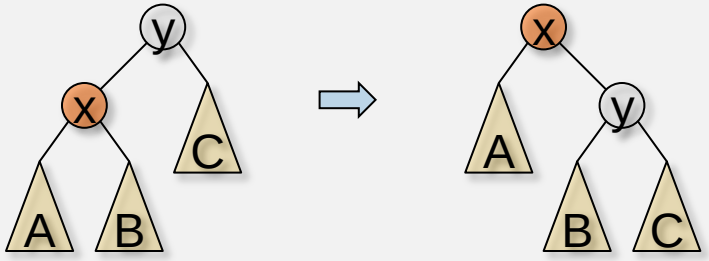
Αντισταθμιστικό κόστος περιστροφών $\hat{c} = c + \Phi'(S) - \Phi(S)$

$\leq 3\big(rank'(x) - rank(x)\big)$

Αρθρωτά δένδρα (splay trees)

Ιδιότητα: Σε ένα αρθρωτό δένδρο με n κόμβους η λειτουργία **splay** εκτελείται σε $O(\log n)$ αντισταθμιστικό χρόνο

Τερματική περίπτωση



Ισχύουν $size'(x) = size(y), size(y) \geq size(x) \Rightarrow$

$rank'(x) = rank(y), rank(y) \geq rank(x)$

και

$$\Phi'(S) - \Phi(S) = \big(rank'(x) - rank(x)\big) + \big(rank'(y) - rank(y)\big)$$

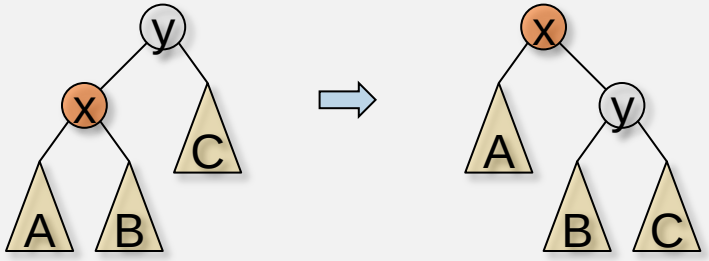
$$= \big(rank(y) - rank(x)\big) + \big(rank'(y) - rank(y)\big)$$

$$= rank'(y) - rank(x)$$

Αρθρωτά δένδρα (splay trees)

Ιδιότητα: Σε ένα αρθρωτό δένδρο με n κόμβους η λειτουργία **splay** εκτελείται σε $O(\log n)$ αντισταθμιστικό χρόνο

Τερματική περίπτωση



Ισχύουν $size'(x) = size(y), size(y) \geq size(x) \Rightarrow$

$rank'(x) = rank(y), rank(y) \geq rank(x)$

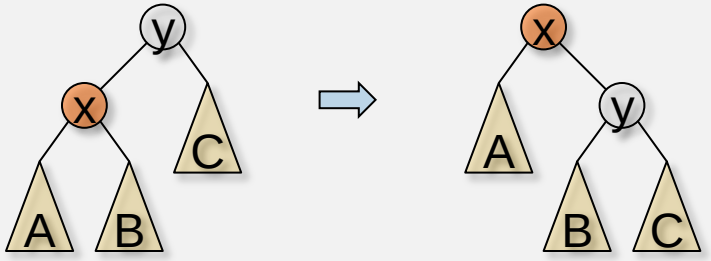
και

$$\begin{aligned}\Phi'(S) - \Phi(S) &= \big(rank'(x) - rank(x)\big) + \big(rank'(y) - rank(y)\big) \\ &= \big(\cancel{rank}(y) - rank(x)\big) + \big(rank'(y) - \cancel{rank}(y)\big) \\ &= rank'(y) - rank(x) \leq rank'(x) - rank(x) \\ &\leq 3\big(rank'(x) - rank(x)\big)\end{aligned}$$

Αρθρωτά δένδρα (splay trees)

Ιδιότητα: Σε ένα αρθρωτό δένδρο με n κόμβους η λειτουργία **splay** εκτελείται σε $O(\log n)$ αντισταθμιστικό χρόνο

Τερματική περίπτωση



Άρα $\Phi'(S) - \Phi(S) \leq 3\big(rank'(x) - rank(x)\big)$

Πραγματικό κόστος περιστροφών $c = 1$

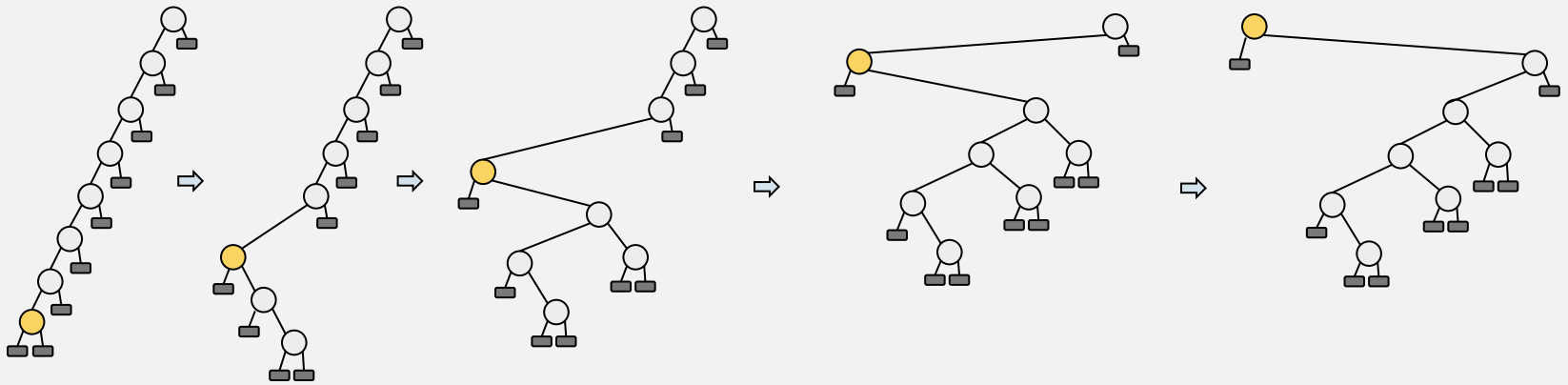
Αντισταθμιστικό κόστος περιστροφών $\hat{c} = c + \Phi'(S) - \Phi(S)$

$\leq 3\big(rank'(x) - rank(x)\big) + 1$

Αρθρωτά δένδρα (splay trees)

Ιδιότητα: Σε ένα αρθρωτό δένδρο με n κόμβους η λειτουργία **splay** εκτελείται σε $O(\log n)$ αντισταθμιστικό χρόνο

Έστω λοιπόν ότι η λειτουργιά **splay** πραγματοποιείται σε p βήματα



Έστω $rank_i(x)$ και $\Phi_i(S)$ οι τιμές μετά το i-οστό βήμα

Το συνολικό αντισταθμιστικό κόστος είναι

$$\hat{c} = \sum_{i=1}^{p} \hat{c}_i \leq \sum_{i=1}^{p} \Big(3\big(rank_i(x) - rank_{i-1}(x)\big)\Big) + 1 = 3\big(rank_p(x) - rank_0(x)\big) + 1$$

$$\leq 3 \lg n + 1$$

Αρθρωτά δένδρα (splay trees)

Γενίκευση : Βάρη στους κόμβους

Έστω ότι δίνουμε ένα βάρος $w(x)$ σε κάθε κόμβο x του αρθρωτού δένδρου S

$S(x) =$ απόγονοι του x (συμπεριλαμβανομένου του x) στο S

$s(x) = \sum_{y \in S(x)} w(y) =$ συνολικό βάρος των απογόνων του x

$r(x) = \lfloor \lg s(x) \rfloor$

Δυναμικό δένδρου S: $\Phi(S) = \sum_{x \in S} r(x)$

Ιδιότητα: Η λειτουργία **splay(x)** εκτελείται σε $O(\log (s(t)/s(x)))$ αντισταθμιστικό χρόνο, όπου t η ρίζα του αρθρωτού δένδρου.

Αρθρωτά δένδρα (splay trees)

Λήμμα Προσπέλασης

Η λειτουργία **splay(x)** εκτελείται σε $O(\log(s(t)/s(x)))$ αντισταθμιστικό χρόνο, όπου t η ρίζα του αρθρωτού δένδρου.

Στατικά Βέλτιστη Προσπέλαση

Έστω $f(i)$ ο αριθμός των προσπελάσεων που γίνονται στο κλειδί i. Αν κάθε κλειδί προσπελαύνεται τουλάχιστον μία φορά, τότε ο συνολικός χρόνος εκτέλεσης μιας ακολουθίας m προσπελάσεων είναι

$$O\left(m + \sum_{i=1}^{n} f(i) \log(m/f(i))\right)$$

Συνεπάγεται από το λήμμα προσπέλασης για $w(i) = f(i)/m$. Έχουμε $s(t) = 1$ και ο αντισταθμιστικός χρόνος προσπέλασης του κλειδιού i είναι $O(\log m/f(i))$

Η συνολική μεταβολή του δυναμικού της δομής είναι $O(\sum_{i=1}^{n} \log(m/f(i)))$

Αρθρωτά δένδρα (splay trees)

Λήμμα Στατικού Σελιδοδείκτη

Έστω k ένα σταθερό κλειδί. Τότε, ο συνολικός χρόνος εκτέλεσης μιας ακολουθίας m προσπελάσεων $i_1, i_2, \ldots, i_m$ είναι

$$O\left(n \log n + m + \sum_{j=1}^{m} \log\left(d(k, i_j) + 1\right)\right)$$

όπου $d(k, i_j)$ η απόσταση των κλειδιών k και i_j στη διατεταγμένη ακολουθία των κλειδιών του αρθρωτού δένδρου.

Συνεπάγεται από το λήμμα προσπέλασης θέτοντας $w(i) = 1/(d(k,i)+1)^2 \geq 1/n^2$ για το βάρος του κλειδιού i. Τότε

$$s(t) = \sum_{i=1}^{n} \frac{1}{(d(k,i)+1)^2} \leq 2 \sum_{j=1}^{\infty} \frac{1}{j^2} = O(1)$$

και ο αντισταθμιστικός χρόνος της j-οστής προσπέλασης είναι $O(\log(d(k, i_j) + 1))$

Η συνολική μεταβολή του δυναμικού της δομής είναι $O(n \log n)$

Αρθρωτά δένδρα (splay trees)

Λήμμα Συνόλου Εργασίας

Έστω μια ακολουθία m προσπελάσεων $i_1, i_2, \ldots, i_m$. Έστω $n(j)$ ο αριθμός των διαφορετικών κλειδιών που προσπελάστηκαν πριν την j-οστή προσπέλαση μέχρι την τελευταία προσπέλαση του i_j . Τότε, ο συνολικός χρόνος εκτέλεσης της ακολουθίας προσπελάσεων είναι

$$O\left(n \log n + m + \sum_{j=1}^{m} \log\left(n(j) + 1\right)\right)$$

Δίνουμε τα βάρη $1, 1/4, 1/9, \ldots, 1/n^2$ στα κλειδιά με τη σειρά ως προς την πρώτη προσπέλαση τους. (Το κλειδί που προσπελάστηκε πρώτο λαμβάνει το μεγαλύτερο βάρος.) Μετά την j-οστή προσπέλαση αλλάζουμε την κατανομή βαρών ως εξής:

Έστω $1/k^2$ το βάρος του i_j κατά την j-οστή προσπέλαση. Μετά την προσπέλαση δίνουμε στο i_j βάρος 1, και σε κάθε κλειδί i με βάρος $1/l^2$, όπου $l < k$, ορίζουμε το νέο βάρος του i ως $1/(l+1)^2$.

Άρα, κατά την j-οστή προσπέλαση έχουμε $w(i_j) = 1/(n(j)+1)^2$