Δομές Αναζήτησης

Χειριζόμαστε ένα σύνολο στοιχείων S όπου το κάθε στοιχείο έχει ένα κλειδί από ολικά διατεταγμένο σύνολο K

Θέλουμε να υποστηρίξουμε δύο βασικές λειτουργίες:

- Εισαγωγή ενός νέου στοιχείου $x \in S$ με κλειδί $k \in K$
- Αναζήτηση στοιχείου με δεδομένο κλειδί $k \in K$

Δομές Αναζήτησης

Χειριζόμαστε ένα σύνολο στοιχείων S όπου το κάθε στοιχείο έχει ένα κλειδί από ολικά διατεταγμένο σύνολο K

Θέλουμε να υποστηρίξουμε δύο βασικές λειτουργίες:

- Εισαγωγή ενός νέου στοιχείου $x \in S$ με κλειδί $k \in K$
- Αναζήτηση στοιχείου με δεδομένο κλειδί $k \in K$

Άλλες χρήσιμες λειτουργίες είναι:

- Διαγραφή ενός καθορισμένου στοιχείου
- Επιλογή j-οστού στοιχείου : Εύρεση στοιχείου με το j-οστό μικρότερο κλειδί
- Ταξινόμηση των στοιχείων σύμφωνα με τα κλειδιά τους
- Εύρεση προκάτοχου ή διάδοχου
- Ένωση δύο δομών αναζήτησης

Δομές Αναζήτησης

Διασύνδεση δομής λεξικού

`Dictionary()`	κατασκευή δομής λεξικού
`void insert(Key k, Item i)`	εισαγωγή αντικειμένου i με κλειδί k
`Item search(Key k)`	επιστροφή αντικειμένου με κλειδί k
`int rank(Key k)`	τάξη κλειδιού k = πλήθος κλειδιών μικρότερων του k
`Key select(int r)`	επιστροφή του κλειδιού τάξης r
`void delete(Key k)`	διαγραφή αντικειμένου με κλειδί k
`boolean isEmpty()`	έλεγχος αν η δομή λεξικού είναι κενή
`int size()`	αριθμός αντικειμένων στη δομή λεξικού
`Iterable<Key> keys()`	όλα τα κλειδιά στη δομή λεξικού

Δομές Αναζήτησης

Διατεταγμένος πίνακας: $O(\log N)$ χρόνος αναζήτησης (δυαδική αναζήτηση)

Αναζήτηση 25

5	8	12	19	25	26	31	33	42	58	59	78	81	90	93

5	8	12	19	25	26	31	33	42	58	59	78	81	90	93

5	8	12	19	25	26	31	33	42	58	59	78	81	90	93

5	8	12	19	25	26	31	33	42	58	59	78	81	90	93

Δομές Αναζήτησης

Διατεταγμένος πίνακας: $O(\log N)$ χρόνος αναζήτησης (δυαδική αναζήτηση)

Αναζήτηση 25

5	8	12	19	25	26	31	33	42	58	59	78	81	90	93

5	8	12	19	25	26	31	33	42	58	59	78	81	90	93

5	8	12	19	25	26	31	33	42	58	59	78	81	90	93

5	8	12	19	25	26	31	33	42	58	59	78	81	90	93

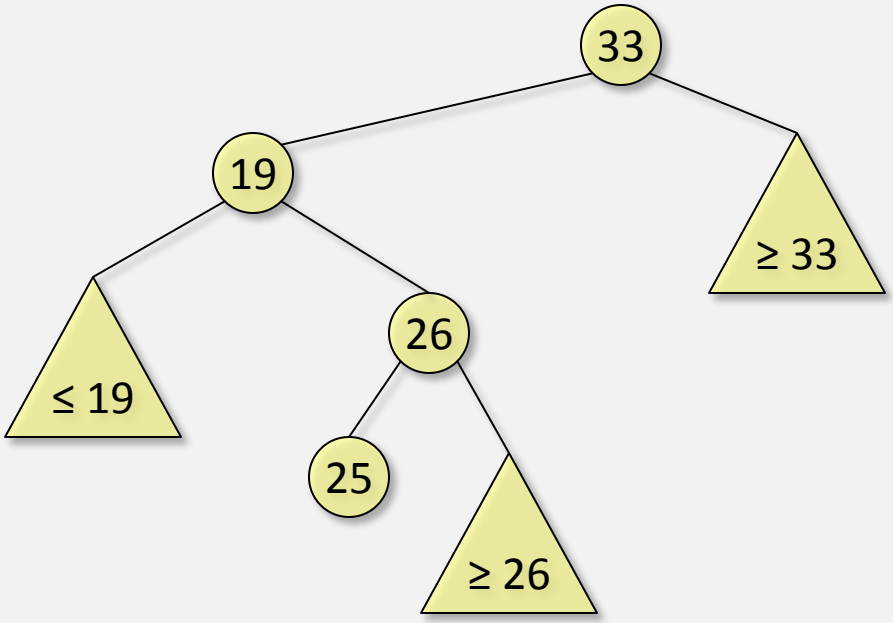
Όμως η εισαγωγή ενός νέου στοιχείου απαιτεί $O(N)$ χρόνο.

Δομές Αναζήτησης

Για να πετύχουμε καλύτερο χρόνο εισαγωγής μπορούμε να χρησιμοποιήσουμε μια δενδρική δομή

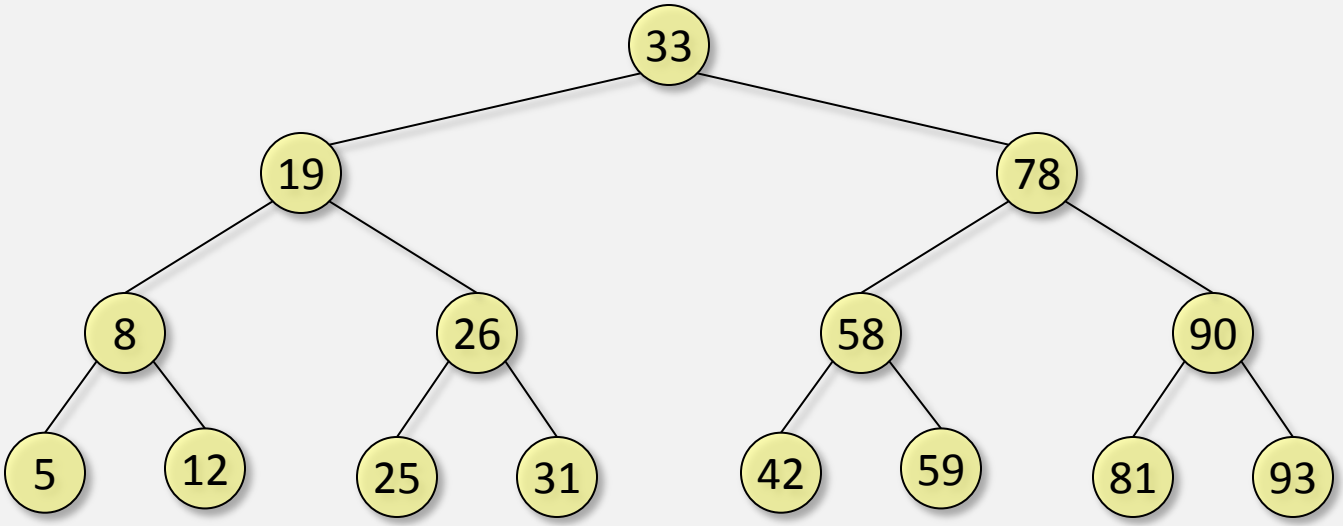
Αναζήτηση 25

5	8	12	19	25	26	31	33	42	58	59	78	81	90	93
5	8	12	19	25	26	31	33	42	58	59	78	81	90	93
5	8	12	19	25	26	31	33	42	58	59	78	81	90	93
5	8	12	19	25	26	31	33	42	58	59	78	81	90	93



Δομές Αναζήτησης

Για να πετύχουμε καλύτερο χρόνο εισαγωγής μπορούμε να χρησιμοποιήσουμε μια δενδρική δομή



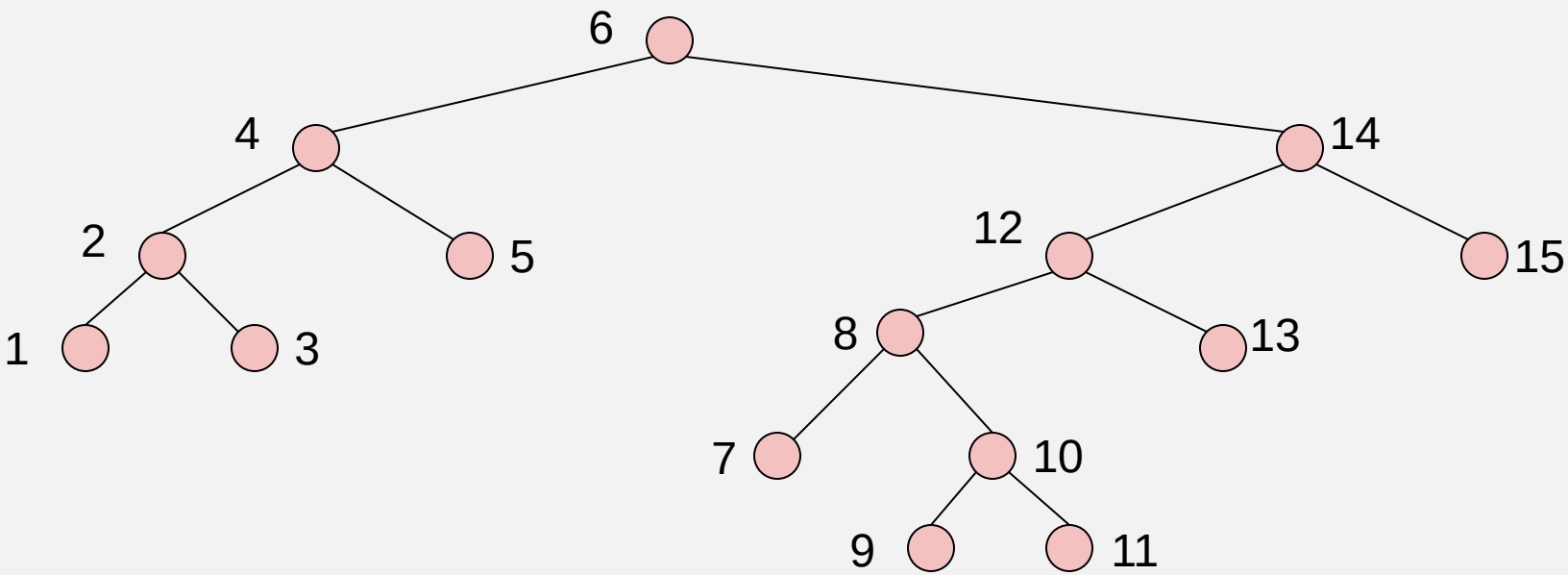
Δυαδικό δένδρο αναζήτησης

Η ενδοδιάτεταγμένη διάσχιση του δένδρου επισκέπτεται τους κόμβους σε αύξουσα σειρά κλειδιού

Διάσχιση Δυαδικού Δένδρου

```
void traverse(Node h) {
        if (h == null) return;
        traverse(h.left);
        h.item.visit();
        traverse(h.right);
}
```

ενδοδιάταξη (inorder)



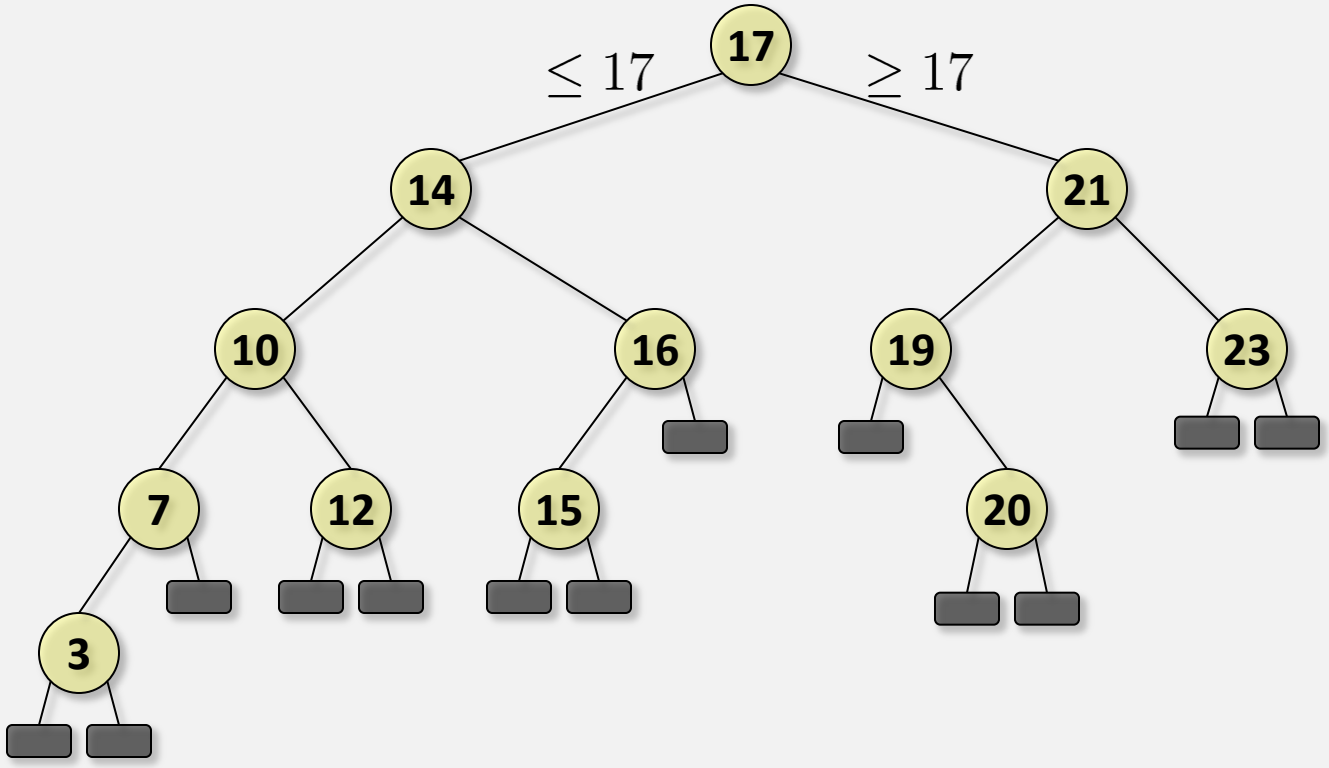
Δομές Αναζήτησης

	χειρότερη περίπτωση			μέση περίπτωση	
	εισαγωγή	αναζήτηση	επιλογή	εισαγωγή	αναζήτηση
διατεταγμένος πίνακας	$O(N)$	$O(\log N)$	$O(1)$	$O(N)$	$O(\log N)$
διατεταγμένη λίστα	$O(N)$	$O(N)$	$O(N)$	$O(N)$	$O(N)$
μη διατεταγμένος πίνακας	$O(1)$	$O(N)$	$O(N \log N)$	$O(1)$	$O(N)$
μη διατεταγμένη λίστα	$O(1)$	$O(N)$	$O(N \log N)$	$O(1)$	$O(N)$
δένδρο αναζήτησης (μη ισορροπημένο)	$O(N)$	$O(N)$	$O(N)$	$O(\log N)$	$O(\log N)$
τυχαιοποιημένο δένδρο	$O(N)^*$	$O(N)^*$	$O(N)^*$	$O(\log N)$	$O(\log N)$
ισορροπημένο δένδρο αναζήτησης	$O(\log N)$	$O(\log N)$	$O(\log N)$	$O(\log N)$	$O(\log N)$
κατακερματισμός	$O(1)$	$O(N)^*$	$O(N \log N)$	$O(1)$	$O(1)$

(*) Συμβαίνει με εξαιρετικά μικρή πιθανότητα

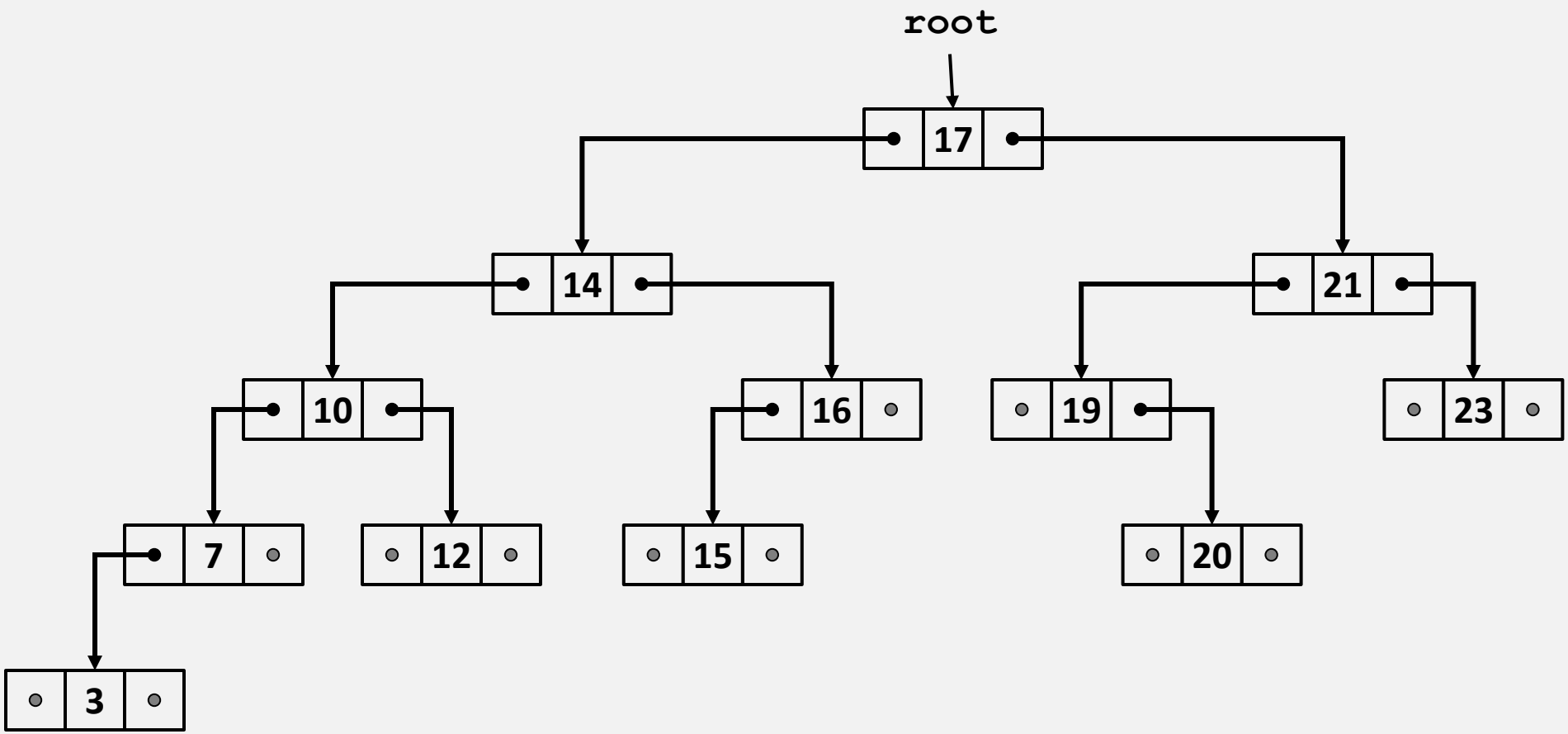
Δυαδικό Δένδρο Αναζήτησης

Το κλειδί οποιουδήποτε εσωτερικού κόμβου είναι μεγαλύτερο (ή ίσο) από όλα τα κλειδιά του αριστερού υποδένδρου και μικρότερο (ή ίσο) από όλα τα κλειδιά του δεξιού υποδένδρου.



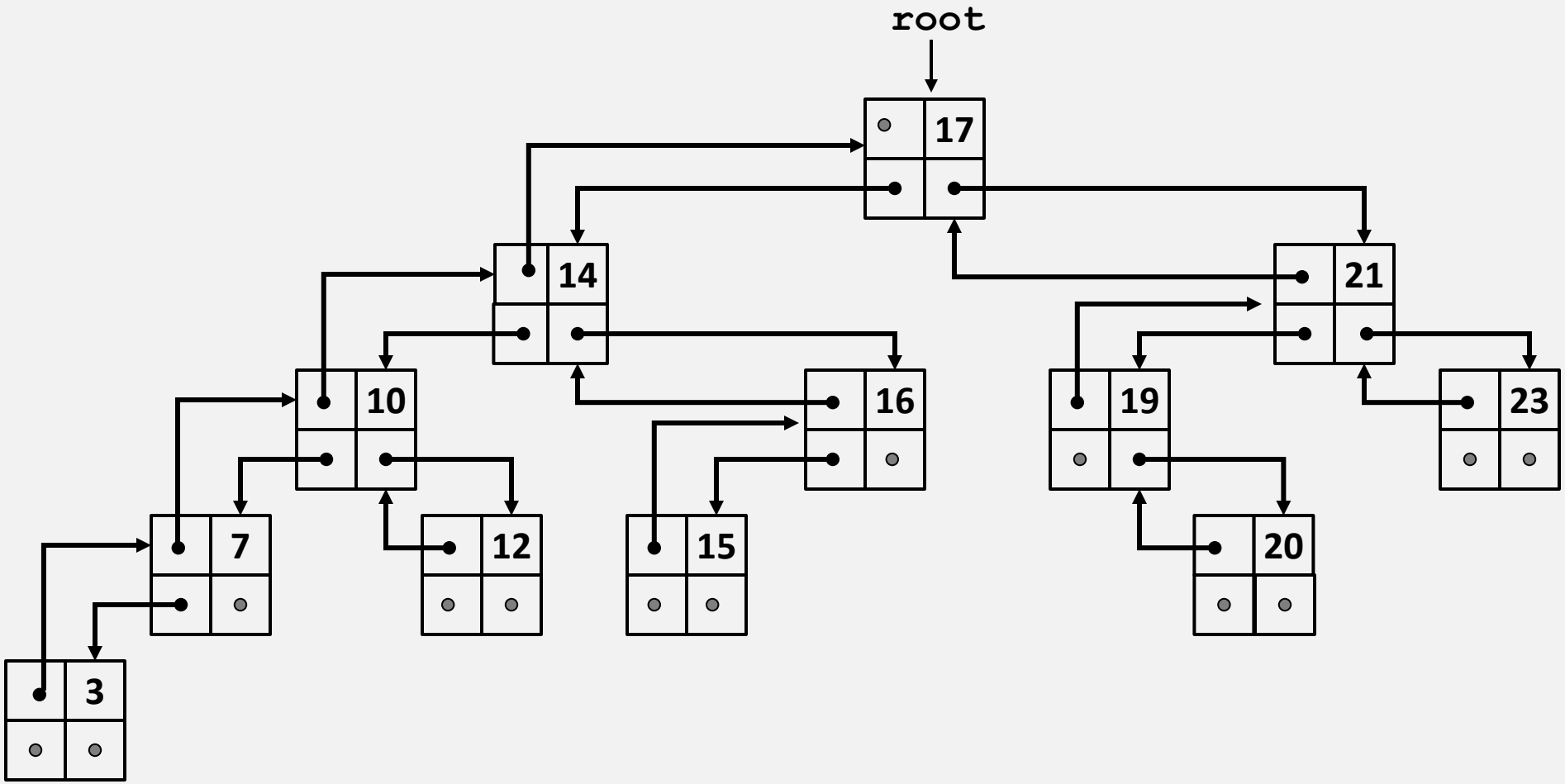
Δυαδικό Δένδρο Αναζήτησης

Το κλειδί οποιουδήποτε εσωτερικού κόμβου είναι μεγαλύτερο (ή ίσο) από όλα τα κλειδιά του αριστερού υποδένδρου και μικρότερο (ή ίσο) από όλα τα κλειδιά του δεξιού υποδένδρου.



Δυαδικό Δένδρο Αναζήτησης

Αν δεν υπάρχει πρόβλημα αποθηκευτικού χώρου που διατίθεται για το δένδρο τότε μπορούμε να έχουμε σε κάθε κόμβο ένα δείκτη προς το γονέα του. Αυτό απλοποιεί την υλοποίηση ορισμένων λειτουργιών.



Δυαδικό Δένδρο Αναζήτησης

```
public class BST<Key extends Comparable<Key>, Item> {
   private Node root;  // ρίζα του δυαδικού δένδρου αναζήτησης

   private class Node {
      private Key key;            // κλειδί
      private Item item;          // αντικείμενο
      private Node left, right;   // σύνδεσμοι σε αριστερό και δεξί παιδί
      private int N;              // πλήθος απογόνων

      public Node(Key key, Item item, int N)
      { this.key = key; this.item = item; this.N = N; }
   }

   private int size(Node x) {
      if (x == null) return 0;
      else return x.N;
   }

   public int size() { return size(root); }

   ...
}
```

Δυαδικό Δένδρο Αναζήτησης

Το κλειδί οποιουδήποτε εσωτερικού κόμβου είναι μεγαλύτερο (ή ίσο) από όλα τα κλειδιά του αριστερού υποδένδρου και μικρότερο (ή ίσο) από όλα τα κλειδιά του δεξιού υποδένδρου.

Αναζήτηση στοιχείου με κλειδί `k`

Ξεκινάμε την αναζήτηση από τον κόμβο `x=root`

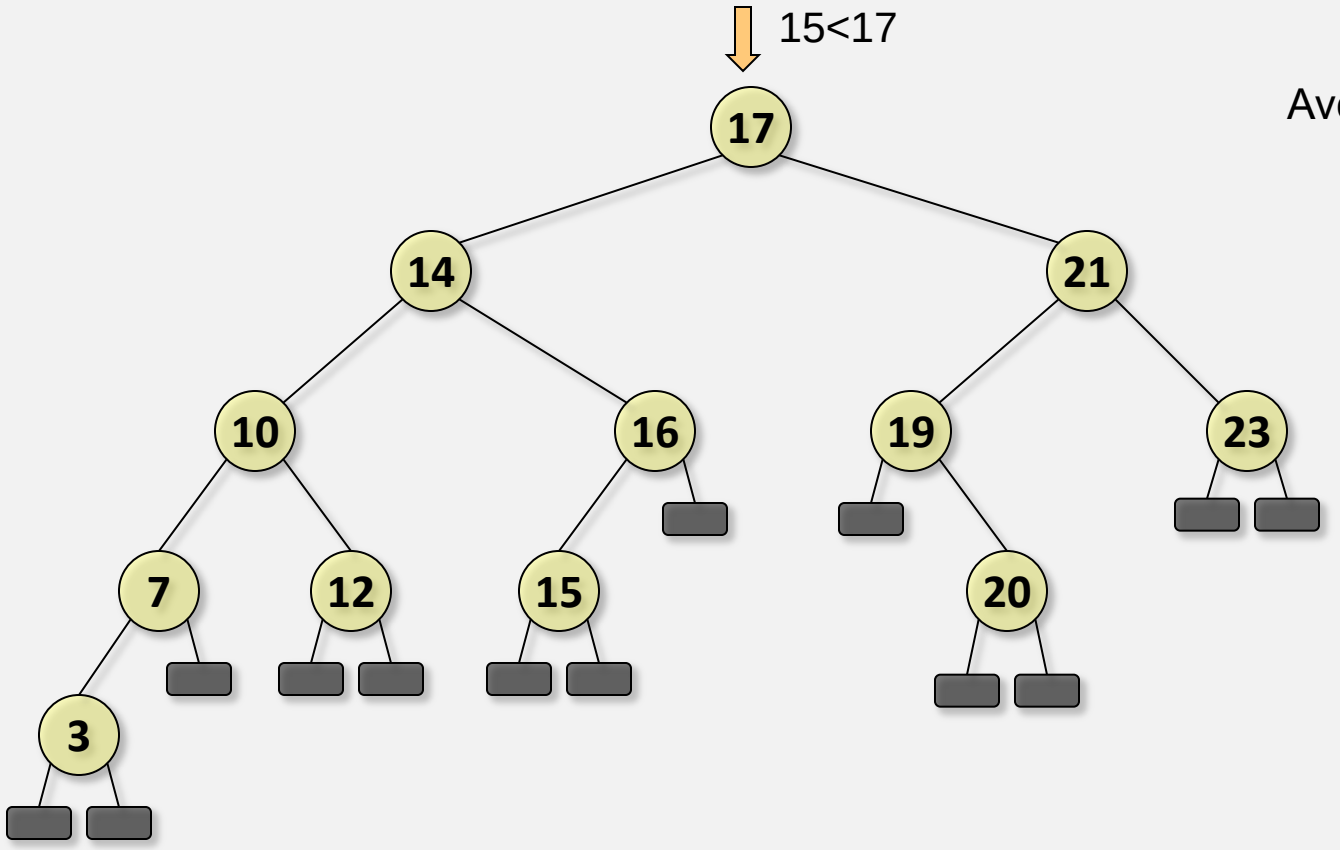
Αν το κλειδί του x είναι `k` ή ο x είναι ο κενός κόμβος τότε επιστρέφουμε το στοιχείο του x

Αν το `k` είναι μικρότερο από το κλειδί του x τότε θέτουμε `x=x.left` και συνεχίζουμε την αναζήτηση στο υποδένδρο με ρίζα x

Αν το `k` είναι μεγαλύτερο από το κλειδί του x τότε θέτουμε `x=x.right` και συνεχίζουμε την αναζήτηση στο υποδένδρο με ρίζα x

Δυαδικό Δένδρο Αναζήτησης

Το κλειδί οποιουδήποτε εσωτερικού κόμβου είναι μεγαλύτερο (ή ίσο) από όλα τα κλειδιά του αριστερού υποδένδρου και μικρότερο (ή ίσο) από όλα τα κλειδιά του δεξιού υποδένδρου.

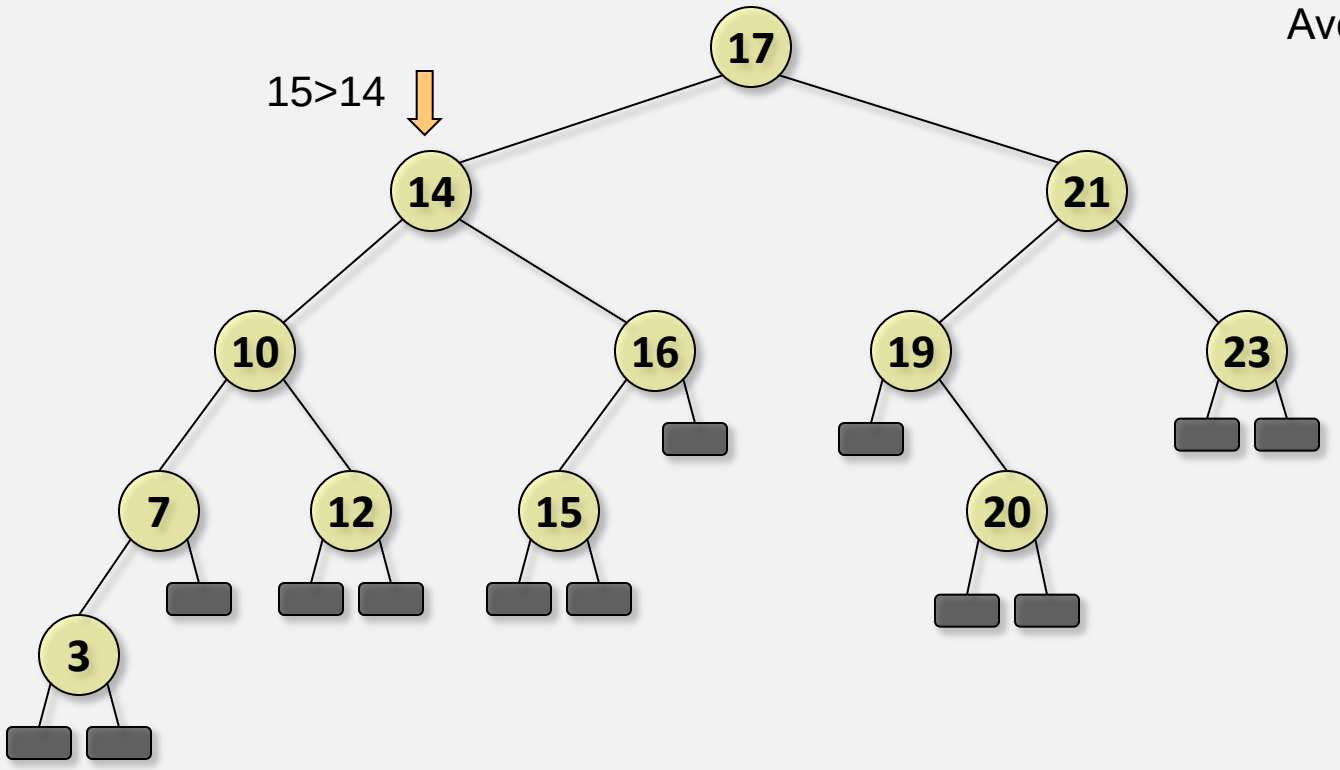


Αναζήτηση 15

Δυαδικό Δένδρο Αναζήτησης

Το κλειδί οποιουδήποτε εσωτερικού κόμβου είναι μεγαλύτερο (ή ίσο) από όλα τα κλειδιά του αριστερού υποδένδρου και μικρότερο (ή ίσο) από όλα τα κλειδιά του δεξιού υποδένδρου.

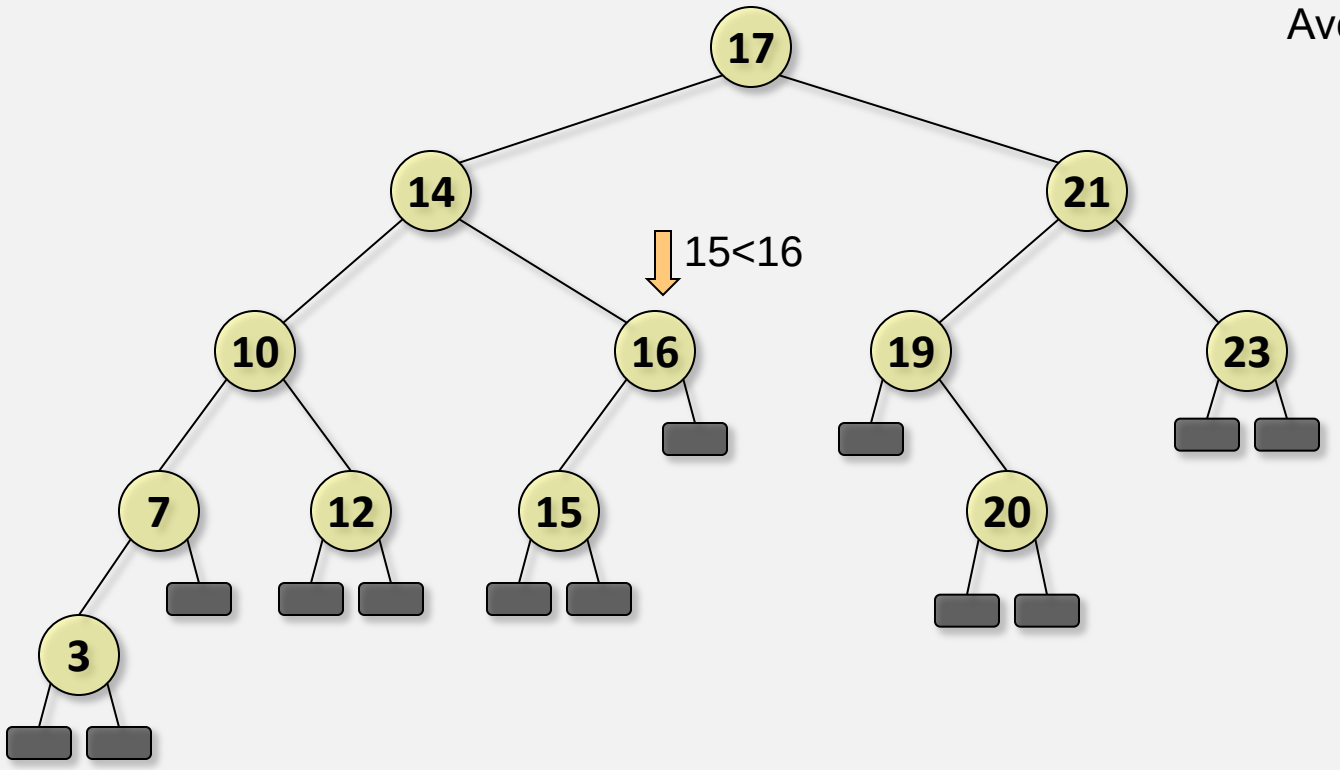
Αναζήτηση 15



Δυαδικό Δένδρο Αναζήτησης

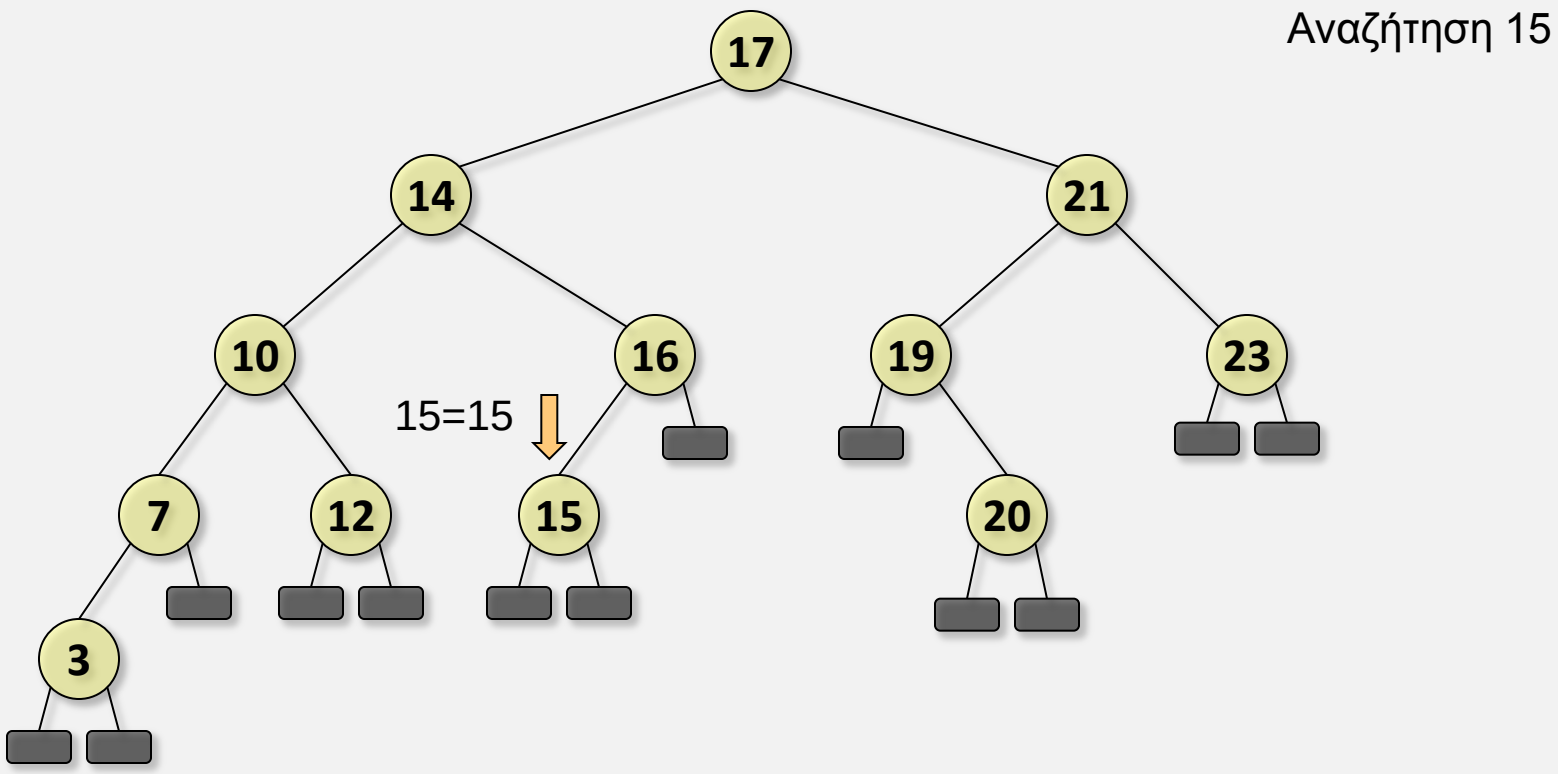
Το κλειδί οποιουδήποτε εσωτερικού κόμβου είναι μεγαλύτερο (ή ίσο) από όλα τα κλειδιά του αριστερού υποδένδρου και μικρότερο (ή ίσο) από όλα τα κλειδιά του δεξιού υποδένδρου.

Αναζήτηση 15



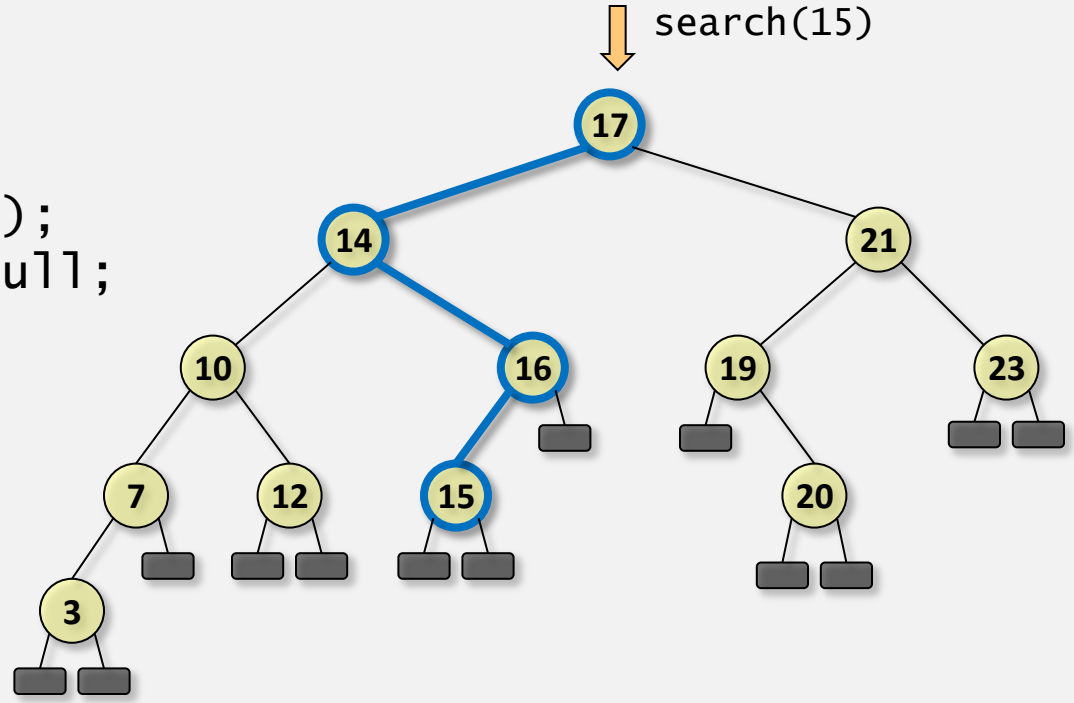
Δυαδικό Δένδρο Αναζήτησης

Το κλειδί οποιουδήποτε εσωτερικού κόμβου είναι μεγαλύτερο (ή ίσο) από όλα τα κλειδιά του αριστερού υποδένδρου και μικρότερο (ή ίσο) από όλα τα κλειδιά του δεξιού υποδένδρου.



Δυαδικό Δένδρο Αναζήτησης

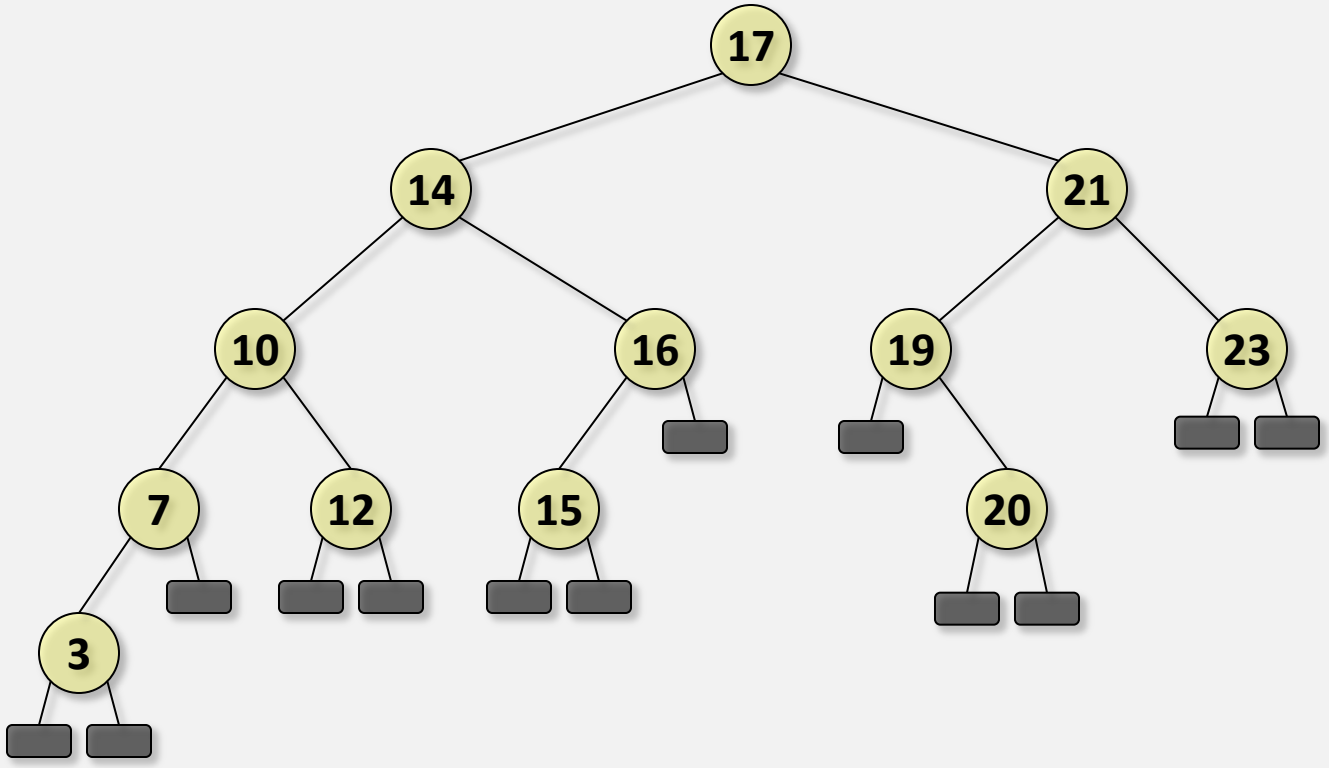
```
private Node search(Node x, Key k)
{
  if (x==null) return null;
  int c = k.compareTo(x.key);  // σύγκριση με το κλειδί του x
  if      (c < 0) return search(x.left,k);
  else if (c > 0) return search(x.right,k);
  else return x;
}

public Item search(Key k)
{
   Node x = search(root, k);
   if (x == null) return null;
   else return x.item;
}
```



Δυαδικό Δένδρο Αναζήτησης

Το κλειδί οποιουδήποτε εσωτερικού κόμβου είναι μεγαλύτερο (ή ίσο) από όλα τα κλειδιά του αριστερού υποδένδρου και μικρότερο (ή ίσο) από όλα τα κλειδιά του δεξιού υποδένδρου.

Εύρεση κόμβων με μέγιστο και ελάχιστο κλειδί



Δυαδικό Δένδρο Αναζήτησης

Το κλειδί οποιουδήποτε εσωτερικού κόμβου είναι μεγαλύτερο (ή ίσο) από όλα τα κλειδιά του αριστερού υποδένδρου και μικρότερο (ή ίσο) από όλα τα κλειδιά του δεξιού υποδένδρου.

Εύρεση κόμβου με μέγιστο κλειδί

Ξεκινάμε την αναζήτηση από τον κόμβο `x=root`

Αν ο x έχει δεξί παιδί τότε θέτουμε `x=x.right` και συνεχίζουμε την αναζήτηση στο υποδένδρο με ρίζα x

Διαφορετικά επιστρέφουμε τον x

Δυαδικό Δένδρο Αναζήτησης

Το κλειδί οποιουδήποτε εσωτερικού κόμβου είναι μεγαλύτερο (ή ίσο) από όλα τα κλειδιά του αριστερού υποδένδρου και μικρότερο (ή ίσο) από όλα τα κλειδιά του δεξιού υποδένδρου.

Εύρεση κόμβου με ελάχιστο κλειδί

Ξεκινάμε την αναζήτηση από τον κόμβο `x=root`

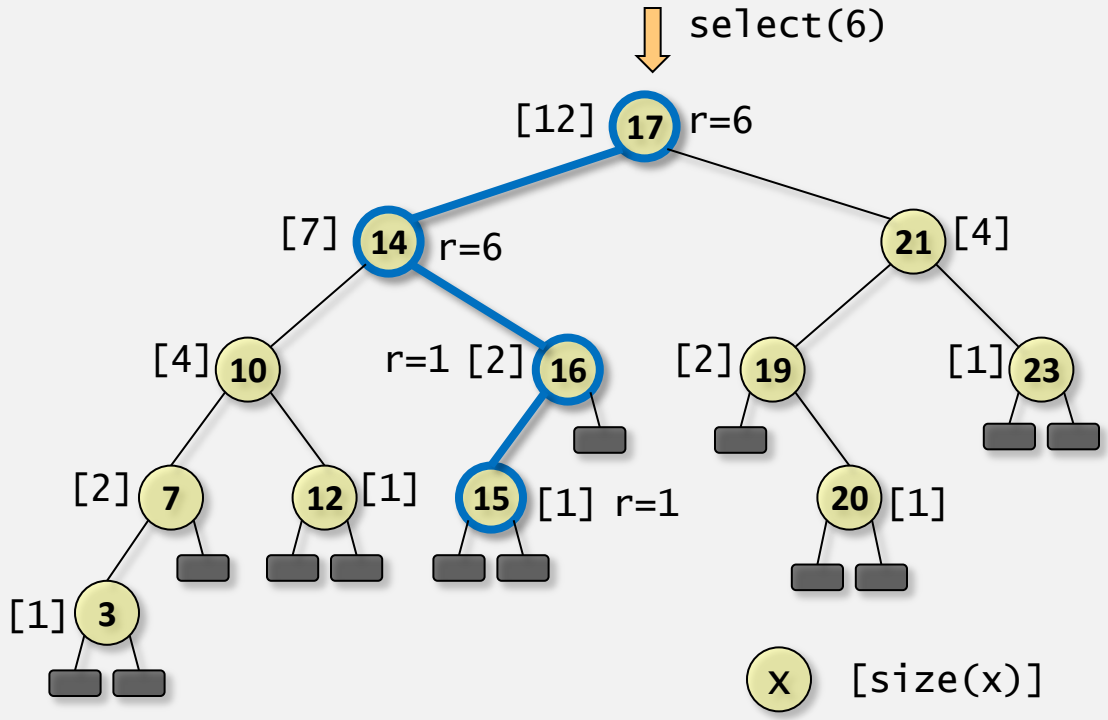
Αν ο x έχει αριστερό παιδί τότε θέτουμε `x=x.left` και συνεχίζουμε την αναζήτηση στο υποδένδρο με ρίζα x

Διαφορετικά επιστρέφουμε τον x

Δυαδικό Δένδρο Αναζήτησης

Επιλογή : Βρίσκει το στοιχείο τάξης r

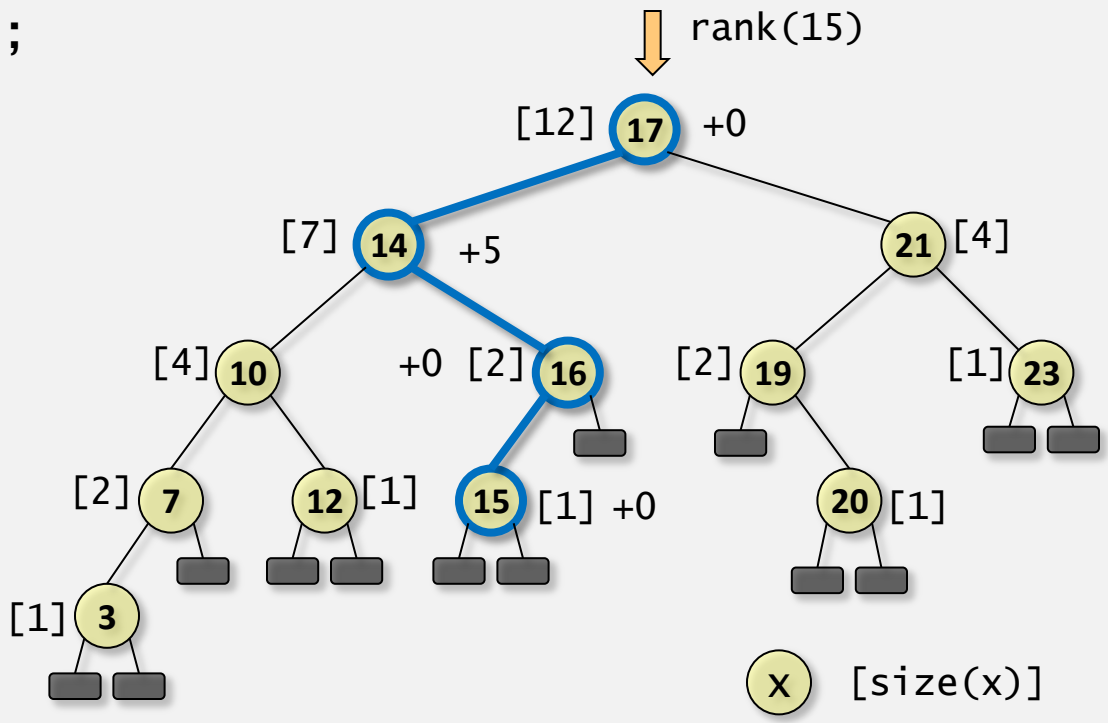
```
private Node select(Node x, int r)
{
  if (x==null) return null;
  int n = size(x.left);
  if      (n > r) return select(x.left,r);
  else if (n < r) return select(x.right,r-n-1);
  else return x;
}

public Key select(int r)
{
  Node x = select(root,r);
  return x.key;
}
```



Δυαδικό Δένδρο Αναζήτησης

Τάξη : Βρίσκει πόσα κλειδιά είναι μεγαλύτερα του k

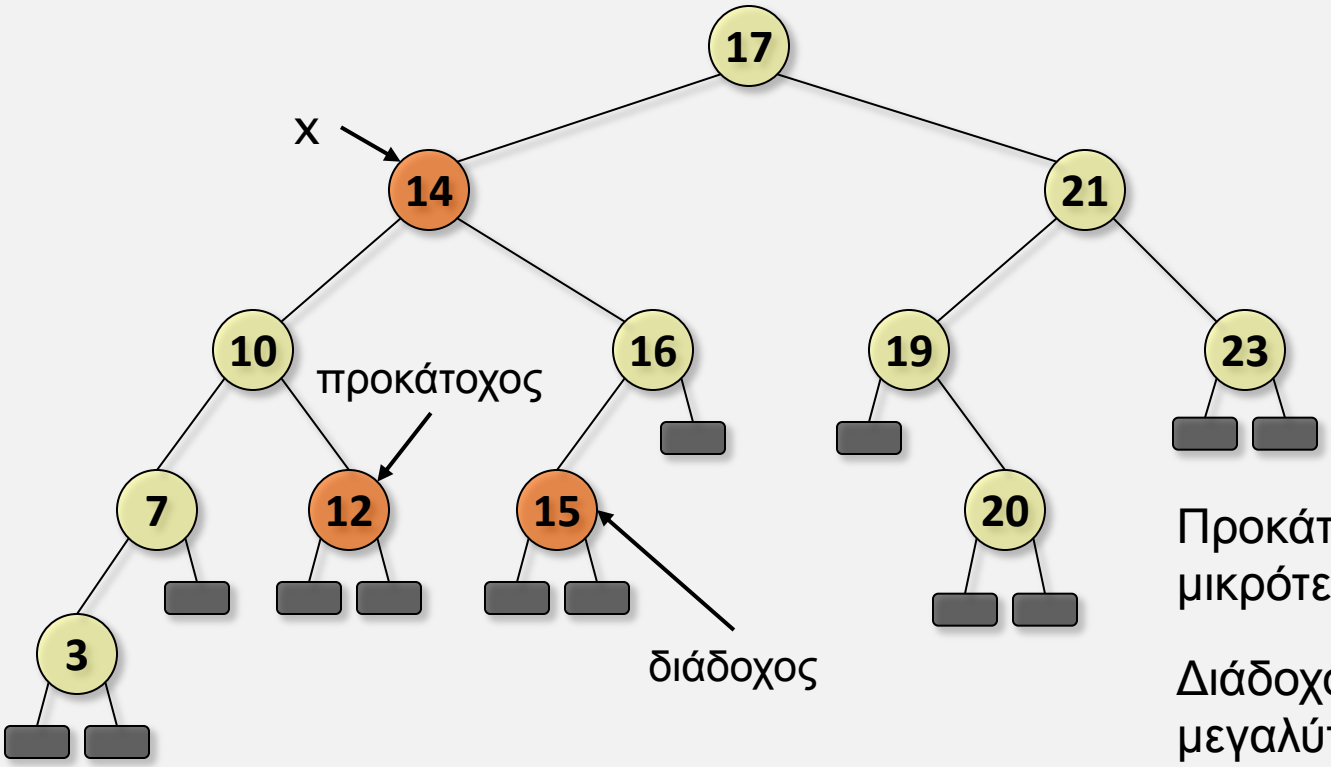
```
private int rank(Key k, Node x)
{
  if (x==null) return 0;
  int c = k.compareTo(x.key);  // σύγκριση με το κλειδί του x
  if      (c < 0) return rank(k,x.left);
  else if (c > 0) return 1 + size(x.left) + rank(k,x.right);
  else return size(x.left);

}

public int rank(Key k)
{
  return rank(k,root);
}
```



Δυαδικό Δένδρο Αναζήτησης

Το κλειδί οποιουδήποτε εσωτερικού κόμβου είναι μεγαλύτερο (ή ίσο) από όλα τα κλειδιά του αριστερού υποδένδρου και μικρότερο (ή ίσο) από όλα τα κλειδιά του δεξιού υποδένδρου.

Εύρεση προκάτοχου και διάδοχου κόμβου του x



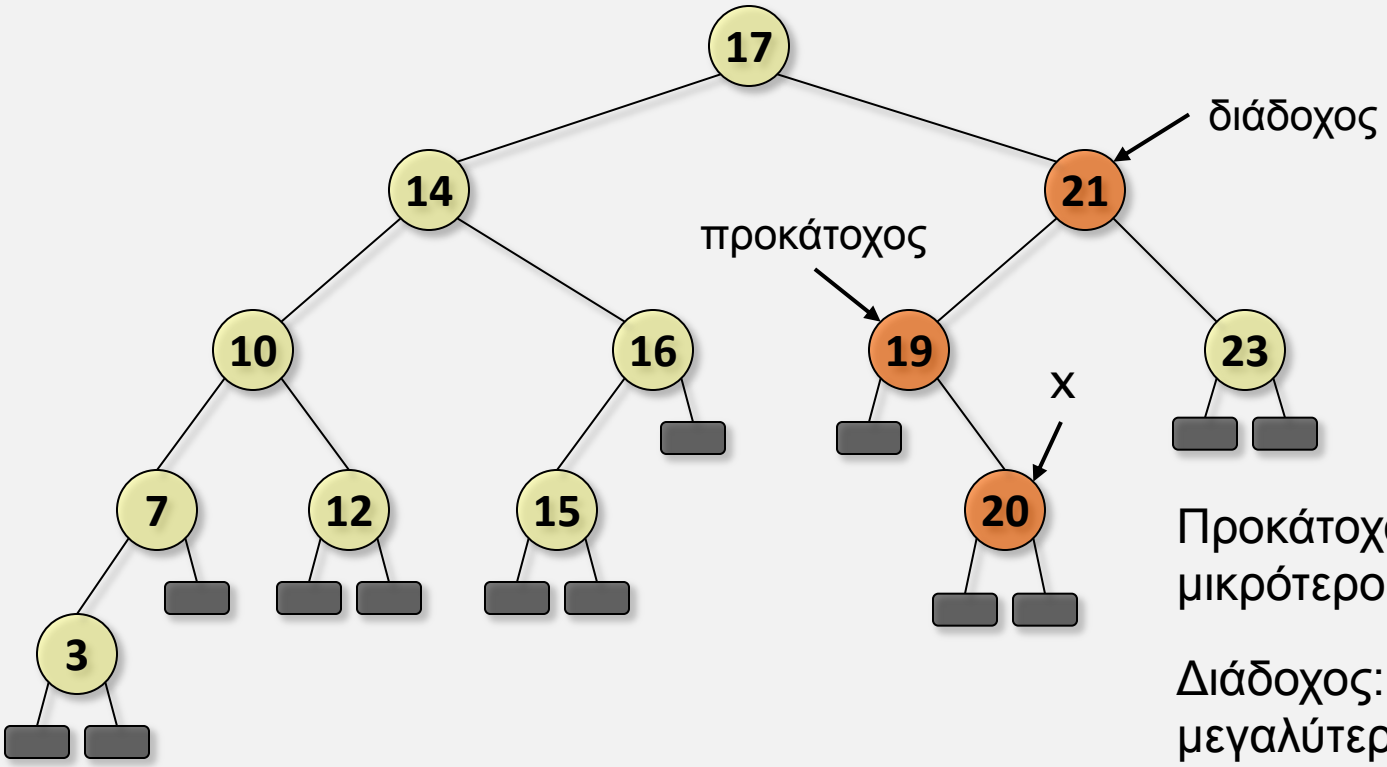
Προκάτοχος: έχει το αμέσως μικρότερο κλειδί

Διάδοχος: έχει το αμέσως μεγαλύτερο κλειδί

Δυαδικό Δένδρο Αναζήτησης

Το κλειδί οποιουδήποτε εσωτερικού κόμβου είναι μεγαλύτερο (ή ίσο) από όλα τα κλειδιά του αριστερού υποδένδρου και μικρότερο (ή ίσο) από όλα τα κλειδιά του δεξιού υποδένδρου.

Εύρεση προκάτοχου και διάδοχου κόμβου του x



Προκάτοχος: έχει το αμέσως μικρότερο κλειδί

Διάδοχος: έχει το αμέσως μεγαλύτερο κλειδί

Δυαδικό Δένδρο Αναζήτησης

Το κλειδί οποιουδήποτε εσωτερικού κόμβου είναι μεγαλύτερο (ή ίσο) από όλα τα κλειδιά του αριστερού υποδένδρου και μικρότερο (ή ίσο) από όλα τα κλειδιά του δεξιού υποδένδρου.

Εύρεση διάδοχου κόμβου του x

Αν ο x έχει δεξί παιδί τότε επιστρέφουμε τον κόμβο με το ελάχιστο κλειδί στο υποδένδρο με ρίζα `x.right`

Διαφορετικά επιστρέφουμε τον κατώτερο πρόγονο y του x το αριστερό παιδί του οποίου (`y.left`) είναι επίσης πρόγονος του x

Δυαδικό Δένδρο Αναζήτησης

Το κλειδί οποιουδήποτε εσωτερικού κόμβου είναι μεγαλύτερο (ή ίσο) από όλα τα κλειδιά του αριστερού υποδένδρου και μικρότερο (ή ίσο) από όλα τα κλειδιά του δεξιού υποδένδρου.

Εισαγωγή νέου στοιχείου με κλειδί k (που δεν υπάρχει ήδη στο δένδρο)

Αναζητούμε το κλειδί k στο δένδρο και βρίσκουμε τον κόμβο y που είναι γονέας του κενού κόμβου στον οποίο καταλήγει η αναζήτηση

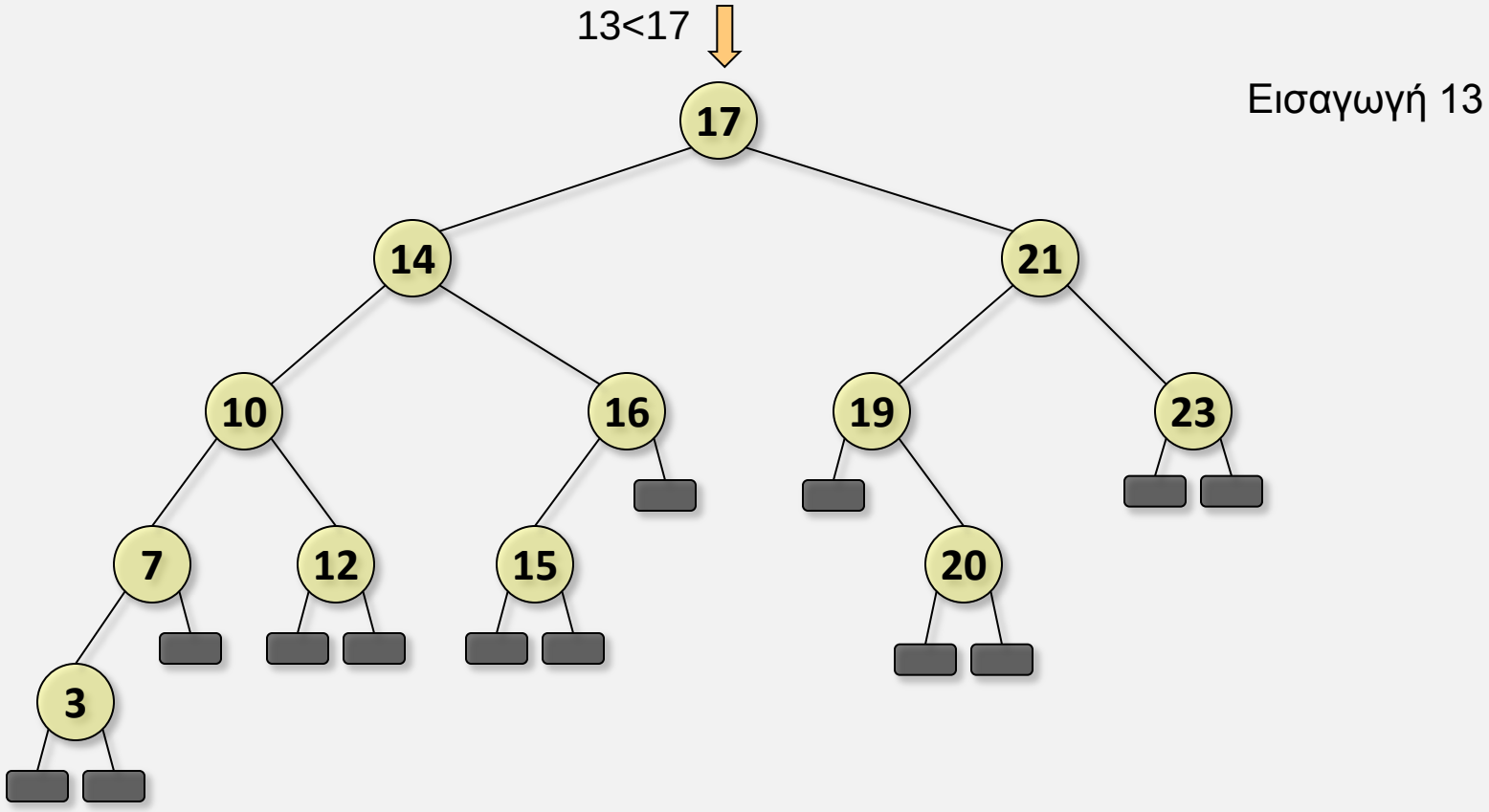
Δημιουργούμε νέο κόμβο x με κενά παιδιά και αποθηκεύουμε το νέο στοιχείο στον x

Αν ο y δεν υπάρχει τότε θέτουμε `root=x`

Διαφορετικά, κάνουμε τον x αριστερό παιδί του y αν η αναζήτηση ακολούθησε τον αριστερό σύνδεσμο του y. Διαφορετικά κάνουμε τον x δεξί παιδί του y

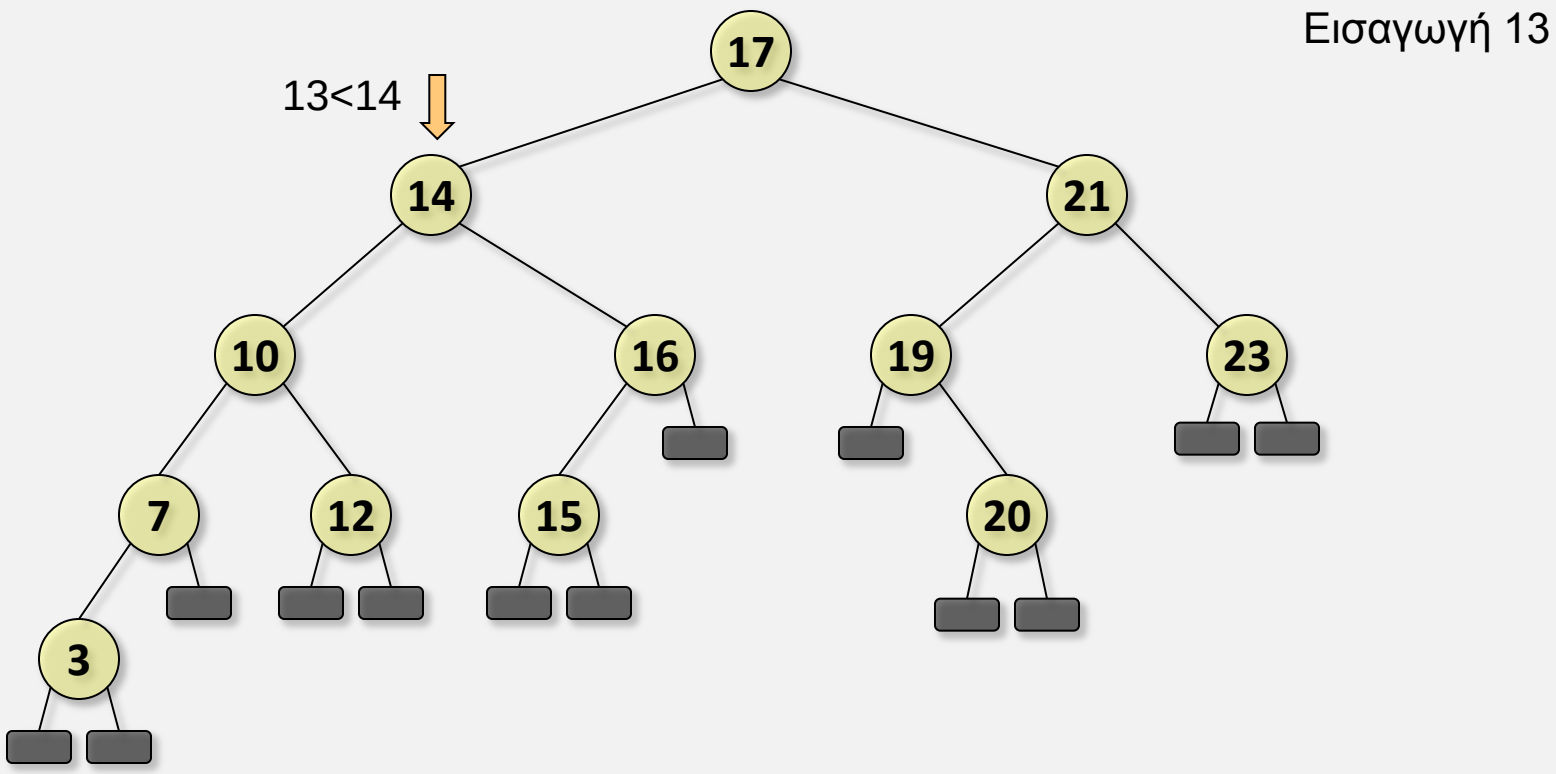
Δυαδικό Δένδρο Αναζήτησης

Το κλειδί οποιουδήποτε εσωτερικού κόμβου είναι μεγαλύτερο (ή ίσο) από όλα τα κλειδιά του αριστερού υποδένδρου και μικρότερο (ή ίσο) από όλα τα κλειδιά του δεξιού υποδένδρου.



Δυαδικό Δένδρο Αναζήτησης

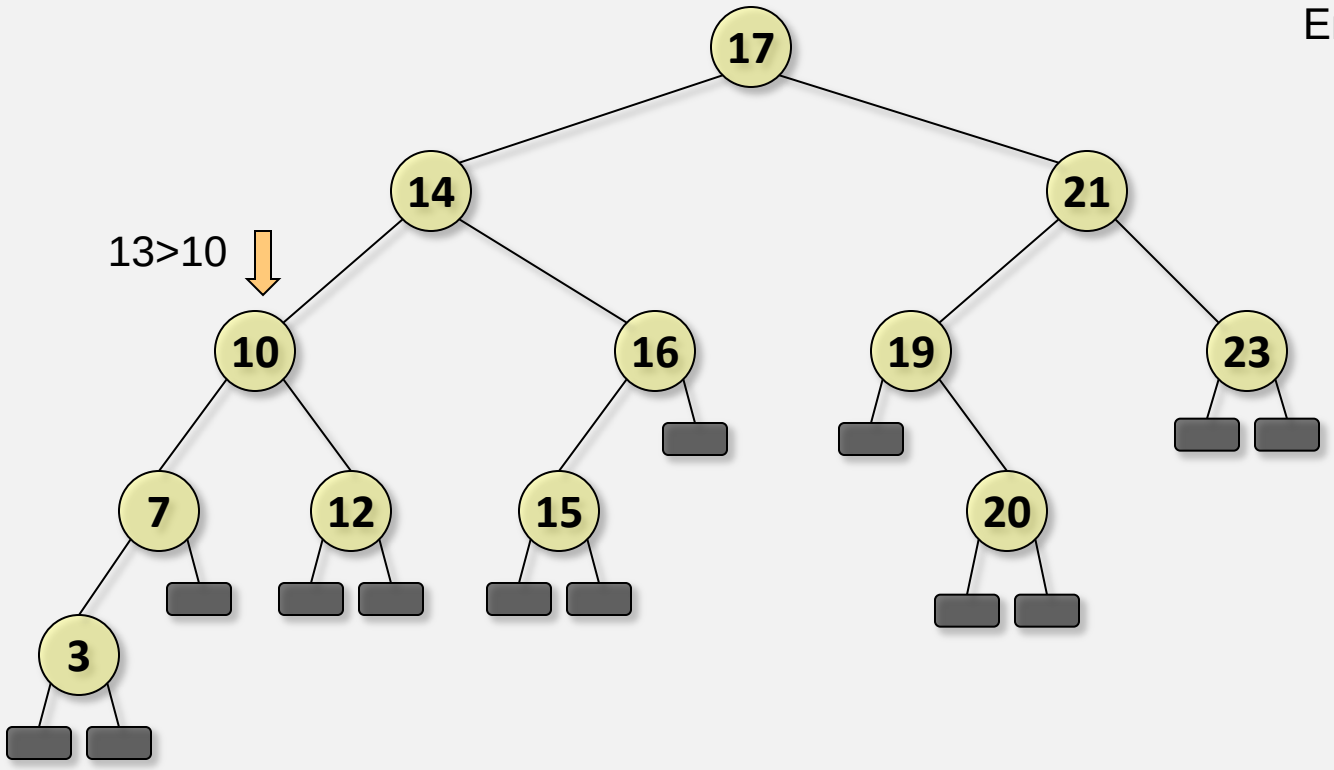
Το κλειδί οποιουδήποτε εσωτερικού κόμβου είναι μεγαλύτερο (ή ίσο) από όλα τα κλειδιά του αριστερού υποδένδρου και μικρότερο (ή ίσο) από όλα τα κλειδιά του δεξιού υποδένδρου.



Δυαδικό Δένδρο Αναζήτησης

Το κλειδί οποιουδήποτε εσωτερικού κόμβου είναι μεγαλύτερο (ή ίσο) από όλα τα κλειδιά του αριστερού υποδένδρου και μικρότερο (ή ίσο) από όλα τα κλειδιά του δεξιού υποδένδρου.

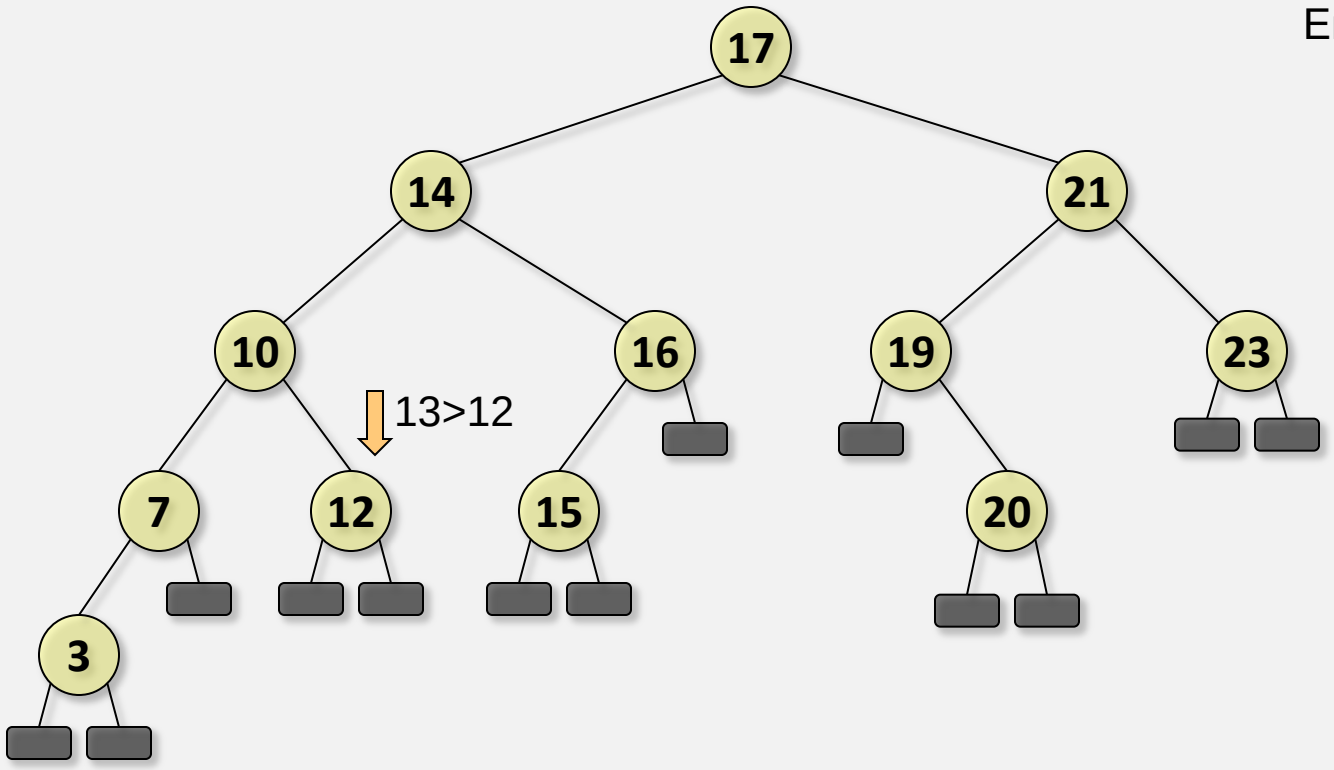
Εισαγωγή 13



Δυαδικό Δένδρο Αναζήτησης

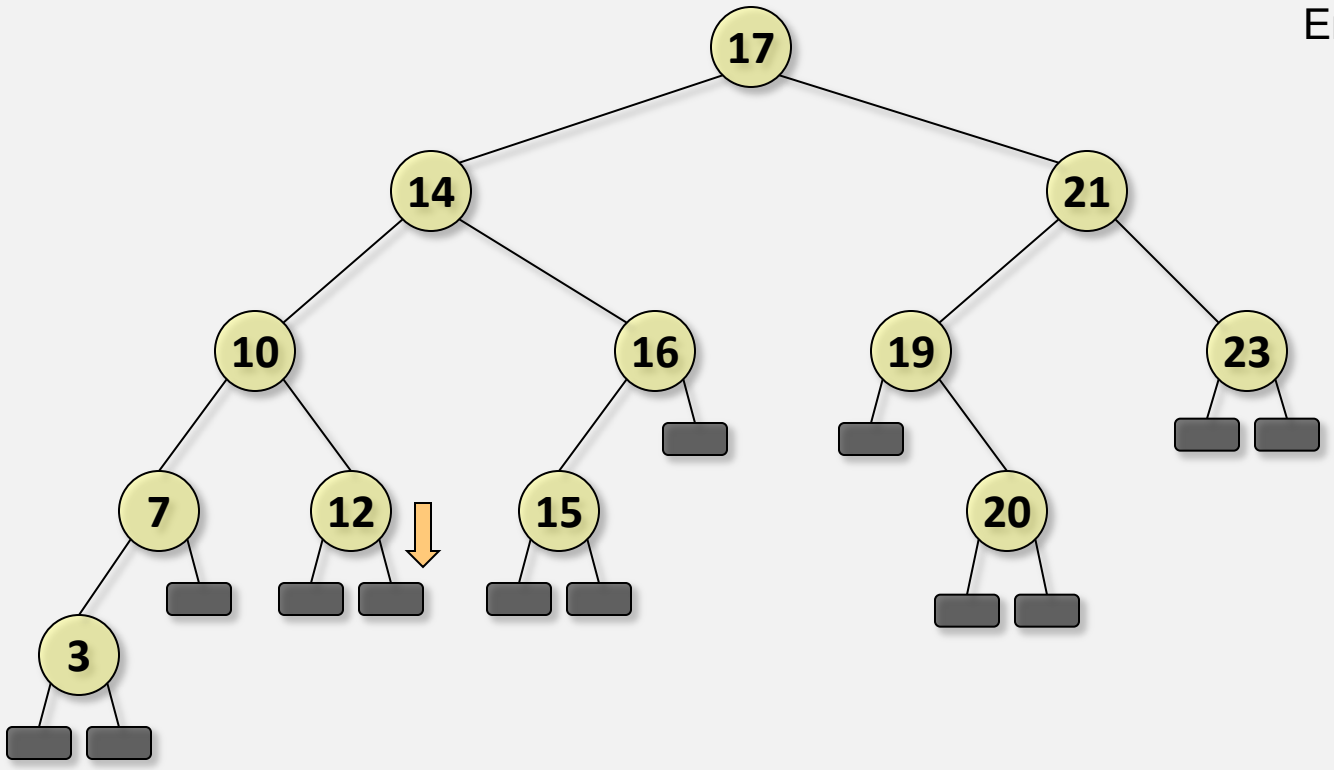
Το κλειδί οποιουδήποτε εσωτερικού κόμβου είναι μεγαλύτερο (ή ίσο) από όλα τα κλειδιά του αριστερού υποδένδρου και μικρότερο (ή ίσο) από όλα τα κλειδιά του δεξιού υποδένδρου.

Εισαγωγή 13



Δυαδικό Δένδρο Αναζήτησης

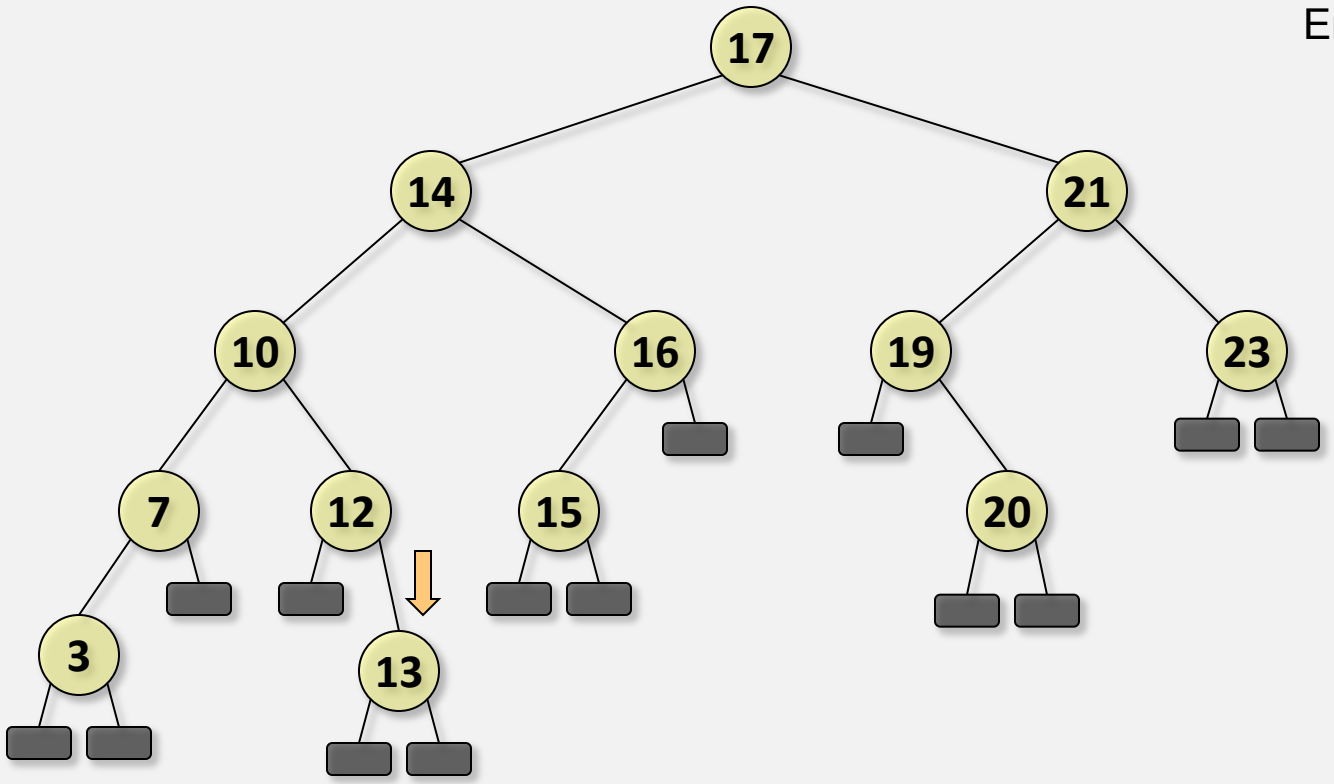
Το κλειδί οποιουδήποτε εσωτερικού κόμβου είναι μεγαλύτερο (ή ίσο) από όλα τα κλειδιά του αριστερού υποδένδρου και μικρότερο (ή ίσο) από όλα τα κλειδιά του δεξιού υποδένδρου.



Δυαδικό Δένδρο Αναζήτησης

Το κλειδί οποιουδήποτε εσωτερικού κόμβου είναι μεγαλύτερο (ή ίσο) από όλα τα κλειδιά του αριστερού υποδένδρου και μικρότερο (ή ίσο) από όλα τα κλειδιά του δεξιού υποδένδρου.

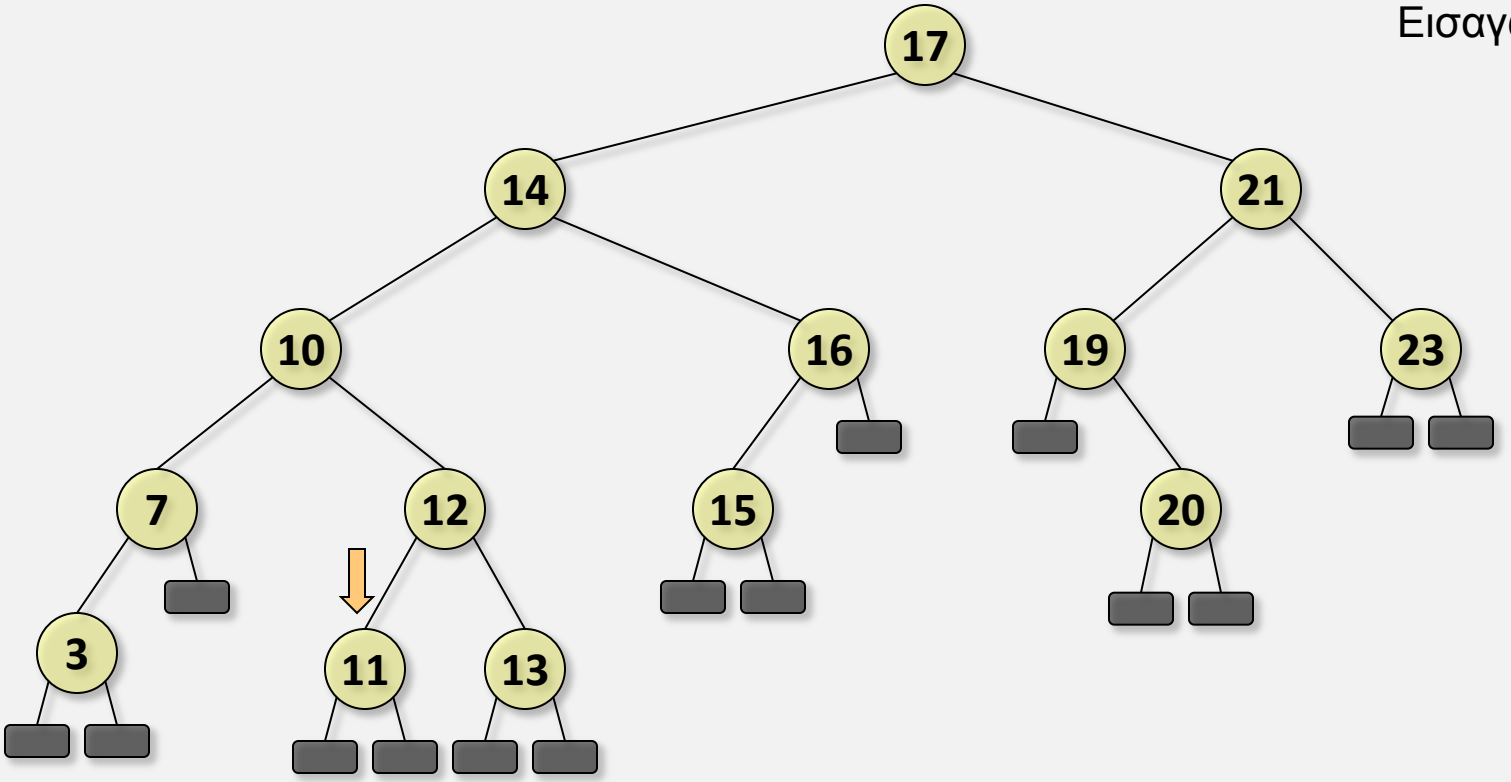
Εισαγωγή 13



Δυαδικό Δένδρο Αναζήτησης

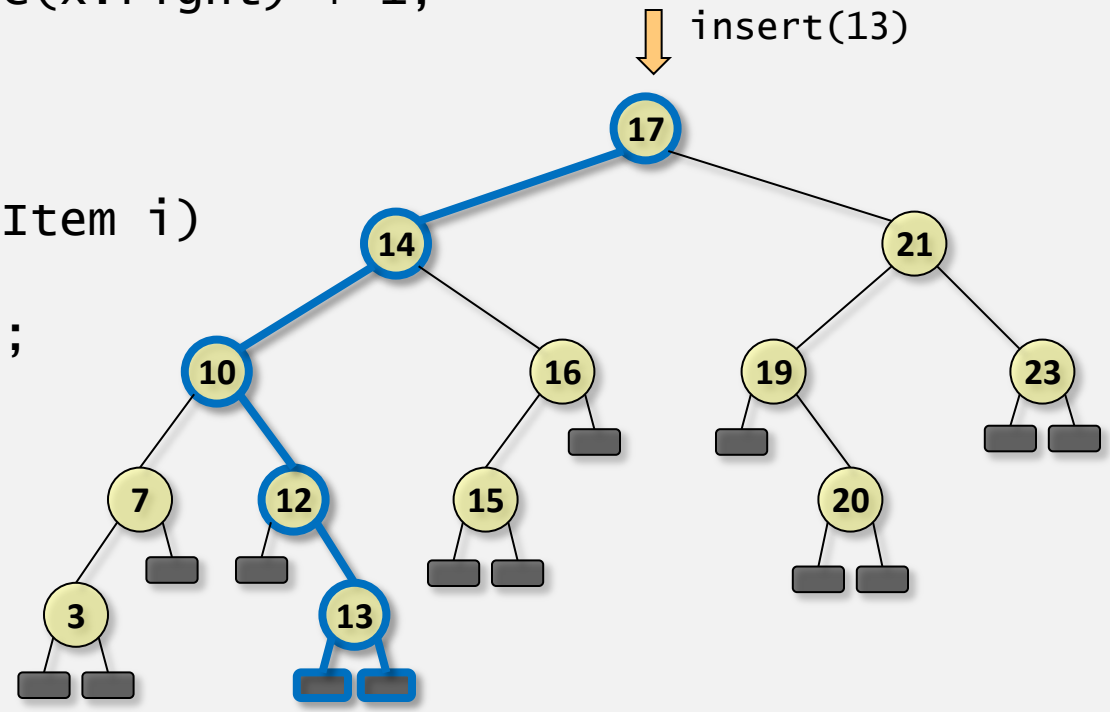
Το κλειδί οποιουδήποτε εσωτερικού κόμβου είναι μεγαλύτερο (ή ίσο) από όλα τα κλειδιά του αριστερού υποδένδρου και μικρότερο (ή ίσο) από όλα τα κλειδιά του δεξιού υποδένδρου.

Εισαγωγή 11



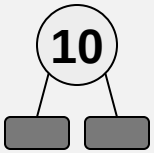
Δυαδικό Δένδρο Αναζήτησης

```
private Node insert(Node x, Key k, Item i)
{
  if (x==null) return new Node(k,i,1);
  int c = k.compareTo(x.key);  // σύγκριση με το κλειδί του x
  if      (c < 0) x.left  = insert(x.left, k,i);
  else if (c > 0) x.right = insert(x.right,k,i);
  else x.item = i;  // το αντικείμενο i έχει κλειδί k==x.key
  x.N = size(x.left) + size(x.right) + 1;
  return x;
}

public void insert(Key k, Item i)
{
   root = insert(root,k,i);
}
```



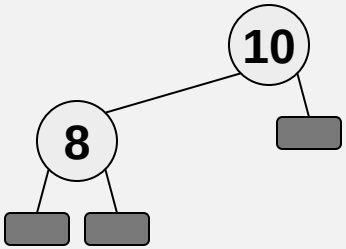
Δυαδικό Δένδρο Αναζήτησης

Εισαγωγή 10



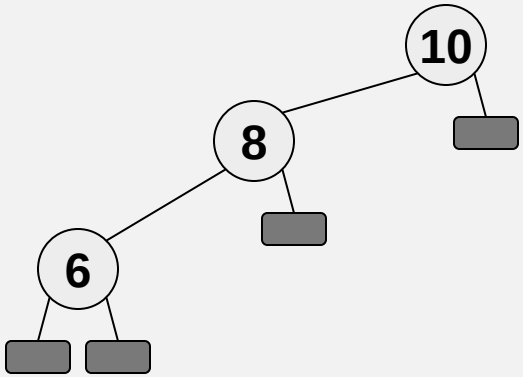
Δυαδικό Δένδρο Αναζήτησης

Εισαγωγή 10, 8



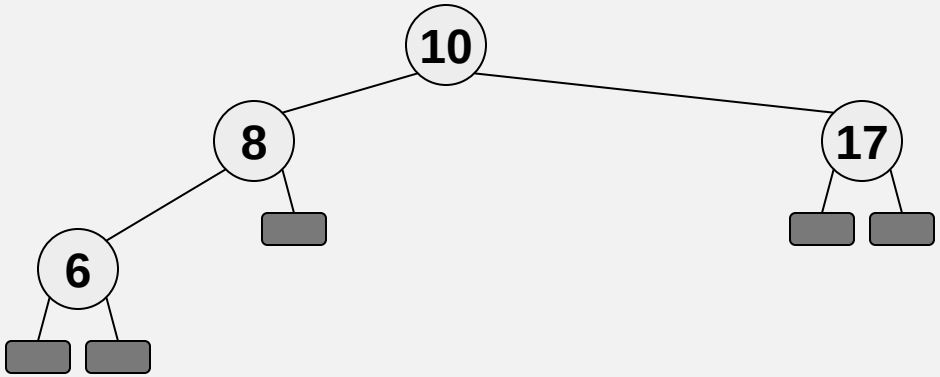
Δυαδικό Δένδρο Αναζήτησης

Εισαγωγή 10, 8, 6



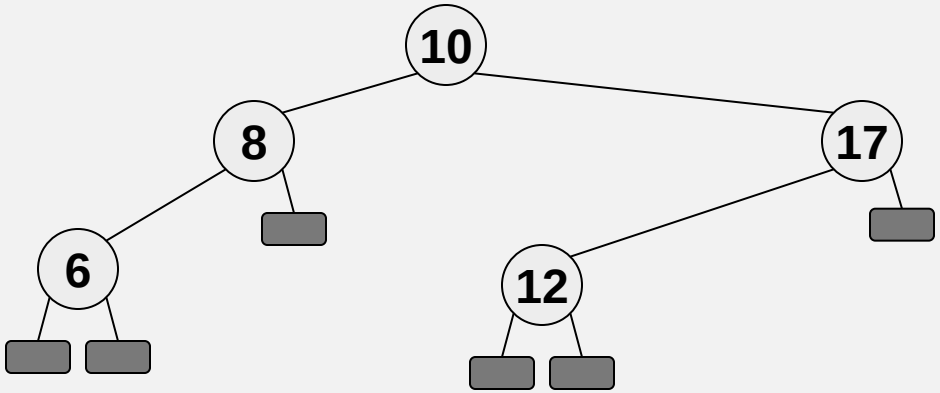
Δυαδικό Δένδρο Αναζήτησης

Εισαγωγή 10, 8, 6, 17



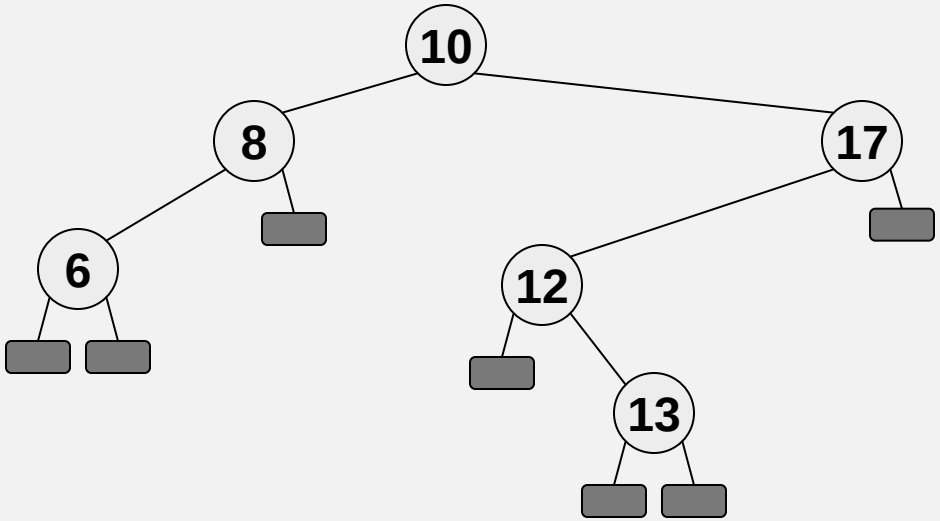
Δυαδικό Δένδρο Αναζήτησης

Εισαγωγή 10, 8, 6, 17, 12



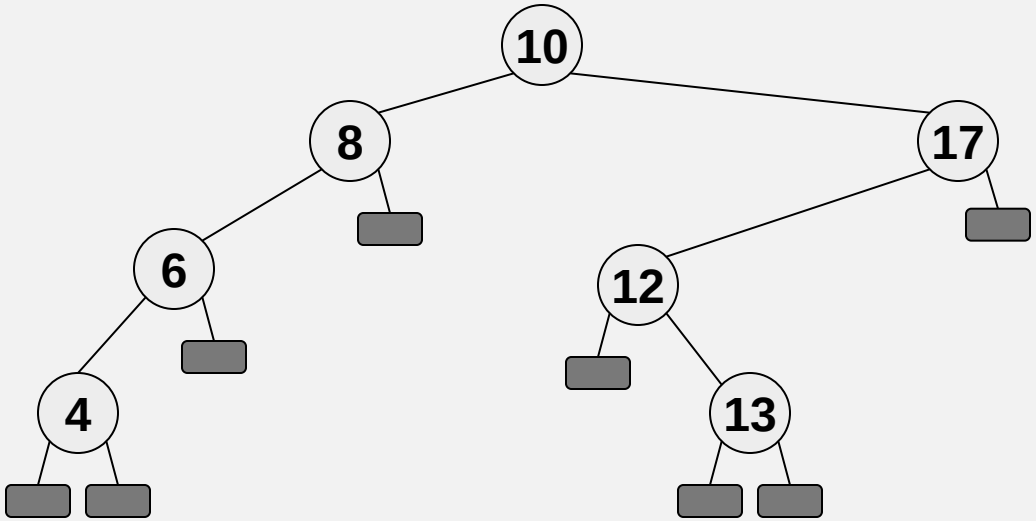
Δυαδικό Δένδρο Αναζήτησης

Εισαγωγή 10, 8, 6, 17, 12, 13



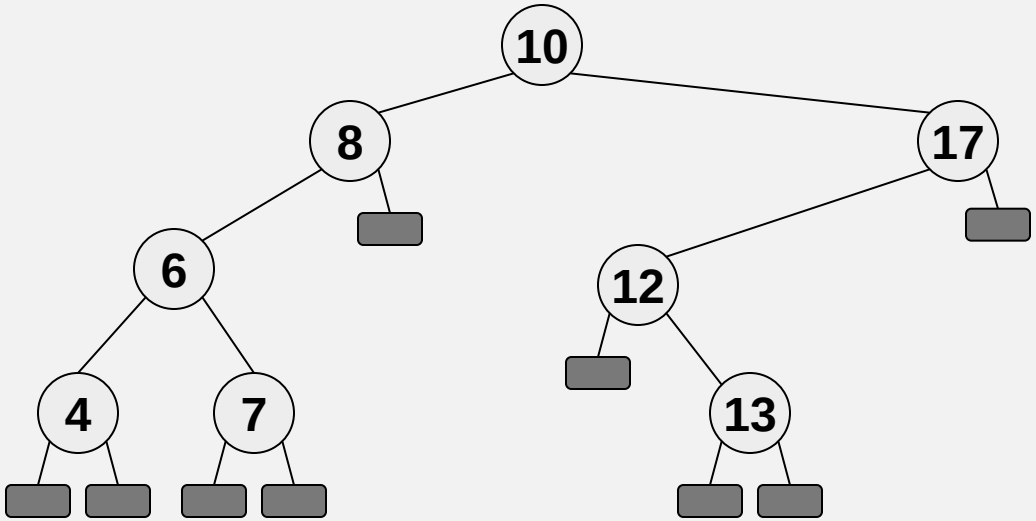
Δυαδικό Δένδρο Αναζήτησης

Εισαγωγή 10, 8, 6, 17, 12, 13, 4



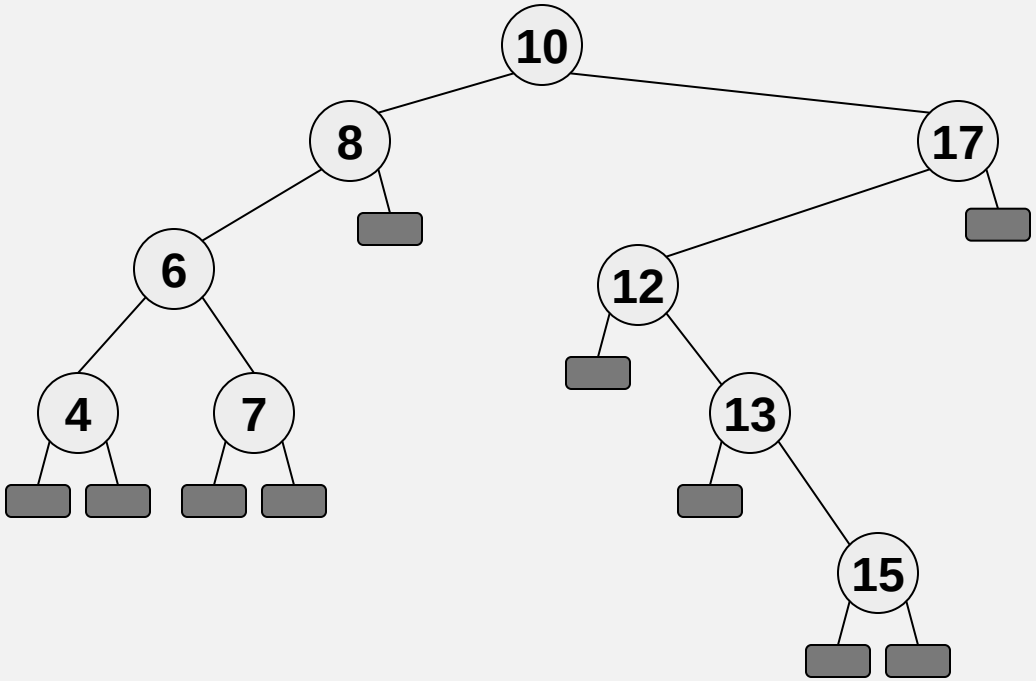
Δυαδικό Δένδρο Αναζήτησης

Εισαγωγή 10, 8, 6, 17, 12, 13, 4, 7



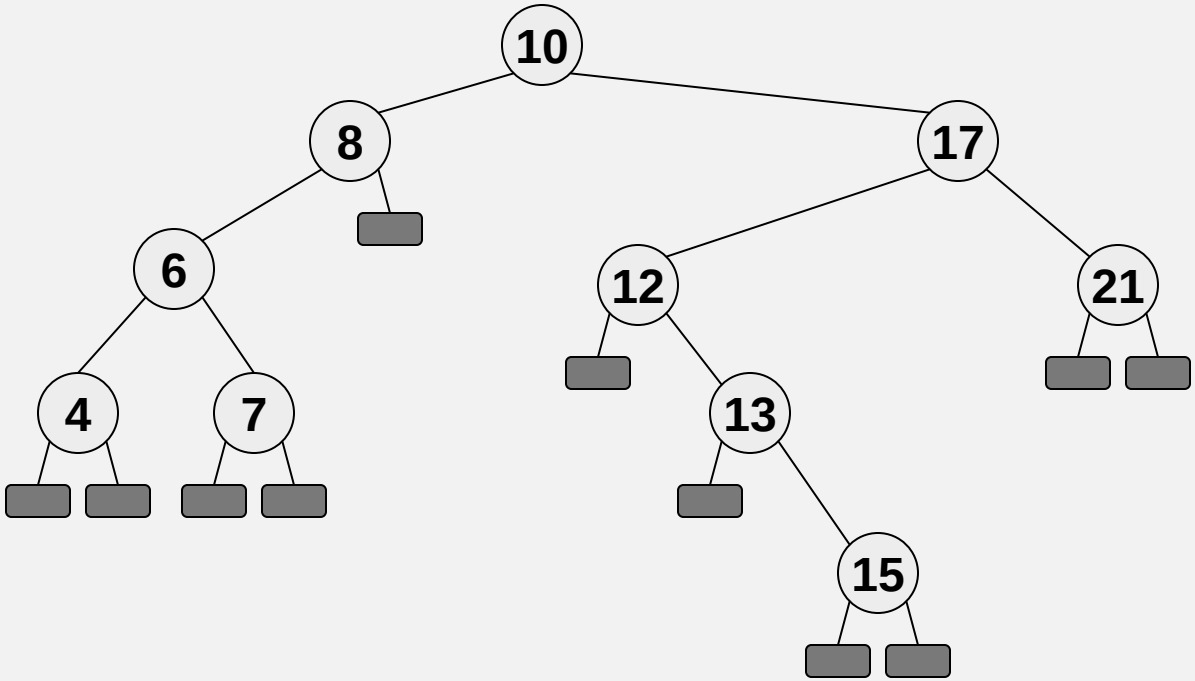
Δυαδικό Δένδρο Αναζήτησης

Εισαγωγή 10, 8, 6, 17, 12, 13, 4, 7, 15



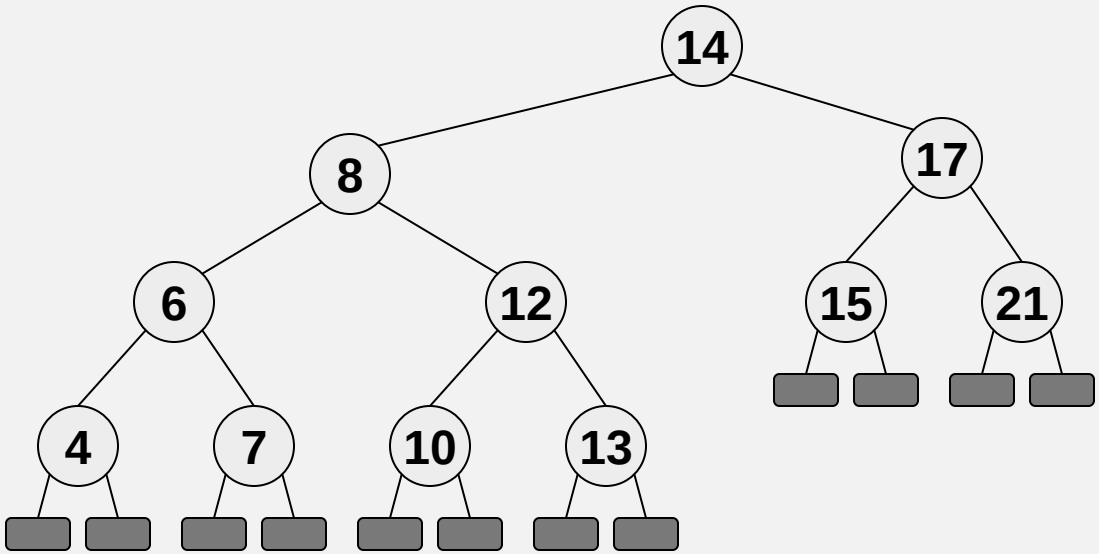
Δυαδικό Δένδρο Αναζήτησης

Εισαγωγή 10, 8, 6, 17, 12, 13, 4, 7, 15, 21



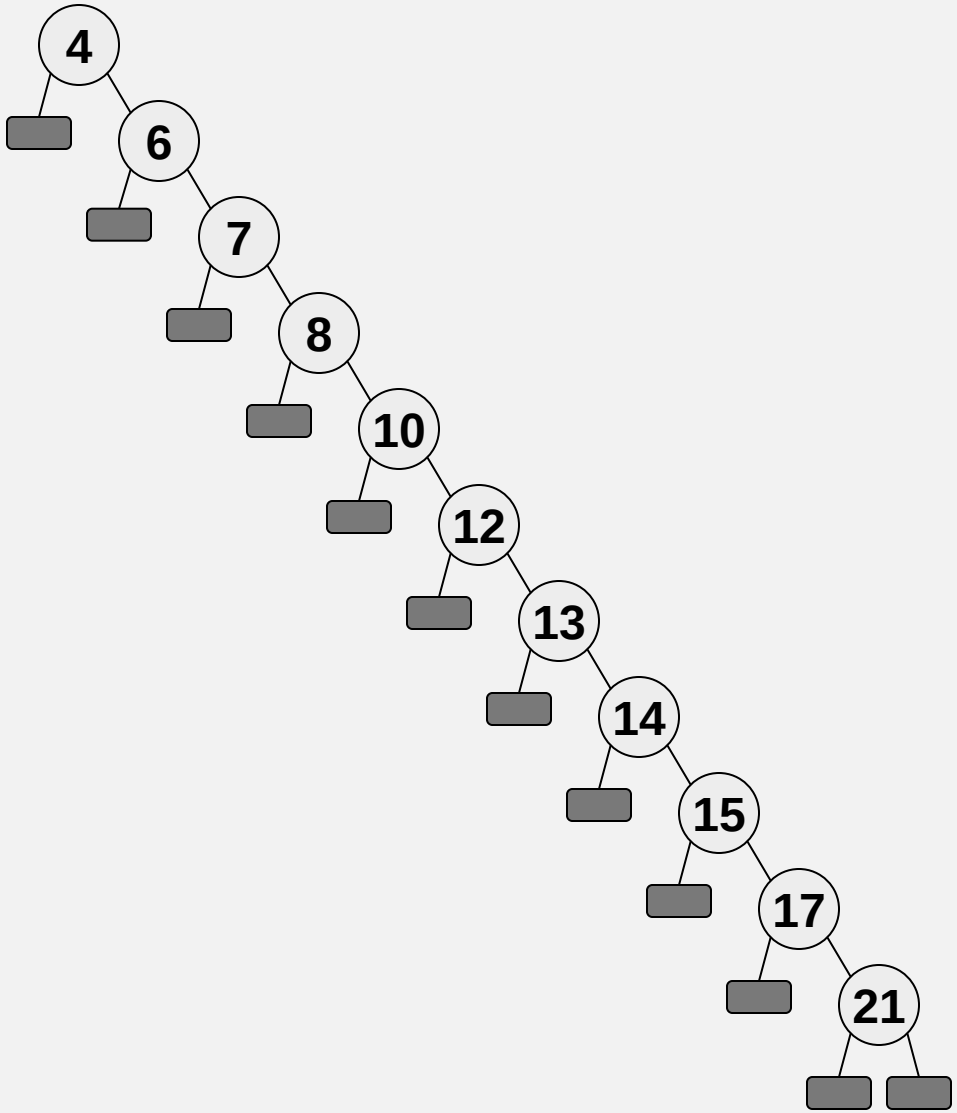
Δυαδικό Δένδρο Αναζήτησης

Εισαγωγή 14, 8, 17, 6, 12, 15, 21, 4, 7, 10, 13



Δυαδικό Δένδρο Αναζήτησης

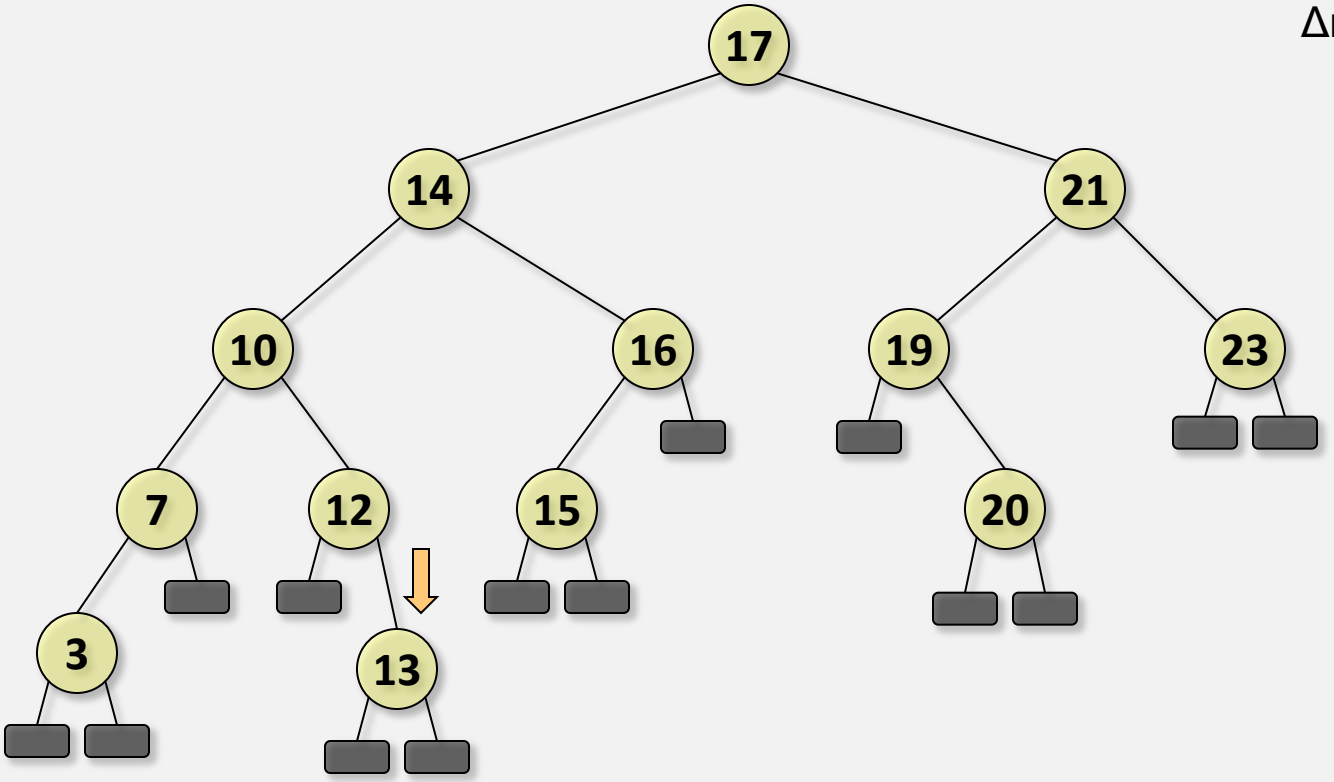
Εισαγωγή 4, 6, 7, 8, 10, 12, 13, 14, 15, 17, 21



Δυαδικό Δένδρο Αναζήτησης

Το κλειδί οποιουδήποτε εσωτερικού κόμβου είναι μεγαλύτερο (ή ίσο) από όλα τα κλειδιά του αριστερού υποδένδρου και μικρότερο (ή ίσο) από όλα τα κλειδιά του δεξιού υποδένδρου.

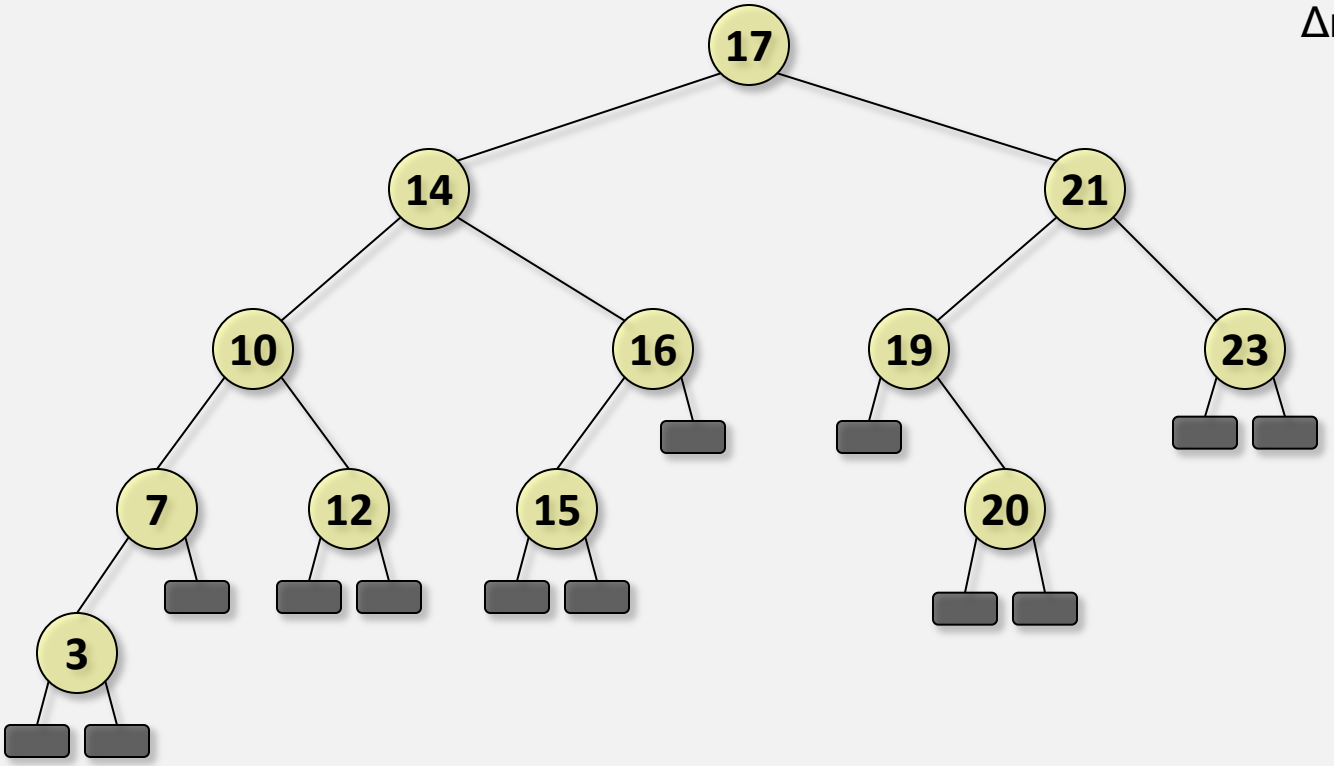
Διαγραφή 13



Δυαδικό Δένδρο Αναζήτησης

Το κλειδί οποιουδήποτε εσωτερικού κόμβου είναι μεγαλύτερο (ή ίσο) από όλα τα κλειδιά του αριστερού υποδένδρου και μικρότερο (ή ίσο) από όλα τα κλειδιά του δεξιού υποδένδρου.

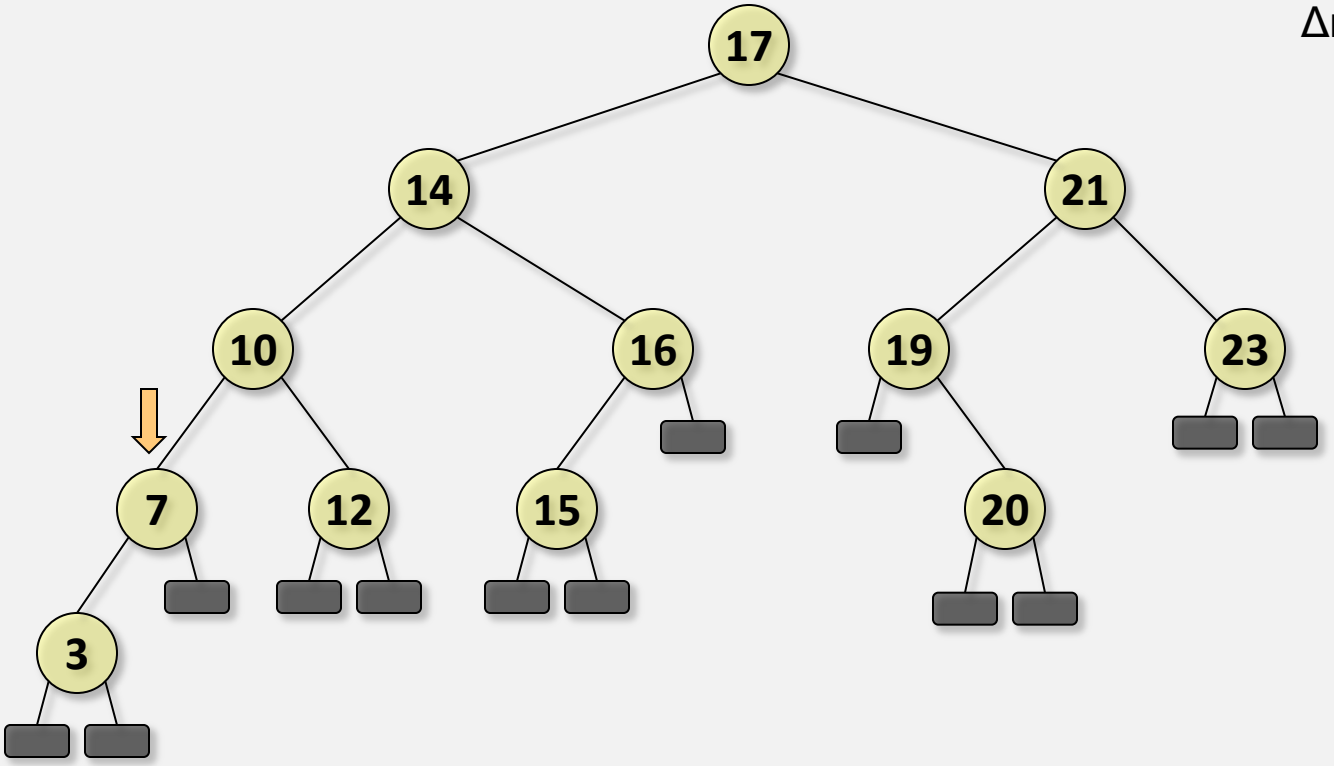
Διαγραφή 13



Δυαδικό Δένδρο Αναζήτησης

Το κλειδί οποιουδήποτε εσωτερικού κόμβου είναι μεγαλύτερο (ή ίσο) από όλα τα κλειδιά του αριστερού υποδένδρου και μικρότερο (ή ίσο) από όλα τα κλειδιά του δεξιού υποδένδρου.

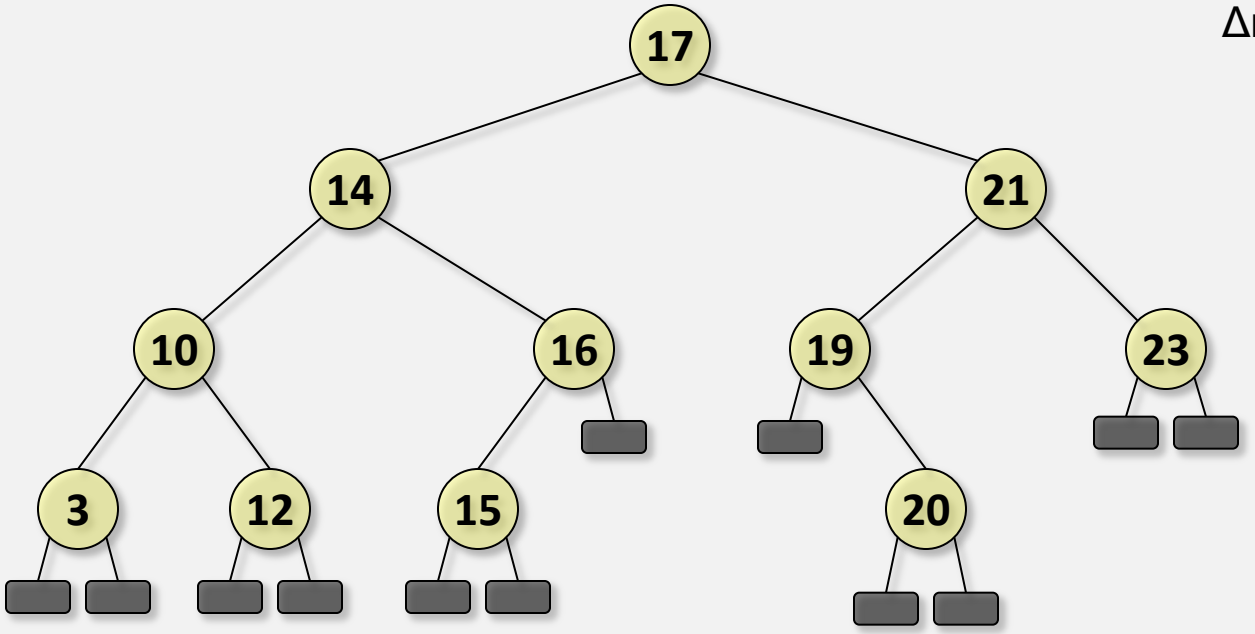
Διαγραφή 7



Δυαδικό Δένδρο Αναζήτησης

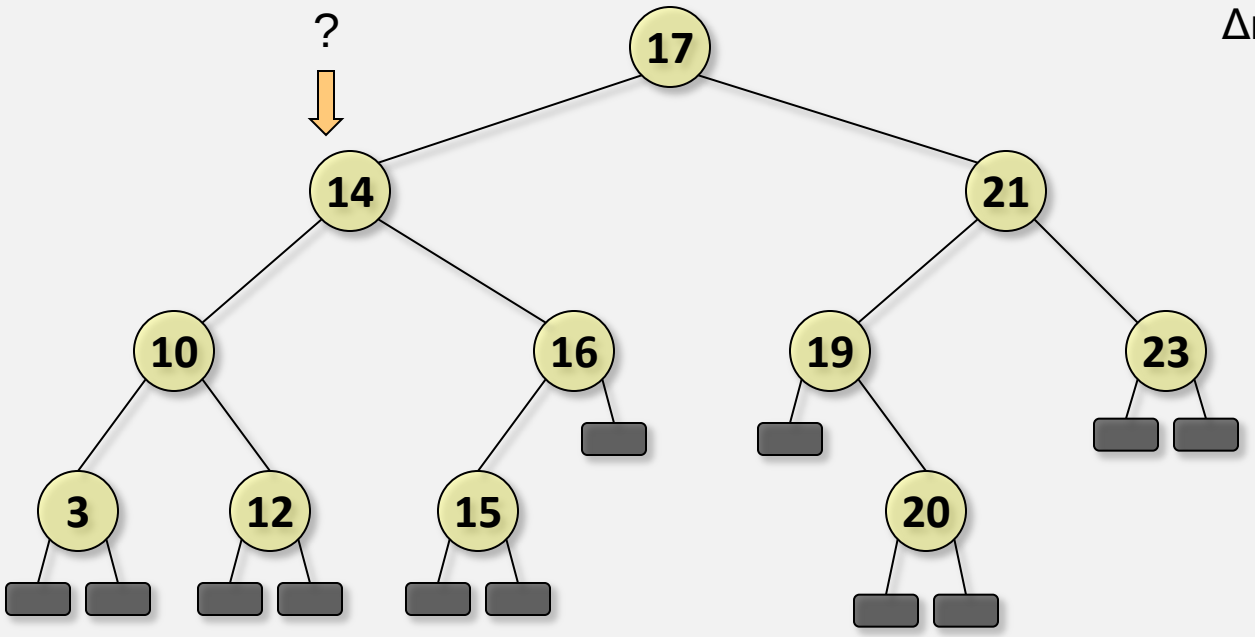
Το κλειδί οποιουδήποτε εσωτερικού κόμβου είναι μεγαλύτερο (ή ίσο) από όλα τα κλειδιά του αριστερού υποδένδρου και μικρότερο (ή ίσο) από όλα τα κλειδιά του δεξιού υποδένδρου.

Διαγραφή 7



Δυαδικό Δένδρο Αναζήτησης

Το κλειδί οποιουδήποτε εσωτερικού κόμβου είναι μεγαλύτερο (ή ίσο) από όλα τα κλειδιά του αριστερού υποδένδρου και μικρότερο (ή ίσο) από όλα τα κλειδιά του δεξιού υποδένδρου.



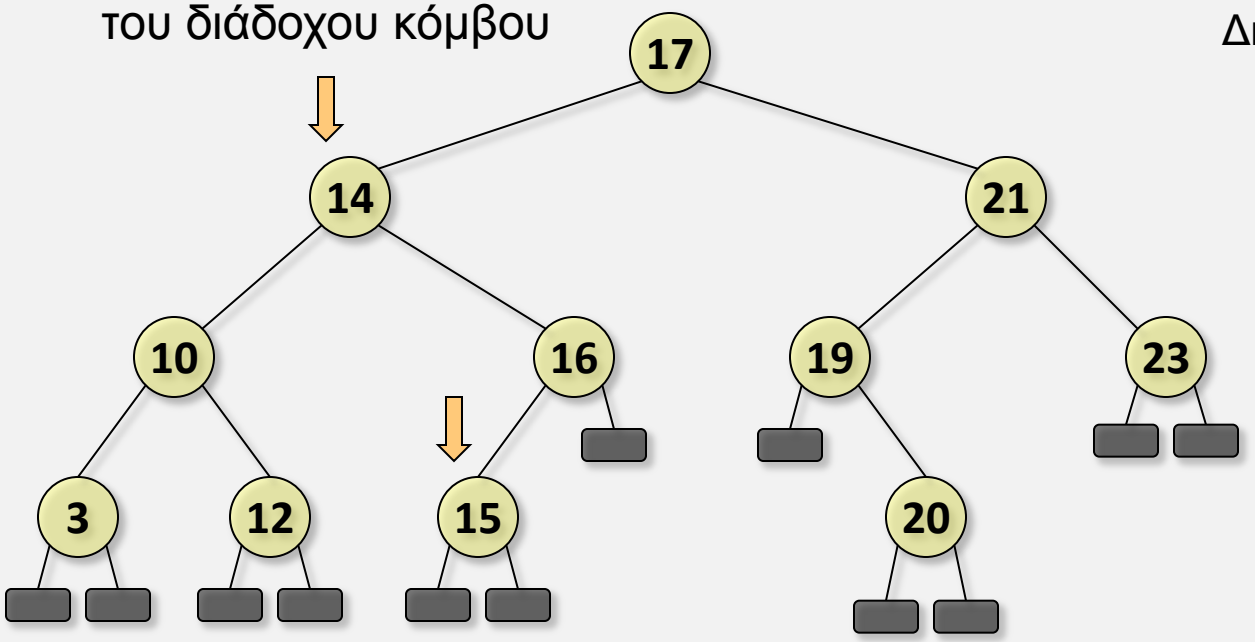
Διαγραφή 14

Δυαδικό Δένδρο Αναζήτησης

Το κλειδί οποιουδήποτε εσωτερικού κόμβου είναι μεγαλύτερο (ή ίσο) από όλα τα κλειδιά του αριστερού υποδένδρου και μικρότερο (ή ίσο) από όλα τα κλειδιά του δεξιού υποδένδρου.

Αντιγράφουμε το κλειδί
του διάδοχου κόμβου

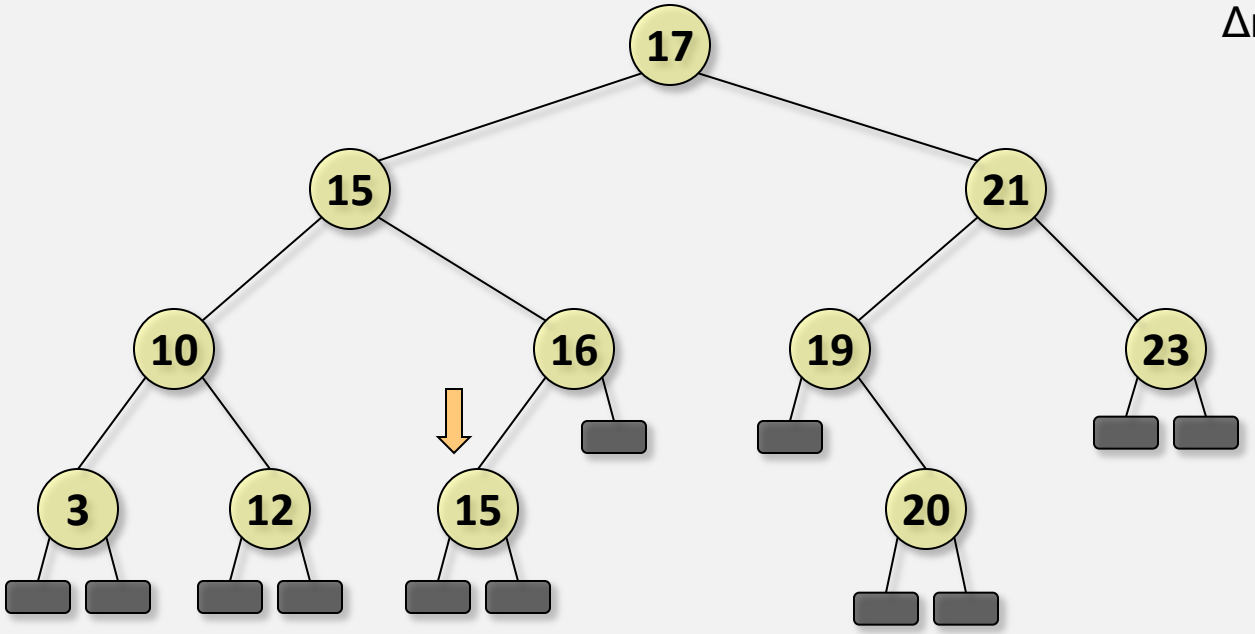
Διαγραφή 14



Δυαδικό Δένδρο Αναζήτησης

Το κλειδί οποιουδήποτε εσωτερικού κόμβου είναι μεγαλύτερο (ή ίσο) από όλα τα κλειδιά του αριστερού υποδένδρου και μικρότερο (ή ίσο) από όλα τα κλειδιά του δεξιού υποδένδρου.

Διαγραφή 14

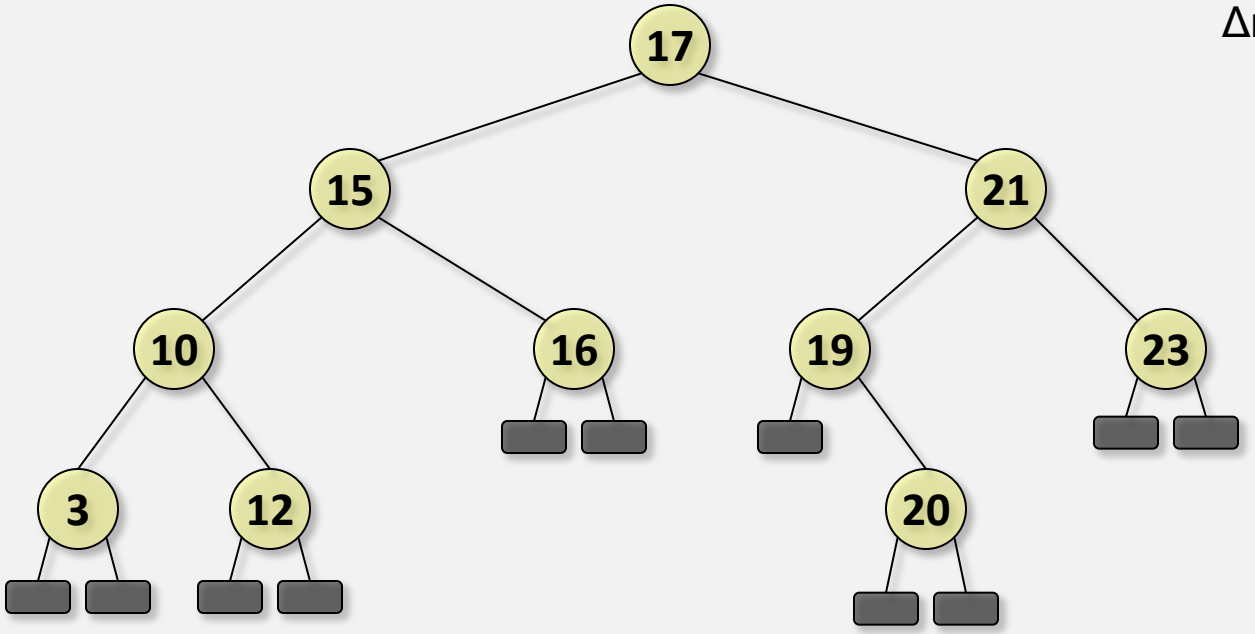


Διαγράφουμε τον διάδοχο κόμβο

Δυαδικό Δένδρο Αναζήτησης

Το κλειδί οποιουδήποτε εσωτερικού κόμβου είναι μεγαλύτερο (ή ίσο) από όλα τα κλειδιά του αριστερού υποδένδρου και μικρότερο (ή ίσο) από όλα τα κλειδιά του δεξιού υποδένδρου.

Διαγραφή 14



Δυαδικό Δένδρο Αναζήτησης

Το κλειδί οποιουδήποτε εσωτερικού κόμβου είναι μεγαλύτερο (ή ίσο) από όλα τα κλειδιά του αριστερού υποδένδρου και μικρότερο (ή ίσο) από όλα τα κλειδιά του δεξιού υποδένδρου.

Διαγραφή κόμβου `x`

Αν ο x έχει κενό παιδί τότε θέτουμε z=x, διαφορετικά θέτουμε z = διάδοχος του x. Αν ο z είναι διαφορετικός από τον x τότε αντιγράφουμε τα δεδομένα (`key, item`) του z στον x.

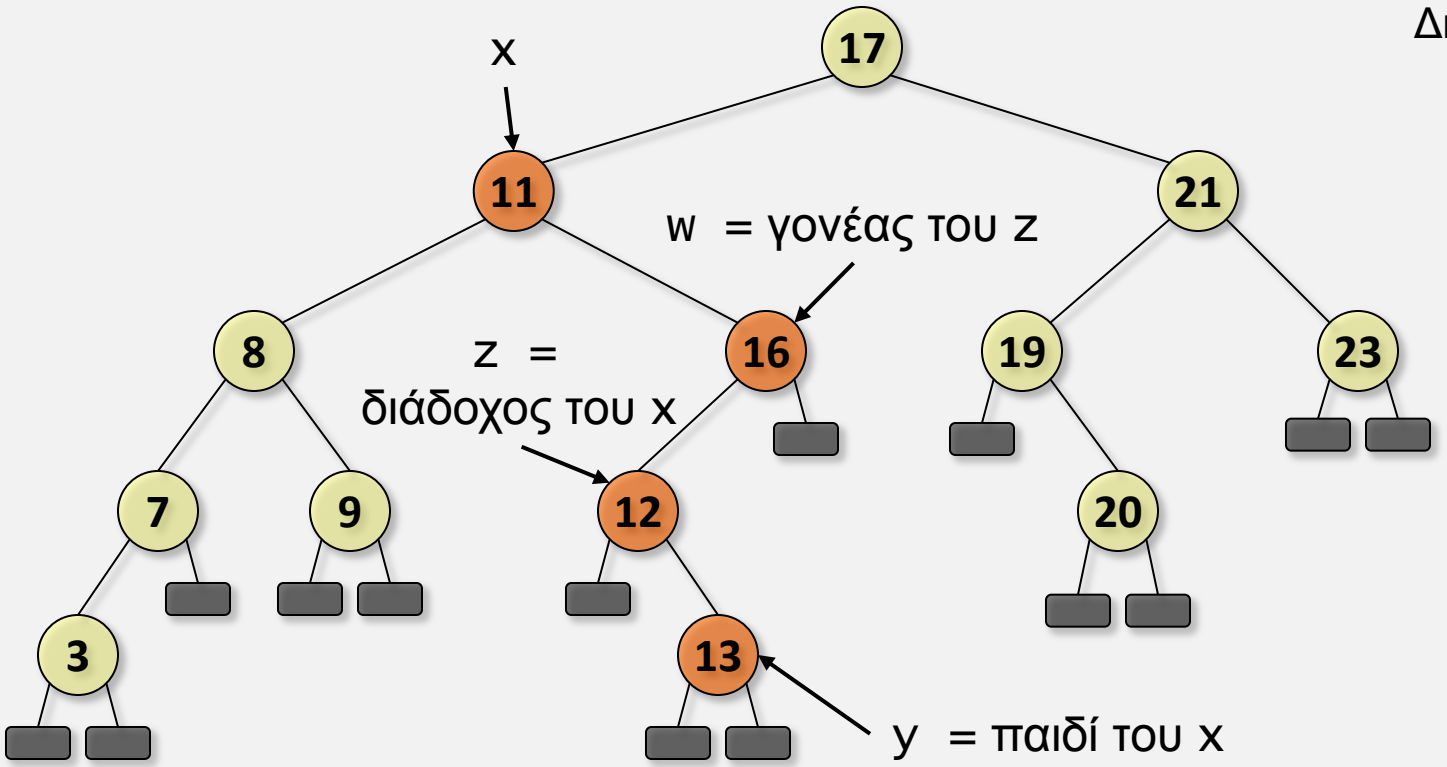
Αν ο z έχει κενό δεξί παιδί τότε θέτουμε `y=z.left`, διαφορετικά θέτουμε `y=z.right`.

Αν ο z δεν έχει γονέα τότε θέτουμε `root=y`. Διαφορετικά έστω w ο γονέας του z. Αν ο z είναι αριστερό παιδί (`w.left==z`) τότε κάνουμε τον y αριστερό παιδί του w, και αν ο z είναι δεξί παιδί (`w.right==z`) τότε κάνουμε τον y δεξί παιδί του w

Δυαδικό Δένδρο Αναζήτησης

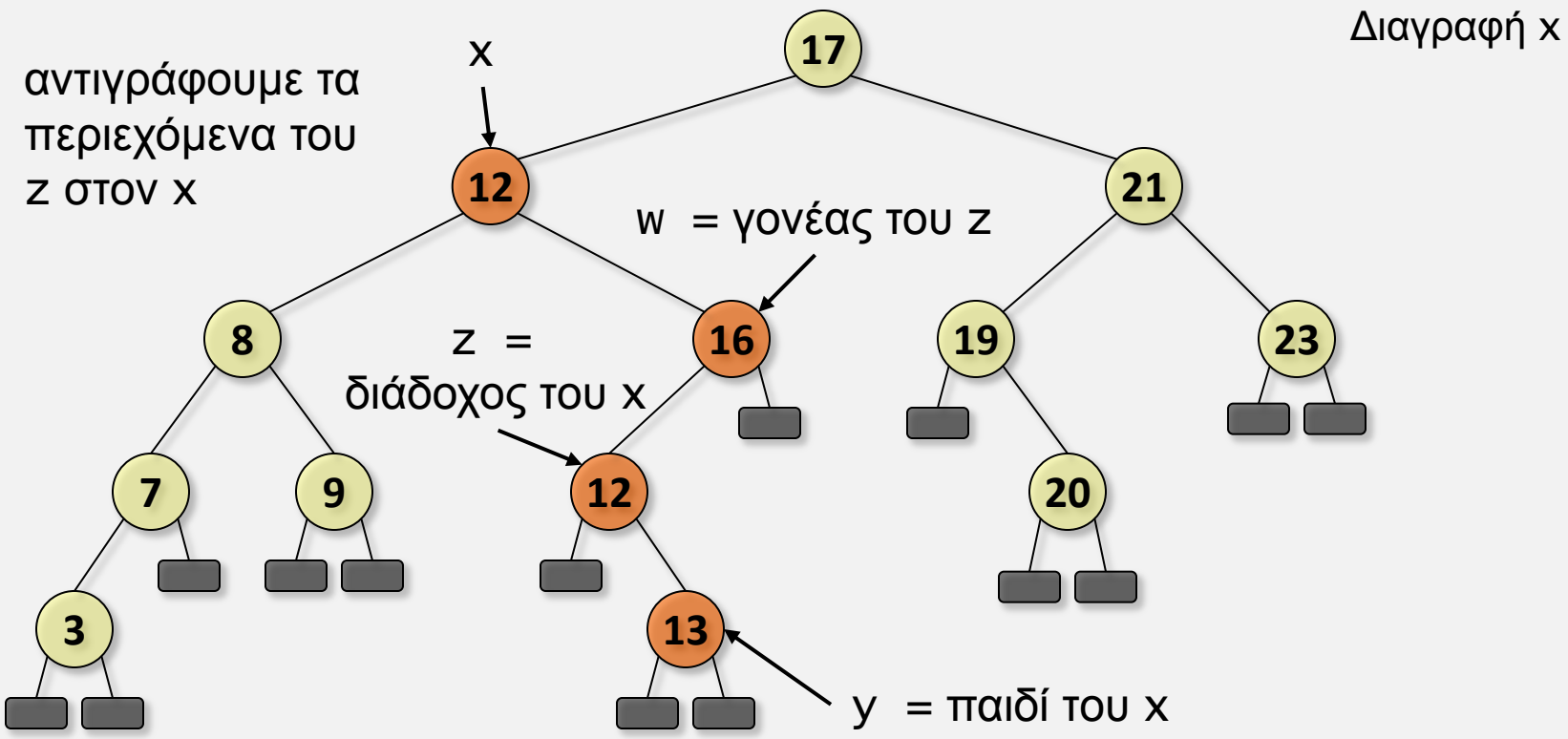
Το κλειδί οποιουδήποτε εσωτερικού κόμβου είναι μεγαλύτερο (ή ίσο) από όλα τα κλειδιά του αριστερού υποδένδρου και μικρότερο (ή ίσο) από όλα τα κλειδιά του δεξιού υποδένδρου.

Διαγραφή x



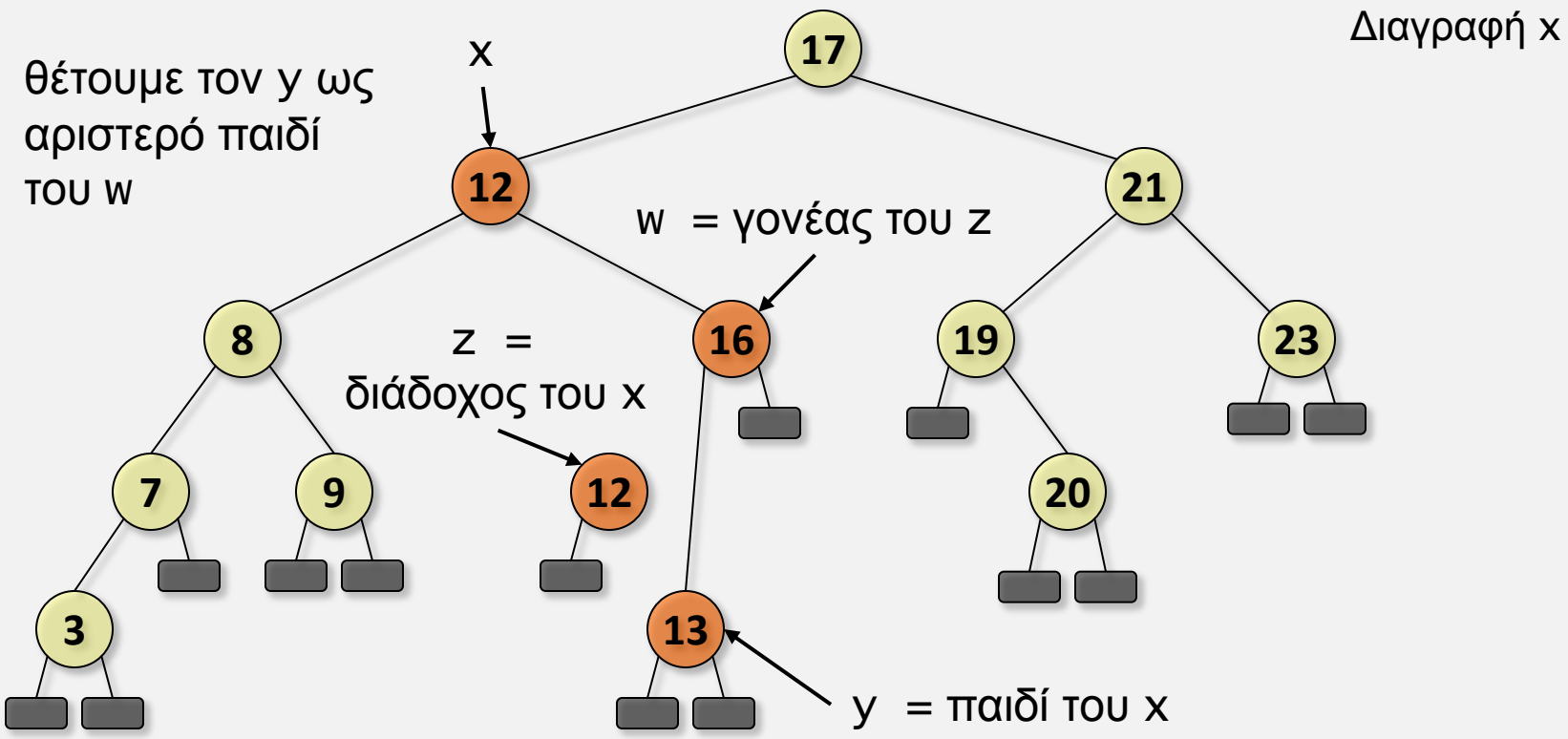
Δυαδικό Δένδρο Αναζήτησης

Το κλειδί οποιουδήποτε εσωτερικού κόμβου είναι μεγαλύτερο (ή ίσο) από όλα τα κλειδιά του αριστερού υποδένδρου και μικρότερο (ή ίσο) από όλα τα κλειδιά του δεξιού υποδένδρου.



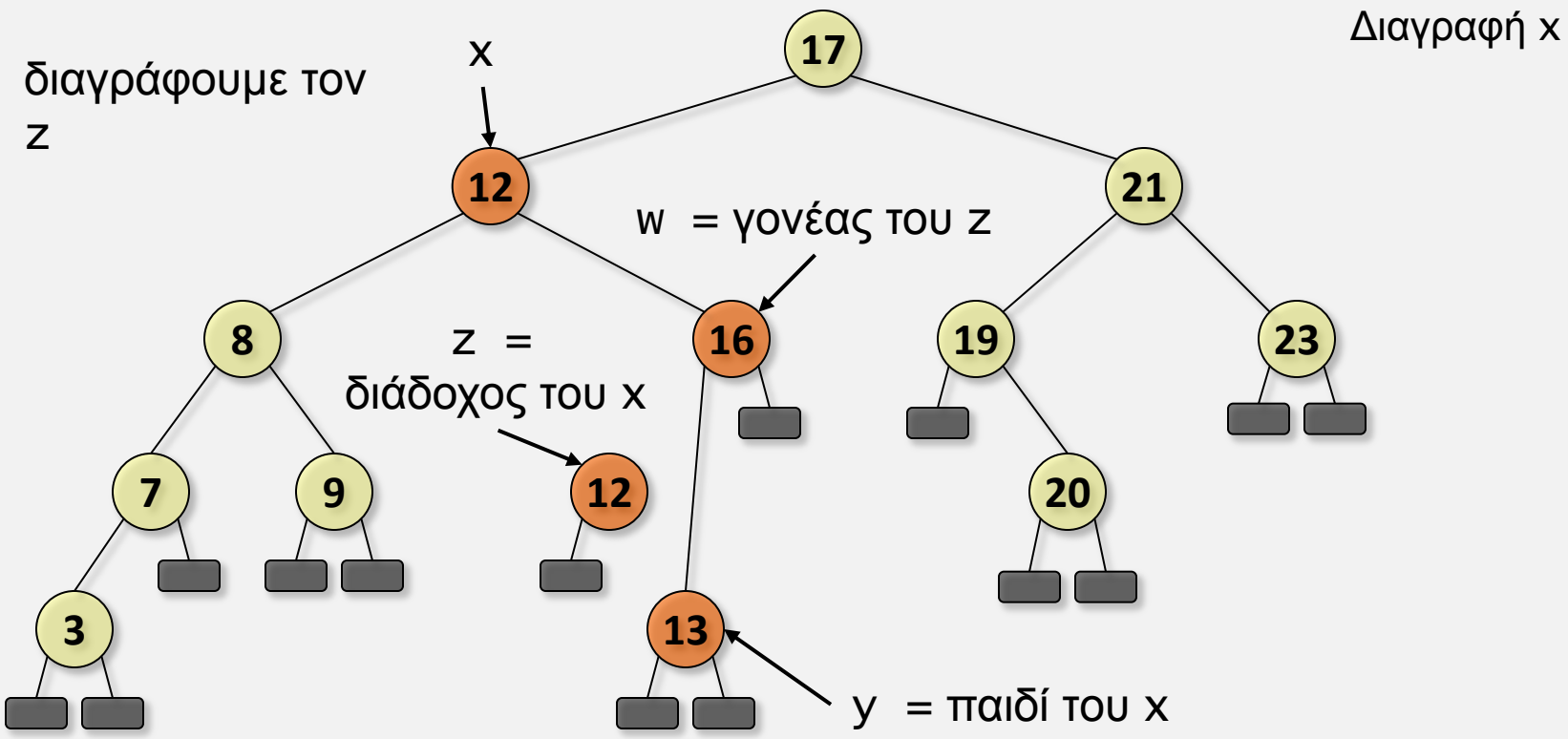
Δυαδικό Δένδρο Αναζήτησης

Το κλειδί οποιουδήποτε εσωτερικού κόμβου είναι μεγαλύτερο (ή ίσο) από όλα τα κλειδιά του αριστερού υποδένδρου και μικρότερο (ή ίσο) από όλα τα κλειδιά του δεξιού υποδένδρου.



Δυαδικό Δένδρο Αναζήτησης

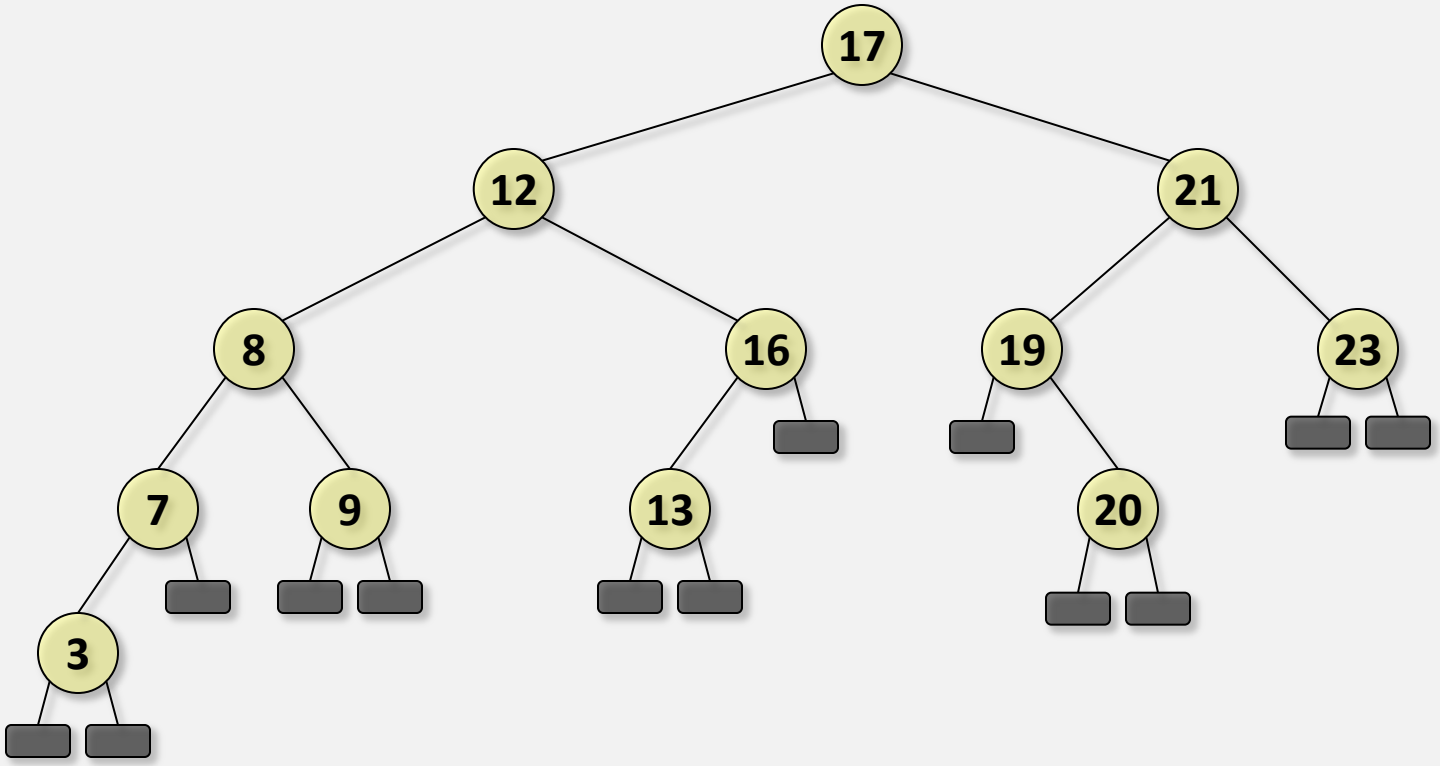
Το κλειδί οποιουδήποτε εσωτερικού κόμβου είναι μεγαλύτερο (ή ίσο) από όλα τα κλειδιά του αριστερού υποδένδρου και μικρότερο (ή ίσο) από όλα τα κλειδιά του δεξιού υποδένδρου.



Διαγραφή x

Δυαδικό Δένδρο Αναζήτησης

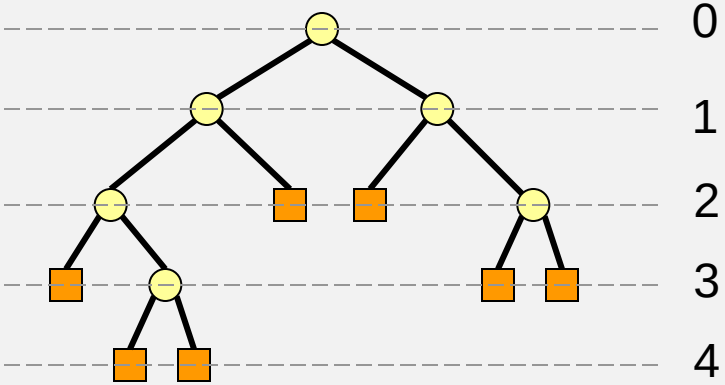
Το κλειδί οποιουδήποτε εσωτερικού κόμβου είναι μεγαλύτερο (ή ίσο) από όλα τα κλειδιά του αριστερού υποδένδρου και μικρότερο (ή ίσο) από όλα τα κλειδιά του δεξιού υποδένδρου.



Δυαδικό Δένδρο Αναζήτησης

επίπεδο ρίζας = 0
επίπεδο κόμβου = επίπεδο γονέα + 1

ύψος δένδρου = μέγιστο επίπεδο



Μήκος διαδρομής = άθροισμα επιπέδου κάθε κόμβου (=30)

Μήκος εσωτερικής διαδρομής = άθροισμα επιπέδου κάθε εσωτερικού κόμβου (=9)

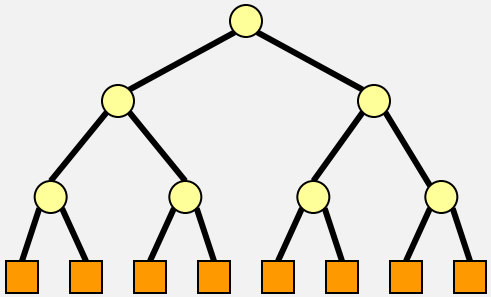
Μήκος εξωτερικής διαδρομής = άθροισμα επιπέδου κάθε εξωτερικού κόμβου (=21)

Ισχύει: μήκος εξωτερικής διαδρομής = μήκος εσωτερικής διαδρομής + 2N

Δυαδικό Δένδρο Αναζήτησης

Ο χρόνος εκτέλεσης των προηγούμενων λειτουργιών είναι ανάλογος του ύψους του δένδρου αναζήτησης

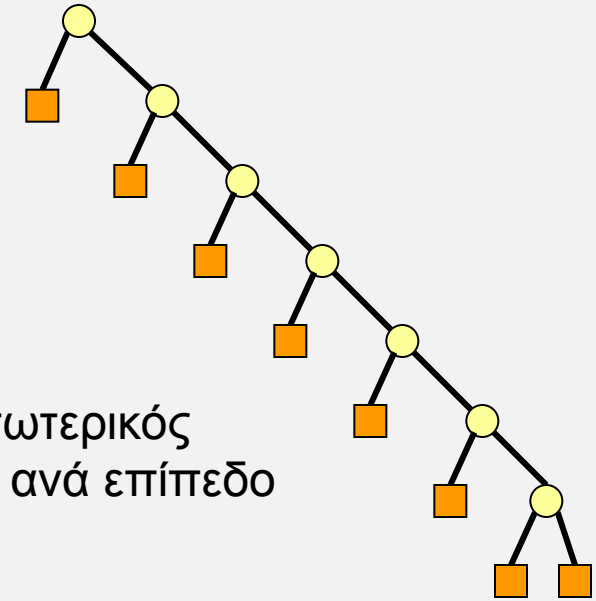
Ένα δυαδικό δένδρο με N εσωτερικούς κόμβους έχει ύψος μεταξύ lgN και N



2^i εσωτερικοί κόμβοι στο επίπεδο i

$$2^{h-1} < N + 1 \leq 2^h$$

μήκος εσωτερικής διαδρομής = $\Theta(N \lg N)$



ένας εσωτερικός κόμβος ανά επίπεδο

μήκος εσωτερικής διαδρομής = $\Theta(N^2)$

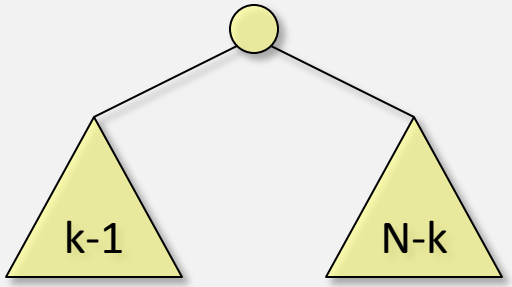
Δυαδικό Δένδρο Αναζήτησης

Ιδιότητα: Οι επιτυχείς αναζητήσεις απαιτούν περίπου 2·lnN συγκρίσεις, κατά μέσο όρο, σε ένα ΔΔΑ κατασκευασμένο από N τυχαία κλειδιά.

Έστω C(N) το μέσο μήκος της εσωτερικής διαδρομής ενός δυαδικού δένδρου αναζήτησης με N κόμβους.

Έχουμε την ακόλουθη αναδρομική σχέση

$$C(N) = (N-1) + \frac{1}{N}\sum_{k=1}^{N}\left[C(k-1) + C(N-k)\right],$$ όπου $C(0) = C(1) = 0$



Δυαδικό Δένδρο Αναζήτησης

Ιδιότητα: Οι επιτυχείς αναζητήσεις απαιτούν περίπου 2·lnN συγκρίσεις, κατά μέσο όρο, σε ένα ΔΔΑ κατασκευασμένο από N τυχαία κλειδιά.

Έστω C(N) το μέσο μήκος της εσωτερικής διαδρομής ενός δυαδικού δένδρου αναζήτησης με N κόμβους.

Έχουμε την ακόλουθη αναδρομική σχέση

$$C(N) = (N-1) + \frac{1}{N}\sum_{k=1}^{N}\left[C(k-1) + C(N-k)\right],$$ όπου $C(0) = C(1) = 0$

$$C(N) = (N-1) + \frac{2}{N}\sum_{k=1}^{N} C(k-1)$$

Δυαδικό Δένδρο Αναζήτησης

Ιδιότητα: Οι επιτυχείς αναζητήσεις απαιτούν περίπου 2·lnN συγκρίσεις, κατά μέσο όρο, σε ένα ΔΔΑ κατασκευασμένο από N τυχαία κλειδιά.

Έστω C(N) το μέσο μήκος της εσωτερικής διαδρομής ενός δυαδικού δένδρου αναζήτησης με N κόμβους.

Έχουμε την ακόλουθη αναδρομική σχέση

$$C(N) = (N-1) + \frac{1}{N}\sum_{k=1}^{N}\left[C(k-1) + C(N-k)\right],$$ όπου $C(0) = C(1) = 0$

$$C(N) = (N-1) + \frac{2}{N}\sum_{k=1}^{N} C(k-1) \;\Rightarrow\; NC(N) = N(N-1) + 2\sum_{k=1}^{N} C(k-1)$$

Ομοίως $$(N-1)C(N-1) = (N-1)(N-2) + 2\sum_{k=1}^{N-1} C(k-1)$$

Δυαδικό Δένδρο Αναζήτησης

Ιδιότητα: Οι επιτυχείς αναζητήσεις απαιτούν περίπου 2·lnN συγκρίσεις, κατά μέσο όρο, σε ένα ΔΔΑ κατασκευασμένο από N τυχαία κλειδιά.

Έστω C(N) το μέσο μήκος της εσωτερικής διαδρομής ενός δυαδικού δένδρου αναζήτησης με N κόμβους.

Επομένως

$$NC(N) = (N+1)C(N-1) + 2(N-1) \;\Rightarrow\; \frac{C(N)}{N+1} = \frac{C(N-1)}{N} + \frac{2(N-1)}{N(N+1)}$$

$$\Rightarrow\; \frac{C(N)}{N+1} \leq \frac{C(N-1)}{N} + \frac{2}{N} = O(\log N)$$

$$\Rightarrow\; C(N) = O(N \log N)$$

Δυαδικό Δένδρο Αναζήτησης

Ιδιότητα: Οι επιτυχείς αναζητήσεις απαιτούν περίπου 2·lnN συγκρίσεις, κατά μέσο όρο, σε ένα ΔΔΑ κατασκευασμένο από N τυχαία κλειδιά.

μήκος εξωτερικής διαδρομής = μήκος εσωτερικής διαδρομής + 2N

Έχουμε λοιπόν

Ιδιότητα: Οι εισαγωγές και οι ανεπιτυχείς αναζητήσεις απαιτούν περίπου 2·lnN συγκρίσεις, κατά μέσο όρο, σε ένα ΔΔΑ κατασκευασμένο από N τυχαία κλειδιά.