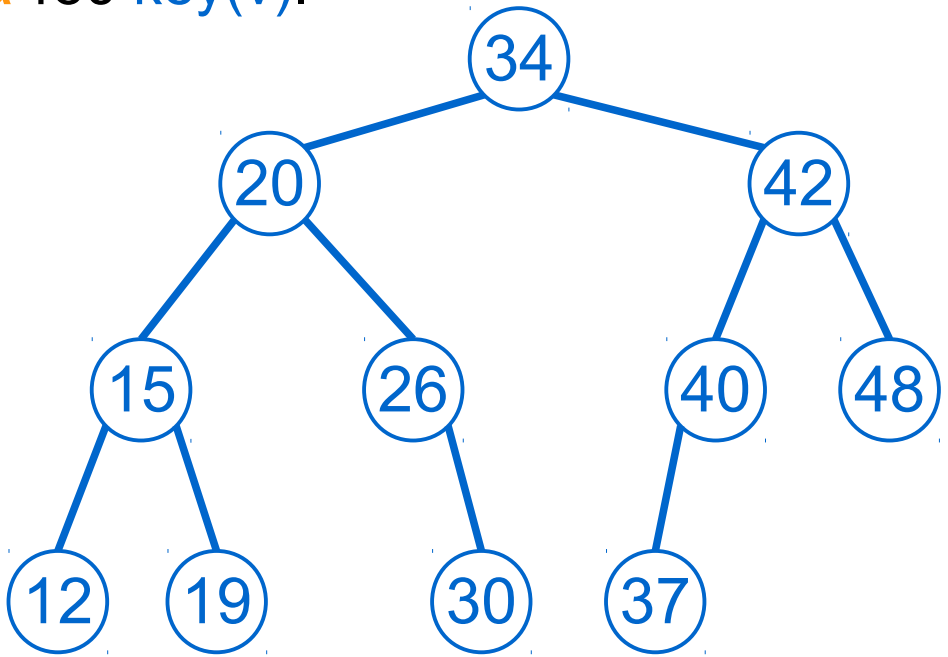
Δυαδικό Δέντρο Αναζήτησης

Δυαδικό Δέντρο Αναζήτησης: Είναι ένα δυαδικό δέντρο όπου για κάθε εσωτερικό κόμβο v με κλειδί key(v):

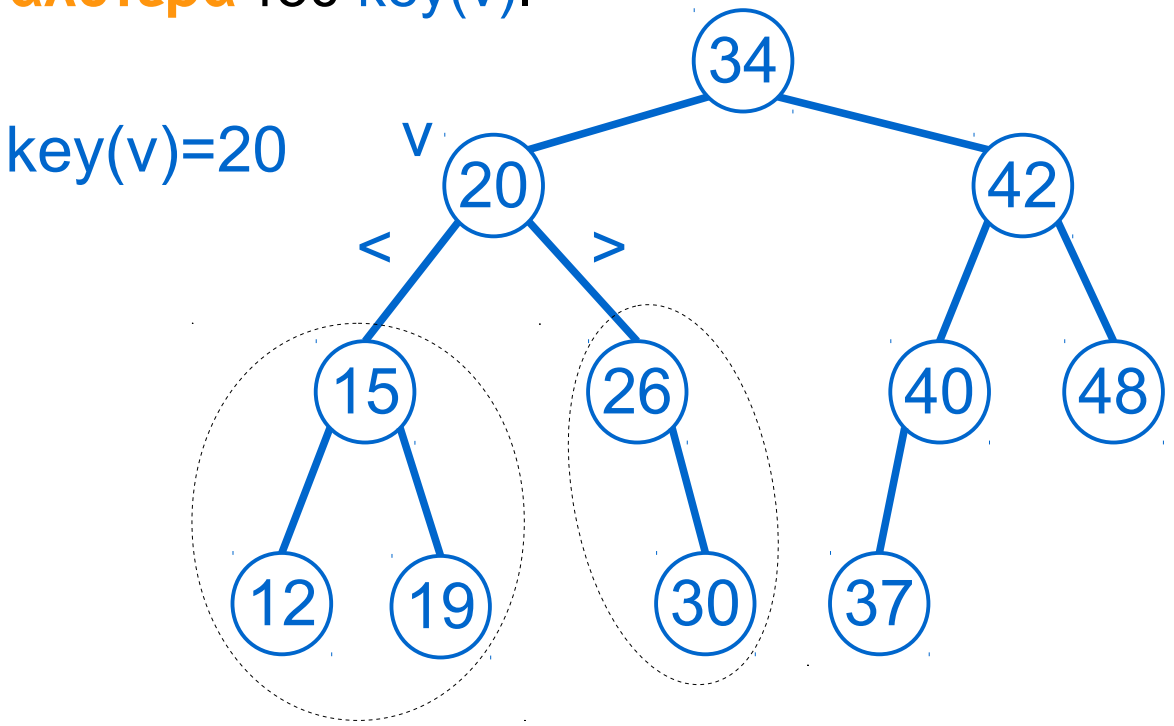
- όλοι οι κόμβοι του **αριστερού υποδέντρου** του v έχουν κλειδιά **μικρότερα** του key(v), και
- όλοι οι κόμβοι του **δεξιού υποδέντρου** του v έχουν κλειδιά **μεγαλύτερα** του key(v).



Δυαδικό Δέντρο Αναζήτησης

Δυαδικό Δέντρο Αναζήτησης: Είναι ένα δυαδικό δέντρο όπου για κάθε εσωτερικό κόμβο v με κλειδί key(v):

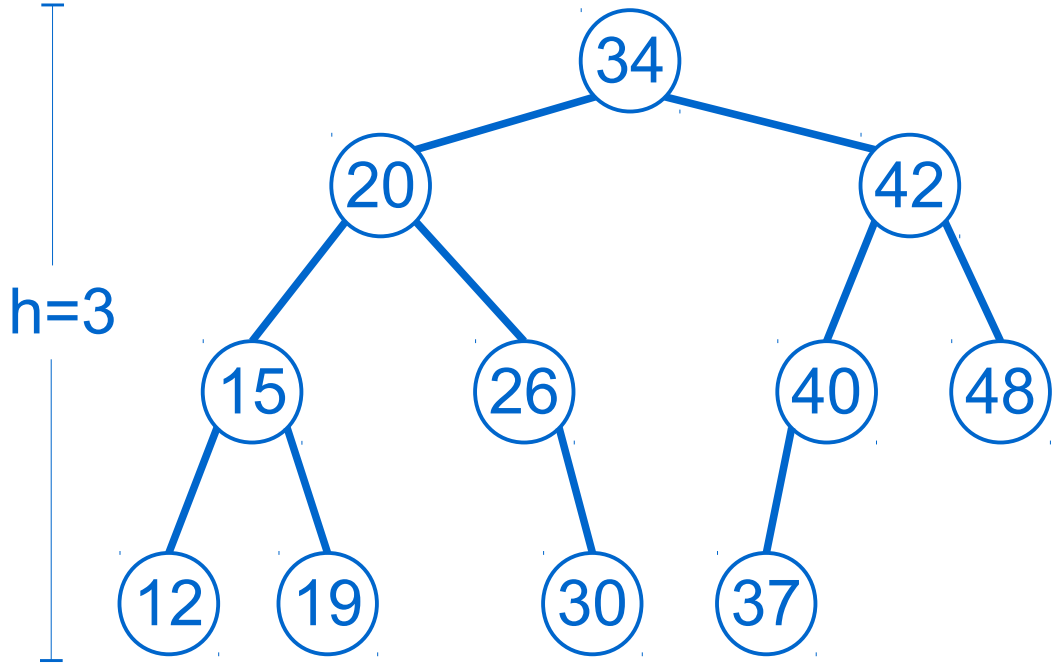
- όλοι οι κόμβοι του **αριστερού υποδέντρου** του v έχουν κλειδιά **μικρότερα** του key(v), και
- όλοι οι κόμβοι του **δεξιού υποδέντρου** του v έχουν κλειδιά **μεγαλύτερα** του key(v).



Δυαδικό Δέντρο Αναζήτησης

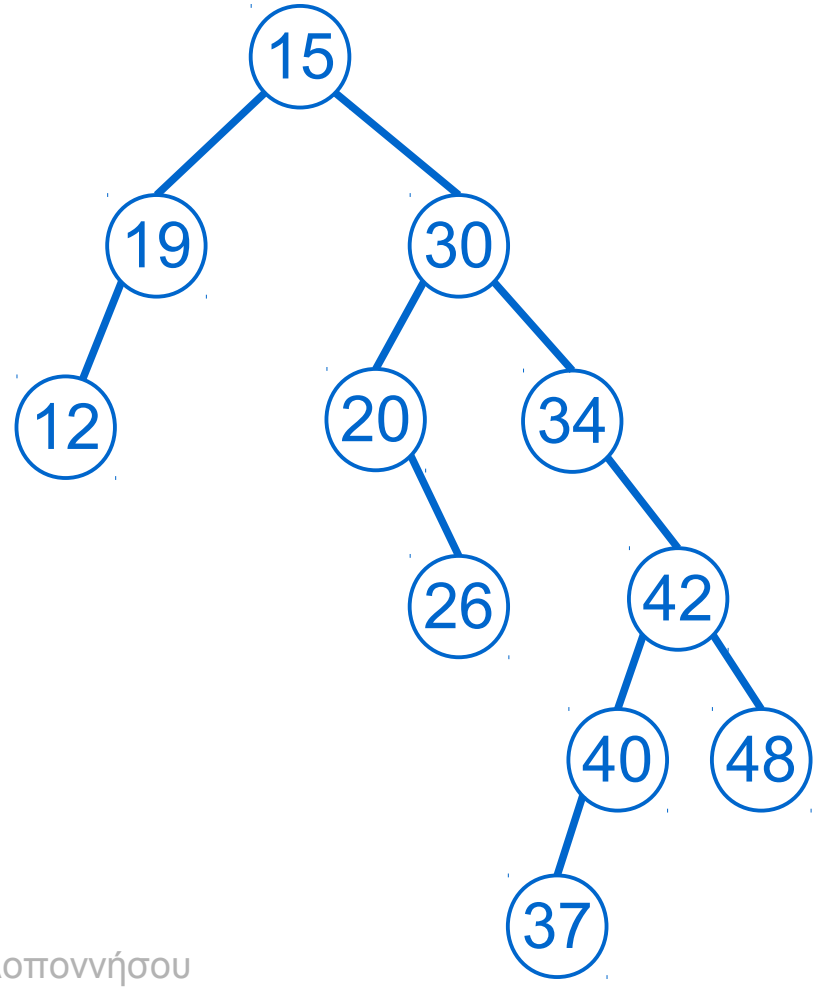
Λειτουργίες: Αναζήτηση, εισαγωγή, διαγραφή, ελάχιστο/μέγιστο, επόμενο/προηγούμενο.

Χρόνος Εκτέλεσης: O(h), όπου h το ύψος του δέντρου.



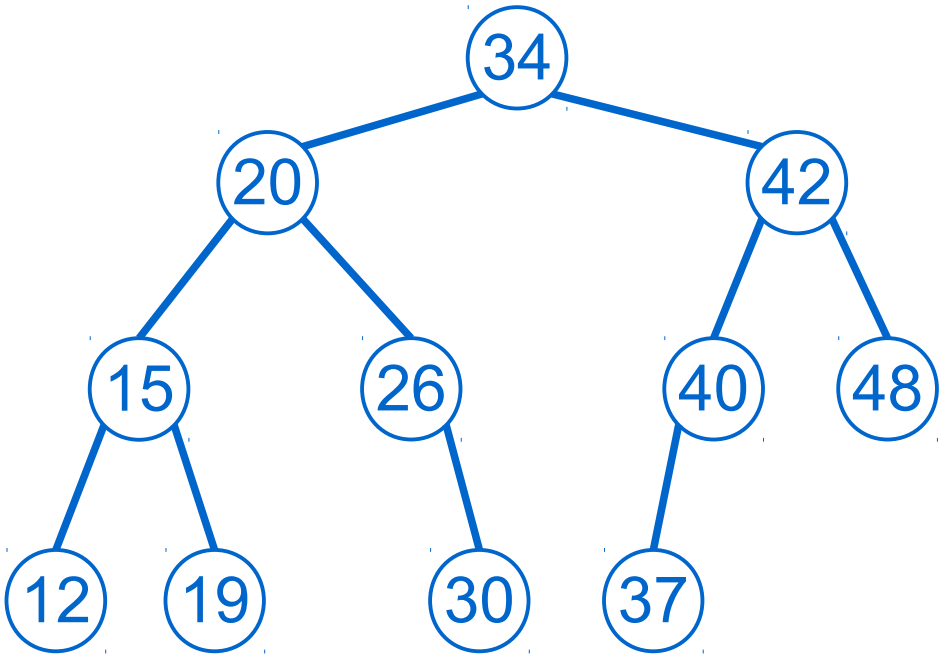
Δυαδικό Δέντρο Αναζήτησης

Μειονέκτημα ΔΔΑ: Το ύψος του δέντρου h στη χειρότερη περίπτωση μπορεί να είναι O(n) και τότε οι λειτουργίες δεν εκτελούνται αποδοτικά.



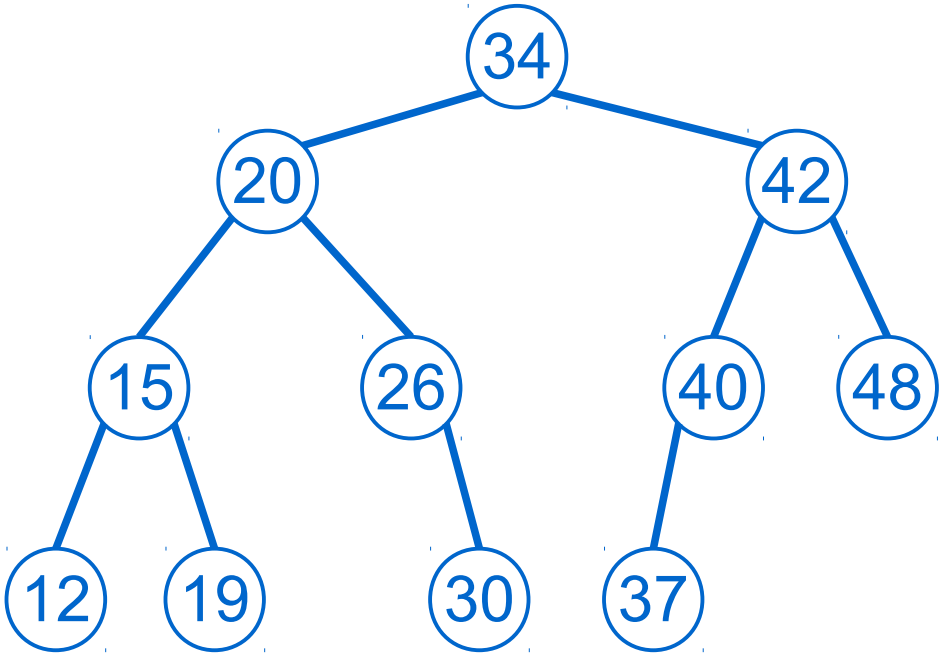
Ζυγισμένο Δέντρο Αναζήτησης

Ζυγισμένο Δέντρο Αναζήτησης: Δέντρο αναζήτησης που διασφαλίζει ότι το ύψος του είναι πάντοτε O(log n), όπου n το τρέχον πλήθος στοιχείων. Κατά συνέπεια, κάθε λειτουργία του απαιτεί χρόνο O(log n) (βλ. επόμενη ενότητα).



Διάσχιση Inorder σε ΔΔΑ

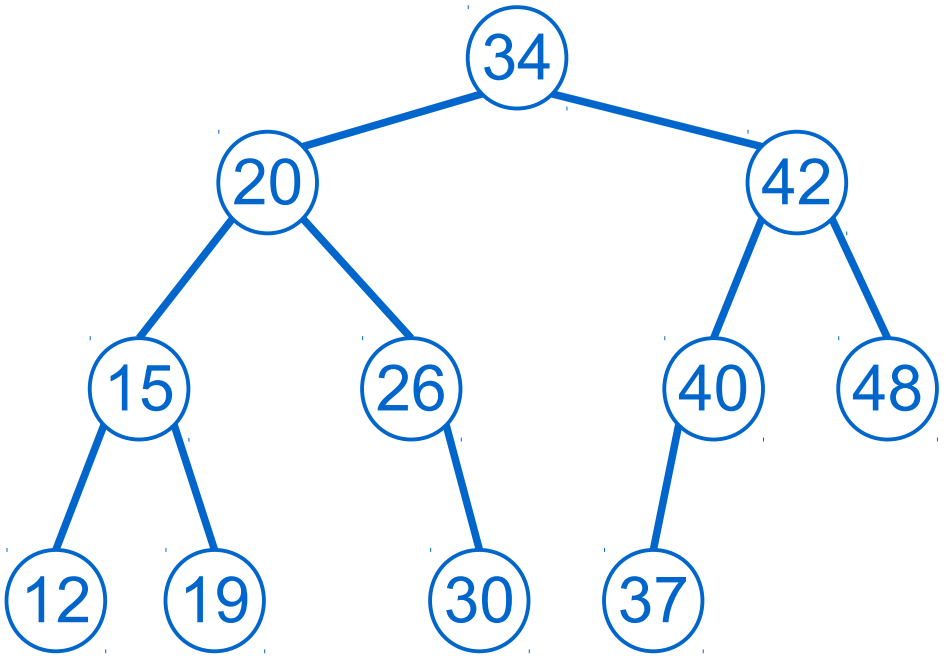
Διάσχιση Inorder σε Δυαδικό Δέντρο Αναζήτησης:
Επισκέπτεται τα κλειδιά κατά **αύξουσα σειρά**.



12<15<19<20<26<30<34<37<40<42<48

Αναζήτηση

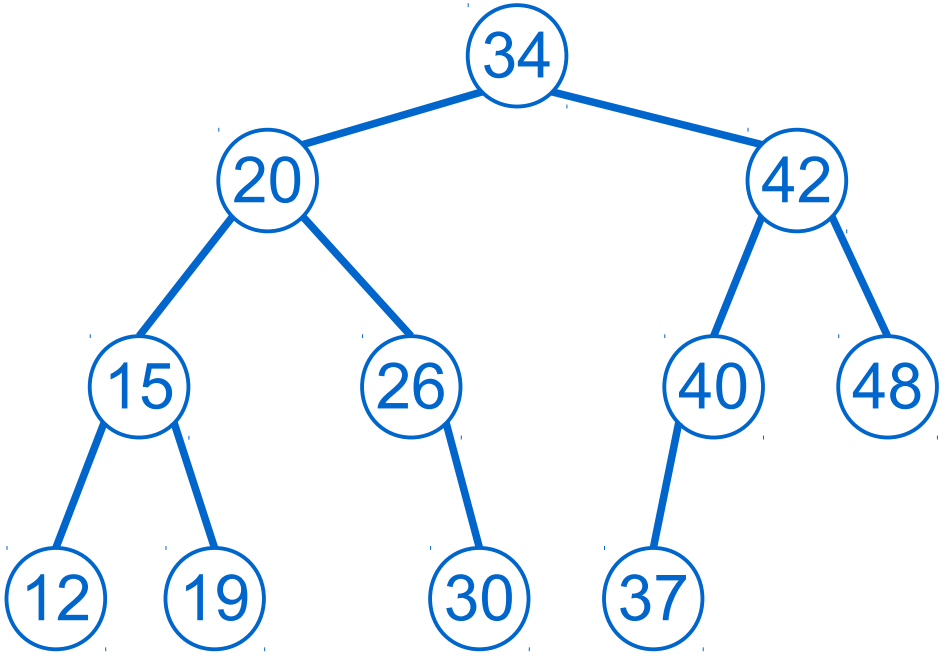
Αναζήτηση: Ξεκινώντας από τη ρίζα σύγκρινε το κλειδί αναζήτησης x με το κλειδί key στον τρέχοντα κόμβο.
Αν x<key, πήγαινε στο αριστερό παιδί του.
Αν x>key, πήγαινε στο δεξί παιδί του.
Αν x=key, επιτυχία. Αν ο τρέχων κόμβος είναι null, αποτυχία.



Αναζήτηση

Αναζήτηση: Ξεκινώντας από τη ρίζα σύγκρινε το κλειδί αναζήτησης x με το κλειδί key στον τρέχοντα κόμβο.
Αν x<key, πήγαινε στο αριστερό παιδί του.
Αν x>key, πήγαινε στο δεξί παιδί του.
Αν x=key, επιτυχία. Αν ο τρέχων κόμβος είναι null, αποτυχία.

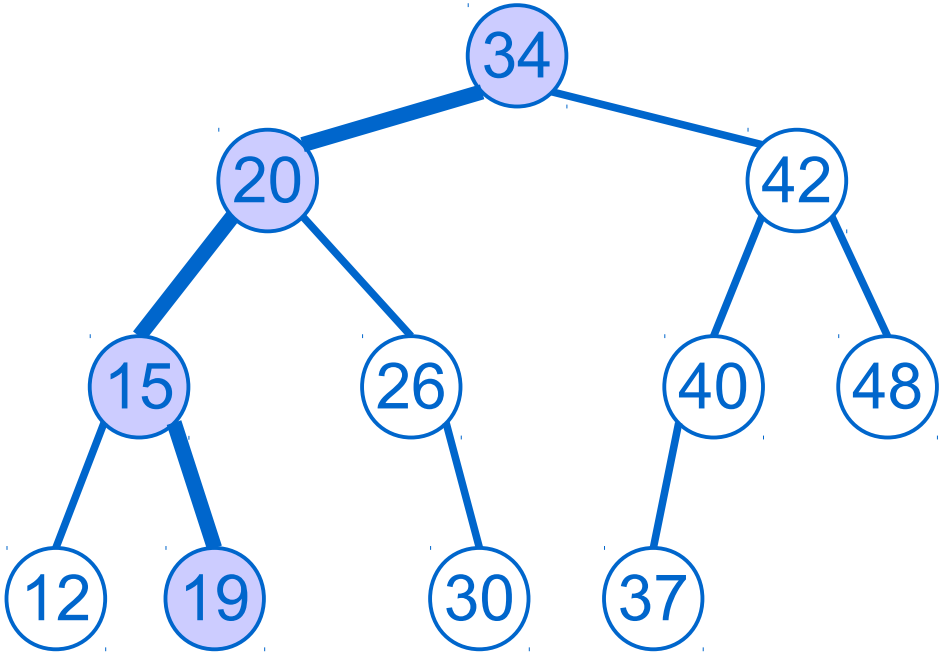
Βρες το 18



Αναζήτηση

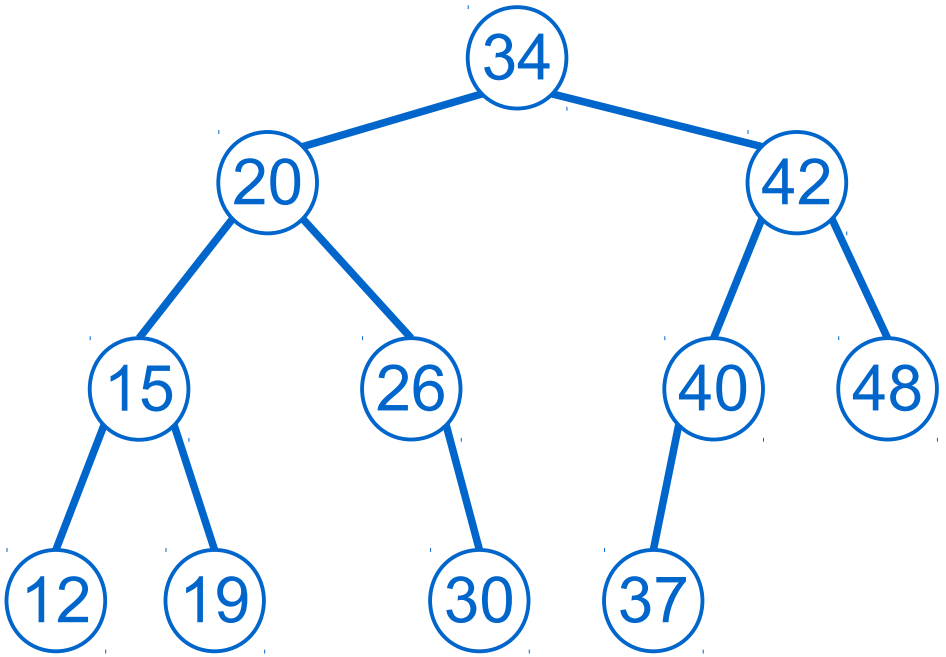
Αναζήτηση: Ξεκινώντας από τη ρίζα σύγκρινε το κλειδί αναζήτησης x με το κλειδί key στον τρέχοντα κόμβο.
Αν x<key, πήγαινε στο αριστερό παιδί του.
Αν x>key, πήγαινε στο δεξί παιδί του.
Αν x=key, επιτυχία. Αν ο τρέχων κόμβος είναι null, αποτυχία.

Βρες το 18



Αναζήτηση

Αναζήτηση: Χρόνος αναζήτησης στη χειρότερη περίπτωση O(h) όπου h το ύψος του δέντρου.



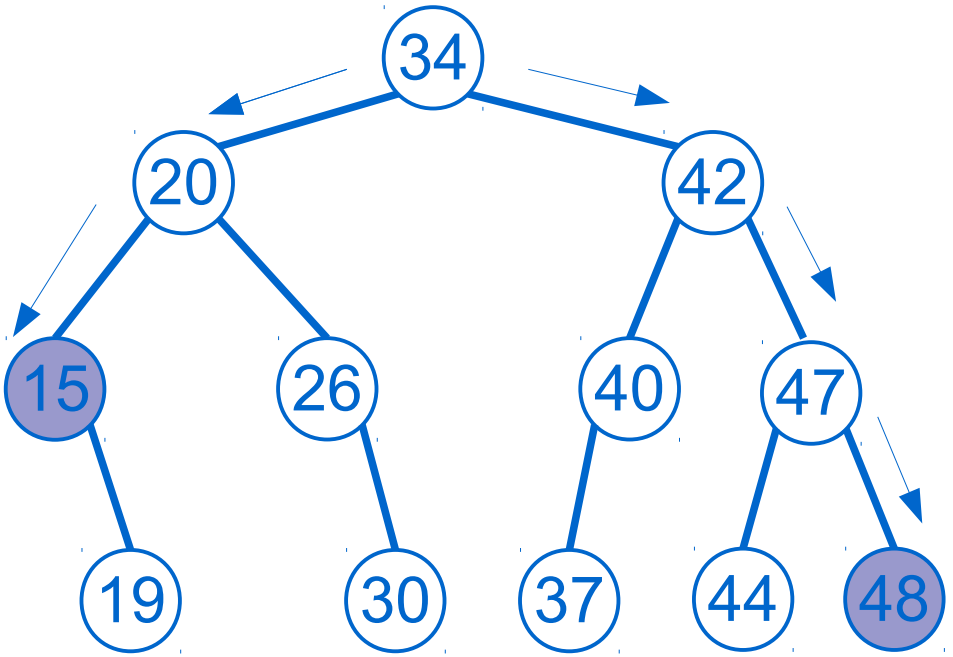
Δυαδική Αναζήτηση

```
bool bsearch(TreeNode *v, int x) {
   // εκδοχή με χρήση αναδρομικής συνάρτησης
   // v είναι ο τρέχων κόμβος
   if (v != NULL) {
     if (x < v->key)
        return bsearch(v->left, x);
     else if (x > v->key)
        return bsearch(v->right, x);
     else return true;  // το κλειδί βρέθηκε
   }
   else return false;  // το κλειδί δεν βρέθηκε
}
```

Αναζήτηση Ελαχίστου ή Μεγίστου

Αναζήτηση Ελαχίστου: Είναι ο αριστερότερος κόμβος.

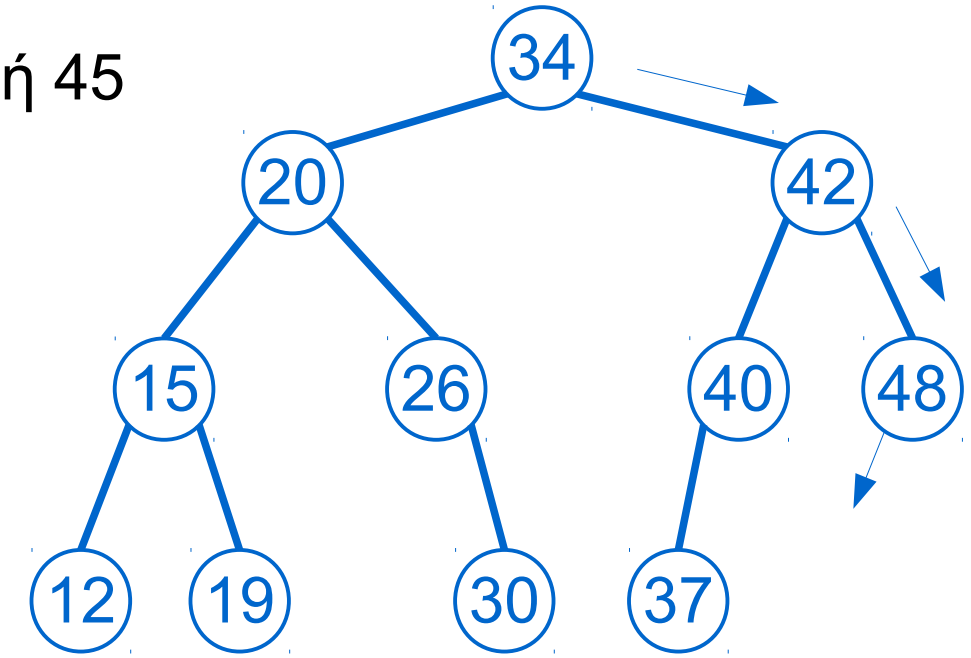
Αναζήτηση Μεγίστου: Είναι ο δεξιότερος κόμβος.



Εισαγωγή

Εισαγωγή: Αναζητούμε το κλειδί x του στοιχείου προς εισαγωγή. Αν υπάρχει ήδη τότε αποτυχία. Διαφορετικά εισάγουμε το στοιχείο σε έναν νέο κόμβο στη θέση ακριβώς που τερματίστηκε η αναζήτηση.

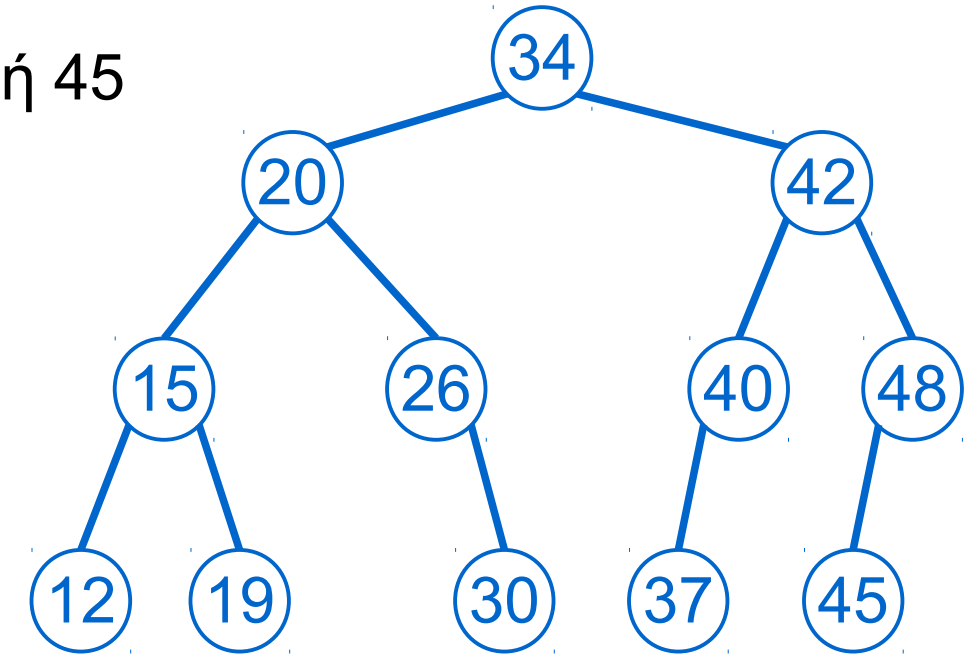
Εισαγωγή 45



Εισαγωγή

Εισαγωγή: Αναζητούμε το κλειδί x του στοιχείου προς εισαγωγή. Αν υπάρχει ήδη τότε αποτυχία. Διαφορετικά εισάγουμε το στοιχείο σε έναν νέο κόμβο στη θέση ακριβώς που τερματίστηκε η αναζήτηση.

Εισαγωγή 45



```
TreeNode* insert(TreeNode *root, int x) {
   TreeNode *v = root;  // ν τρέχων κόμβος
   TreeNode *pv = NULL; // pv γονέας ν
   while (v != NULL) {  // επαναληπτική εκδοχή
      pv = v;
      if (x < v->key) v=v->left;
      else if (x > v->key) v=v->right;
           else { // υπάρχει ήδη
                  printf("error: duplicate");
                  exit(1);}
   }
   TreeNode *tmp = malloc(sizeof(TreeNode));
   tmp->key=x; tmp->left=tmp->right=NULL;
   if (root != NULL) {
      if (x < pv->key) pv->left=tmp;
         else pv->right=tmp;
   } else root=tmp;
   return root; // επιστροφή ρίζας δέντρου
}
```

Διαγραφή

Διαγραφή: Διακρίνουμε **τρεις περιπτώσεις** για τον κόμβο v που πρόκειται να διαγραφεί:

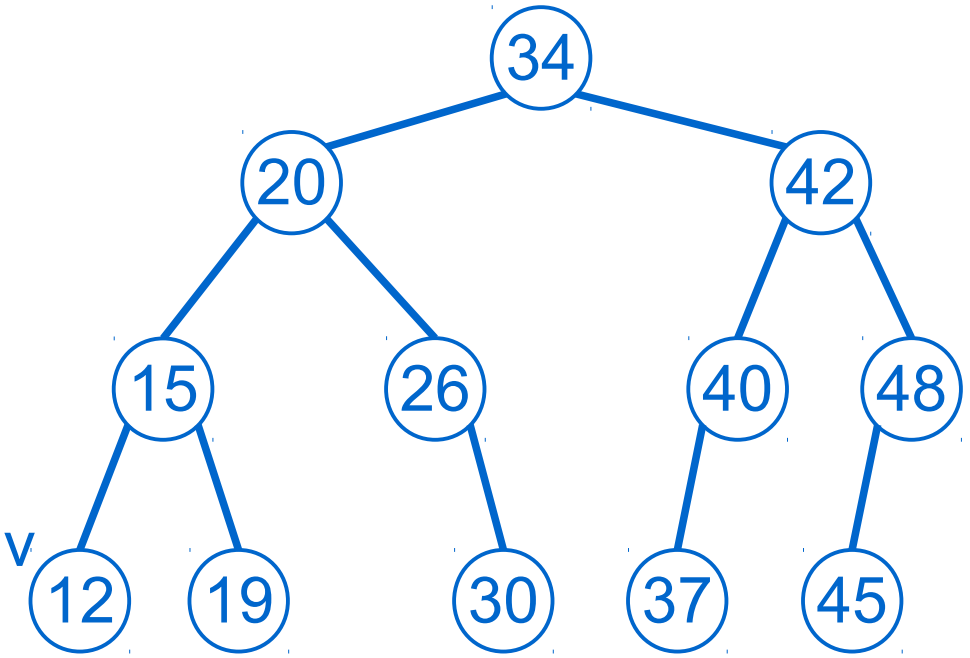
(α) ο v είναι φύλλο

(β) ο v έχει ένα μοναδικό παιδί

(γ) ο v έχει δύο παιδιά

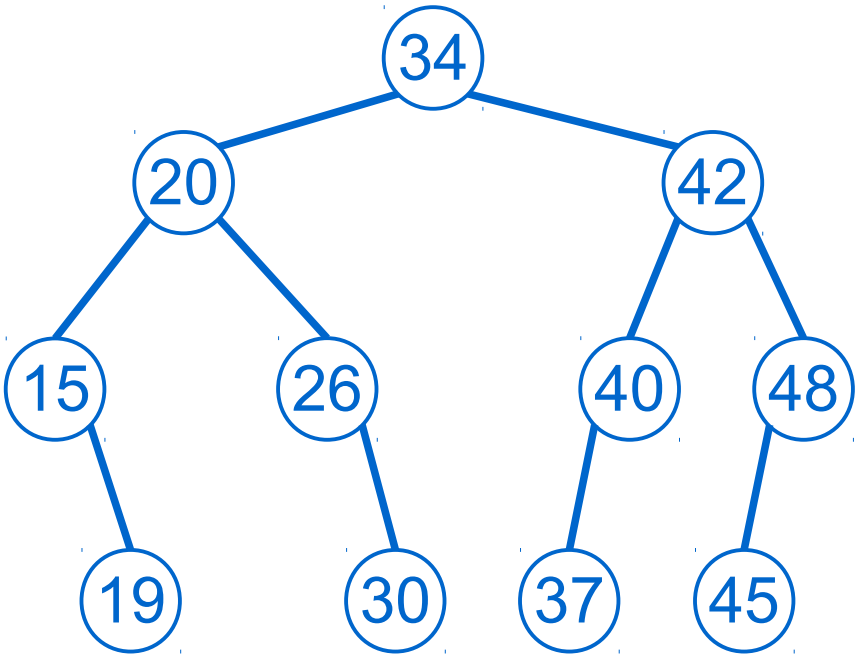
Διαγραφή (α)

Διαγραφή (α): Απλά διαγράφουμε το φύλλο. (Αν είναι το μοναδικό φύλλο το δέντρο γίνεται κενό.) Πχ. διαγραφή 12.



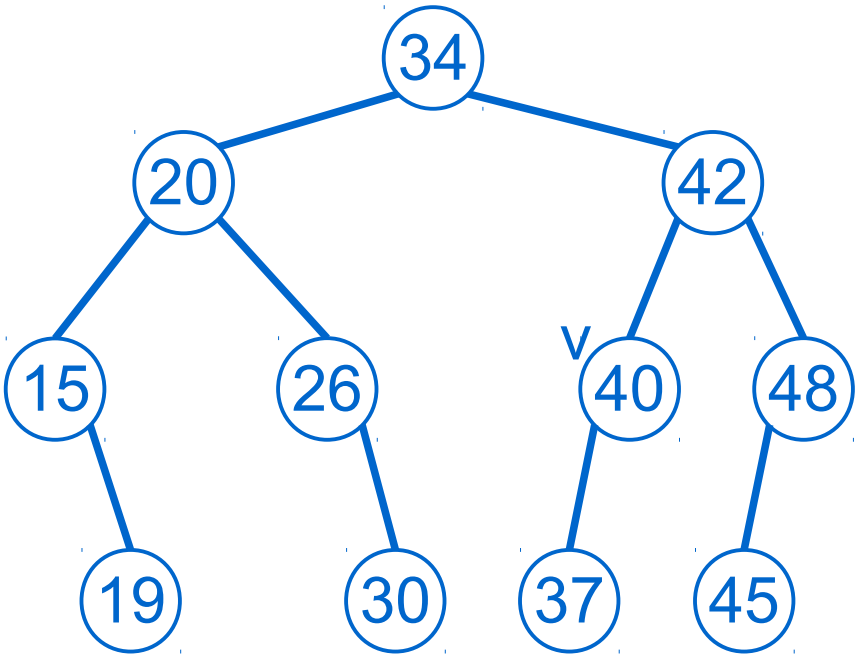
Διαγραφή (α)

Διαγραφή (α): Απλά διαγράφουμε το φύλλο. (Αν είναι το μοναδικό φύλλο το δέντρο γίνεται κενό.) Πχ. διαγραφή 12.



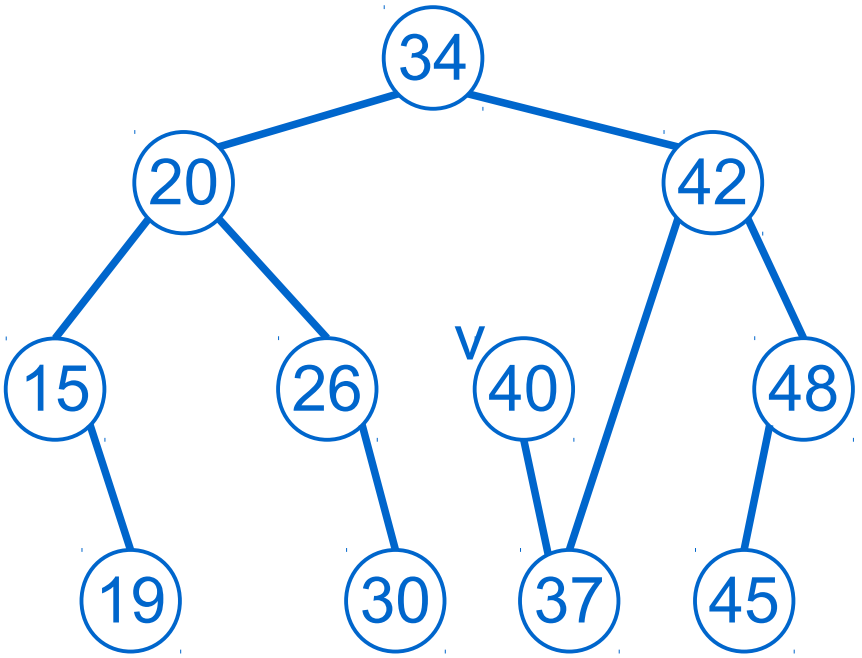
Διαγραφή (β)

Διαγραφή (β): Υποπερίπτωση: ο v δεν είναι η ρίζα.
Ο γονέας του v υιοθετεί το μοναδικό παιδί του v και μετά ο v διαγράφεται. Πχ. διαγραφή 40.



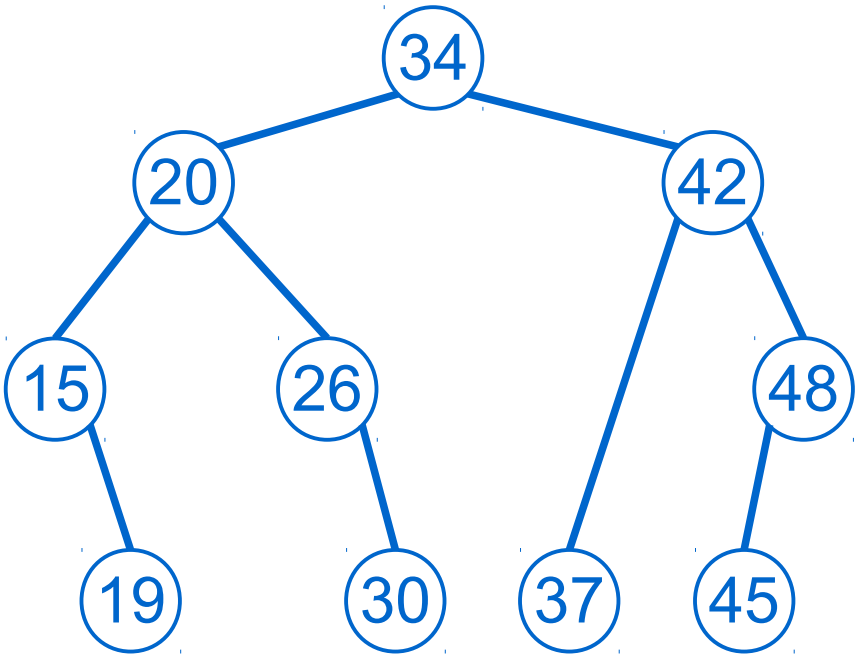
Διαγραφή (β)

Διαγραφή (β): Υποπερίπτωση: ο v δεν είναι η ρίζα.
Ο γονέας του v υιοθετεί το μοναδικό παιδί του v και μετά ο v διαγράφεται. Πχ. διαγραφή 40.



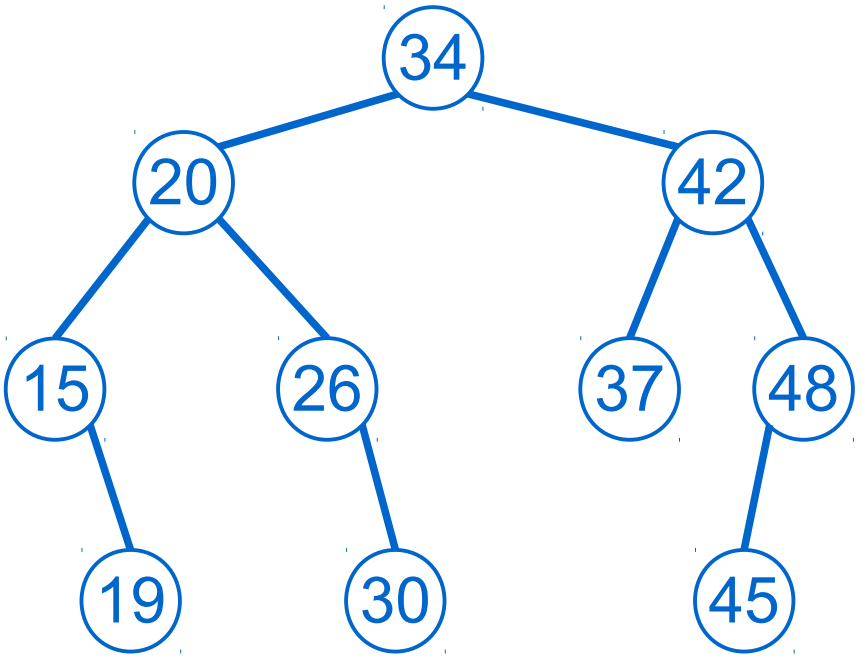
Διαγραφή (β)

Διαγραφή (β): Υποπερίπτωση: ο v δεν είναι η ρίζα.
Ο γονέας του v υιοθετεί το μοναδικό παιδί του v και μετά ο v διαγράφεται. Πχ. διαγραφή 40.



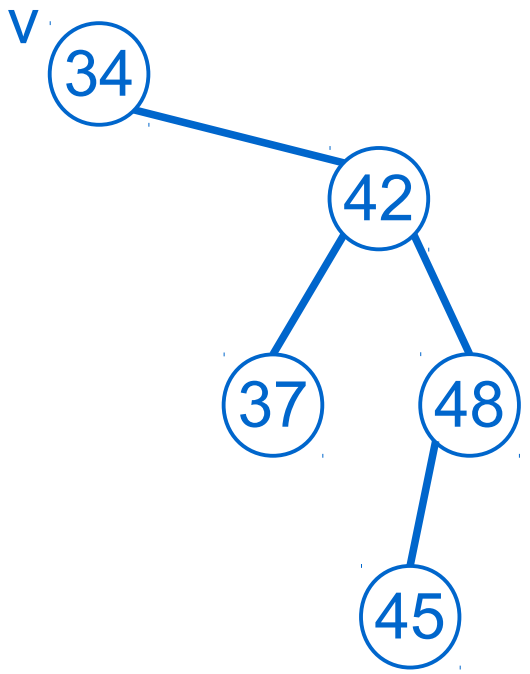
Διαγραφή (β)

Διαγραφή (β): Υποπερίπτωση: ο v δεν είναι η ρίζα.
Ο γονέας του v υιοθετεί το μοναδικό παιδί του v και μετά ο v διαγράφεται. Πχ. διαγραφή 40.



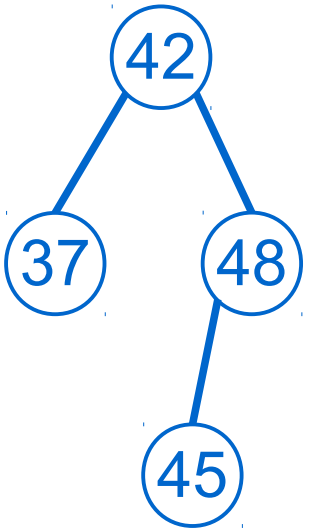
Διαγραφή (β)

Διαγραφή (β): Υποπερίπτωση: ο v είναι η ρίζα.
Το μοναδικό παιδί του v γίνεται η νέα ρίζα του δέντρου και ο v διαγράφεται.



Διαγραφή (β)

Διαγραφή (β): Υποπερίπτωση: ο v είναι η ρίζα.
Το μοναδικό παιδί του v γίνεται η νέα ρίζα του δέντρου και ο v διαγράφεται.



Διαγραφή (γ)

Διαγραφή (γ): Αντικαθιστούμε πρώτα το στοιχείο του κόμβου v προς διαγραφή με:

(1) το στοιχείο με το **μεγαλύτερο** κλειδί στο **αριστερό** υποδέντρο του v,

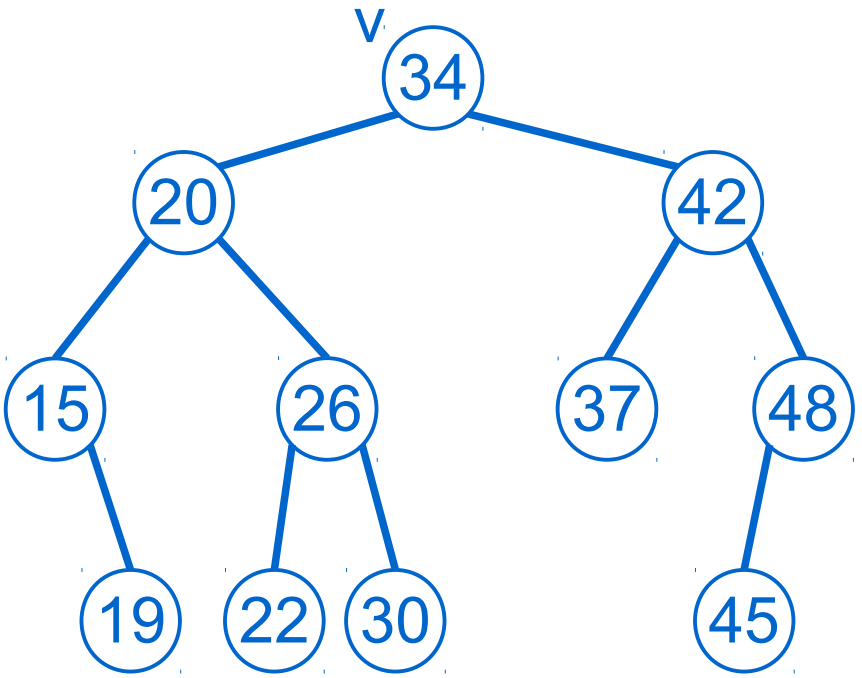
είτε, ισοδύναμα, με

(2) το στοιχείο με το **μικρότερο** κλειδί στο **δεξί** υποδέντρο του v.

- Και τα δύο αυτά στοιχεία βρίσκονται σε κόμβο u με κανένα ή μοναδικό παιδί.
- Οπότε στη συνέχεια διαγράφουμε τον u όπως στην διαγραφή (α) ή (β).

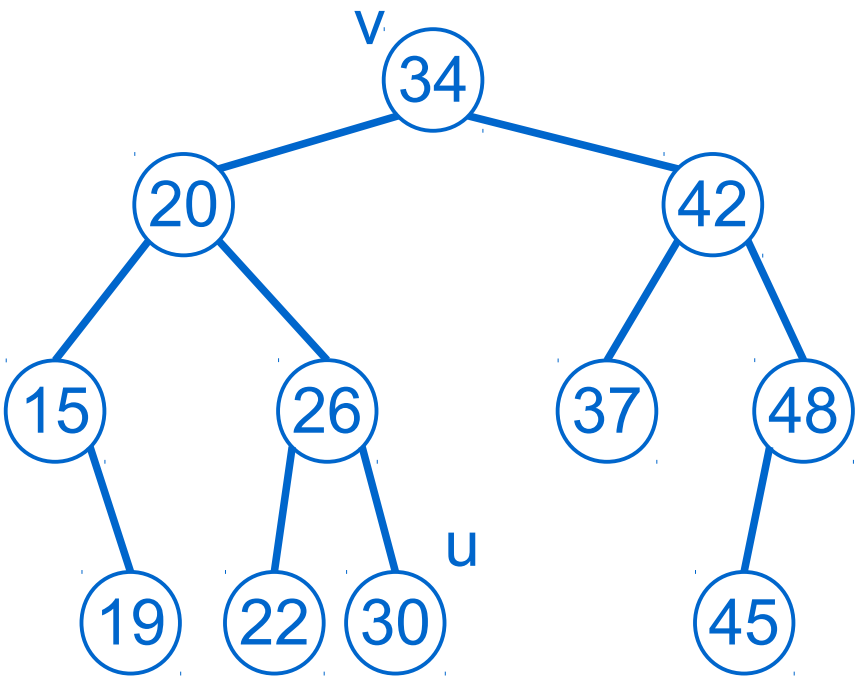
Διαγραφή (γ)

Διαγραφή (γ): Αντικαθιστούμε το στοιχείο του κόμβου v προς διαγραφή με: (1) το στοιχείο με το μεγαλύτερο κλειδί στο αριστερό υποδέντρο του v.



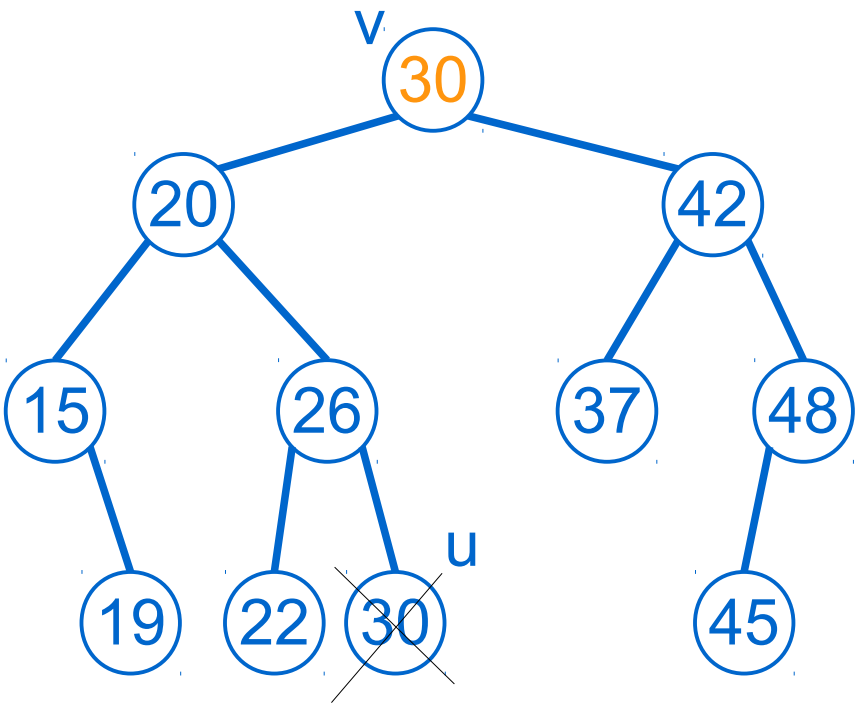
Διαγραφή (γ)

Διαγραφή (γ): Αντικαθιστούμε το στοιχείο του κόμβου v προς διαγραφή με: (1) το στοιχείο με το μεγαλύτερο κλειδί στο αριστερό υποδέντρο του v.



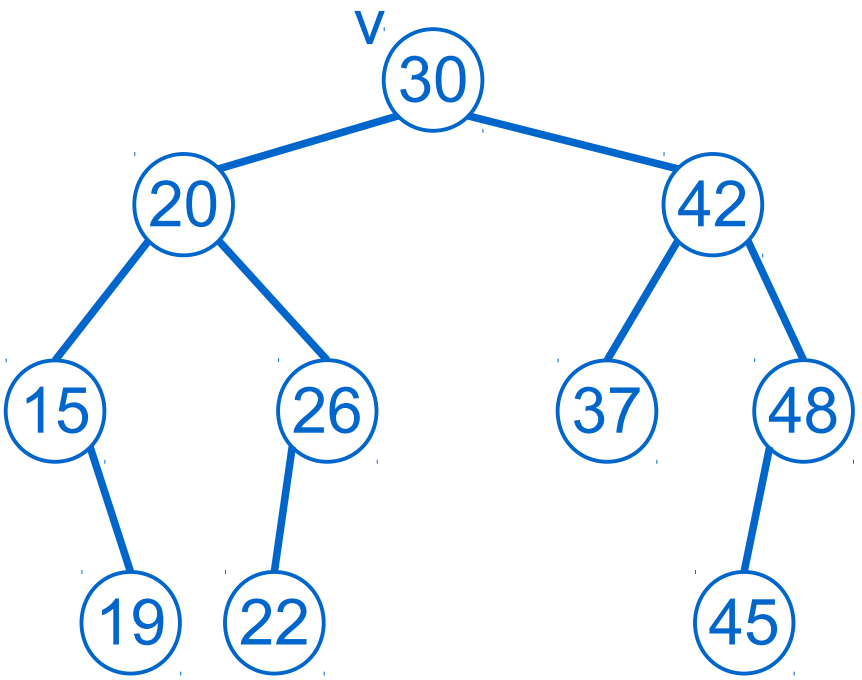
Διαγραφή (γ)

Διαγραφή (γ): Αντικαθιστούμε το στοιχείο του κόμβου v προς διαγραφή με: (1) το στοιχείο με το μεγαλύτερο κλειδί στο αριστερό υποδέντρο του v.



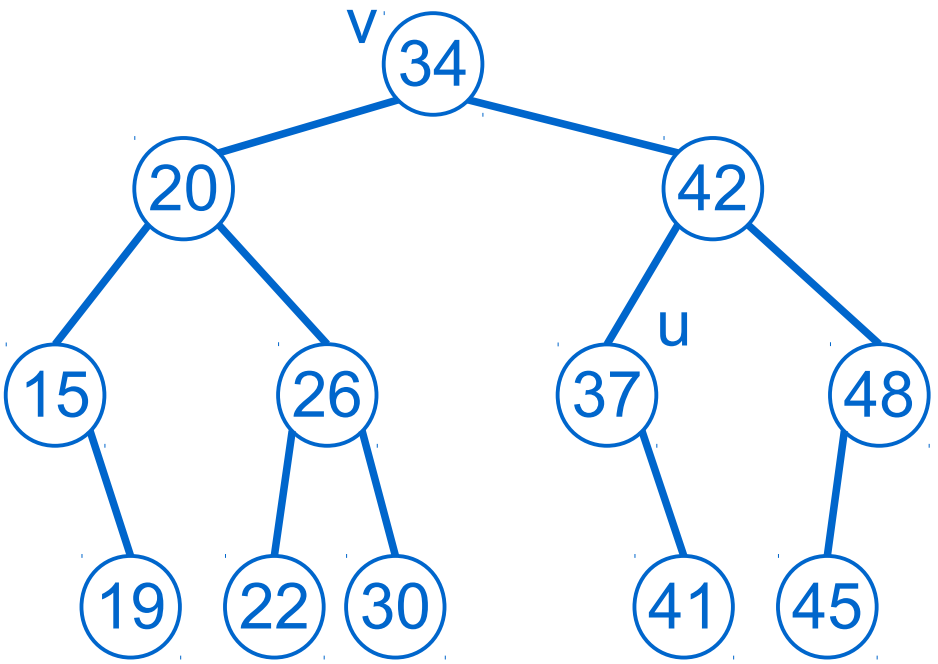
Διαγραφή (γ)

Διαγραφή (γ): Αντικαθιστούμε το στοιχείο του κόμβου v προς διαγραφή με: (1) το στοιχείο με το μεγαλύτερο κλειδί στο αριστερό υποδέντρο του v.



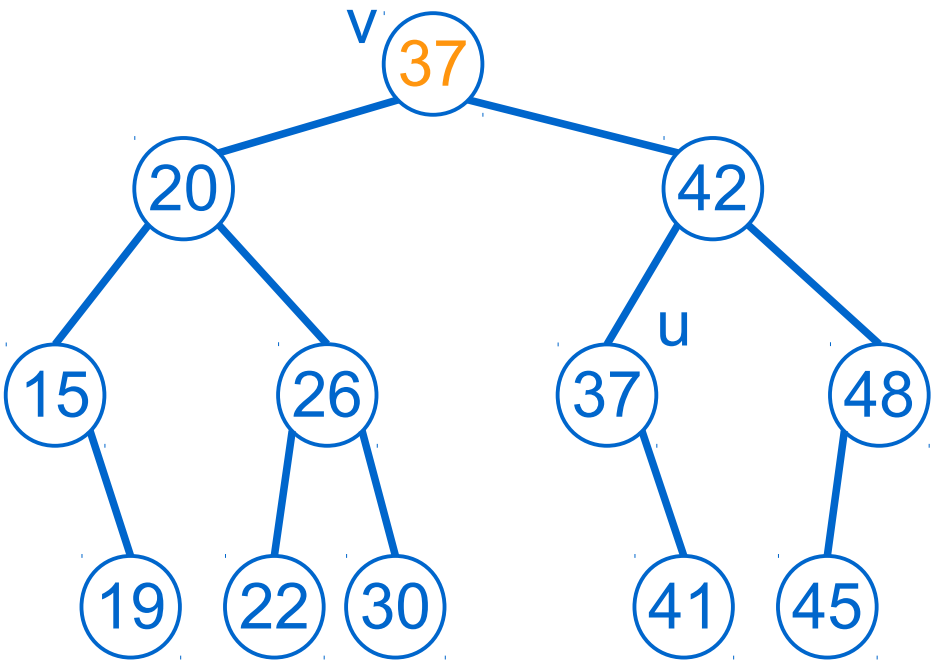
Διαγραφή (γ)

Διαγραφή (γ): Αντικαθιστούμε το στοιχείο του κόμβου v προς διαγραφή με: (2) το στοιχείο με το μικρότερο κλειδί στο δεξί υποδέντρο του v.



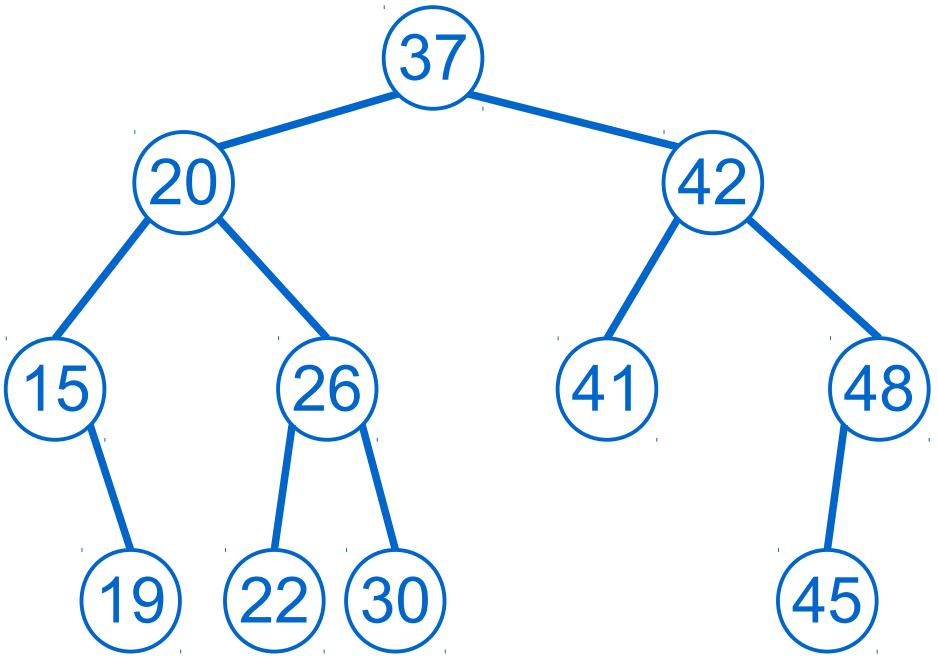
Διαγραφή (γ)

Διαγραφή (γ): Αντικαθιστούμε το στοιχείο του κόμβου v προς διαγραφή με: (2) το στοιχείο με το μικρότερο κλειδί στο δεξί υποδέντρο του v.



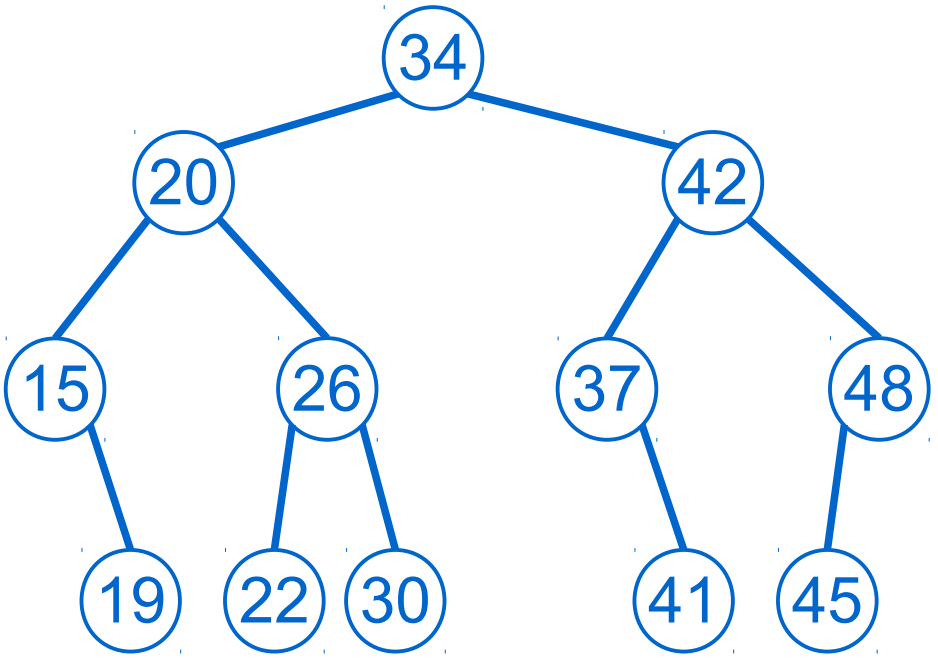
Διαγραφή (γ)

Διαγραφή (γ): Αντικαθιστούμε το στοιχείο του κόμβου v προς διαγραφή με: (2) το στοιχείο με το μικρότερο κλειδί στο δεξί υποδέντρο του v.



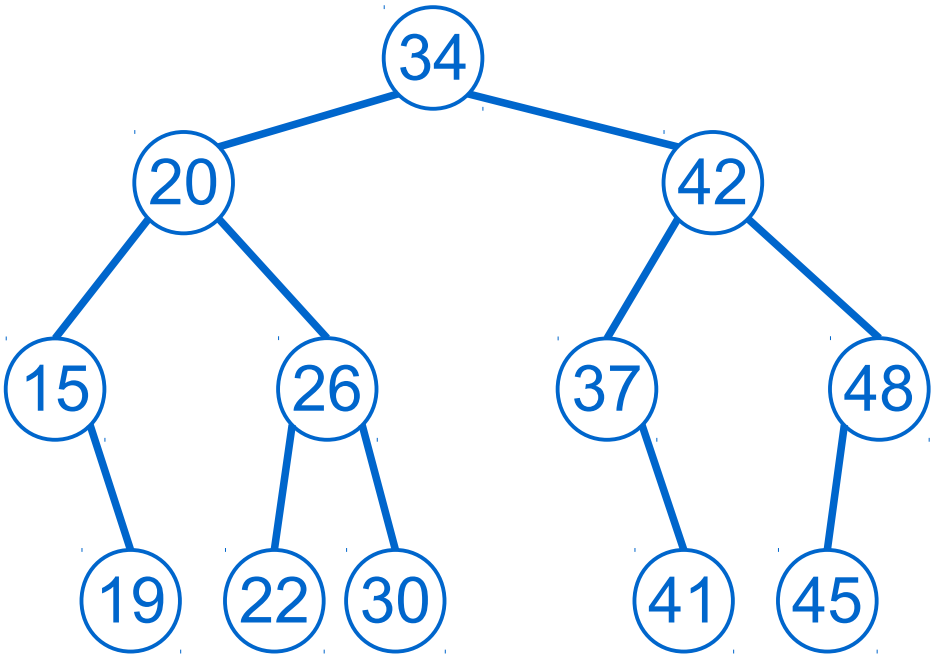
Επόμενος

Επόμενος: Έστω ένα στοιχείο με κλειδί key στον κόμβο v. Αναζητούμε το στοιχείο με το αμέσως μεγαλύτερο κλειδί από το key.



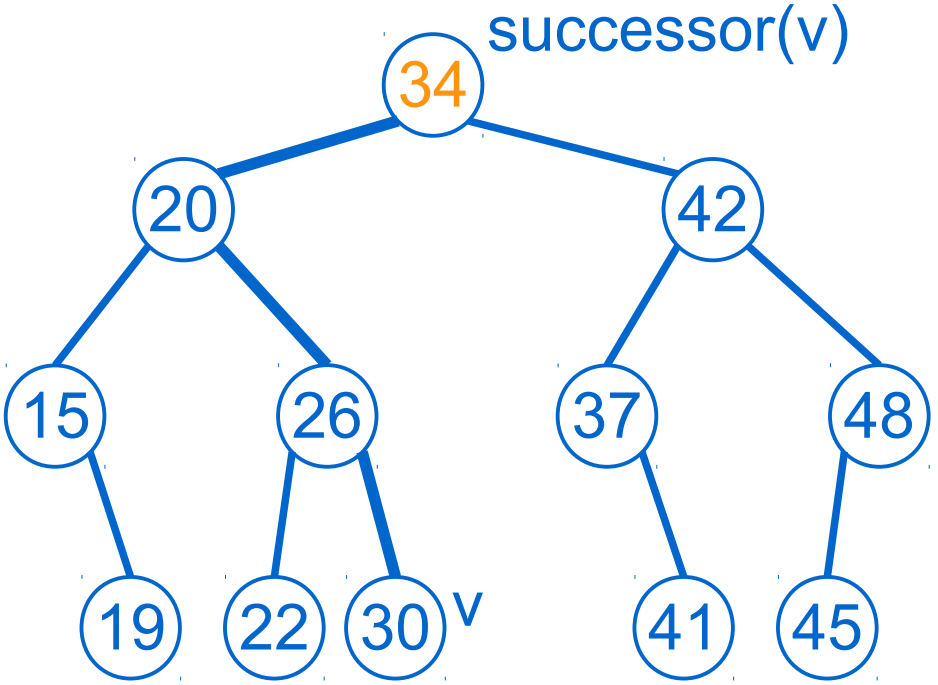
Επόμενος

Επόμενος: (α) αν ο v έχει δεξί υποδέντρο τότε επέλεξε τον αριστερότερο κόμβο στο δεξί υποδέντρο.
(β) αν όχι, επέλεξε τον πρώτο κόμβο στο μονοπάτι προς τη ρίζα με αριστερό παιδί πρόγονο του v, ή με άλλα λόγια τον κόμβο εκκίνησης της χαμηλότερης αριστερής ακμής στο ίδιο μονοπάτι.



Επόμενος

Επόμενος: (α) αν ο v έχει δεξί υποδέντρο τότε επέλεξε τον αριστερότερο κόμβο στο δεξί υποδέντρο.
(β) αν όχι, επέλεξε τον πρώτο κόμβο στο μονοπάτι προς τη ρίζα με αριστερό παιδί πρόγονο του v.



Προηγούμενος

Προηγούμενος: Έστω ένα στοιχείο με κλειδί key στον κόμβο v. Αναζητούμε το στοιχείο με το αμέσως μικρότερο κλειδί από το key. Συμμετρική υλοποίηση της λειτουργίας επόμενος.