Δένδρα

Στην ενότητα αυτή θα μελετηθούν τα εξής επιμέρους θέματα:

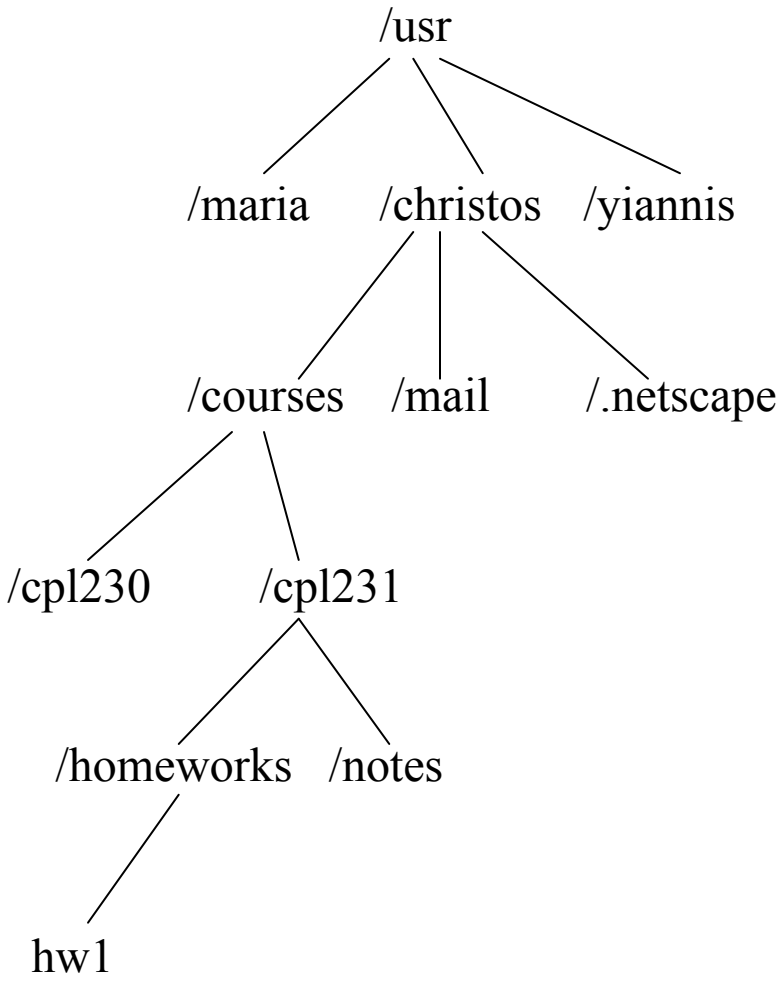
Εισαγωγή σε δενδρικές δομές δεδομένων, ορισμοί, πράξεις και αναπαράσταση στη μνήμη

Δυαδικά Δένδρα και Δυαδικά Δένδρα Αναζήτησης

Δένδρα

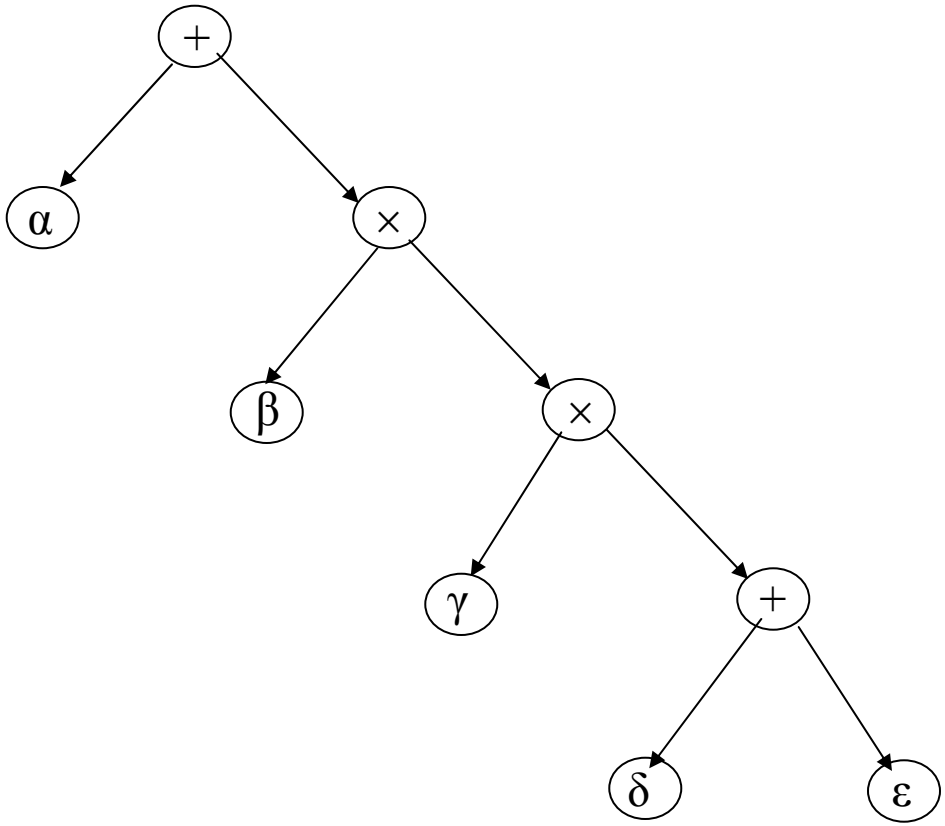
- Στις λίστες η εύρεση κόμβου έχει χρόνο εκτέλεσης O(N), όπου N το μήκος της λίστας.
- Στα δένδρα τα δεδομένα είναι 'καλύτερα' οργανωμένα ώστε οι διαδικασίες αναζήτησης, εισαγωγής και εξαγωγής κόμβου να είναι πιο αποδοτικές.
- ***Δένδρο*** είναι ένα σύνολο κόμβων που συνδέονται από ακμές ή τόξα. Ορίζεται αναδρομικά ως εξής:
  1. Ένα δένδρο είναι είτε κενό είτε
  2. αποτελείται από *μια ρίζα* (root), δηλαδή ένα κόμβο στον οποίο δεν καταλήγουν αλλά μόνο ξεκινούν ακμές, και *0 ή περισσότερα υποδένδρα* T_1, $T_{2,}$ …, T_k, το καθένα ξεχωριστό από τα άλλα και από τη ρίζα. Υπάρχει μια ακμή από τη ρίζα στις ρίζες των T_1, $T_{2,}$ …, T_k.
- Από κάθε κόμβο ξεκινούν 0 ή περισσότερες ακμές. Σε κάθε κόμβο εκτός της ρίζας καταλήγει μία μόνο ακμή. Στη ρίζα δεν καταλήγει καμιά ακμή.
- Οι κόμβοι περιέχουν δεδομένα. Οι ακμές επιβάλλουν μια ιεραρχική δομή στα δεδομένα.

Παράδειγμα: Unix File System



Παράδειγμα: Αναπαράσταση Αριθμητικών Παραστάσεων

$\alpha + \beta \times \gamma \times (\delta + \varepsilon)$



Ορισμοί

- Όταν υπάρχει ακμή από ένα κόμβο u σε ένα κόμβο v τότε ο u είναι ο **πατέρας** ή ο προκάτοχος (parent) του v, και ο v είναι **παιδί** (child) του u.
- Η ορολογία αυτή γενικεύεται ώστε να μιλούμε για **προγόνους** και **απογόνους** κόμβων.
- Κόμβοι που έχουν τον ίδιο προκάτοχο ονομάζονται **αδέλφια.**
- Κόμβοι που δεν έχουν παιδιά λέγονται **φύλλα**. Οι υπόλοιποι λέγονται **εσωτερικοί**.
- **Βαθμός (degree) ενός κόμβου** είναι ο αριθμός των παιδιών του. **Βαθμός ενός δένδρου** είναι ο μέγιστος από τους βαθμούς των κόμβων του.
- Δένδρα βαθμού n λέγονται n-αδικά. Δένδρα που χρησιμοποιούνται συχνά είναι τα δυαδικά (binary), τετραδικά (quadtrees), οκταδικά (octtrees).

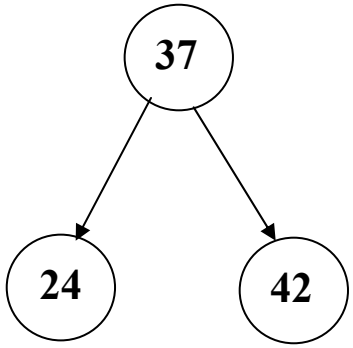
Βάθος και ύψος ενός δένδρου

- **Επίπεδο** ή **βάθος** ενός κόμβου ονομάζουμε τον αριθμό των προγόνων του + 1.
  π.χ. το βάθος της ρίζας ενός δένδρου είναι 1.
- **Βάθος** ενός δένδρου ονομάζουμε το μέγιστο επίπεδο κόμβων του δένδρου.
- **Μονοπάτι** ή **διαδρομή** (path) ενός δένδρου είναι μια ακολουθία κόμβων v_1, $v_{2,}$ …, v_k, όπου κάθε v_i είναι πατέρας του v_{i+1}. Το **μήκος** του μονοπατιού v_1, $v_{2,}$ …, v_k είναι k-1.
- **Ύψος** ενός κόμβου ονομάζουμε το μήκος του μέγιστου μονοπατιού από τον κόμβο προς κάποιο φύλλο (μετρούμε ακμές).
  π.χ. το ύψος οποιουδήποτε φύλλου είναι 0.
- **Ύψος** ενός δένδρου είναι το ύψος της ρίζας του δένδρου.
- **Διατεταγμένο δένδρο** ονομάζεται ένα δένδρο όταν για κάθε παιδί u κάποιου κόμβου v γνωρίζουμε αν ο u είναι το k-οστό παιδί του v.

Παράδειγμα

Ύψος: 0

Βάθος:1



Ύψος: 1

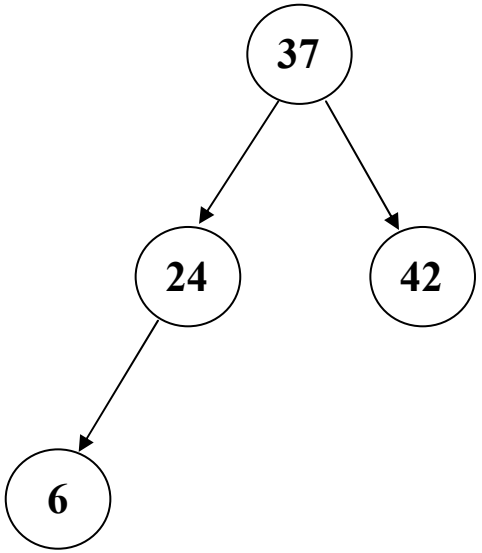
Βάθος: 2

Ύψος κόμβου 37: 1

Βάθος κόμβου 37: 1

Ύψος κόμβου 24: 0

Βάθος κόμβου 24: 2



Ύψος: 2

Βάθος: 3

Ύψος κόμβου 24: 1

Βάθος κόμβου 24: 2

Ύψος κόμβου 6: 0

Βάθος κόμβου 6: 3

Χρήσιμες Πράξεις

`Parent(u)`	δώσε τον πατέρα του u
`Children(u)`	δώσε τα παιδιά του u
`FirstChild(u)`	δώσε το πρώτο παιδί του u
`RightSibling(u)`	δώσε τον κόμβο στα δεξιά του u
`LeftSibling(u)`	δώσε τον κόμβο στα αριστερά του u
`IsLeaf(u)`	αν το u είναι φύλλο τότε δώσε true, διαφορετικά δώσε false
`IsRoot(u)`	αν το u είναι η ρίζα του δένδρου δώσε true, διαφορετικά δώσε false
`Depth(u)`	δώσε το βάθος του u στο δένδρο.

Αναπαράσταση Δένδρων στη Μνήμη

- Η πιο απλή και φυσική στρατηγική είναι η χρήση δυναμικής χορήγησης μνήμης.

- Κάθε κόμβος είναι ένα καταχώρημα μνήμης που αποτελείται από κάποιο αριθμό πεδίων:
  1. το πρώτο από τα οποία περιέχει τα *πραγματικά δεδομένα του κόμβου* (το κλειδί), και
  2. τα υπόλοιπα περιέχουν *στοιχεία σχετικά με τη δομή* του δένδρου, δηλαδή περιγράφουν με κάποιο τρόπο τη σχέση του συγκεκριμένου κόμβου με άλλους κόμβους του δένδρου.

- Για παράδειγμα, αν θέλουμε να αναπαραστήσουμε δένδρο βαθμού n κάθε κόμβος μπορεί να έχει την πιο κάτω μορφή:

Info	p_1	p_2	…	p_n

όπου ο p_i είναι δείκτης στο i-οστό παιδί του κόμβου.

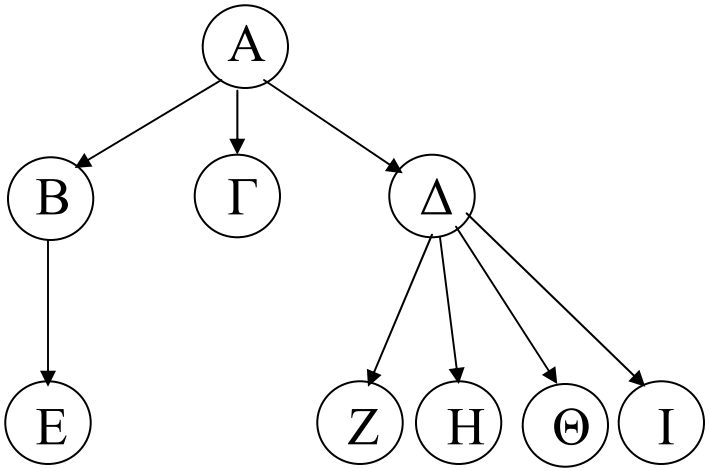
Αναπαράσταση Δένδρων στη Μνήμη

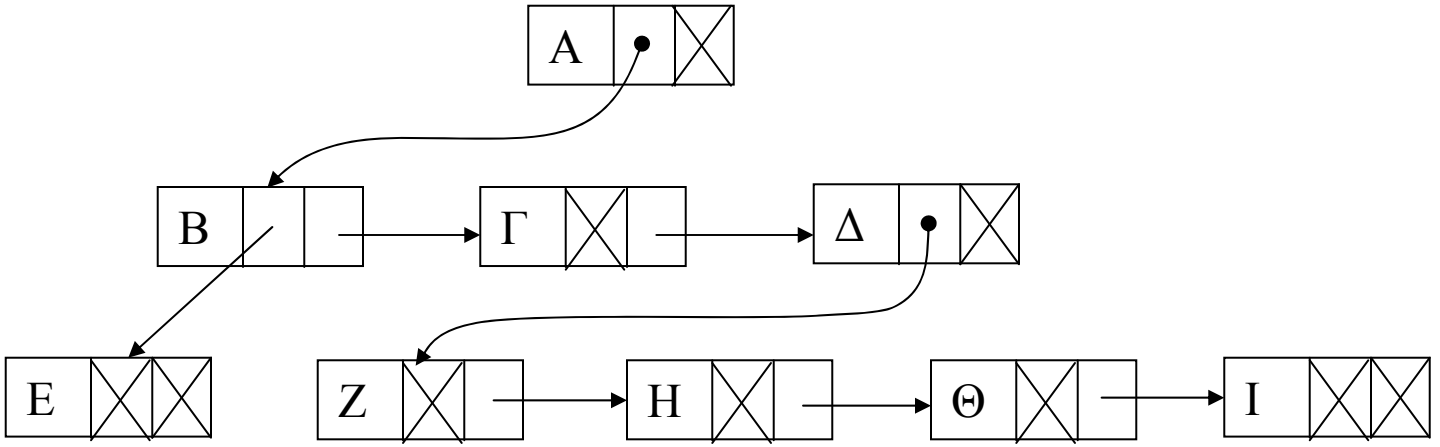
- Προβλήματα αυτού του είδους αναπαράστασης είναι
  1. αν το δένδρο δεν είναι πλήρες, τότε η μνήμη δεν αξιοποιείται αποτελεσματικά.
  2. πιθανόν ο βαθμός του δένδρου να είναι άγνωστος.

- Λύση: κάθε κόμβος u περιέχει
  1. τα δεδομένα (κλειδί),
  2. δείκτη στο πρώτο παιδί του u,
  3. δείκτη στον επί δεξιά αδελφό του u, αν υπάρχει.

- *Νέο πρόβλημα:* Πώς μπορούμε να επισκεφθούμε όλα τα παιδιά κάποιου κόμβου u;

Παράδειγμα αναπαράστασης δένδρου





Διάσχιση Δένδρων

- Αν θέλουμε να επισκεφθούμε όλους τους κόμβους ενός δένδρου, μπορούμε να χρησιμοποιήσουμε ένα από τους πιο κάτω τρόπους, οι οποίοι διαφέρουν στη σειρά με την οποία εξετάζουν τους κόμβους.

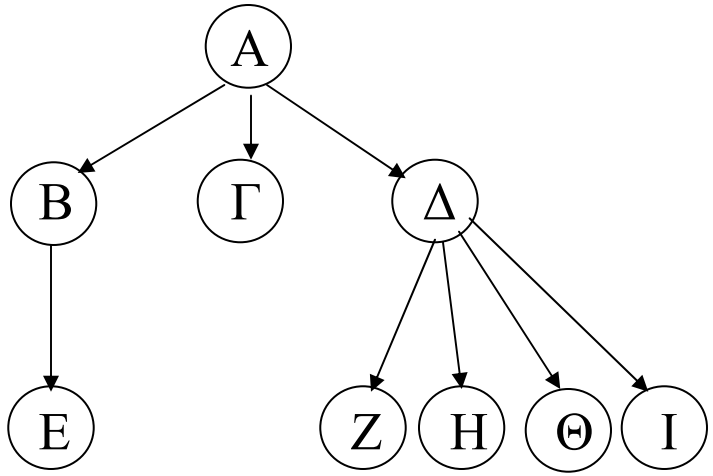
1. ***Προθεματική Διάσχιση***: (preorder traversal) επισκεπτόμαστε πρώτα τη ρίζα και ύστερα τα παιδιά της. Αναδρομικά η πράξη ορίζεται ως εξής:

```
Print_Preorder(*treenode u)
    Output data at u;
    for each child v of u
        Print_Preorder(v)
```

2. ***Μεταθεματική Διάσχιση***: (postorder traversal) επισκεπτόμαστε πρώτα τα παιδιά και ύστερα τη ρίζα του δένδρου. Αναδρομικά:

```
Print_Postorder(*treenode u)
    for each child v of u
        Print_Postorder(v)
    output data at u;
```

Παράδειγμα Διάσχισης δένδρου



Προθεματική Διάσχιση: Α Β Ε Γ Δ Ζ Η Θ Ι

Μεταθεματική Διάσχιση: Ε Β Γ Ζ Η Θ Ι Δ Α

Δυαδικά Δένδρα

- Ένα δένδρο είναι **δυαδικό** αν όλοι οι κόμβοι του έχουν βαθμό ≤ 2.
- Ορισμός: Δυαδικό δένδρο λέγεται ένα δένδρο το οποίο είτε είναι κενό, είτε αποτελείται από μια ρίζα και δύο δυαδικά υπόδενδρα, το καθένα διακριτό από το άλλο και από τη ρίζα. Αναφερόμαστε στα δύο υπόδενδρα ως το αριστερό και το δεξιό υπόδενδρο.
- Το ύψος ενός δυαδικού δένδρου με n κόμβους μπορεί να είναι το πολύ $n - 1$ και το λιγότερο $\lfloor \lg n \rfloor$.
- Ένα δυαδικό δένδρο είναι **γεμάτο** (full), αν κάθε εσωτερικός του κόμβος έχει δύο παιδιά.
- Ένα δυαδικό δένδρο είναι **τέλειο** (perfect), αν είναι γεμάτο και όλα τα φύλλα έχουν το ίδιο βάθος.

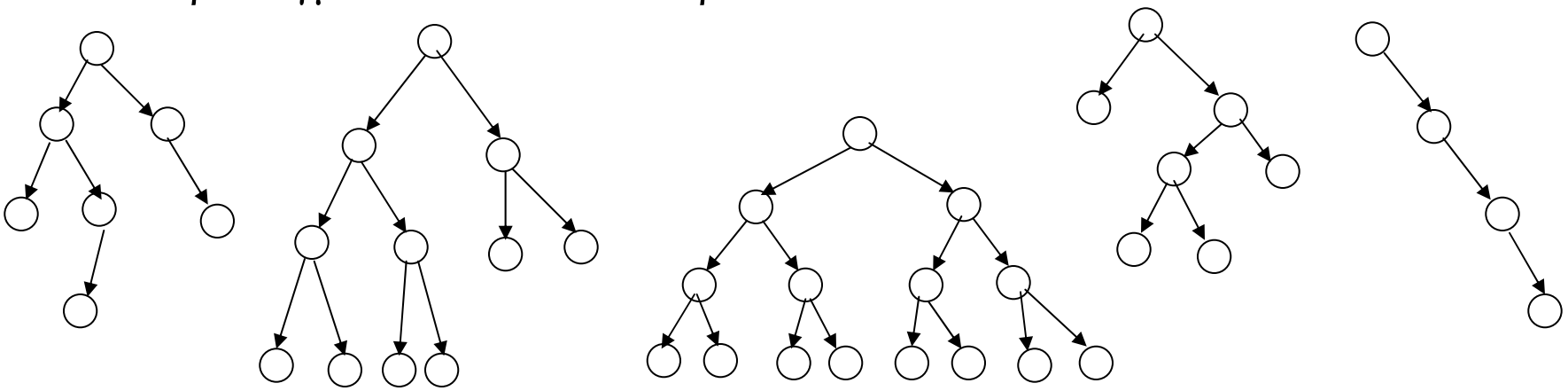
Δυαδικά Δένδρα

- Ένα δυαδικό δένδρο είναι **πλήρες** (complete) αν
  1. έχει ύψος 0 και ένα κόμβο,
  2. έχει ύψος 1 και η ρίζα του έχει είτε δύο παιδιά είτε ένα αριστερό παιδί.
  3. έχει ύψος h και η ρίζα του έχει ένα τέλειο αριστερό υπόδενδρο ύψους h-1 και ένα πλήρες δεξιό υπόδενδρο ύψους h-1,

     ή

     ένα πλήρες αριστερό υπόδενδρο ύψους h-1 και ένα τέλειο δεξιό υπόδενδρο ύψους h-2.

- Παραδείγματα δυαδικών δένδρων



Θεώρημα

1. *Ένα γεμάτο δυαδικό (μη άδειο) δένδρο με n εσωτερικούς κόμβους, έχει n+1 φύλλα.*
2. *Κάθε δυαδικό δένδρο με n κόμβους έχει n+1 null δείκτες*

Απόδειξη (2) με τη μέθοδο της μαθηματικής επαγωγής:

Βάση της επαγωγής: n = 0

Το δένδρο αποτελείται από ένα NULL δείκτη και το ζητούμενο έπεται.

Υπόθεση της επαγωγής: Έστω ότι κάθε δυαδικό δένδρο με k κόμβους, k < m, έχει k+1 NULL δείκτες.

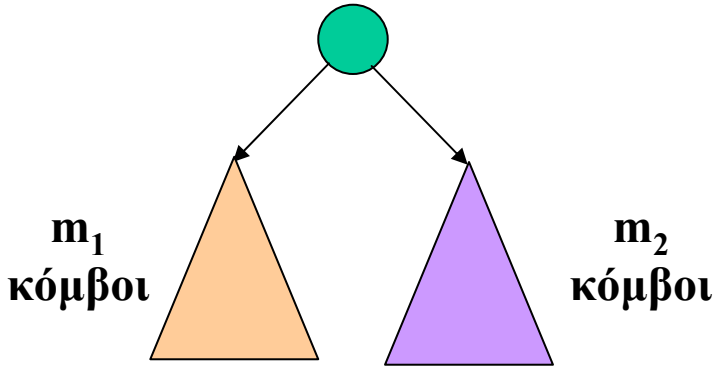
Βήμα της επαγωγής:

Έστω δυαδικό δένδρο με m κόμβους.

Θέλουμε να δείξουμε πως περιέχει m+1 NULL δείκτες.

Θεώρημα

Ένα οποιοδήποτε δυαδικό δένδρο έχει την πιο κάτω μορφή:

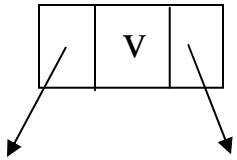


$$m_1 + m_2 + 1 = m$$

Ο αριθμός NULL δεικτών του δένδρου είναι
= ο αριθμός NULL δεικτών του αριστερού υποδένδρου
+
ο αριθμός NULL δεικτών του δεξιού υποδένδρου
= {από την υπόθεση της επαγωγής και αφού $m_1, m_2 < m$}
$(m_1+1) + (m_2+ 1) = (m_1+ m_2+1) + 1$
= $m + 1$

Υλοποίηση Δυαδικών Δένδρων

- Αφού κάθε κόμβος σε ένα δυαδικό δένδρο έχει το πολύ δύο παιδιά, μπορούμε να κρατούμε δείκτες στο καθένα από αυτά. Δηλαδή, ένας κόμβος μπορεί να υλοποιηθεί ως μια εγγραφή BTnode με τρία πεδία (παρόμοια με κόμβο διπλά συνδεδεμένης λίστας).
  1. `val`, όπου αποθηκεύουμε την πληροφορία του κόμβου,
  2. `left`, τύπου pointer, ο οποίος δείχνει το αριστερό, υπόδενδρο που ριζώνει στον συγκεκριμένο κόμβο, και
  3. `right`, τύπου pointer, ο οποίος δείχνει το δεξιό υπόδενδρο που ριζώνει στον συγκεκριμένο κόμβο.



- Έτσι, ένα δυαδικό δένδρο υλοποιείται ως ένας δείκτης στη ρίζα του δένδρου, δηλαδή δείκτης σε εγγραφή τύπου BTnode.

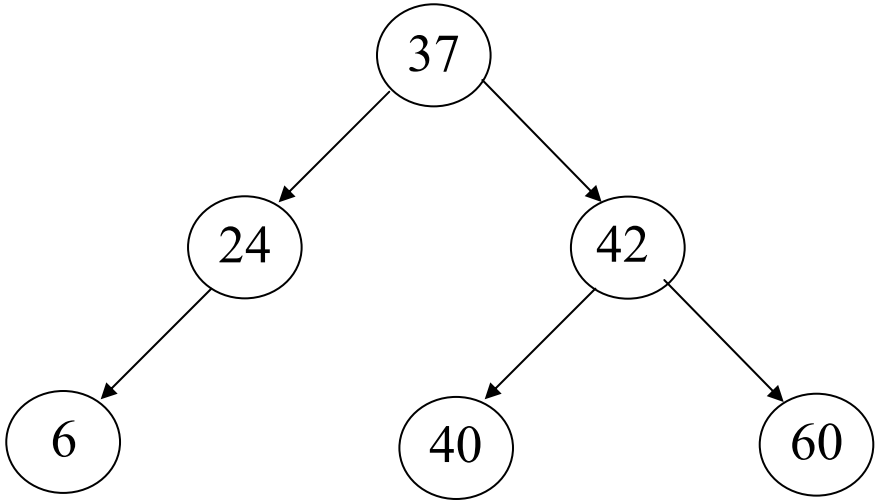
Δυαδικά Δένδρα Αναζήτησης (Binary Search Trees)

- Το πιο σημαντικό πλεονέκτημα της χρήσης δυαδικών δένδρων είναι η αποδοτική αναζήτηση σε ένα σύνολο στοιχείων.
- Υποθέτουμε την ύπαρξη μιας σχέσης στο σύνολο των στοιχείων που επεξεργαζόμαστε, έστω τη σχέση < πάνω στο σύνολο των ακεραίων.
- Ένα ***δυαδικό δένδρο αναζήτησης*** (ΔΔΑ) είναι ένα δυαδικό δένδρο κάθε κόμβος u του οποίου ικανοποιεί τα εξής:
  1. τα κλειδιά του αριστερού υποδένδρου του u είναι μικρότερα από το κλειδί του u
  2. τα κλειδιά του δεξιού υποδένδρου του u είναι μεγαλύτερα από το κλειδί του u.

```
struct  BSTnode{
        int   val;
        BSTnode     *left;
        BSTnode     *right;
}
```

Κτίζοντας ένα ΔΔΑ

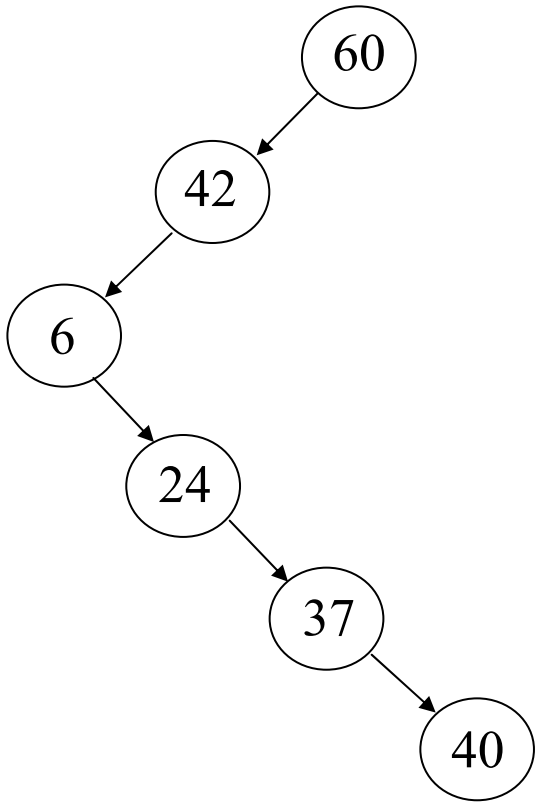
Εισαγωγή 37:
Εισαγωγή 24:
Εισαγωγή 42:
Εισαγωγή 6:
Εισαγωγή 40:
Εισαγωγή 60:



Κτίζοντας ένα ΔΔΑ

Δένδρο με τα ίδια στοιχεία τοποθετημένα με διαφορετική σειρά: 60, 42, 6, 24, 37, 40.

Εισαγωγή 60:
Εισαγωγή 42:
Εισαγωγή 6:
Εισαγωγή 24:
Εισαγωγή 37:
Εισαγωγή 40:



Διαδικασία Εύρεσης Στοιχείου

- Απλή αναδρομική στρατηγική: συγκρίνουμε το στοιχείο που μας ενδιαφέρει α με το στοιχείο της ρίζας του δένδρου β (αν υπάρχει) και
  1. αν α=β, σταματούμε,
  2. αν α<β, προχωρούμε στο αριστερό υπόδενδρο,
  3. αν α>β προχωρούμε στο δεξιό υπόδενδρο.

```
BSTnode* Find(BSTnode* r,int n){
    if (r == null) return null;
    else
        if (n < (*r).val)
            return Find((*r).left, n);
        else if (n == (*r).val) )
            return r;
        else return Find((*r).right,n);
}
```

Διαδικασία ??

```
BSTnode* X(BSTnode* p, int n){
    while (p != null){
          if (n < (*p).val)
                p = (*p).left;
          else
                if (n == (*p).val)
                      break;
                else
                      p = (*p).right;
    return p;
}
```

Διαδικασία Εισαγωγής Κόμβου

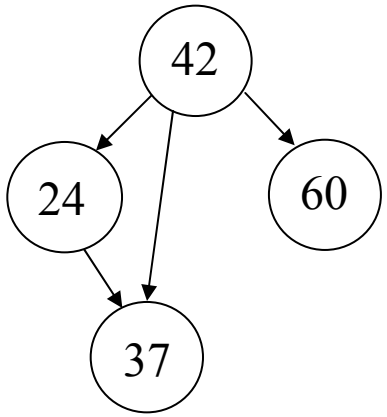
```
BSTnode* InsertNode(BSTnode* r, int n){

if (r == NULL)
      p=(BSTNODE *)malloc(sizeof(BSTNODE));
      (*p).val = n;
      (*p).left = NULL;
      (*p).right = NULL;
      return p;
if (n < (*r).val))
      (*r).left = InsertNode((*r).left, n);
if (n > (*r).val))
      (*r).right = InsertNode((*r).right, n);

return r;
}
```

Εξαγωγή του μικρότερου κόμβου

1. Ακολουθούμε τους αριστερούς δείκτες όσο μπορούμε, φθάνοντας στον κόμβο με το μικρότερο στοιχείο, u.
2. Βρίσκουμε τον πατέρα v του u και αλλάζουμε τον αριστερό δείκτη του v ώστε να δείχνει στο δεξιό παιδί του u.



Εξαγωγή του μικρότερου κόμβου

```
BSTnode* deleteMin(BSTnode* r){
  if r!= null
     if ((*r).left != null)
        (*r).left = deleteMin((*r).left);
        return r;
     else
        rightchild = (*r).right;
        free r;
        return rightchild;
}
```

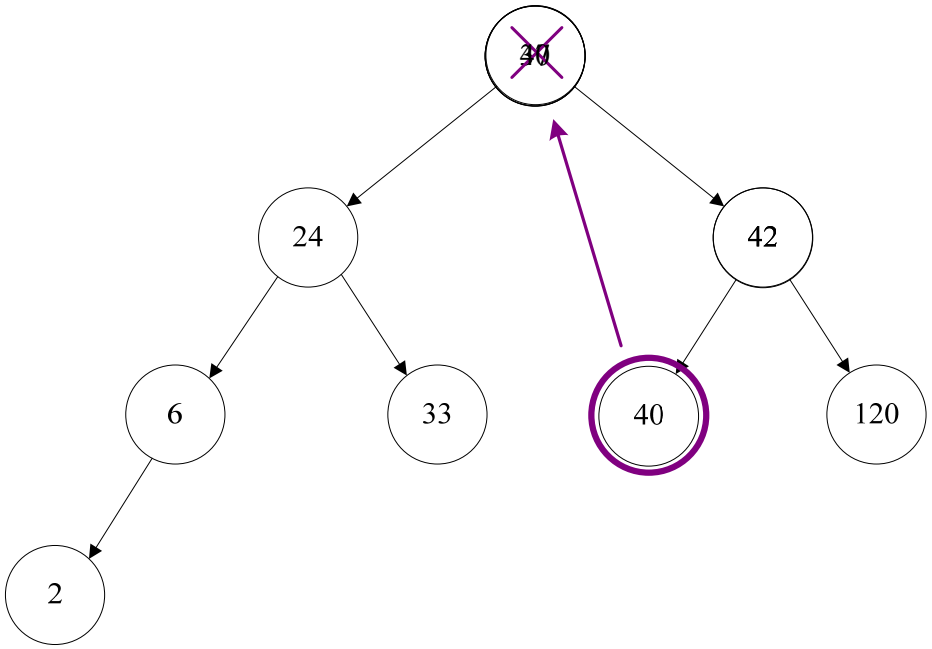
Διαγραφή κόμβου

Για να αφαιρέσουμε ένα κλειδί i από ένα ΔΔΑ:

1. Βρίσκουμε τον κόμβο u που περιέχει το i. Ας υποθέσουμε πως v είναι ο πατέρας του u.
2. Αν u είναι φύλλο, τότε αλλάζουμε τον δείκτη του v που δείχνει το u, ώστε να γίνει null.
3. Αν ο u έχει ένα παιδί, τότε αλλάζουμε τον δείκτη του v που δείχνει τον u, ώστε να δείχνει στο παιδί του u.
4. Αν ο u έχει δύο παιδιά,
   - αλλάζουμε το κλειδί του u ώστε να γίνει το μικρότερο από τα κλειδιά όλων των απογόνων του που έχουν κλειδιά μεγαλύτερα του i.
   - καλούμε τη διαδικασία DeleteMin στο δεξιό παιδί του u.

Παράδειγμα Διαγραφής Κόμβου

- Αφαίρεση του στοιχείου 37:



Διαγραφή Κόμβου

- Το κόστος κάθε διαδικασίας είναι ανάλογο του βάθους/ύψους του δένδρου.
  - Θ(N) στη χειρότερη περίπτωση,
  - Θ(log N) στη μέση περίπτωση, υποθέτοντας πως όλα τα δένδρα είναι εξίσου πιθανά.
- Σημειώστε πως η διαδικασία διαγραφής κόμβου που έχουμε περιγράψει αφαιρεί πάντα κόμβο δεξιού υποδένδρου. Έτσι βοηθά την ύπαρξη μιας ασυμμετρίας στη δομή των ΔΔΑ (πιο βαθιά στα αριστερά).
- *Ερώτημα: Μπορούμε, κατά τις εισαγωγές και εξαγωγές στοιχείων, να οργανώνουμε ένα δένδρο έτσι ώστε το ύψος του να παραμένει όσο το δυνατό πιο μικρό (τάξεως O(log n));*