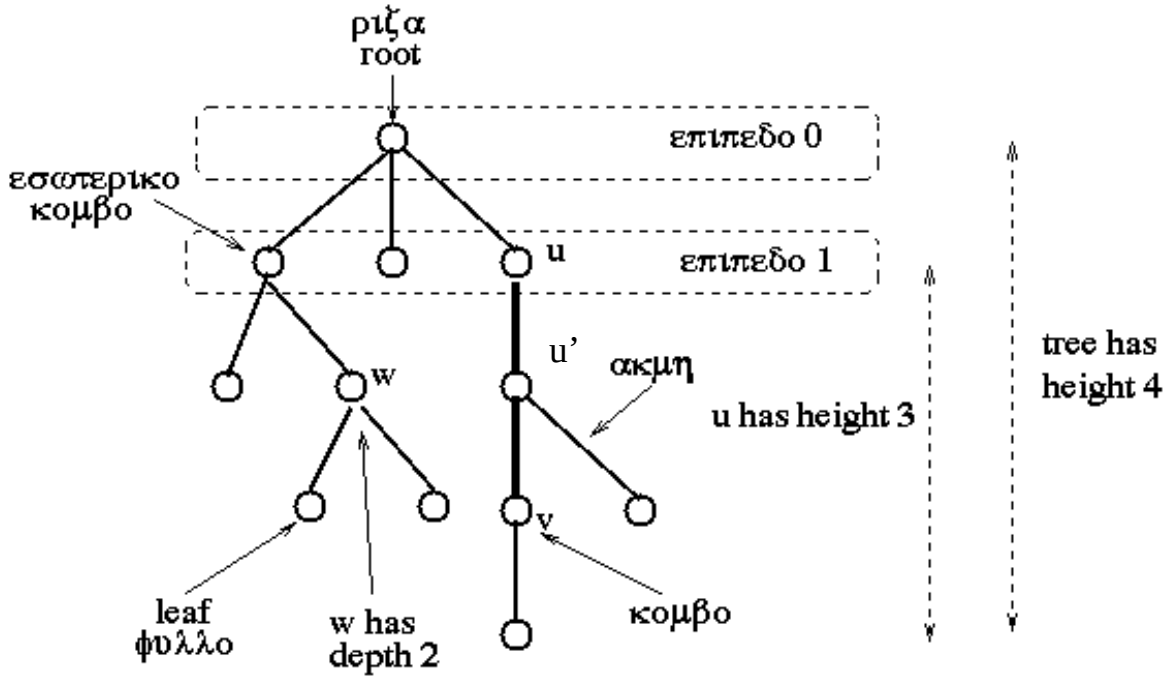
ΕΝΟΤΗΤΑ 4
ΔΕΝΔΡΑ

Δένδρα



- Κόμβοι (nodes)
- Ακμές (edges)
- Ουρά και κεφαλή ακμής (tail, head)
- Γονέας – Παιδί – Αδελφικός κόμβος (parent, child, sibling)
- Μονοπάτι (path)
- Πρόγονος – απόγονος (ancestor, descendant)
- Φύλλο – Εσωτερικός Κόμβος (leaf, non-leaf)

Βαθμός Κόμβου (node degree)

Ο αριθμός των υποδένδρων που αρχίζουν από ένα κόμβο.

Βαθμός Δένδρου (tree degree)

Μέγιστος βαθμός των κόμβων του δένδρου.

Επίπεδο (level)

Η ρίζα βρίσκεται στο επίπεδο 0. Ένας κόμβος βρίσκεται στο επίπεδο k αν η απόσταση του από τη ρίζα είναι k. Το επίπεδο είναι επομένως ένα σύνολο από κόμβους.

Ύψος Κόμβου (node height)

Μήκος μακρύτερου μονοπατιού από τον κόμβο σε οποιοδήποτε φύλλο.

Ύψος Δένδρου (tree height)

Μέγιστο ύψος μεταξύ των κόμβων του δένδρου.

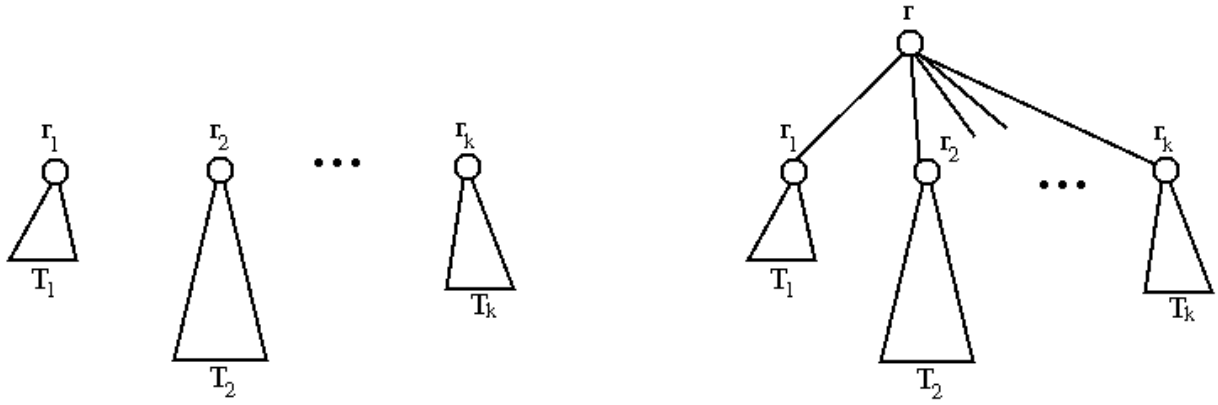
Βάθος Κόμβου (node depth)

Μήκος μονοπατιού από τη ρίζα στον κόμβο.

Βάθος Δένδρου (tree depth)

Μέγιστο βάθος μεταξύ των κόμβων του δένδρου.

Αναδρομικός Ορισμός



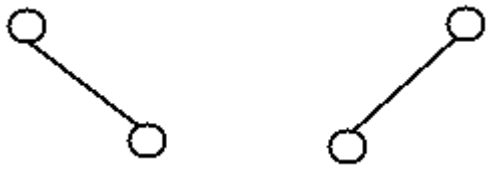
Ένα δένδρο Τ είναι ένα πεπερασμένο σύνολο από έναν ή περισσότερους κόμβους:

- Ένας μόνο κόμβος (χωρίς καμία ακμή) αποτελεί ένα δένδρο. Ο κόμβος αυτός είναι και ρίζα του δένδρου.
- Έστω ότι T_1, ..., T_κ ($\kappa > 0$) είναι δένδρα που δεν μοιράζονται κόμβους και έστω r_1, ..., r_k οι ρίζες τους. Έστω r ένας νέος κόμβος. Αν το Τ αποτελείται από τους κόμβους και τις ακμές των T_1, ..., T_κ, το νέο κόμβο r και τις νέες ακμές $<r,r_1>$, $<r,r_2>$, …, $<r,r_k>$, τότε το Τ είναι δένδρο. Η ρίζα του Τ είναι το r. Τα T_1, ..., T_κ είναι υποδένδρα του Τ.

Είδη Δένδρων

Διατεταγμένο Δένδρο

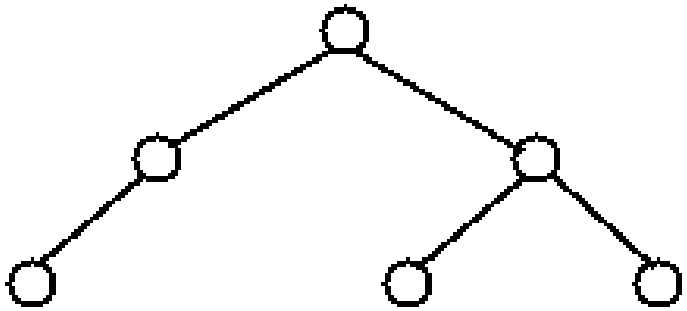
Δένδρο στο οποίο έχει οριστεί μια διάταξη στα παιδιά κάθε κόμβου.



Δυαδικό δένδρο

Διατεταγμένο δένδρο του οποίου κάθε κόμβος έχει το πολύ δύο παιδιά (ένα αριστερό και ένα δεξί).

Λ (nill ή NULL): άδειο δυαδικό δένδρο (που δεν περιέχει κανένα κόμβο και άρα και καμία ακμή)



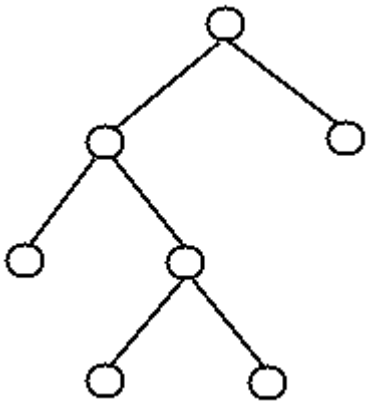
Δάσος

Πεπερασμένο σύνολο από δένδρα.

Είδη και Ιδιότητες Δυαδικών Δένδρων

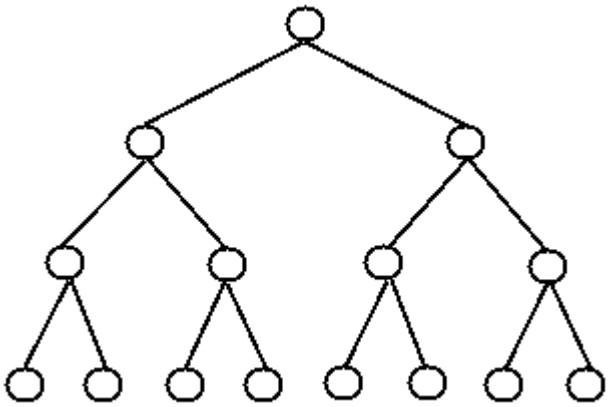
Γεμάτο Δυαδικό Δένδρο (full binary tree)

Δεν υπάρχει κόμβος με μόνο 1 παιδί στο δένδρο.



Τέλειο Δυαδικό Δένδρο (perfect binary tree)

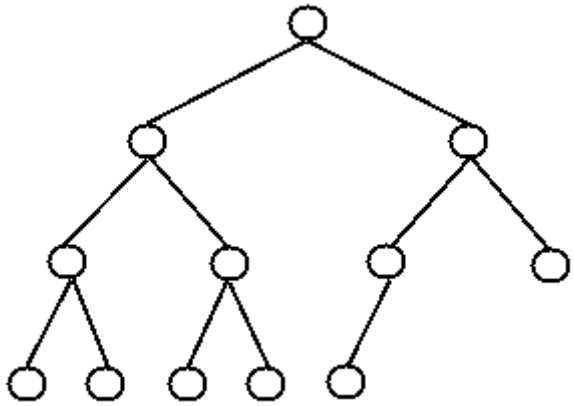
Γεμάτο δυαδικό δένδρο στο οποίο όλα τα φύλλα έχουν το ίδιο βάθος.



Είδη και Ιδιότητες Δυαδικών Δένδρων

Πλήρες Δυαδικό Δένδρο Ύψους h (complete binary tree of height h)

Αποτελείται από ένα τέλειο δυαδικό δένδρο ύψους h-1 στο οποίο έχουν προστεθεί ένα ή περισσότερα φύλλα με ύψος h. Τα φύλλα αυτά έχουν τοποθετηθεί στις αριστερότερες θέσεις του δένδρου.



Αναδρομικός Ορισμός

- Ένα πλήρες δυαδικό δένδρο ύψους 0 αποτελείται από ένα μόνο κόμβο.
- Ένα πλήρες δυαδικό δένδρο ύψους 1 είναι ένα δένδρο ύψους 1 στο οποίο η ρίζα έχει είτε 2 παιδιά ή ένα μόνο **αριστερό** παιδί.
- Ένα πλήρες δυαδικό δένδρο ύψους h>1, αποτελείται από μια ρίζα και 2 υποδένδρα τ.ω:
  - είτε το αριστερό υποδένδρο είναι τέλειο ύψους h-1 και το δεξιό είναι πλήρες ύψους h-1, ή
  - το αριστερό υποδένδρο είναι πλήρες ύψους h-1 και το δεξιό είναι τέλειο ύψους h-2.

Ιδιότητες Δυαδικών Δένδρων

Πρόταση

Ένα τέλειο δυαδικό δένδρο ύψους h έχει $2^{h+1} - 1$ κόμβους, εκ των οποίων 2^h είναι φύλλα και $2^h - 1$ είναι εσωτερικοί κόμβοι.

Απόδειξη

Με επαγωγή στο h.

Βάση επαγωγής, h = 0

Το τέλειο δυαδικό δένδρο ύψους 0 αποτελείται μόνο από ένα κόμβο-ρίζα και άρα έχει 1 κόμβο που είναι φύλλο και 0 εσωτερικούς κόμβους. Πράγματι:

$2^{h+1} - 1 = 2^{0+1} - 1 = 2 - 1 = 1$ κόμβος

$2^h = 2^0 = 1$ φύλλο

$2^h - 1 = 0$ εσωτερικοί κόμβοι

Επαγωγική Υπόθεση

Έστω ότι ένα τέλειο δυαδικό δένδρο ύψους κ έχει $2^{\kappa+1} - 1$ κόμβους, εκ των οποίων 2^κ είναι φύλλα και $2^\kappa - 1$ είναι εσωτερικοί κόμβοι, $\kappa >= 0$.

Απόδειξη Πρότασης (Συνέχεια)

Επαγωγικό βήμα

Θα δείξουμε ότι ο ισχυρισμός είναι σωστός για δένδρα ύψους k+1.

Ένα τέλειο δένδρο Τ ύψους k+1 αποτελείται από 2 τέλεια δένδρα ύψους k (έστω T_1, T_2) και τη ρίζα του. Από επαγωγική υπόθεση καθένα από τα T_1, T_2, έχει $2^{\kappa+1} - 1$ κόμβους, εκ των οποίων 2^{κ} είναι φύλλα και $2^{\kappa} - 1$ είναι εσωτερικοί κόμβοι. Άρα το Τ έχει

- $2*(2^{\kappa+1} - 1) + 1$ κόμβους = $2^{\kappa+2} - 1$ κόμβους (όπως απαιτείται),

εκ των οποίων

- $2^{\kappa} + 2^{\kappa} = 2^{\kappa+1}$ είναι φύλλα (όπως απαιτείται), και

- $2*(2^{\kappa} - 1) + 1 = 2^{\kappa+1} - 1$ είναι εσωτερικοί κόμβοι (όπως απαιτείται).

Λειτουργίες σε Δένδρα

Parent(v): επιστρέφει τον κόμβο γονέα του v ή nill αν ο v είναι η ρίζα

Children(v): επιστρέφει το σύνολο των παιδιών του v ή το άδειο σύνολο αν ο v είναι φύλλο

FirstChild(v): επιστρέφει το πρώτο παιδί του v ή nill αν ο v είναι φύλλο

RightSibling(v): επιστρέφει το δεξιό αδελφικό κόμβο του v ή nill αν ο v είναι η ρίζα ή το δεξιότερο παιδί του γονικού του κόμβου

LeftSibling(v): επιστρέφει τον αριστερό αδελφικό κόμβο του v ή nill αν ο v είναι η ρίζα ή το αριστερότερο παιδί του γονικού του κόμβου

LeftChild(v), RightChild(v): επιστρέφει το αριστερό/δεξιό παιδί του v (ή nill)

IsLeaf(v): επιστρέφει true αν ο v είναι φύλλο, false διαφορετικά

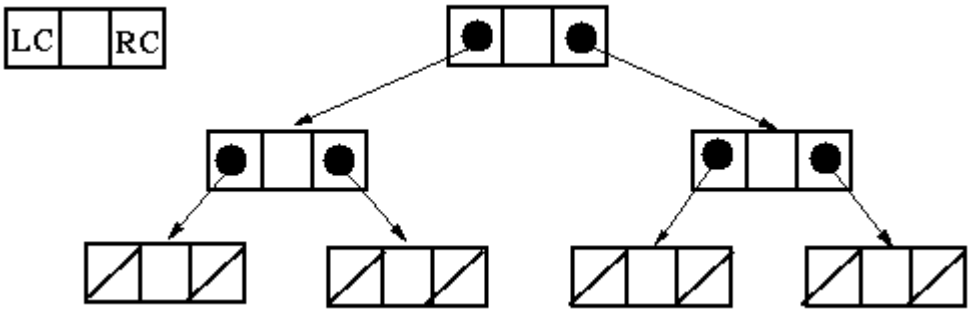
Depth(v): επιστρέφει το βάθος του v στο δένδρο

Height(v): επιστρέφει το ύψος του v στο δένδρο

Υλοποίηση Δένδρων

Υλοποίηση Δυαδικών Δένδρων

Κάθε κόμβος έχει ένα πεδίο Info και 2 δείκτες LC (Left Child) και RC (Right Child) που δείχνουν στο αριστερό και στο δεξιό παιδί του κόμβου αντίστοιχα.



Οι λειτουργίες LeftChild() και RightChild() υλοποιούνται πολύ εύκολα σε Θ(1) χρόνο.

Είναι το ίδιο αλήθεια για τη λειτουργία Parent()?

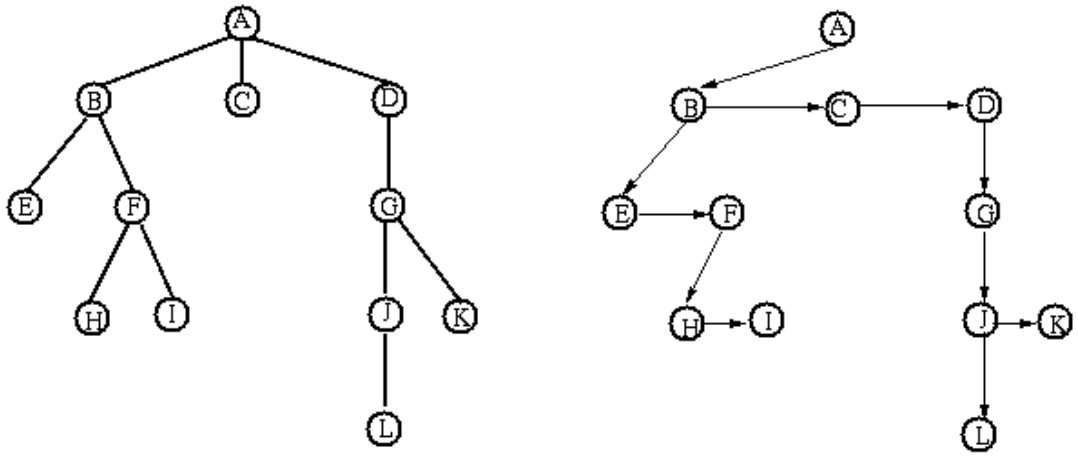
Αποδοτική Υλοποίηση της Parent()

Κρατάμε και ένα τρίτο δείκτη σε κάθε κόμβο που δείχνει στον πατρικό του κόμβο («Διπλά Συνδεδεμένο» δένδρο).

Υλοποίηση Διατεταγμένων Δένδρων

Τι γίνεται αν δεν γνωρίζουμε τον αριθμό των παιδιών που μπορεί να έχει κάποιος κόμβος?

Απεικόνιση Διατεταγμένου σαν Δυαδικό Δένδρο



Έχουμε χάσιμο πληροφορίας?

Μπορούμε να ξαναχτίσουμε το αρχικό δένδρο από το δυαδικό δένδρο?

Ένα δυαδικό δένδρο μπορεί να απεικονίσει και ένα δάσος από διατεταγμένα δένδρα (δεξιός αδελφικός κόμβος της ρίζας είναι σε αυτή την περίπτωση η ρίζα του επόμενου δένδρου στο δάσος)

Ύψος δυαδικού σε σχέση με το αρχικό δένδρο?

Πολυπλοκότητα:

FirstChild(), RightSibling(): Θ(1) χρόνο

K^{th}-child(k, v): εύρεση του κ-οστού παιδιού του v σε Θ(k) χρόνο.

Η Parent() δεν υποστηρίζεται αποδοτικά.

Απεικόνιση Πλήρων Δυαδικών Δένδρων

Υπάρχει μόνο ένα πλήρες δυαδικό δένδρο με n κόμβους και το υλοποιούμε με ένα πίνακα n στοιχείων.

Αριθμούμε τους κόμβους 1...n και αποθηκεύουμε τον κόμβο i στο στοιχείο T[i] του πίνακα.

Θέλουμε να κάνουμε την αρίθμηση με τέτοιο τρόπο ώστε να πετύχουμε την εκτέλεση χρήσιμων λειτουργιών στο δένδρο σε σταθερό χρόνο.

Αρίθμηση:

- Η ρίζα είναι ο κόμβος 0.
- Το αριστερό παιδί του κόμβου i αριθμείται ως κόμβος 2i+1, ενώ το δεξί παιδί του ως κόμβος 2i+2.

Υλοποίηση Λειτουργιών

IsLeaf(i): return (2i+1>n);

LeftChild(i): if (2i+1 < n); return (2i+1) else return nill;

RightChild(i): if (2i+2<n) return(2i+2); else return nill;

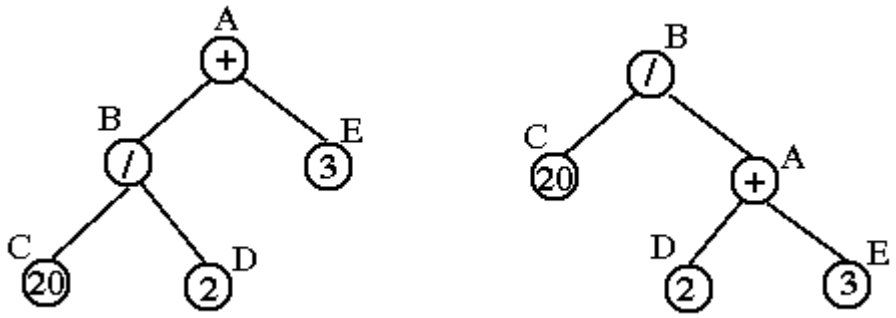
LeftSibling(i): if (i != 0 and i not odd) return (i-1);

RightSibling(i): if (i != n-1 and i not even) return(i+1);

Parent(i): if (i != 0) return($\lfloor (i-1)/2 \rfloor$);

Χρονική πολυπλοκότητα κάθε λειτουργίας: Θ(1)

Δένδρα Αριθμητικών Εκφράσεων



Υπολογισμός Αριθμητικής Έκφρασης

Label(v): ο αριθμός ή η πράξη που αναγράφεται στον v

ApplyOp(op: operation, x,y: numbers): υπολογίζει την έκφραση x <op> y, ανάλογα με το τι είναι το op.

function *Evaluate(pointer P):* **integer**

```
/* Return value of the expression represented by
   the tree with root P */

   if IsLeaf(P) then return Label(P)
   else
        x_l = Evaluate(LeftChild(P))
        x_r = Evaluate(RightChild(P))
        op = Label(P)
        return  ApplyOp(op, x_l, x_r)
```

Διάσχιση Δένδρων

Visit(P): αυθαίρετη λειτουργία που εφαρμόζεται στον κόμβο στον οποίο δείχνει ο δείκτης P

Προδιατεταγμένη Διάσχιση:

Επίσκεψη της ρίζας

Επίσκεψη του αριστερού υποδένδρου

Επίσκεψη του δεξιού υποδένδρου

```
Procedure Preorder(pointer P):
/* P is a pointer to the root of a binary tree */

    Visit(P)
    foreach child Q of P, in order, do
        Preorder(Q)
```

Μεταδιατεταγμένη διάσχιση:

Επίσκεψη του αριστερού υποδένδρου

Επίσκεψη του δεξιού υποδένδρου

Επίσκεψη της ρίζας

```
Procedure Postorder(pointer P):
/* P is a pointer to the root of a binary tree */

    foreach child Q of P, in order, do
        Postorder(Q)
    Visit(P)
```

Διάσχιση Δένδρων

Ενδοδιατεταγμένη διάσχιση:

Επίσκεψη του αριστερού υποδένδρου

Επίσκεψη της ρίζας

Επίσκεψη του δεξιού υποδένδρου

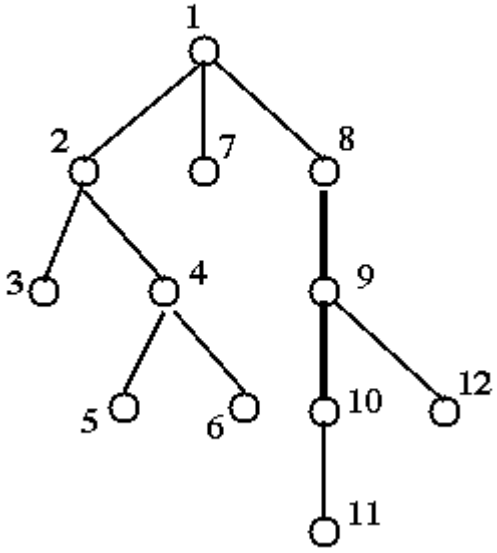
```
Procedure Inorder(pointer P):
/* P is a pointer to the root of a binary tree */

    if P = NULL then return
    else
        Inorder(P->LC)
        Visit(P)
        Inorder(P->RC)
```

Μνημονικός Κανόνας

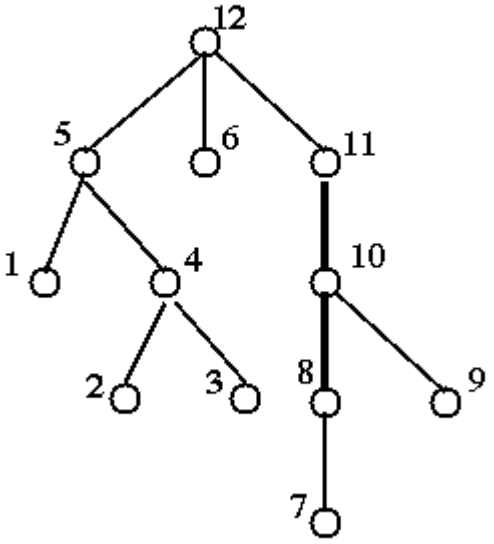
Στην προδιατεταγμένη μορφή επισκεπτόμαστε τον κόμβο πριν από τα παιδιά του, στη μεταδιατεταγμένη μορφή επισκεπτόμαστε τον κόμβο μετά από τα παιδιά του και στην ενδοδιατεταγμένη μορφή επισκεπτόμαστε τον κόμβο μεταξύ των παιδιών του.

Διάσχιση Δένδρων

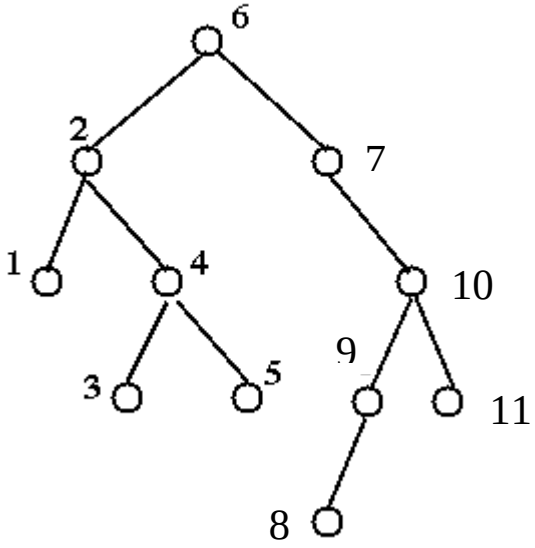
Προδιατεταγμένη



Μεταδιατεταγμένη



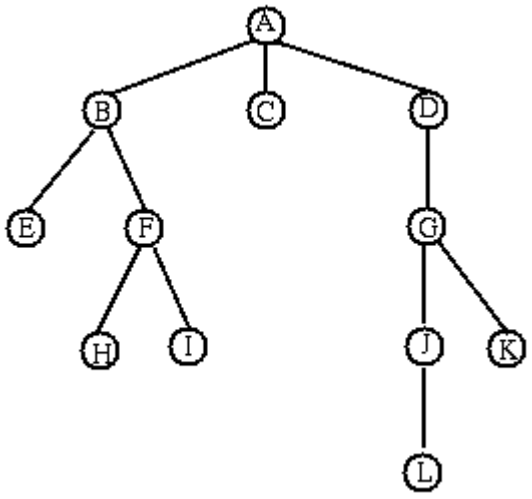
Ενδοδιατεταγμένη



Διάσχιση Δένδρων

Διάσχιση Δένδρου κατά Επίπεδα (κατά πλάτος)

Επισκέπτεται τους κόμβους κατά αύξον βάθος και τους κόμβους του ίδιου επιπέδου από τα αριστερά προς τα δεξιά.



Υλοποιήσεις Διάσχισης Δένδρων

Με Χρήση Στοίβας

Αναδρομικές λύσεις έχουν ήδη συζητηθεί.

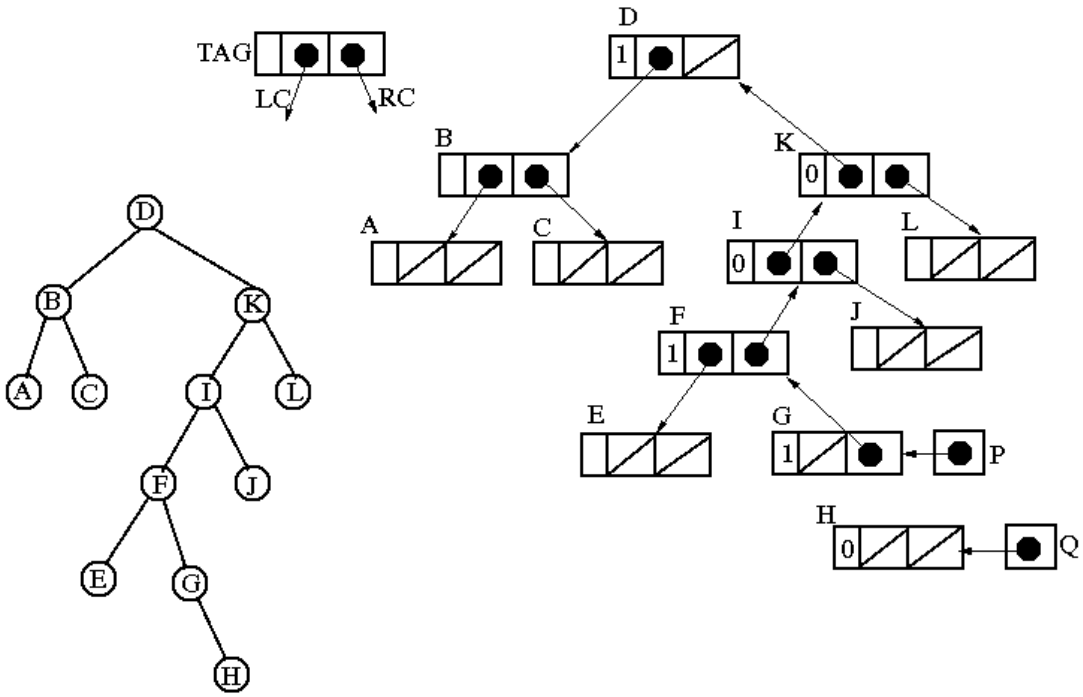
Χρόνος Διάσχισης: O(n), n: αριθμός κόμβων

Μνήμη?

Μέγεθος στοίβας ανάλογο του ύψους του δένδρου.

Άρα, οι αναδρομικοί αλγόριθμοι είναι πολύ απαιτητικοί σε μνήμη.

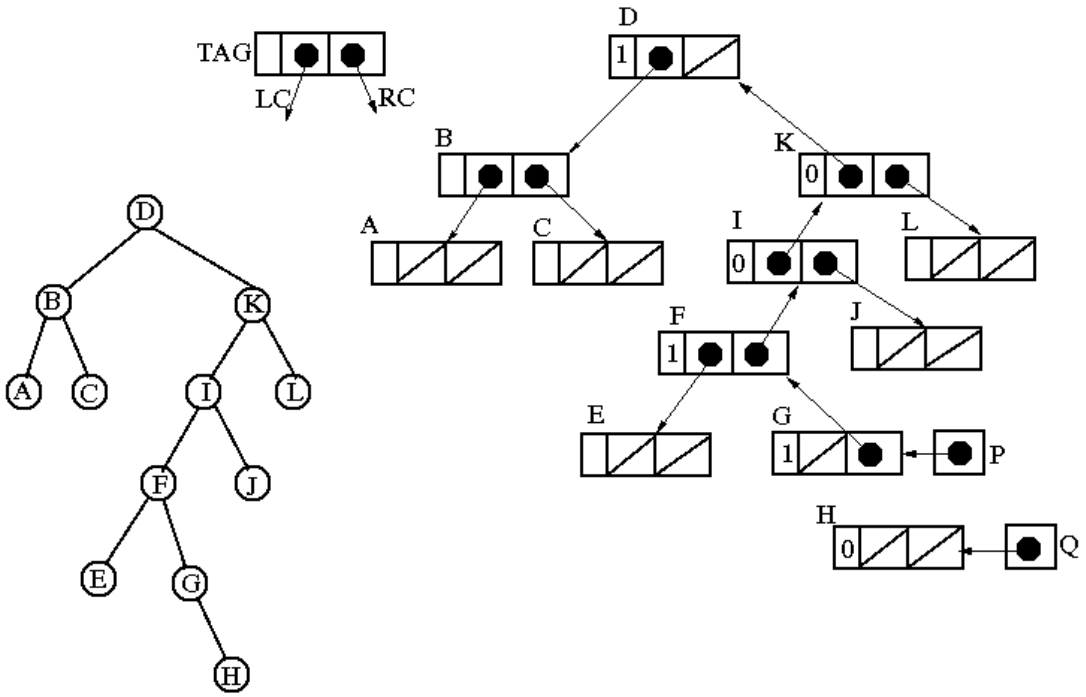
Με Αναστροφή Δεικτών



Procedure *LinkInversionTraversal(pointer Q):*
/* Initially Q points to the root of the tree to be traversed */

```
P = NULL;
while (1)
        while (Q != NULL) do
                Visit(Q)
                Tag(Q) = 0
                descend to left
        while (P != NULL and Tag(P) = 1) do
                ascend from right
                Visit(Q)
        if (P == NULL) then return
        else
                ascend from left
                Visit(Q)
                Tag(Q) = 1
                descend to right
```

Με Αναστροφή Δεικτών



descend to left: descend to right:

$$\begin{pmatrix} P \\ Q \\ Q->LC \end{pmatrix} \leftarrow \begin{pmatrix} Q \\ Q->LC \\ P \end{pmatrix} \qquad \begin{pmatrix} P \\ Q \\ Q->RC \end{pmatrix} \leftarrow \begin{pmatrix} Q \\ Q->RC \\ P \end{pmatrix}$$

ascend from left: ascend from right:

$$\begin{pmatrix} Q \\ P \\ P->LC \end{pmatrix} \leftarrow \begin{pmatrix} P \\ P->LC \\ Q \end{pmatrix} \qquad \begin{pmatrix} Q \\ P \\ P->RC \end{pmatrix} \leftarrow \begin{pmatrix} P \\ P->RC \\ Q \end{pmatrix}$$

Έχουμε βελτίωση στην απαιτούμενη μνήμη?

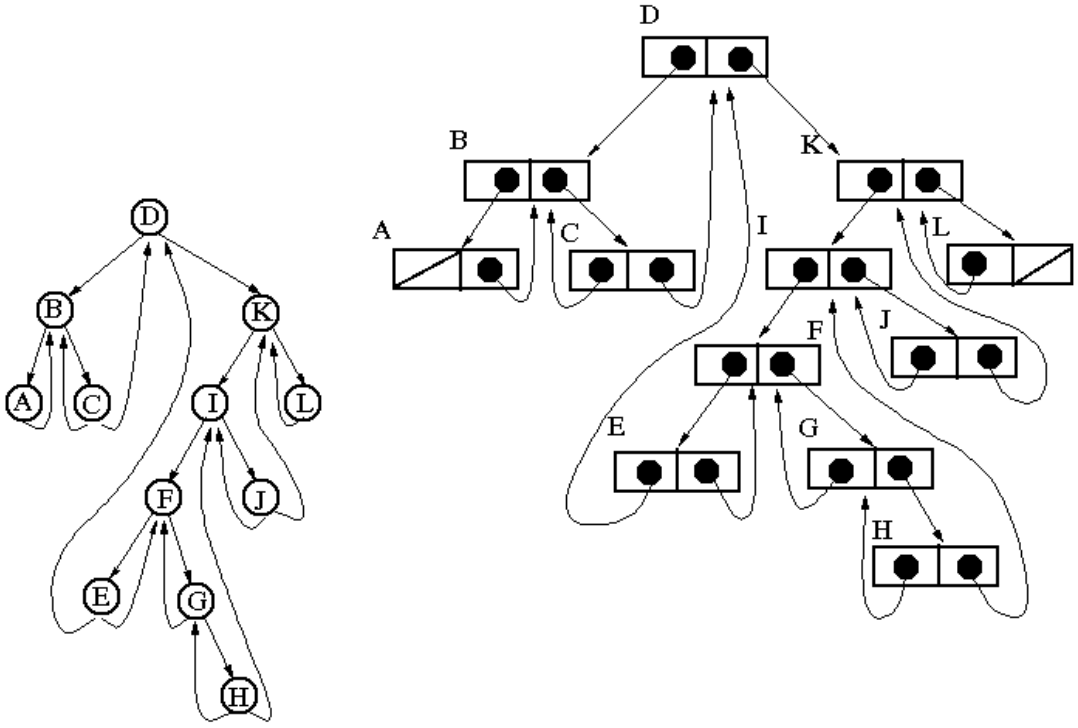
Νηματικά Δυαδικά Δένδρα

- ✓ Σε ένα δυαδικό δένδρο οι μισοί περίπου δείκτες έχουν την τιμή NULL.
- ✓ Η αναδρομική διάσχιση είναι ακριβή σε μνήμη.

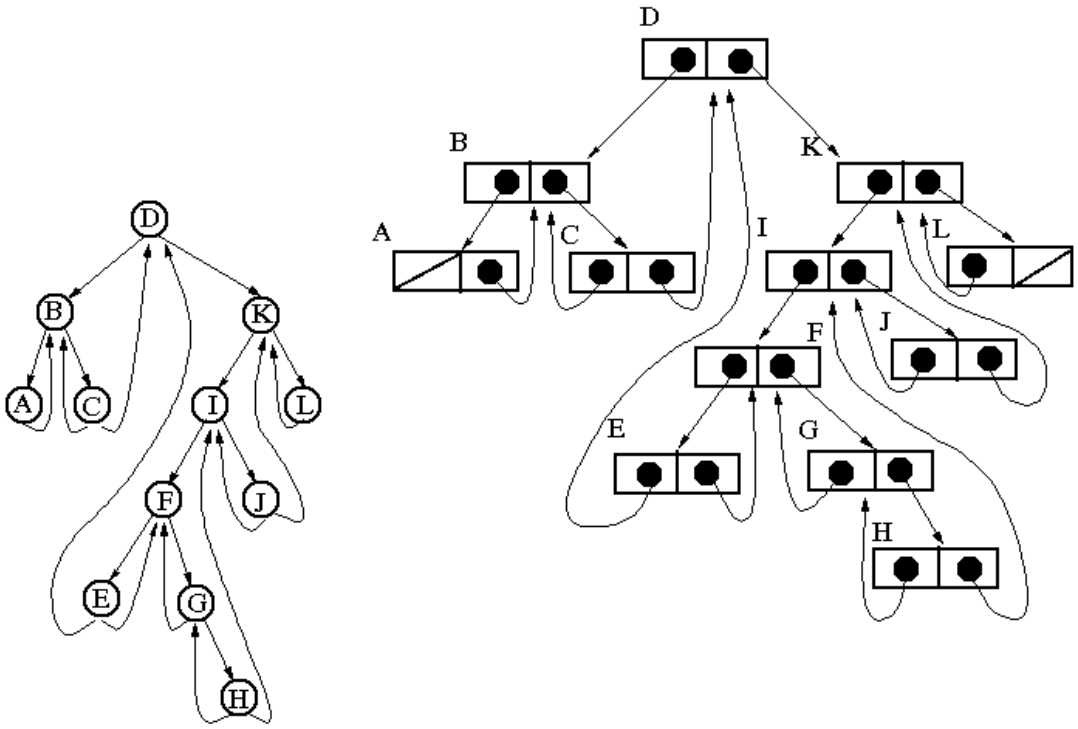
Συχνά οι δείκτες NULL χρησιμοποιούνται για να δείχνουν σε άλλους κόμβους. Τα δένδρα που υλοποιούνται με αυτόν τον τρόπο λέγονται **νηματικά**.

Ο RC δείκτης ενός κόμβου χωρίς δεξιό παιδί δείχνει στον επόμενό του κόμβο στην ενδοδιατεταγμένη διάταξη, ενώ ο LC δείχνει στον προηγούμενό του κόμβο στην ενδοδιατεταγμένη διάταξη.

Χρειαζόμαστε ένα ακόμη bit ανά δείκτη για να ξεχωρίζουμε τους νηματικούς από τους κανονικούς δείκτες.



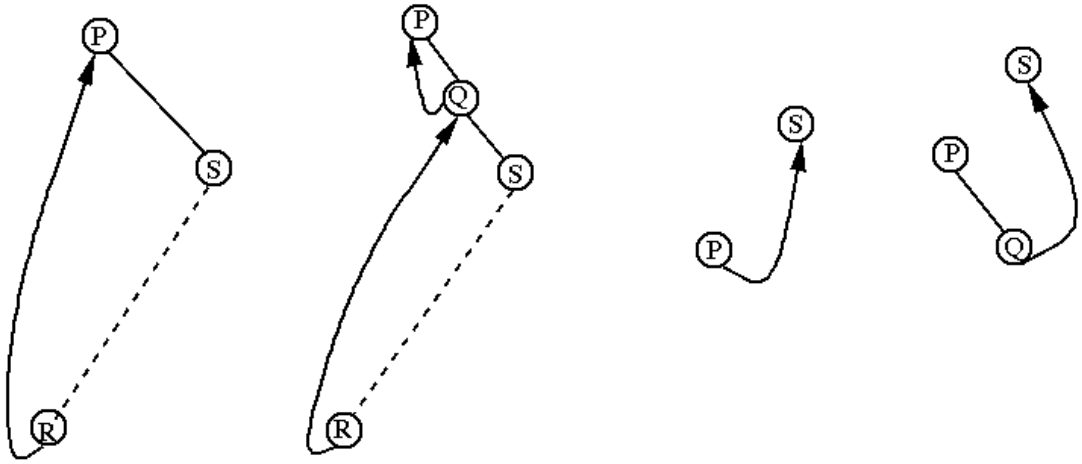
Νηματικά Δυαδικά Δένδρα



function *InorderSuccessor(pointer N):* **pointer**
/* returns the inorder successor of node N, or Λ if N has none */

```
P = N->RC
if (P = NULL) then return NULL
else if (P is not a thread) then
        while (P->LC is not a thread or NULL) do
              P = P->LC
return P
```

Εισαγωγή Κόμβου σε Νηματικό Δυαδικό Δένδρο



procedure *ThreadedInsert(pointer P,Q):*

/* Make node Q the inorder successor of node P */

$$\begin{pmatrix} P->RC \\ Q->LC \\ Q->RC \end{pmatrix} \leftarrow \begin{pmatrix} (child)Q \\ (thread)P \\ P->RC \end{pmatrix}$$

if Q->RC is not a thread then

R = InorderSuccessor(Q);

R->LC = (thread) Q

Υλοποίηση Διάσχισης κατά Επίπεδα

Χρήση Ουράς

- Αρχικά η ουρά περιέχει μόνο τη ρίζα.
- Στη συνέχεια επαναληπτικά: κάνουμε Deque ένα στοιχείο της ουράς και προσθέτουμε τα παιδιά από αριστερά προς τα δεξιά του στοιχείου αυτού.

Παράδειγμα

Περιεχόμενα Ουράς

A
B, C, D
C, D, E, F
D, E, F
E, F, G
F, G
G, H, I
H, I, J, K
I, J, K
J, K
K, L
L
<empty>