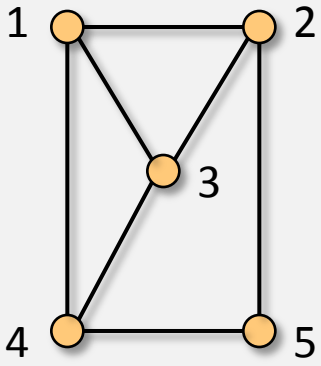
Γράφημα

Συνδυαστικό αντικείμενο που αποτελείται από 2 σύνολα:

- Σύνολο κορυφών (vertex set)
- Σύνολο ακμών (edge set)



$$G = (V, E)$$

$$V = \{1, 2, 3, 4, 5\}$$

$$E = \{\{1, 2\}, \{1, 3\}, \{1, 4\}, \{2, 3\}, \{2, 5\}, \{3, 4\}, \{4, 5\}\}$$

$|V| = N$ πλήθος κορυφών

$|E| = M$ πλήθος ακμών

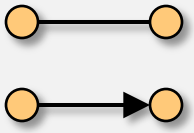
Γράφημα

Συνδυαστικό αντικείμενο που αποτελείται από 2 σύνολα:

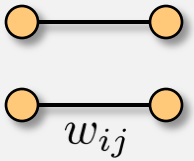
- Σύνολο κορυφών (vertex set)
- Σύνολο ακμών (edge set)

Μερικά είδη γραφημάτων:

- Κατεύθυνση ακμών
  - μη κατευθυνόμενα
  - κατευθυνόμενα



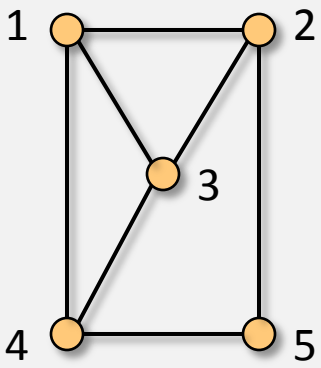
- Βάρος ακμών
  - μη σταθμισμένα
  - σταθμισμένα



Αναπαράσταση Γραφήματος

Μήτρα γειτνίασης (adjacency matrix)

Χρησιμοποιούμε έναν $N \times N$ πίνακα A, όπου $A[i,j] = \begin{cases} 1, & i = j \\ 1, & \{i,j\} \in E \\ 0, & \{i,j\} \notin E \end{cases}$



$$A = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 \end{bmatrix}$$

Χώρος: N^2 bits

συμμετρικός πίνακας

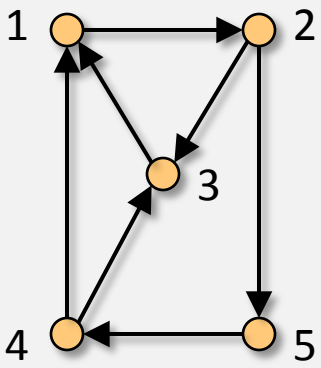
Ελέγχουμε αν $\{i,j\} \in E$ σε $\Theta(1)$ χρόνο

Επεξεργαζόμαστε όλες τις ακμές σε $\Theta(N^2)$ χρόνο

Αναπαράσταση Γραφήματος

Μήτρα γειτνίασης (adjacency matrix)

Χρησιμοποιούμε έναν $N \times N$ πίνακα A, όπου $A[i,j] = \begin{cases} 1, & i = j \\ 1, & \{i,j\} \in E \\ 0, & \{i,j\} \notin E \end{cases}$



$$A = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 \end{bmatrix}$$

Χώρος: N^2 bits

Ελέγχουμε αν $\{i,j\} \in E$ σε $\Theta(1)$ χρόνο

Επεξεργαζόμαστε όλες τις ακμές σε $\Theta(N^2)$ χρόνο

Αναπαράσταση Γραφήματος

Μήτρα γειτνίασης (adjacency matrix)

$$A = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 \end{bmatrix}$$

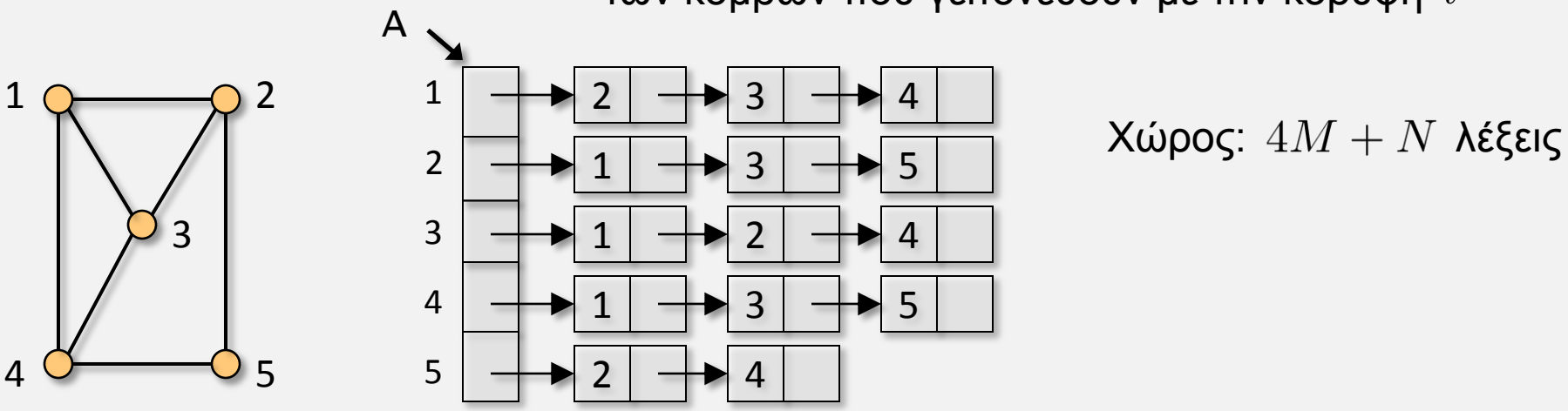
```
class AdjacencyMatrix {

        public static void main(String[] args) {
                int N = Integer.parseInt(args[0]);
                int M = Integer.parseInt(args[1]);
                boolean adj[][] = new boolean[N][N];
                for (int i = 0; i < N; i++)
                        for (int j = 0; j < N; j++)
                                adj[i][j] = false;
                for (int i = 0; j < N; j++) adj[i][i] = true;
                for (In.init(); !In.empty();) {
                        int i = In.getInt(), j = In.getInt();
                        adj[i][j] = true; adj[j][i] = true;
                }
        }
}
```

Διαβάζει **μη κατευθυνόμενο** γράφημα

Αναπαράσταση Γραφήματος

Λίστες γειτνίασης (adjacency lists)

Χρησιμοποιούμε έναν $N \times 1$ πίνακα A, όπου $A[i]$ είναι αναφορά σε λίστα των κόμβων που γειτονεύουν με την κορυφή i



Χώρος: $4M + N$ λέξεις

Κάθε ακμή εμφανίζεται 2 φορές

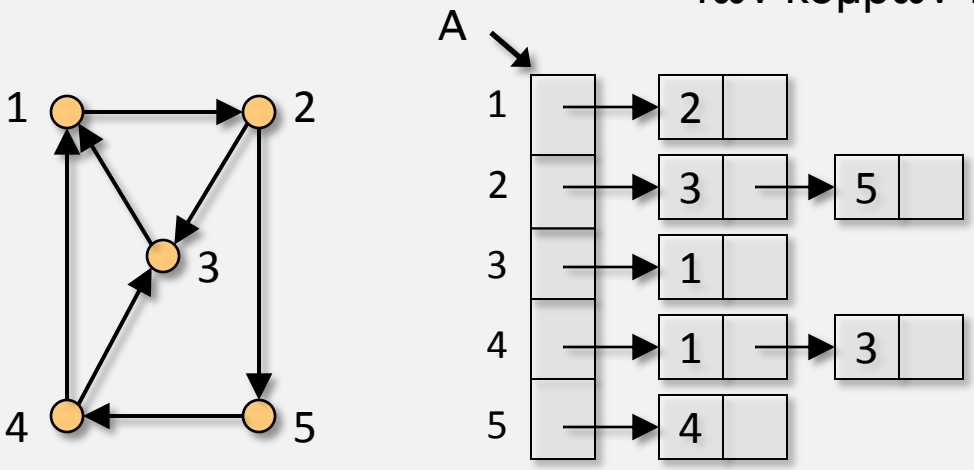
Ελέγχουμε αν $\{i, j\} \in E$ σε $\Theta(N)$ χρόνο

Επεξεργαζόμαστε όλες τις ακμές σε $\Theta(M)$ χρόνο

Αναπαράσταση Γραφήματος

Λίστες γειτνίασης (adjacency lists)

Χρησιμοποιούμε έναν $N \times 1$ πίνακα A, όπου $A[i]$ είναι αναφορά σε λίστα των κόμβων που γειτονεύουν με την κορυφή i

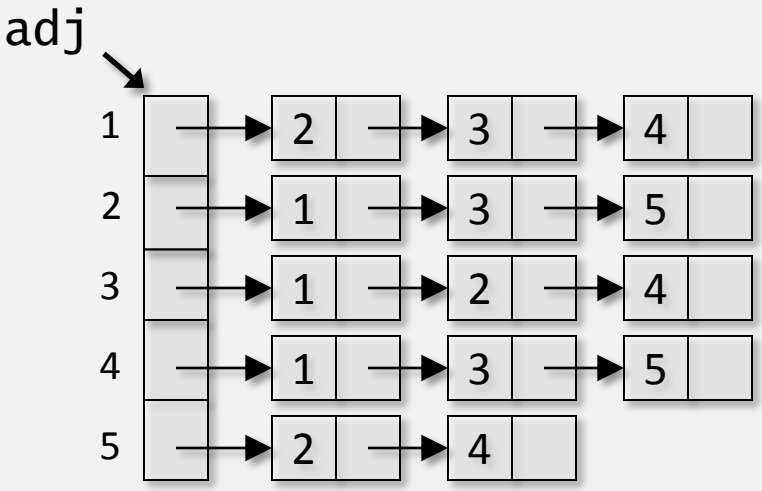


Χώρος: $2M + N$ λέξεις

Ελέγχουμε αν $\{i, j\} \in E$ σε $\Theta(N)$ χρόνο

Επεξεργαζόμαστε όλες τις ακμές σε $\Theta(M)$ χρόνο

Αναπαράσταση Γραφήματος



Λίστες γειτνίασης (adjacency lists)

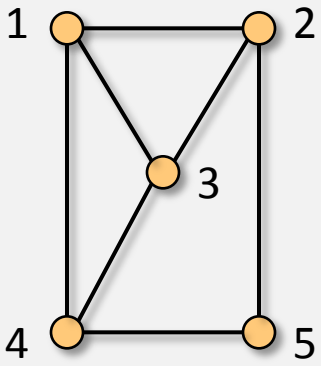
```
class AdjacencyLists {
        static class Node {
                int v; Node next;
                Node(int v, Node t)
                { this.v = v; next = t; }
        }

        public static void main(String[] args) {
                int N = Integer.parseInt(args[0]);
                int M = Integer.parseInt(args[1]);
                Node adj[] = new Node[V];
                for (int i = 0; i < N; i++) adj[i] = null;
                for (In.init(); !In.empty();) {
                        int i = In.getInt(), j = In.getInt();
                        adj[j] = new Node(i, adj[j]);
                        adj[i] = new Node(j, adj[i]);
                }
        }
}
```

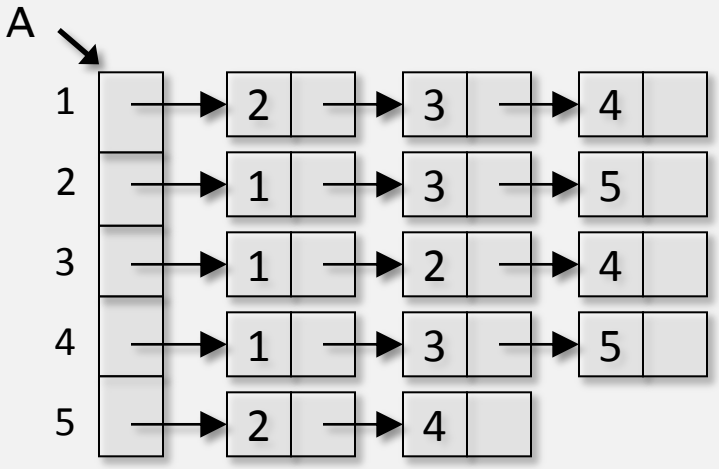
Διαβάζει **μη κατευθυνόμενο** γράφημα

Αναπαράσταση Γραφήματος

Λίστες γειτνίασης (adjacency lists)

Χρησιμοποιούμε έναν $N \times 1$ πίνακα A, όπου $A[i]$ είναι αναφορά σε λίστα των κόμβων που γειτονεύουν με την κορυφή i





Χώρος: $4M + N$ λέξεις

Αν το γράφημα είναι στατικό (δεν έχουμε εισαγωγές ή διαγραφές ακμών) τότε μπορούμε να αναπαραστήσουμε τις λίστες γειτνίασης με 2 μονοδιάστατους πίνακες.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14
targetVertex =	2	3	4	1	3	5	1	2	4	1	3	5	2	4

	1	2	3	4	5
firstTarget =	1	4	7	10	13

Αναπαράσταση Γραφήματος

Λίστες γειτνίασης (adjacency lists)

Αν το γράφημα είναι στατικό (δεν έχουμε εισαγωγές ή διαγραφές ακμών) τότε μπορούμε να αναπαραστήσουμε τις λίστες γειτνίασης με 2 μονοδιάστατους πίνακες.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14
$\text{targetVertex} =$	2	3	4	1	3	5	1	2	4	1	3	5	2	4

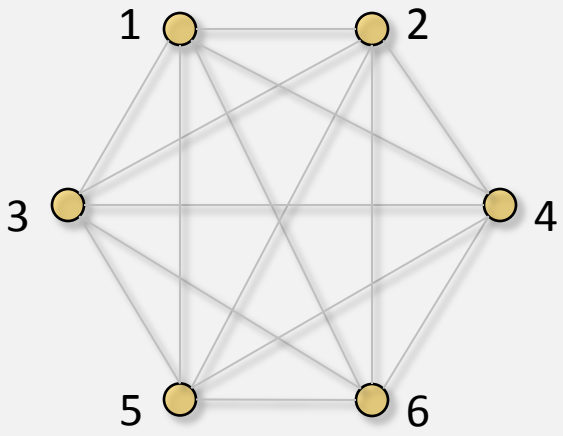
	1	2	3	4	5
$\text{firstTarget} =$	1	4	7	10	13

Οι γείτονες του κόμβου i βρίσκονται στις θέσεις $\text{targetVertex}[k : l-1]$ όπου $k = \text{firstTarget}[i]$ και $l = \text{firstTarget}[i+1]$

Χώρος: $2M + N$ λέξεις για μη κατευθυνόμενο γράφημα, $M + N$ για κατευθυνόμενο

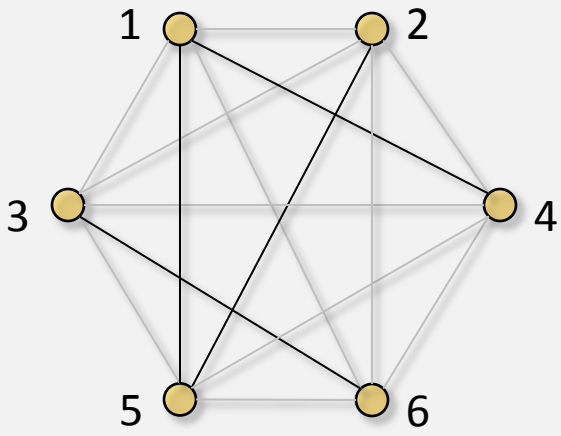
Τυχαία Γραφήματα

Επιλέγουμε m ακμές τυχαία (από κάποια κατανομή) σε γράφημα με n κόμβους



Τυχαία Γραφήματα

Επιλέγουμε m ακμές τυχαία (από κάποια κατανομή) σε γράφημα με n κόμβους



Τυχαία Γραφήματα

Ομοιόμορφα τυχαία επιλογή κόμβων

Επιλέγουμε ομοιόμορφα τυχαία τους δύο κόμβους της κάθε ακμής από τους n κόμβους του γραφήματος

```
import java.util.Random;
...
Random rand = new Random(seed);
for(int i=0; i<m; i++)
{
        int j = rand.nextInt(n) + 1;
        int k = rand.nextInt(n) + 1;
        addEdge(j,k);
}
```

Κατασκευάζει γράφημα με n κόμβους και m ακμές αλλά μπορεί να περιέχει βρόχους και πολλαπλές (επαναλαμβανόμενες) ακμές

Τυχαία Γραφήματα

Ομοιόμορφη δειγματοληψία

Το παρακάτω πρόγραμμα επιλέγει κάθε μια από τις $n(n-1)/2$ δυνατές ακμές με πιθανότητα $p = \dfrac{m}{n(n-1)/2}$

```
import java.util.Random;
...
Random rand = new Random(seed);
double p = (double) 2*m/(n*(n-1));

for (int k=1; k<=n; k++)
    for (int j=1; j<k; j++)
        if ( rand.nextDouble() < p ) addEdge(j,k);
```

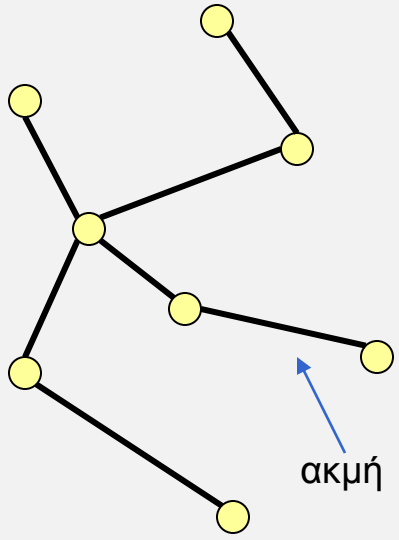
Κατασκευάζει γράφημα με n κόμβους και m αναμενόμενο αριθμό ακμών

Δένδρα

Μαθηματικά (συνδυαστικά) αντικείμενα.

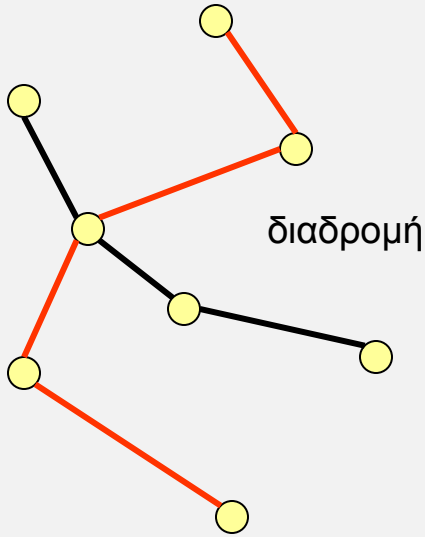
Έχουν κεντρικό ρόλο στην επιστήμη των υπολογιστών :

- Ανάλυση αλγορίθμων (π.χ. δένδρα αναδρομής)
- Δομές δεδομένων (π.χ. δένδρα αναζήτησης)

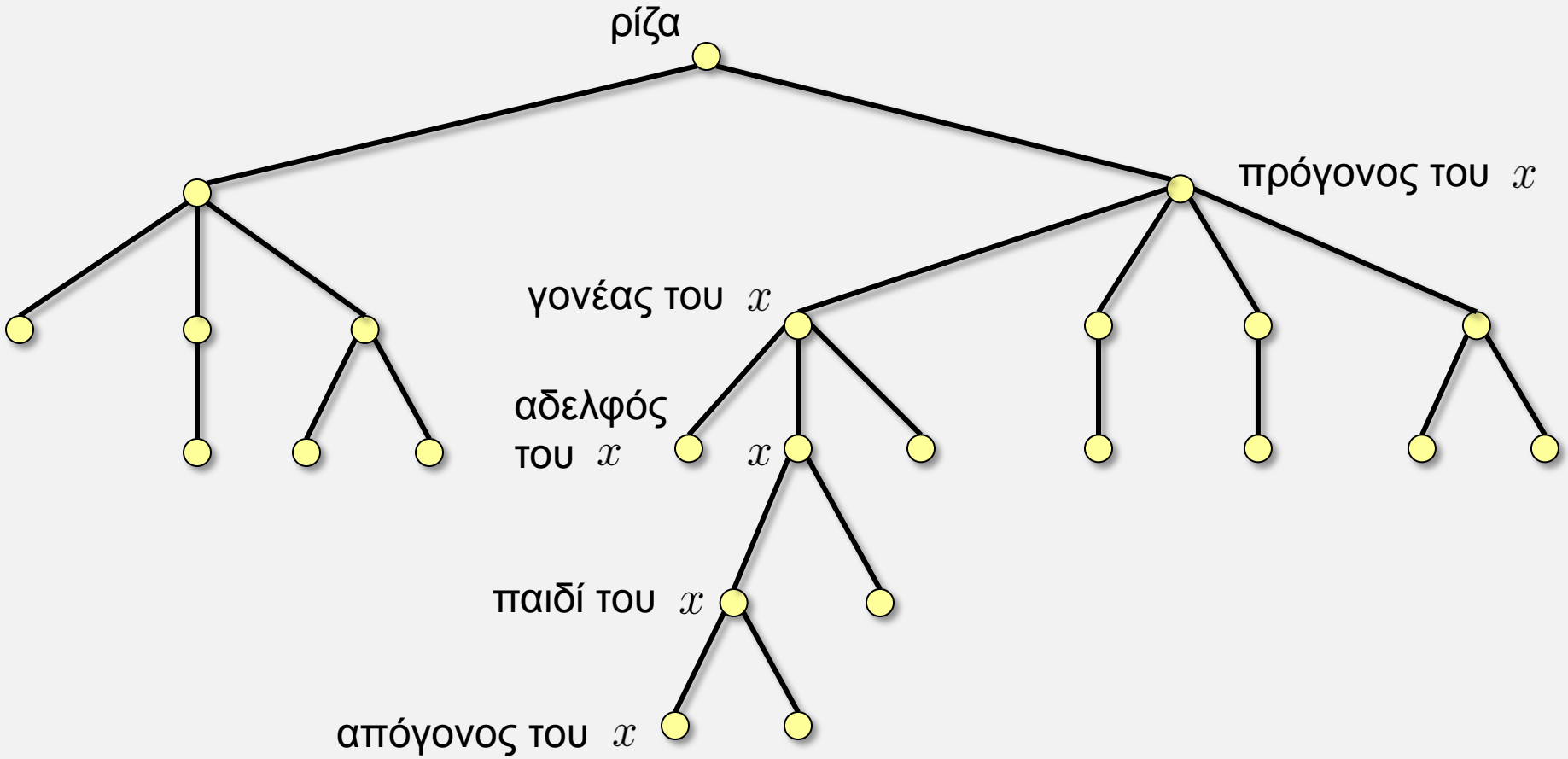


Κατηγορίες (αύξουσα σειρά γενικότητας) :

- Δυαδικά και Μ-αδικά δένδρα
- Διατεταγμένα δένδρα
- Δένδρα με ρίζα
- Ελεύθερα δένδρα



Δένδρα με Ρίζα

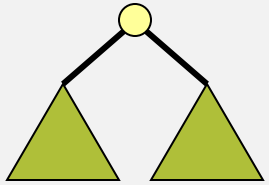


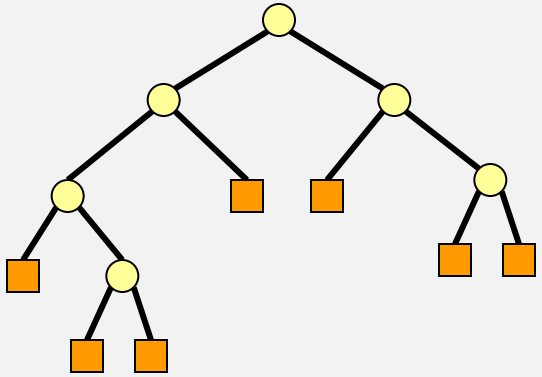
Ο y είναι πρόγονος του x (ο x απόγονος του y) αν βρίσκεται στο μονοπάτι από τη ρίζα στο x

Δυαδικά Δένδρα

Δυαδικό δένδρο (αναδρομικός ορισμός)  =

εξωτερικός κόμβος, ή εσωτερικός κόμβος που συνδέεται με ένα δυαδικό δένδρο στα αριστερά και ένα δυαδικό δένδρο στα δεξιά.

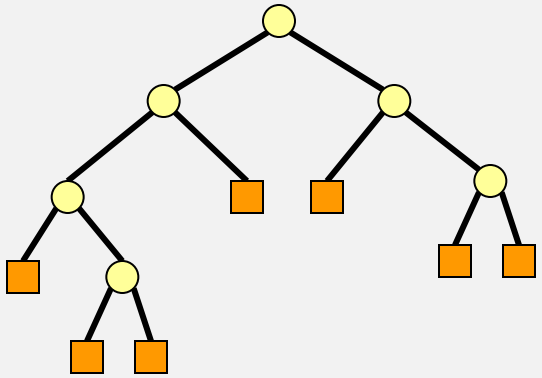


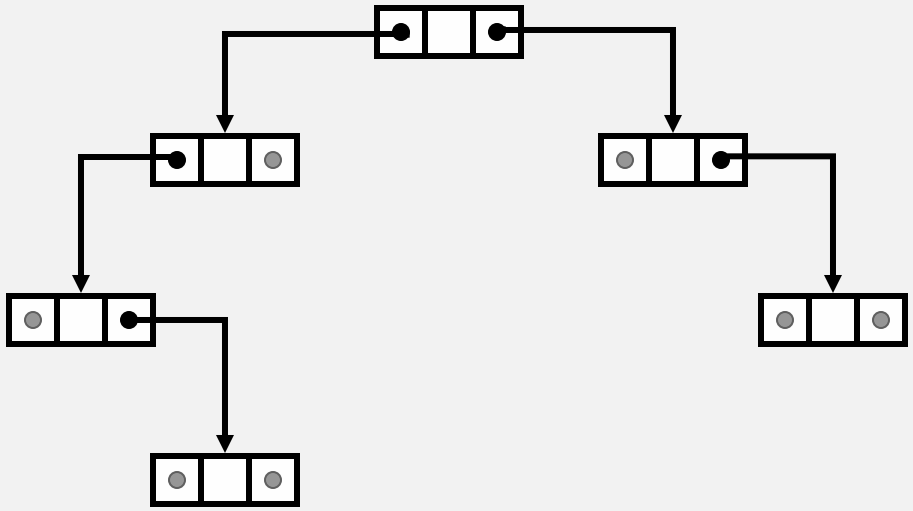


Δυαδικά Δένδρα

Δυαδικό δένδρο - Υλοποίηση

```
class Node {
        Item item; Node l; Node r;
        Node(Item v, Node l, Node r) {
                this.item = v;
                this.l = l;
                this.r = r;
        }
}
```





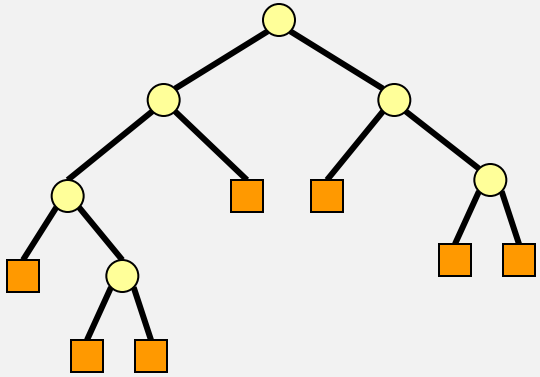
Οι εξωτερικοί κόμβοι αντιστοιχούν σε μηδενικές (null) αναφορές

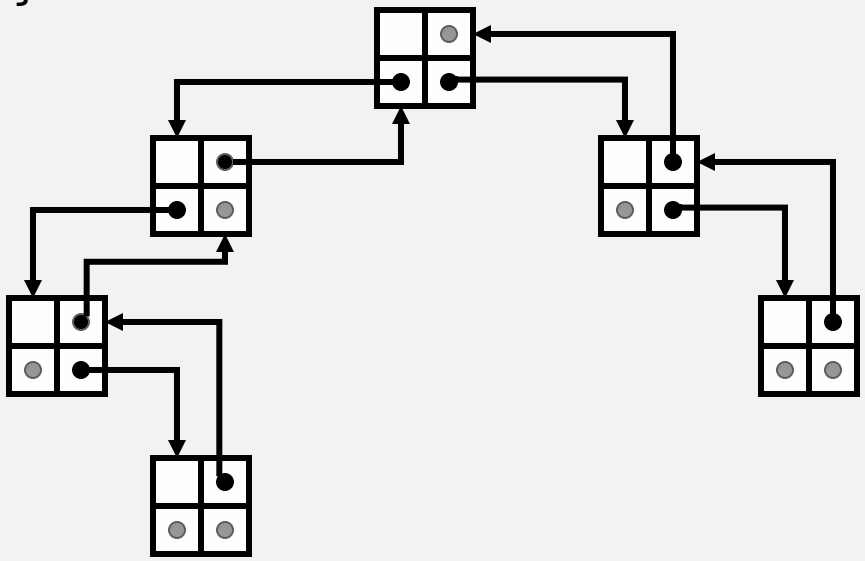
Δυαδικά Δένδρα

Δυαδικό δένδρο - Υλοποίηση

Αν δεν υπάρχει πρόβλημα χώρου τότε μπορούμε να έχουμε σε κάθε κόμβο μια αναφορά προς το γονέα του. Αυτό απλοποιεί την υλοποίηση ορισμένων λειτουργιών.

```
class Node {
    Item item; Node l; Node r; Node p;
    Node(Item v, Node l, Node r, Node p) {
            this.item = v;
            this.l = l; this.r = r;
            this.p = p;
    }
}
```



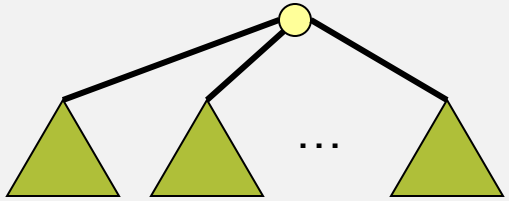


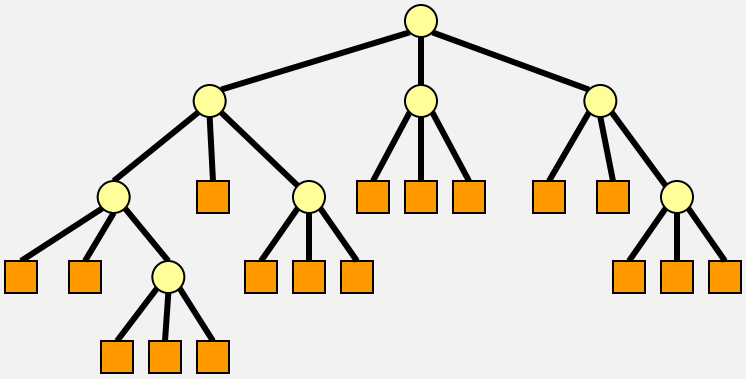
Οι εξωτερικοί κόμβοι αντιστοιχούν σε μηδενικές (null) αναφορές

Μ-αδικά Δένδρα

Μ-αδικό δένδρο (αναδρομικός ορισμός)  =

εξωτερικός κόμβος, ή εσωτερικός κόμβος που συνδέεται με διατεταγμένη ακολουθία Μ-αδικών δένδρων.

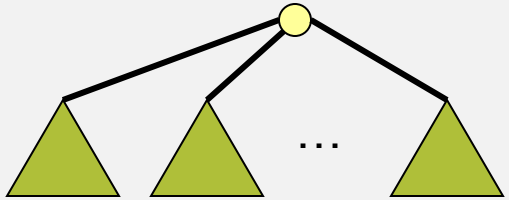


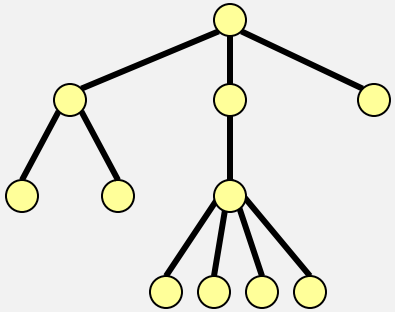


Διατεταγμένα Δένδρα

Διατεταγμένο δένδρο (αναδρομικός ορισμός)  =

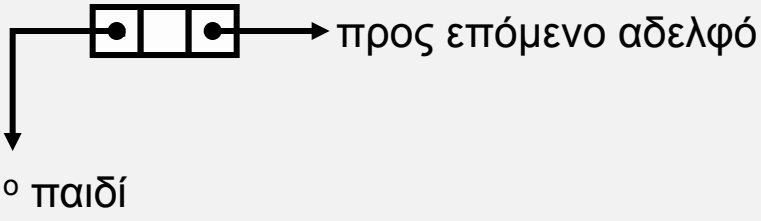
κόμβος (ρίζα του δένδρου) που συνδέεται με διατεταγμένη ακολουθία διατεταγμένων δένδρων.

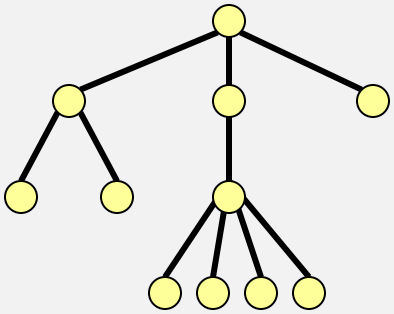


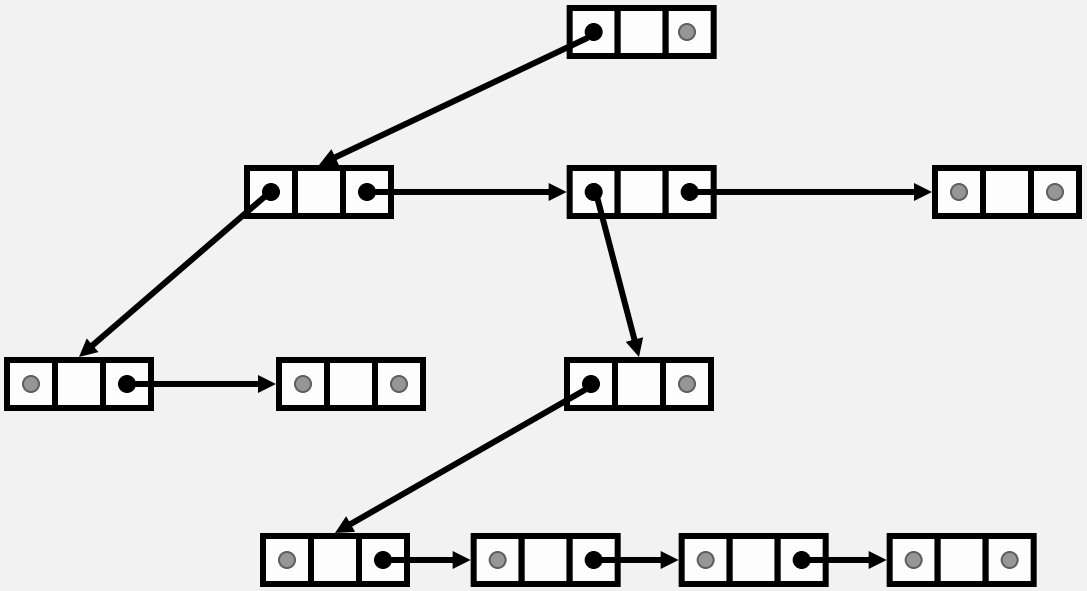


Διατεταγμένα Δένδρα

Διατεταγμένο δένδρο - Υλοποίηση

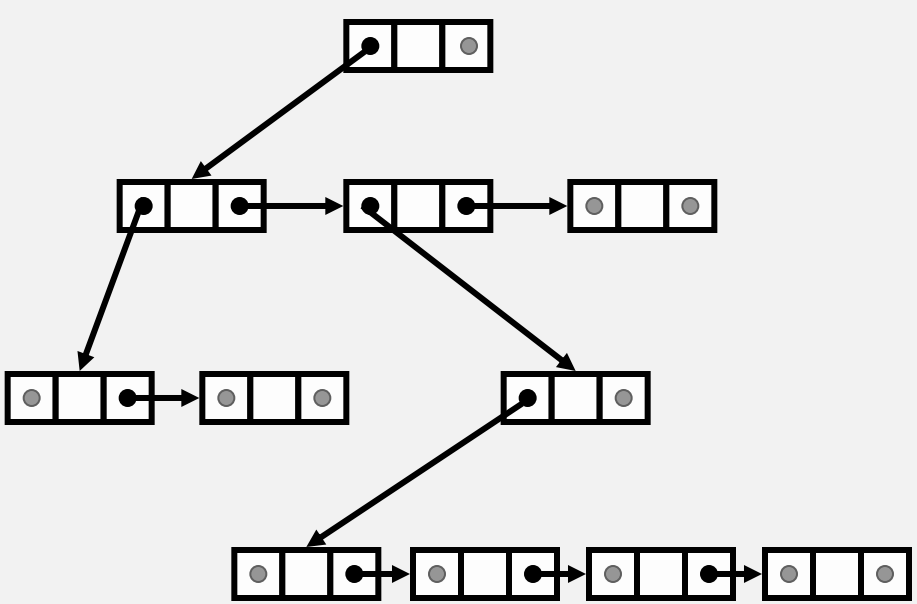


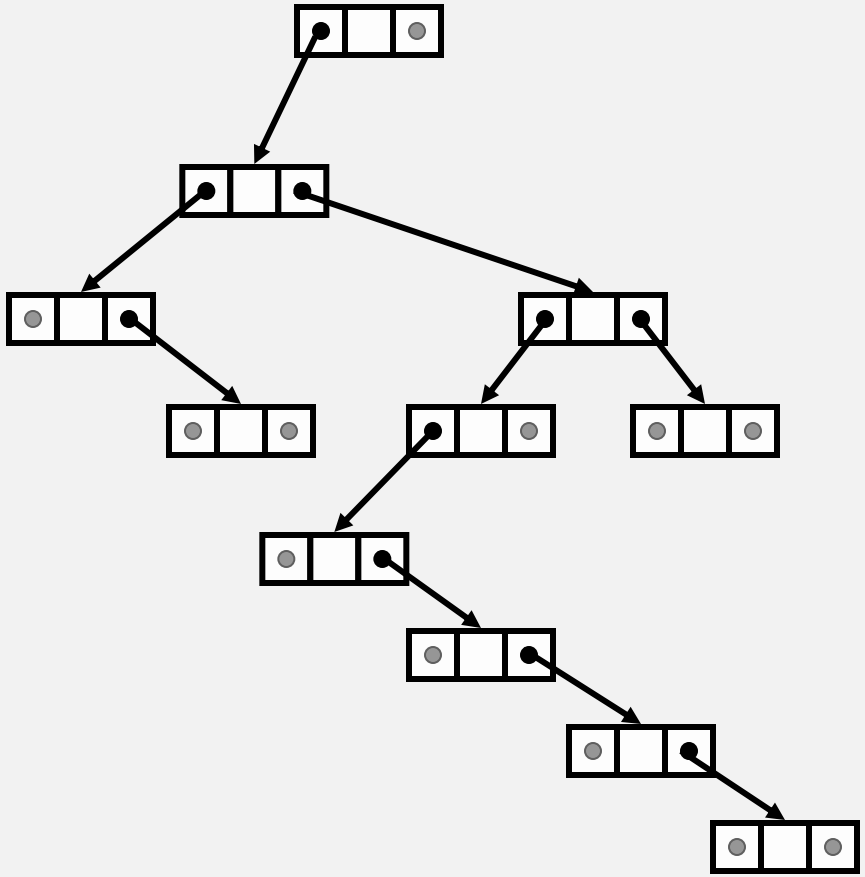




Διατεταγμένα Δένδρα

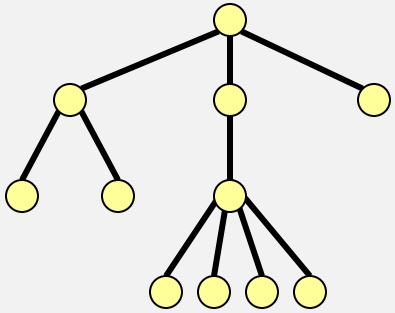
Διατεταγμένο δένδρο – Μετατροπή σε δυαδικό δένδρο

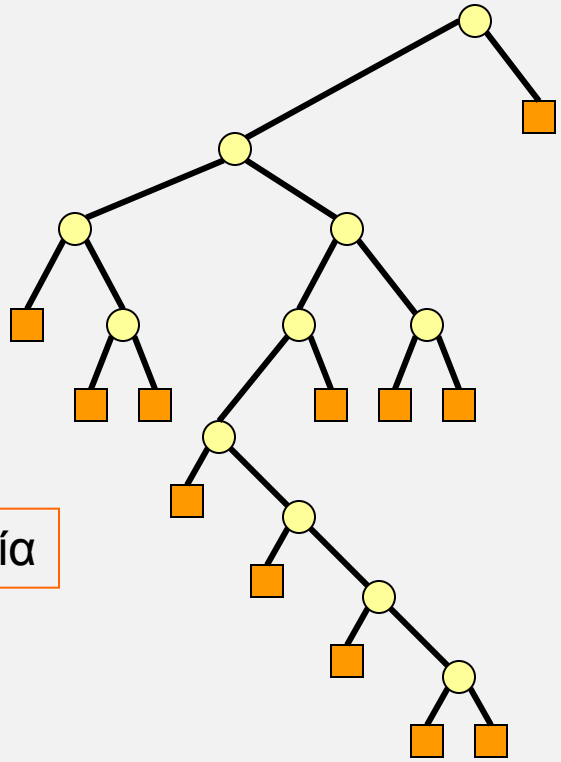




Διατεταγμένα Δένδρα

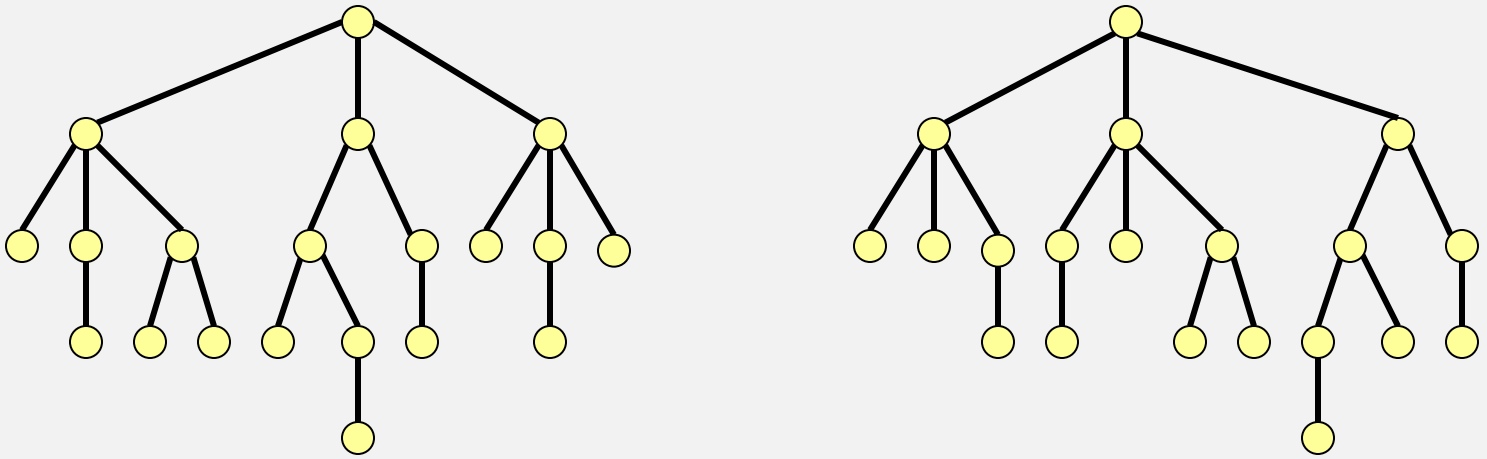
Διατεταγμένο δένδρο – Μετατροπή σε δυαδικό δένδρο





1-προς-1 αντιστοιχία

Ισομορφικά Δένδρα

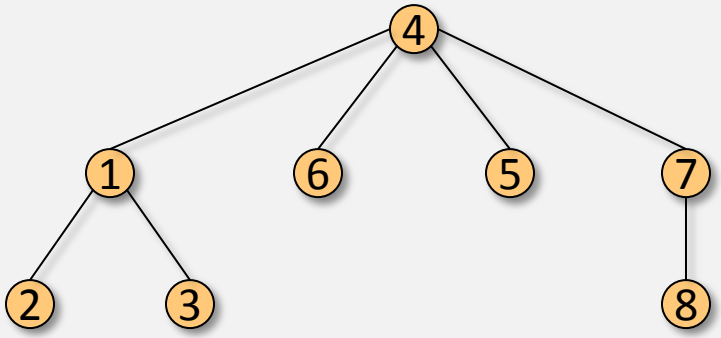


Διαφορετικά διατεταγμένα δένδρα μπορεί να αντιστοιχούν στο ίδιο μη διατεταγμένο δένδρο.

Αναπαραστάσεις Ειδικού Σκοπού

Αναπαράσταση με συνάρτηση γονέα

Αποθηκεύουμε τον γονέα κάθε κόμβου, π.χ. σε ένα πίνακα `parent[1:N]`



	1	2	3	4	5	6	7	8
parent =	4	1	1	4	4	4	4	7

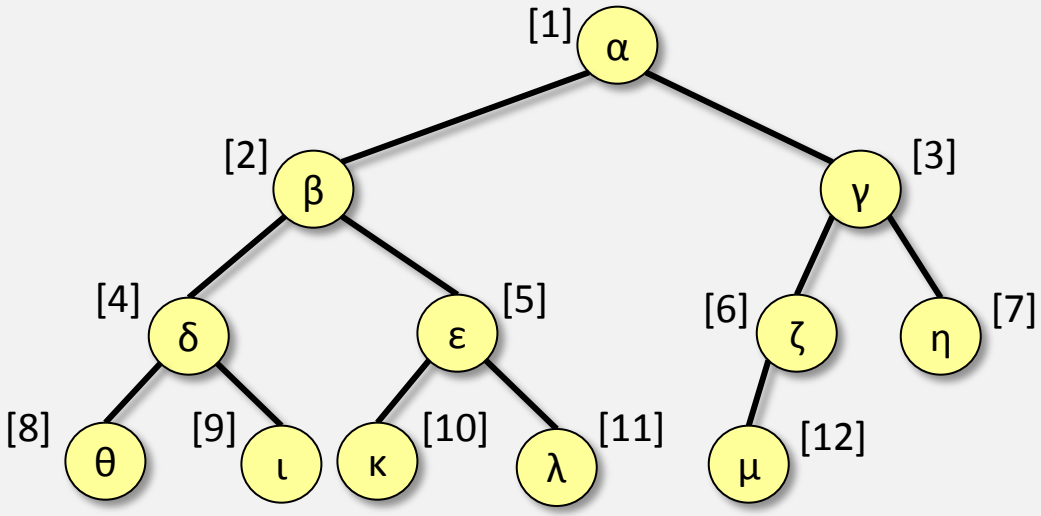
Για τη ρίζα `j` του δένδρου θέτουμε `parent[j]=j`.

Αναπαραστάσεις Ειδικού Σκοπού

Αναπαράσταση πλήρους δυαδικού δένδρου με πίνακα θέσης

Ένα πλήρες δυαδικό δένδρο μπορεί να αποθηκευτεί σε ένα πίνακα `b[1:N]` :

Το στοιχείο στη θέση `i` είναι ο γονέας των στοιχείων στις θέσεις `2i` και `2i+1`.

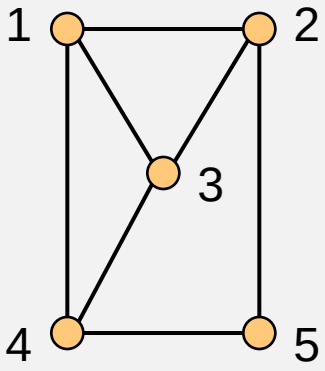


	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]	[11]	[12]
b =	α	β	γ	δ	ε	ζ	η	θ	ι	κ	λ	μ

Γράφημα

Συνδυαστικό αντικείμενο που αποτελείται από 2 σύνολα:

- Σύνολο κορυφών (vertex set)
- Σύνολο ακμών (edge set)



$$G = (V, E)$$

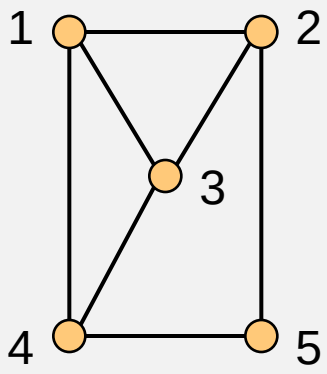
$$V = \{1, 2, 3, 4, 5\}$$

$$E = \{\{1, 2\}, \{1, 3\}, \{1, 4\}, \{2, 3\}, \{2, 5\}, \{3, 4\}, \{4, 5\}\}$$

Γράφημα

Συνδυαστικό αντικείμενο που αποτελείται από 2 σύνολα:

- Σύνολο κορυφών (vertex set)
- Σύνολο ακμών (edge set)



$(1, 2, 3, 1, 4, 5)$ διαδρομή

$(1, 3, 2, 5, 4)$ απλή διαδρομή

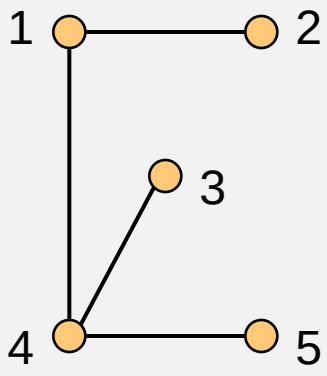
$(2, 3, 4, 5, 2)$ (απλός) κύκλος

Γράφημα

Κάθε δένδρο είναι ένα γράφημα.

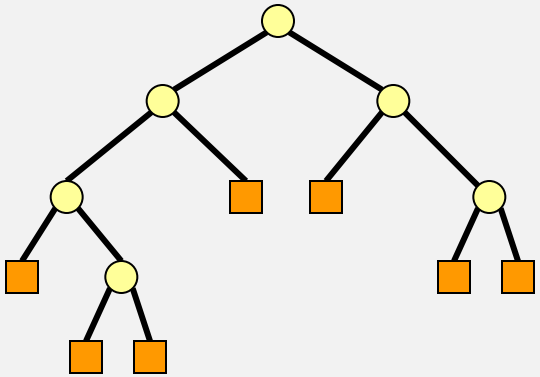
Ένα γράφημα είναι δένδρο ένα ισχύει ένα από τα παρακάτω:

- έχει N-1 ακμές και κανένα κύκλο
- έχει N-1 ακμές και είναι συνδεδεμένο
- έχει μοναδική απλή διαδρομή για κάθε ζεύγος κόμβων
- είναι συνδεδεμένο αλλά παύει να είναι μετά την αφαίρεση κάποιας ακμής



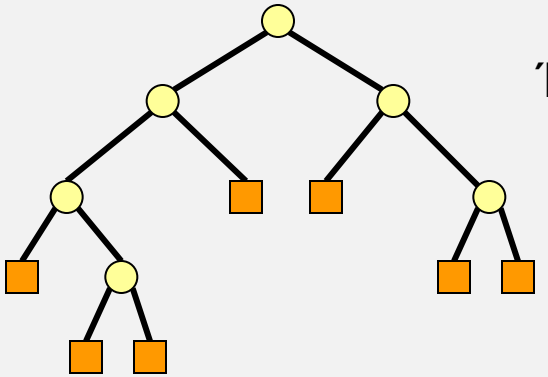
Ιδιότητες Δυαδικών Δένδρων

Ένα δυαδικό δένδρο με Ν εσωτερικούς κόμβους έχει Ν+1 εξωτερικούς κόμβους



Ιδιότητες Δυαδικών Δένδρων

Ένα δυαδικό δένδρο με Ν εσωτερικούς κόμβους έχει Ν+1 εξωτερικούς κόμβους



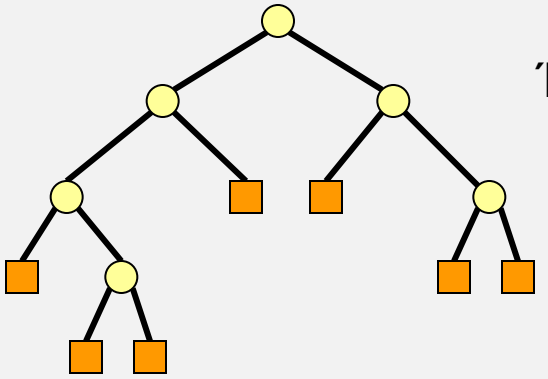
Απόδειξη με επαγωγή.

Για Ν=0 το δένδρο είναι ένας εξωτερικός κόμβος.

Έστω ότι ισχύει για λιγότερους από Ν εσωτερικούς κόμβους.

Ιδιότητες Δυαδικών Δένδρων

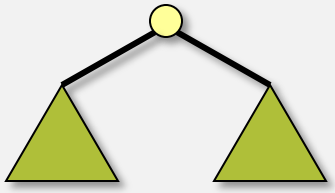
Ένα δυαδικό δένδρο με N εσωτερικούς κόμβους έχει N+1 εξωτερικούς κόμβους



Απόδειξη με επαγωγή.

Για N=0 το δένδρο είναι ένας εξωτερικός κόμβος.

Έστω ότι ισχύει για λιγότερους από N εσωτερικούς κόμβους.



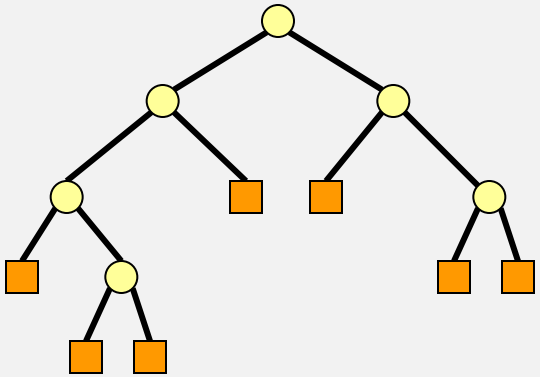
$k-1$ $N-k$

Ο συνολικός αριθμός εξωτερικών κόμβων είναι

$$((k-1)+1)+((N-k)+1)=N+1$$

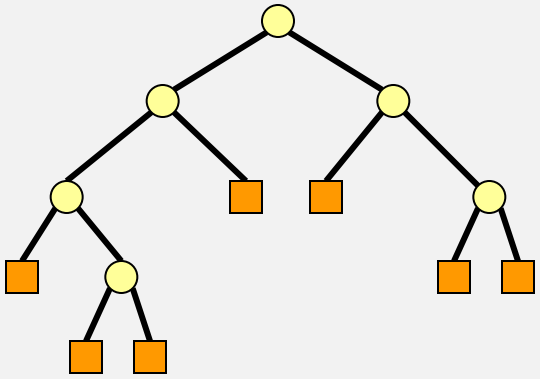
Ιδιότητες Δυαδικών Δένδρων

Ένα δυαδικό δένδρο με N εσωτερικούς κόμβους έχει 2N ακμές: N-1 συνδέουν εσωτερικούς κόμβους και N+1 συνδέουν εσωτερικό με εξωτερικό κόμβο.



Ιδιότητες Δυαδικών Δένδρων

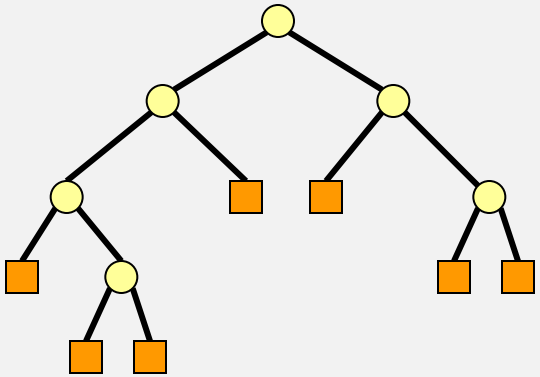
Ένα δυαδικό δένδρο με Ν εσωτερικούς κόμβους έχει 2Ν ακμές: Ν-1 συνδέουν εσωτερικούς κόμβους και Ν+1 συνδέουν εσωτερικό με εξωτερικό κόμβο.



Κάθε κόμβος εκτός από τη ρίζα έχει μοναδικό γονέα. Άρα έχουμε Ν-1 εσωτερικούς κόμβους που συνδέονται με το γονέα τους. Αντίστοιχα, καθένας από τους Ν+1 εξωτερικούς κόμβους συνδέεται με τον γονέα του που είναι εσωτερικός κόμβος.

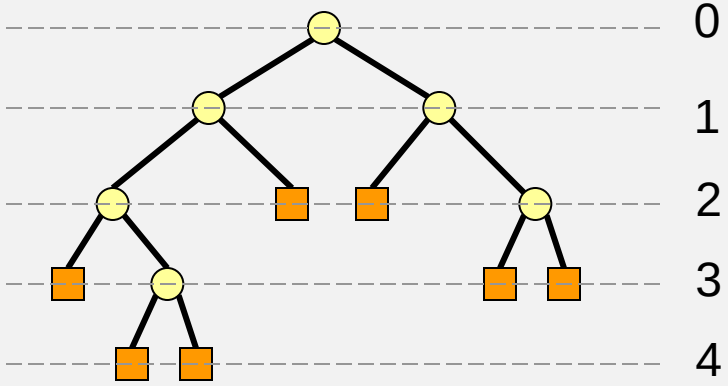
Ιδιότητες Δυαδικών Δένδρων

επίπεδο ρίζας = 0
επίπεδο κόμβου = επίπεδο γονέα + 1



Ιδιότητες Δυαδικών Δένδρων

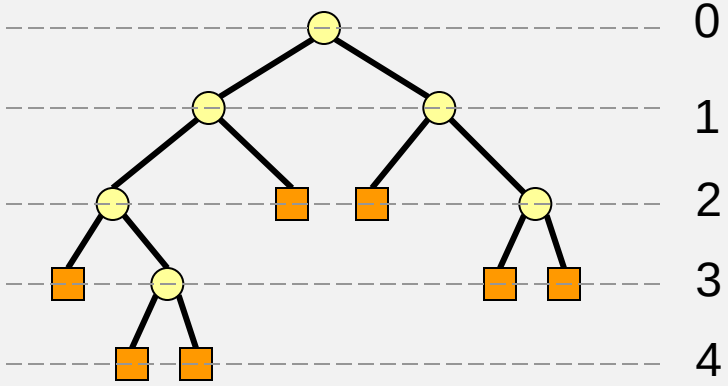
επίπεδο ρίζας = 0
επίπεδο κόμβου = επίπεδο γονέα + 1



Ιδιότητες Δυαδικών Δένδρων

επίπεδο ρίζας = 0
επίπεδο κόμβου = επίπεδο γονέα + 1

ύψος δένδρου = μέγιστο επίπεδο



Μήκος διαδρομής = άθροισμα επιπέδου κάθε κόμβου (=30)

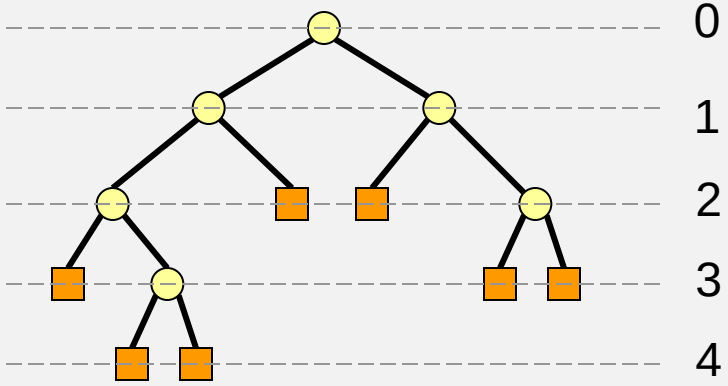
Μήκος εσωτερικής διαδρομής = άθροισμα επιπέδου κάθε εσωτερικού κόμβου (=9)

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

Ιδιότητες Δυαδικών Δένδρων

επίπεδο ρίζας = 0
επίπεδο κόμβου = επίπεδο γονέα + 1

ύψος δένδρου = μέγιστο επίπεδο



Μήκος διαδρομής = άθροισμα επιπέδου κάθε κόμβου (=30)

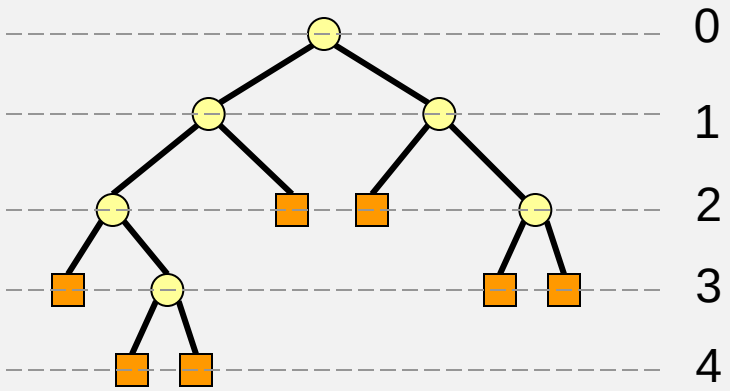
Μήκος εσωτερικής διαδρομής = άθροισμα επιπέδου κάθε εσωτερικού κόμβου (=9)

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

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

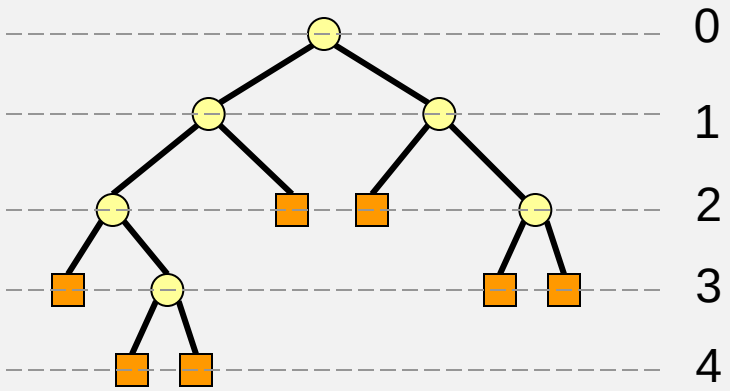
Ιδιότητες Δυαδικών Δένδρων

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



Ιδιότητες Δυαδικών Δένδρων

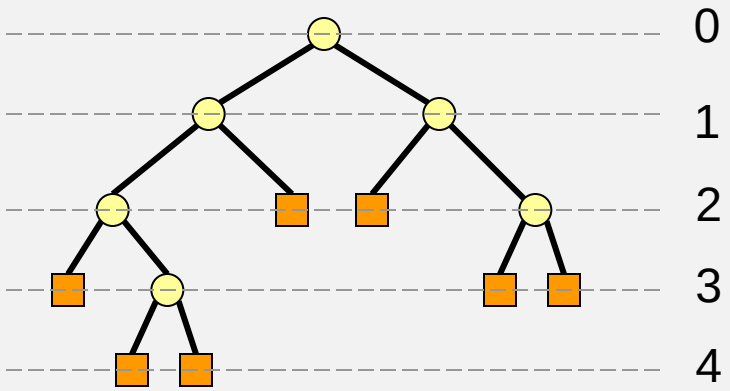
Ισχύει: μήκος εξωτερικής διαδρομής = μήκος εσωτερικής διαδρομής + 2N



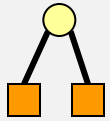
ξεκινάμε με εξωτερικό κόμβο

Ιδιότητες Δυαδικών Δένδρων

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

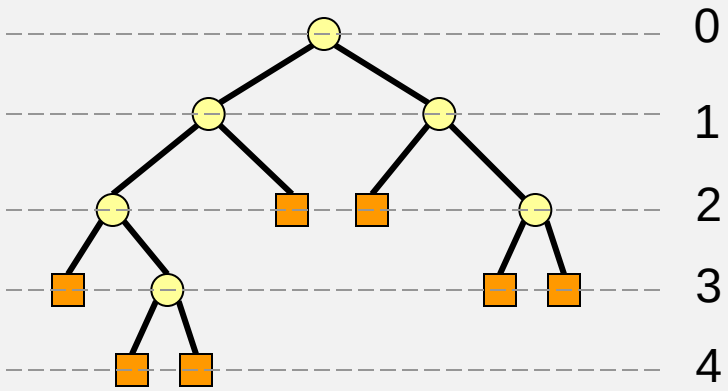


ξεκινάμε με εξωτερικό κόμβο

σε κάθε βήμα  

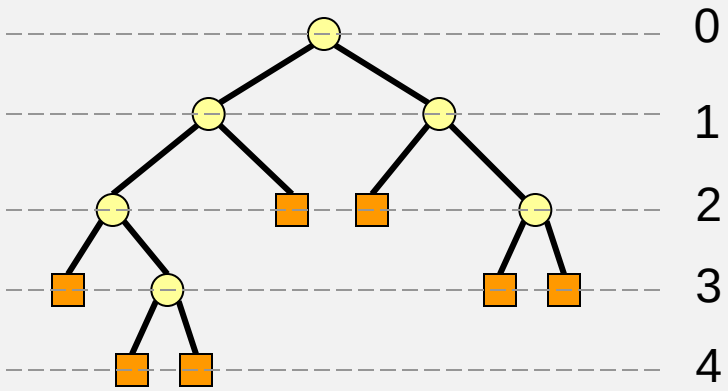
Ιδιότητες Δυαδικών Δένδρων

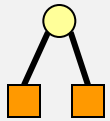
Ισχύει: μήκος εξωτερικής διαδρομής = μήκος εσωτερικής διαδρομής + 2N



Ιδιότητες Δυαδικών Δένδρων

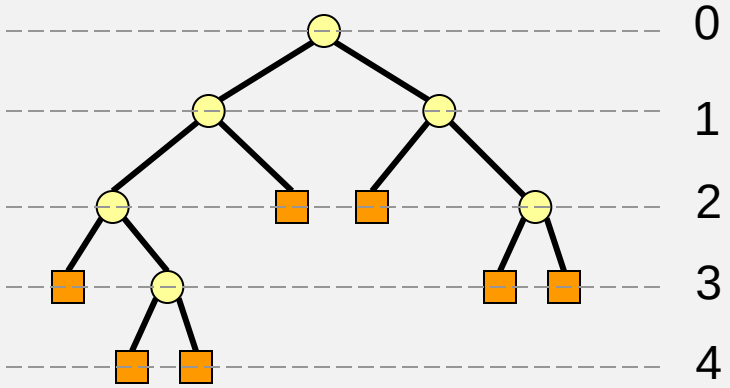
Ισχύει: μήκος εξωτερικής διαδρομής = μήκος εσωτερικής διαδρομής + 2N

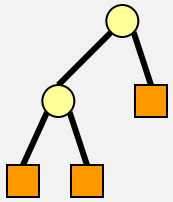




Ιδιότητες Δυαδικών Δένδρων

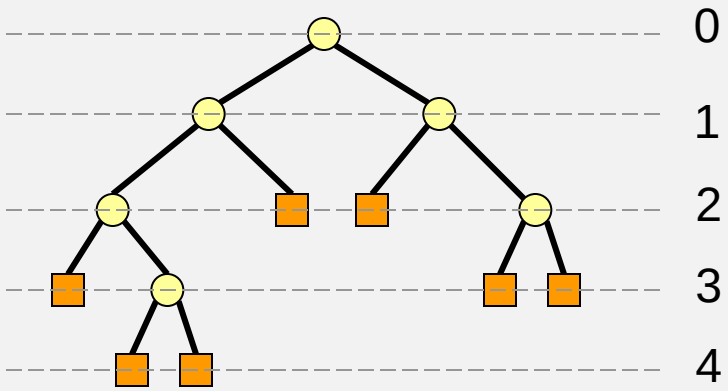
Ισχύει: μήκος εξωτερικής διαδρομής = μήκος εσωτερικής διαδρομής + 2N

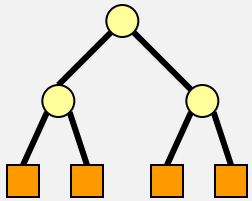




Ιδιότητες Δυαδικών Δένδρων

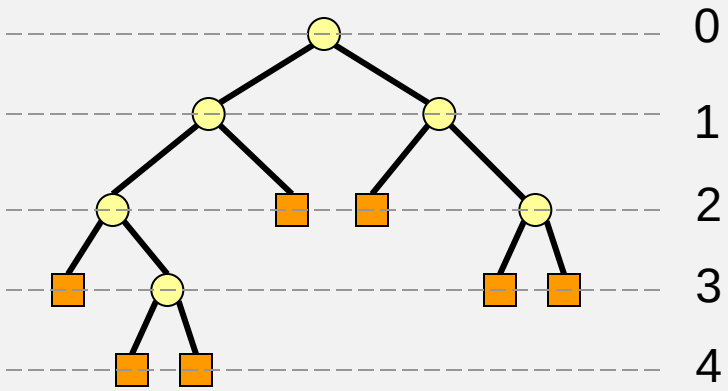
Ισχύει: μήκος εξωτερικής διαδρομής = μήκος εσωτερικής διαδρομής + 2N

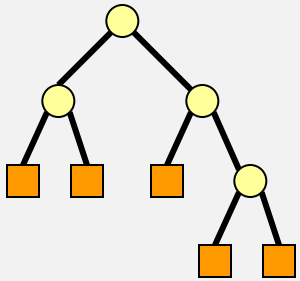




Ιδιότητες Δυαδικών Δένδρων

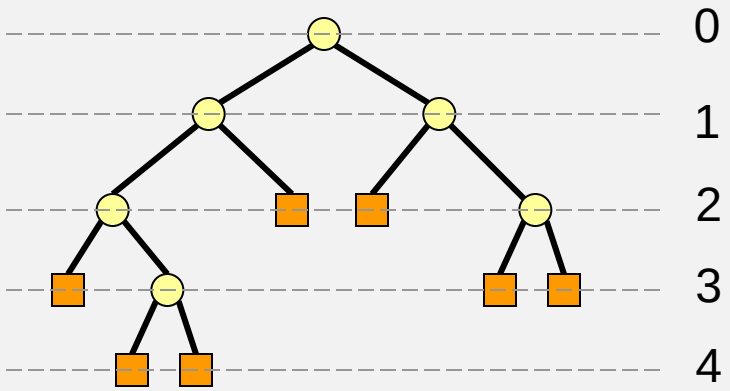
Ισχύει: μήκος εξωτερικής διαδρομής = μήκος εσωτερικής διαδρομής + 2N

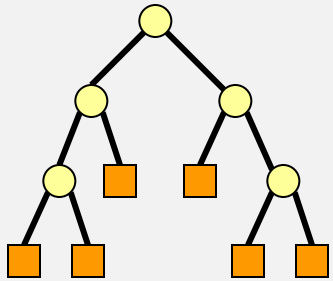




Ιδιότητες Δυαδικών Δένδρων

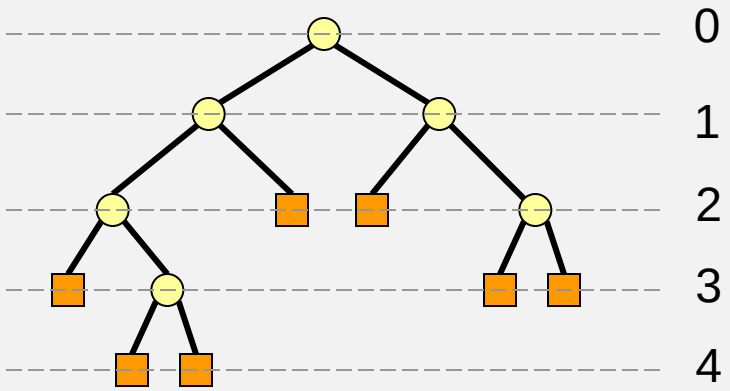
Ισχύει: μήκος εξωτερικής διαδρομής = μήκος εσωτερικής διαδρομής + 2N

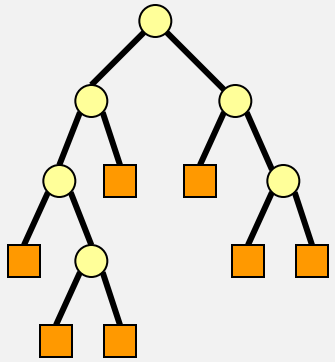




Ιδιότητες Δυαδικών Δένδρων

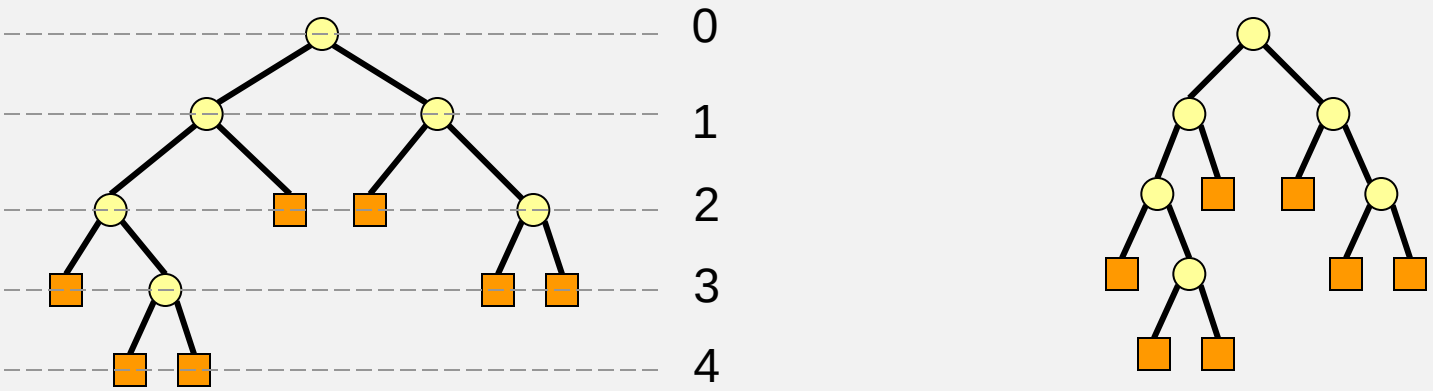
Ισχύει: μήκος εξωτερικής διαδρομής = μήκος εσωτερικής διαδρομής + 2N





Ιδιότητες Δυαδικών Δένδρων

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



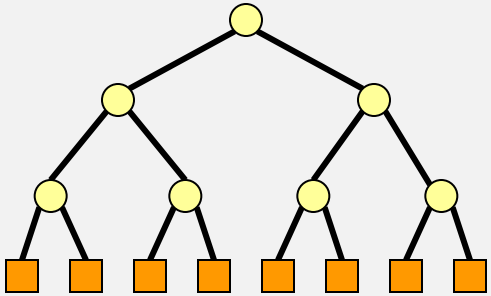
Αντικατάσταση εσωτερικού κόμβου στο επίπεδο k $\Rightarrow$

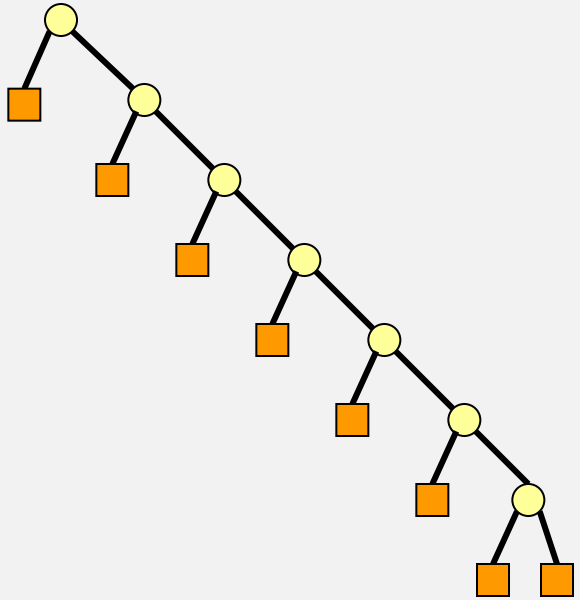
μήκος εσωτερικής διαδρομής αυξάνει κατά k

μήκος εξωτερικής διαδρομής αυξάνει κατά k+2

Ιδιότητες Δυαδικών Δένδρων

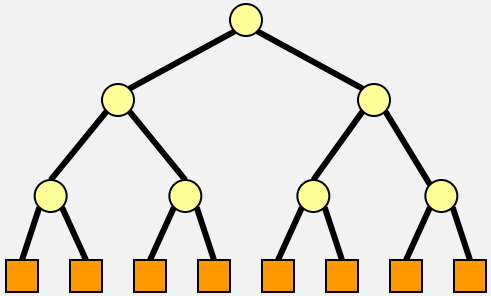
Ένα δυαδικό δένδρο με N εσωτερικούς κόμβους έχει ύψος μεταξύ lgN και N



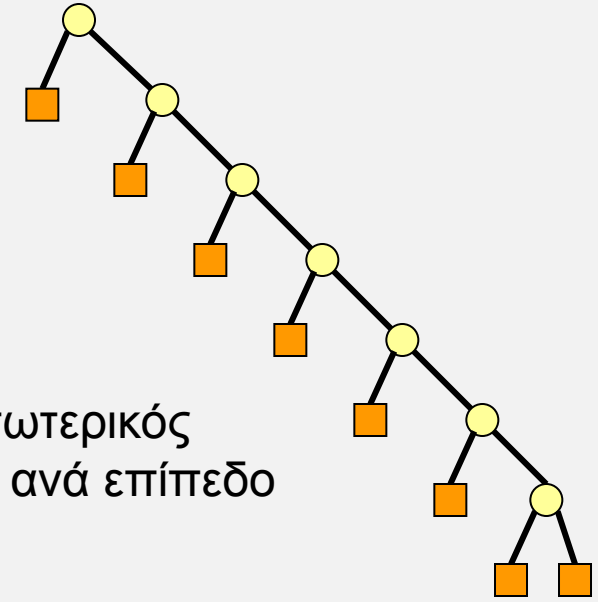


Ιδιότητες Δυαδικών Δένδρων

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



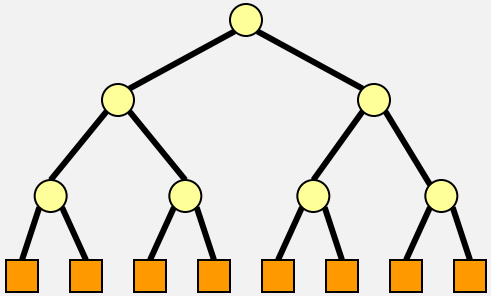
2^i εσωτερικοί κόμβοι στο επίπεδο i



ένας εσωτερικός
κόμβος ανά επίπεδο

Ιδιότητες Δυαδικών Δένδρων

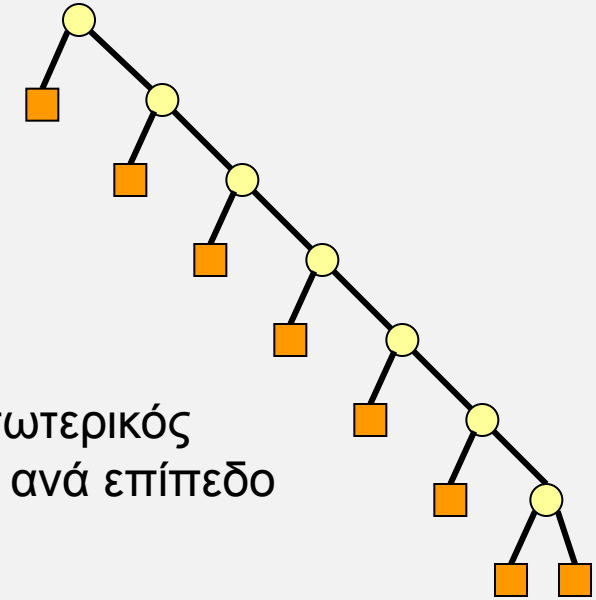
Ένα δυαδικό δένδρο με N εσωτερικούς κόμβους έχει ύψος μεταξύ lgN και N



2^i εσωτερικοί κόμβοι στο επίπεδο i

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

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

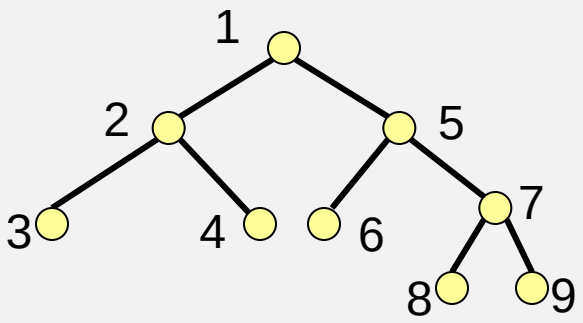


ένας εσωτερικός κόμβος ανά επίπεδο

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

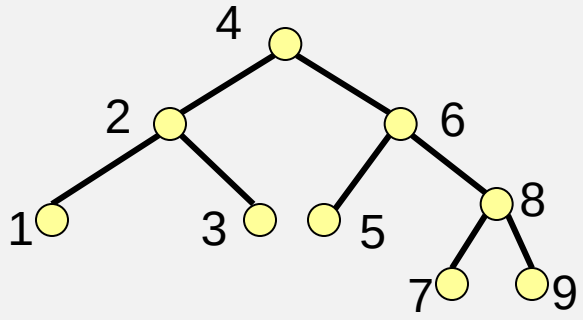
Διάσχιση Δυαδικού Δένδρου

προδιάταξη (preorder)



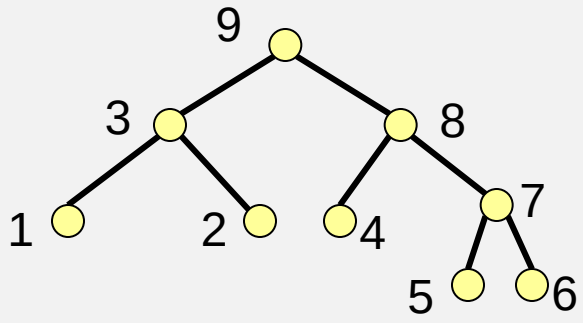
σειρά επεξεργασίας :
1. γονέας
2. αριστερό παιδί
3. δεξί παιδί

ενδοδιάταξη (inorder)



σειρά επεξεργασίας :
1. αριστερό παιδί
2. γονέας
3. δεξί παιδί

μεταδιάταξη (postorder)



σειρά επεξεργασίας :
1. αριστερό παιδί
2. δεξί παιδί
3. γονέας

Διάσχιση Δυαδικού Δένδρου

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

προδιάταξη (preorder)
σειρά επεξεργασίας :
1. γονέας
2. αριστερό παιδί
3. δεξί παιδί

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

ενδοδιάταξη (inorder)
σειρά επεξεργασίας :
1. αριστερό παιδί
2. γονέας
3. δεξί παιδί

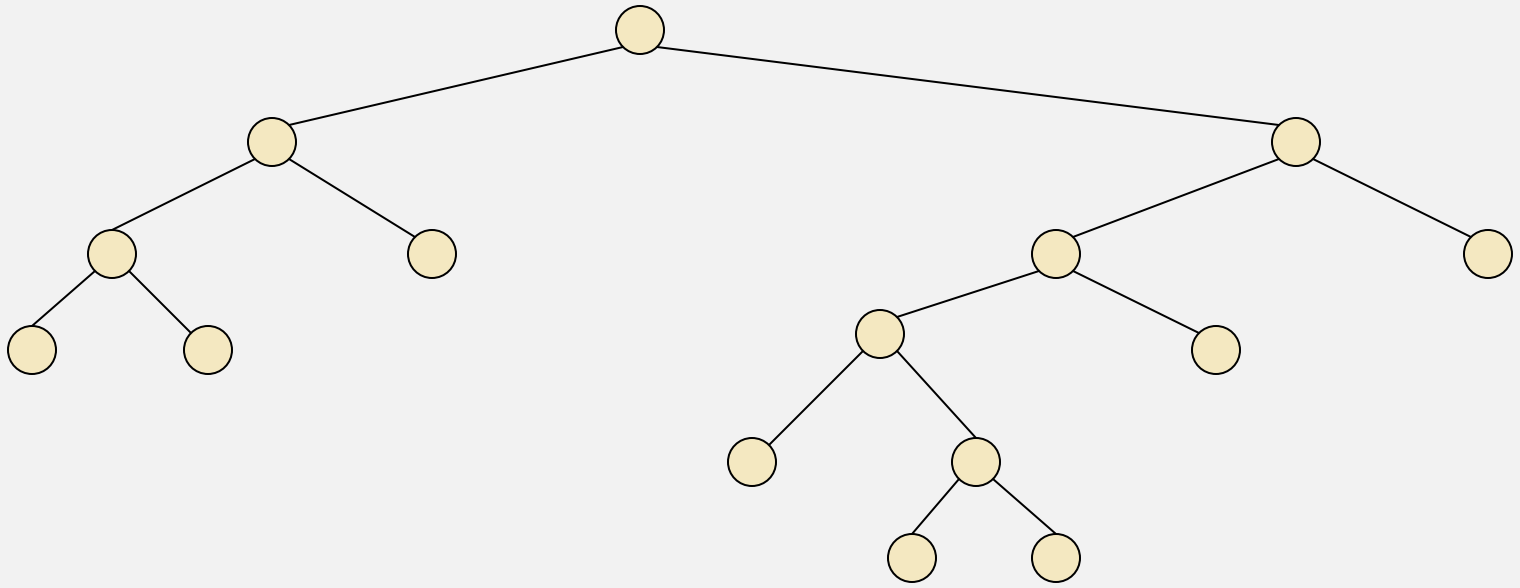
```
void traverse(Node h) {
        if (h == null) return;
        traverse(h.l);
        traverse(h.r);
        h.item.visit();
}
```

μεταδιάταξη (postorder)
σειρά επεξεργασίας :
1. αριστερό παιδί
2. δεξί παιδί
3. γονέας

Διάσχιση Δυαδικού Δένδρου

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

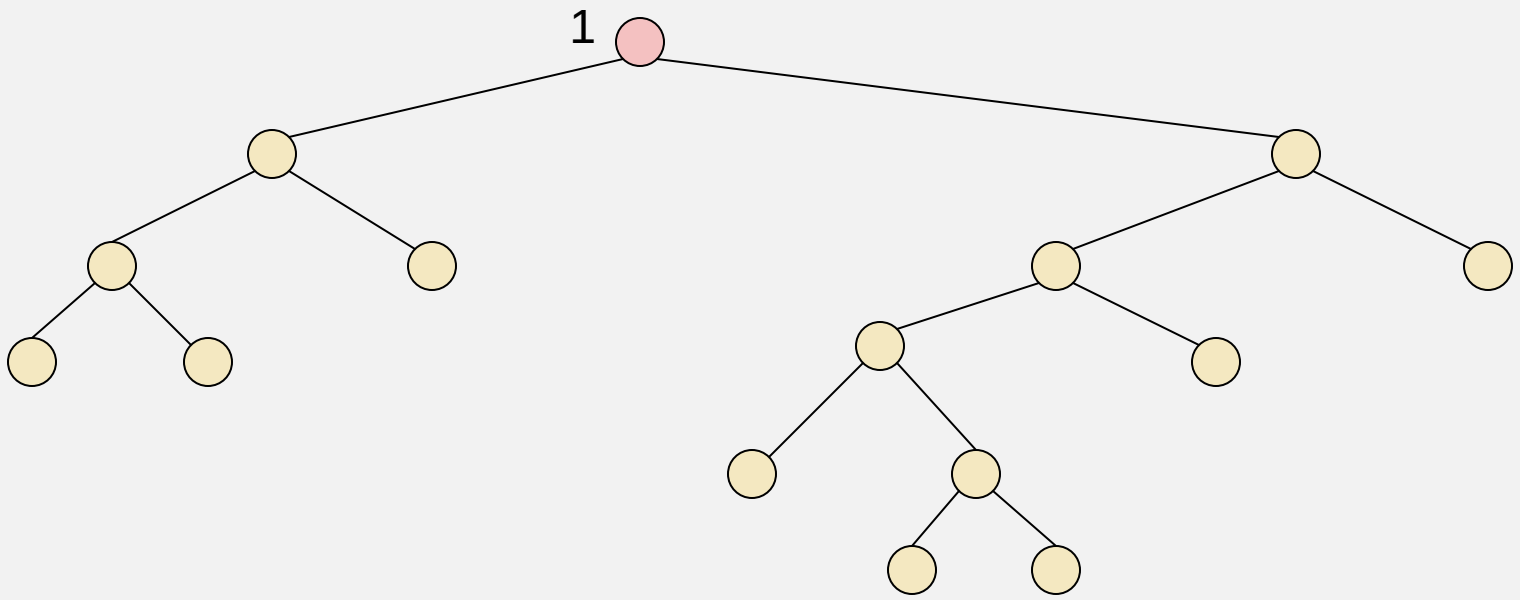
προδιάταξη (preorder)



Διάσχιση Δυαδικού Δένδρου

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

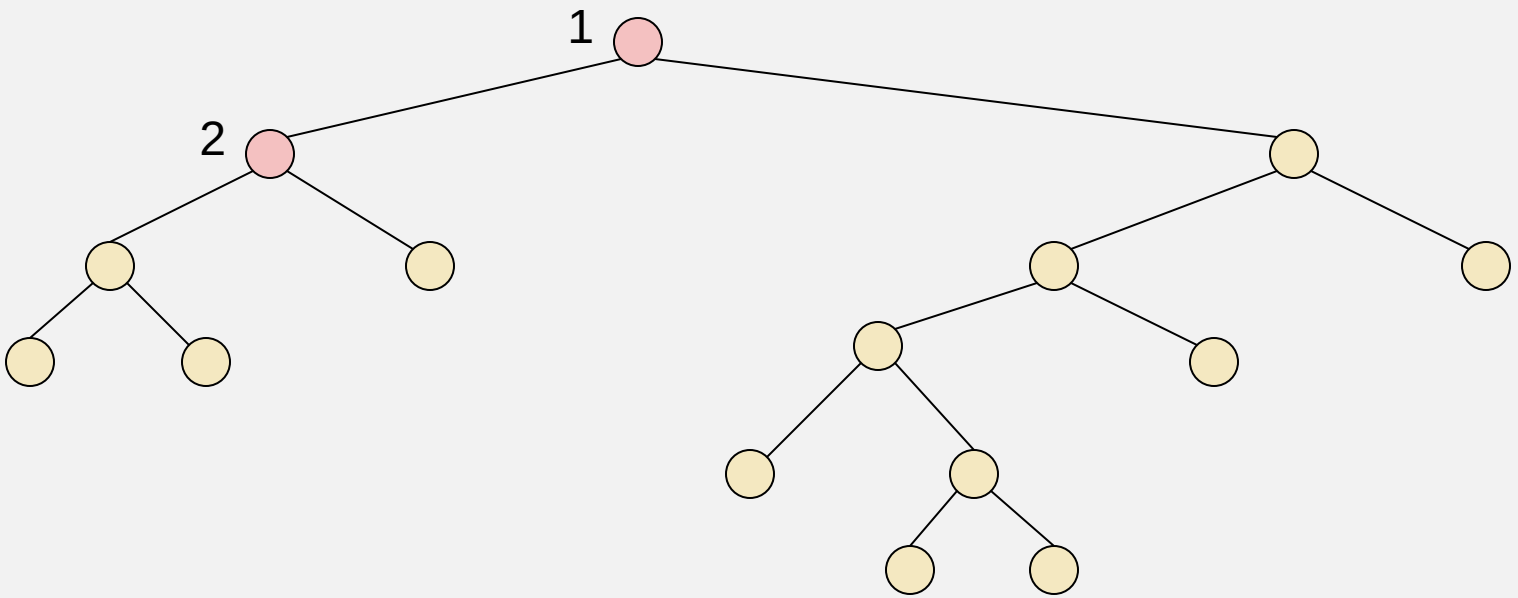
προδιάταξη (preorder)



Διάσχιση Δυαδικού Δένδρου

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

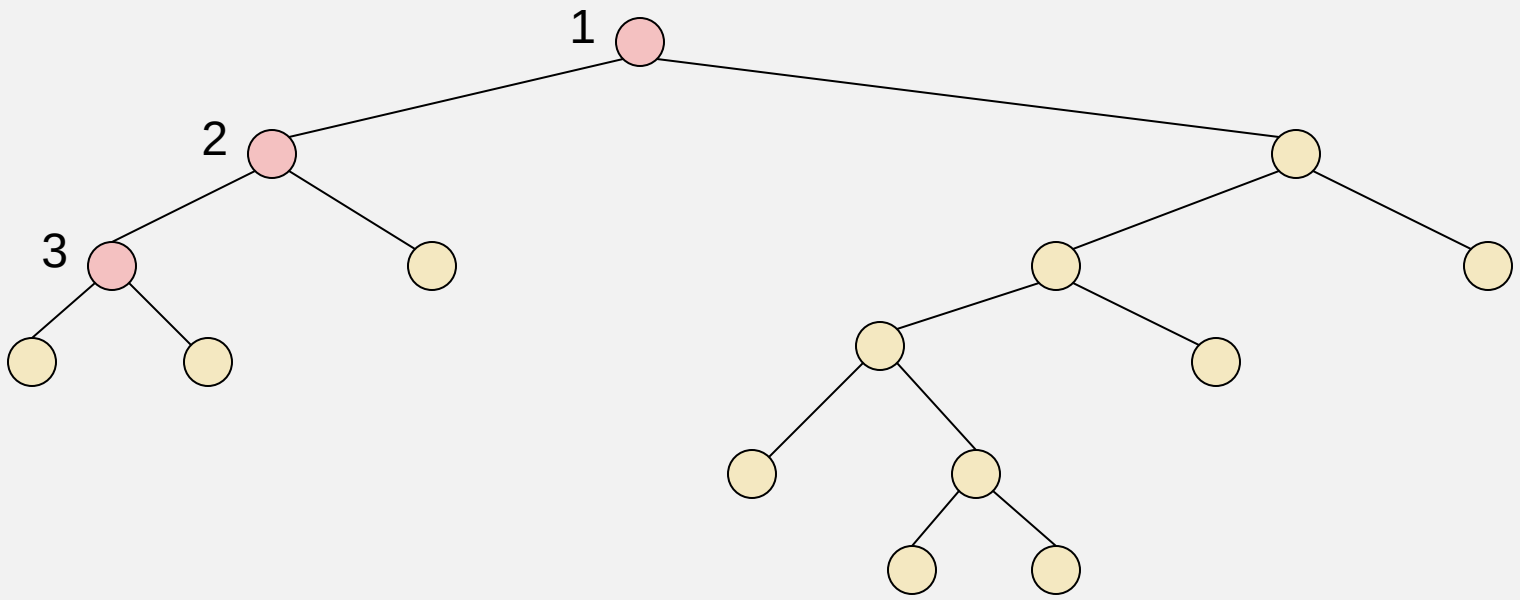
προδιάταξη (preorder)



Διάσχιση Δυαδικού Δένδρου

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

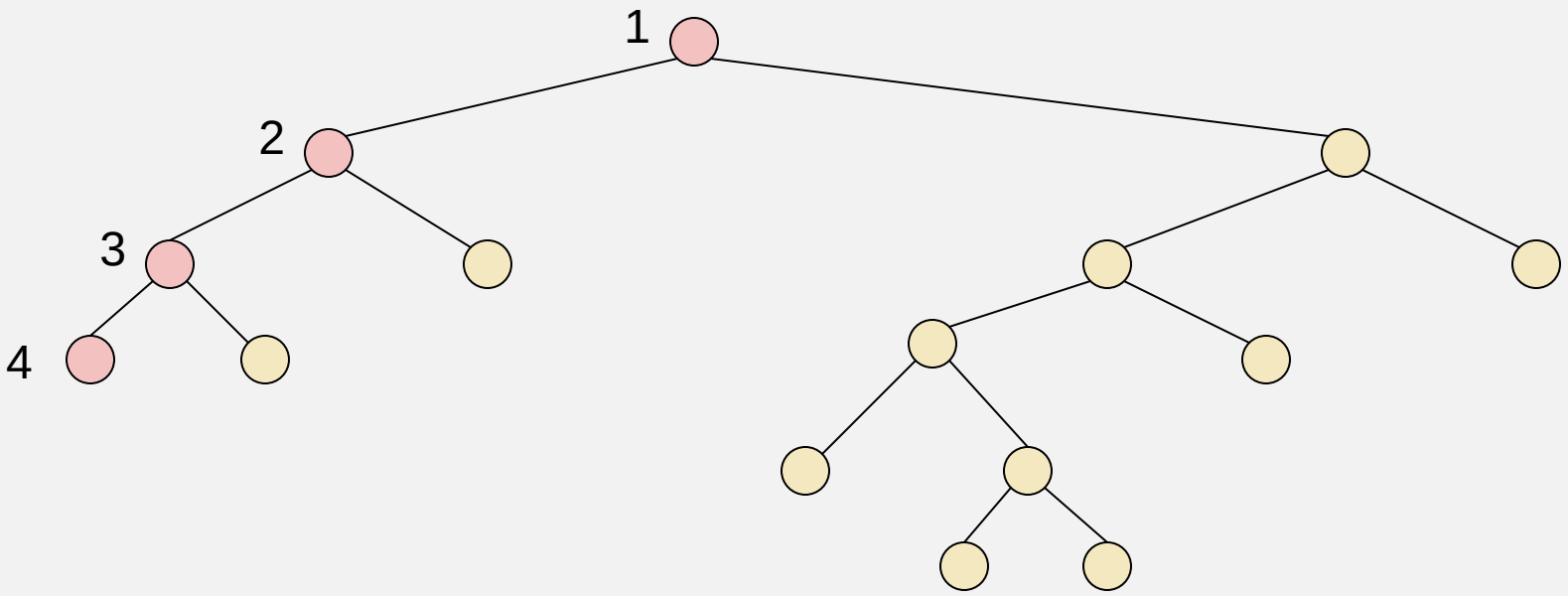
προδιάταξη (preorder)



Διάσχιση Δυαδικού Δένδρου

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

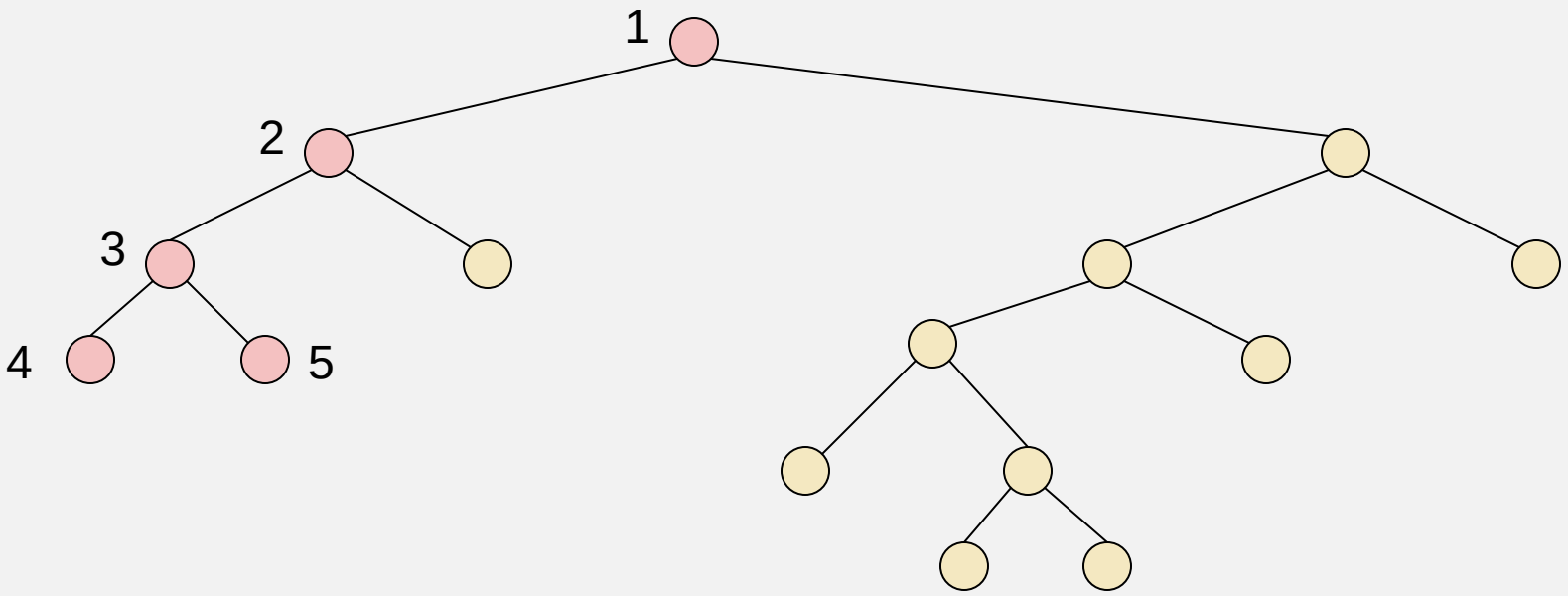
προδιάταξη (preorder)



Διάσχιση Δυαδικού Δένδρου

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

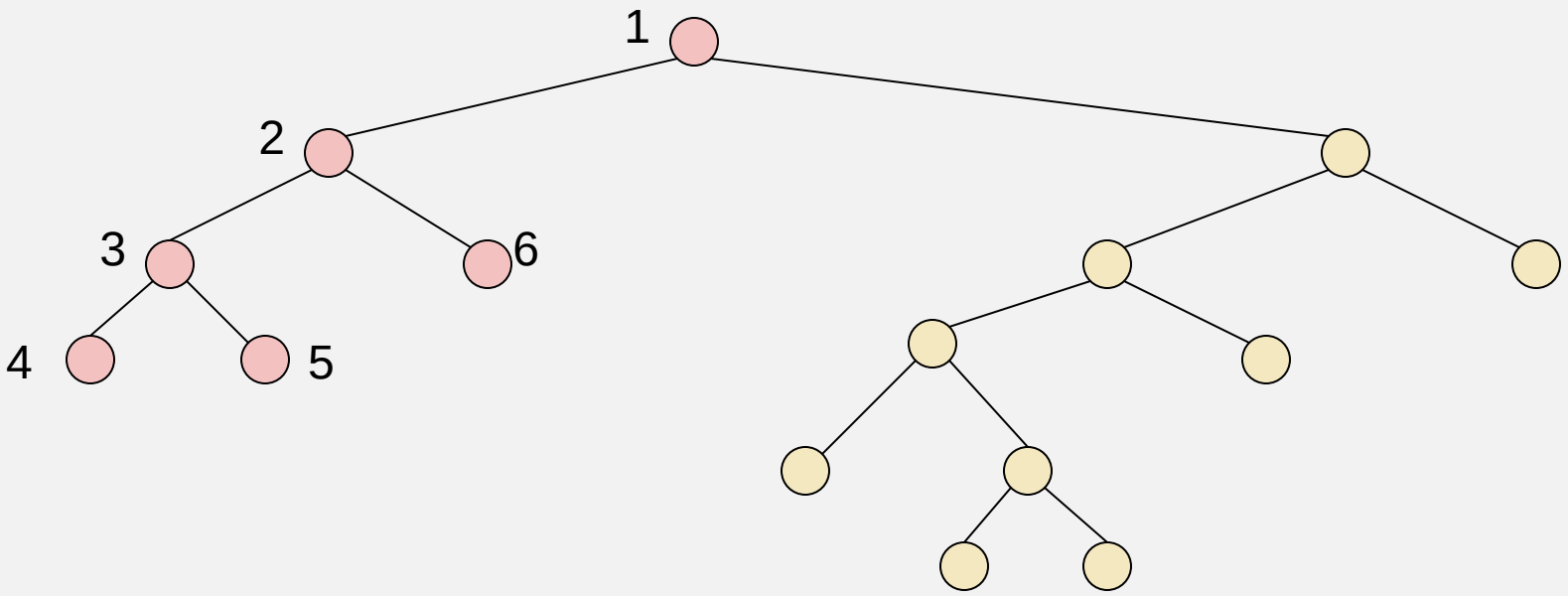
προδιάταξη (preorder)



Διάσχιση Δυαδικού Δένδρου

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

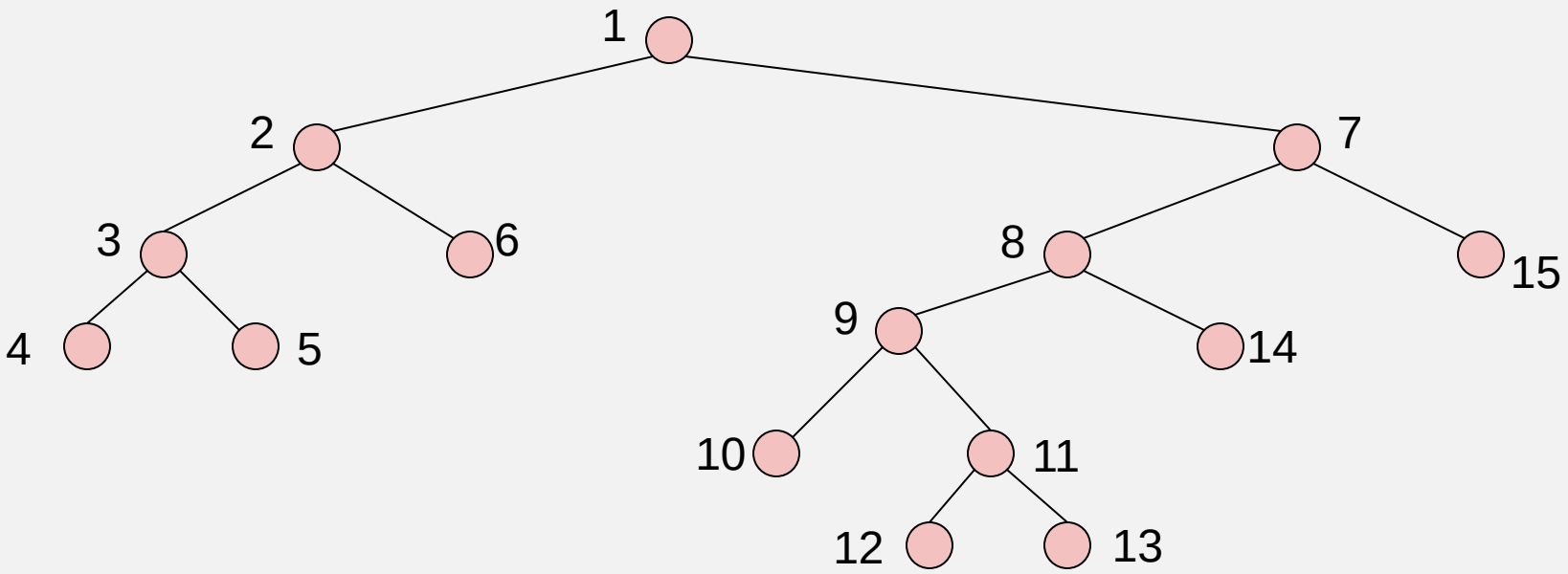
προδιάταξη (preorder)



Διάσχιση Δυαδικού Δένδρου

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

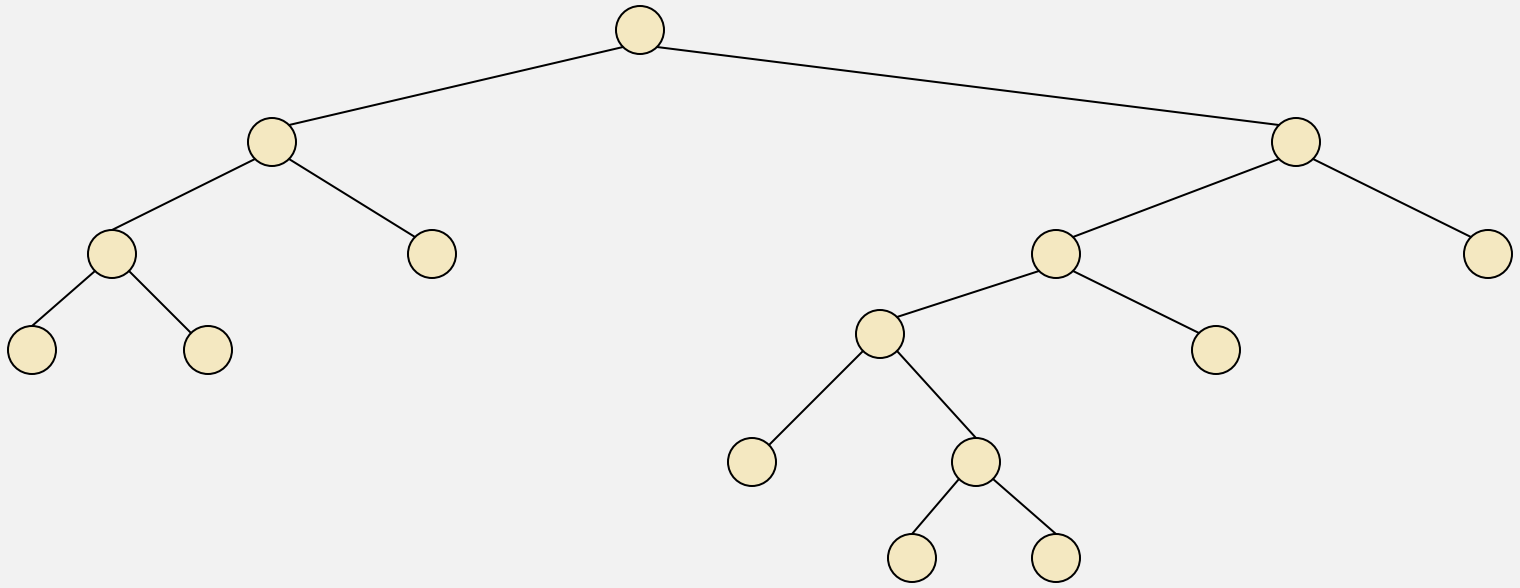
προδιάταξη (preorder)



Διάσχιση Δυαδικού Δένδρου

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

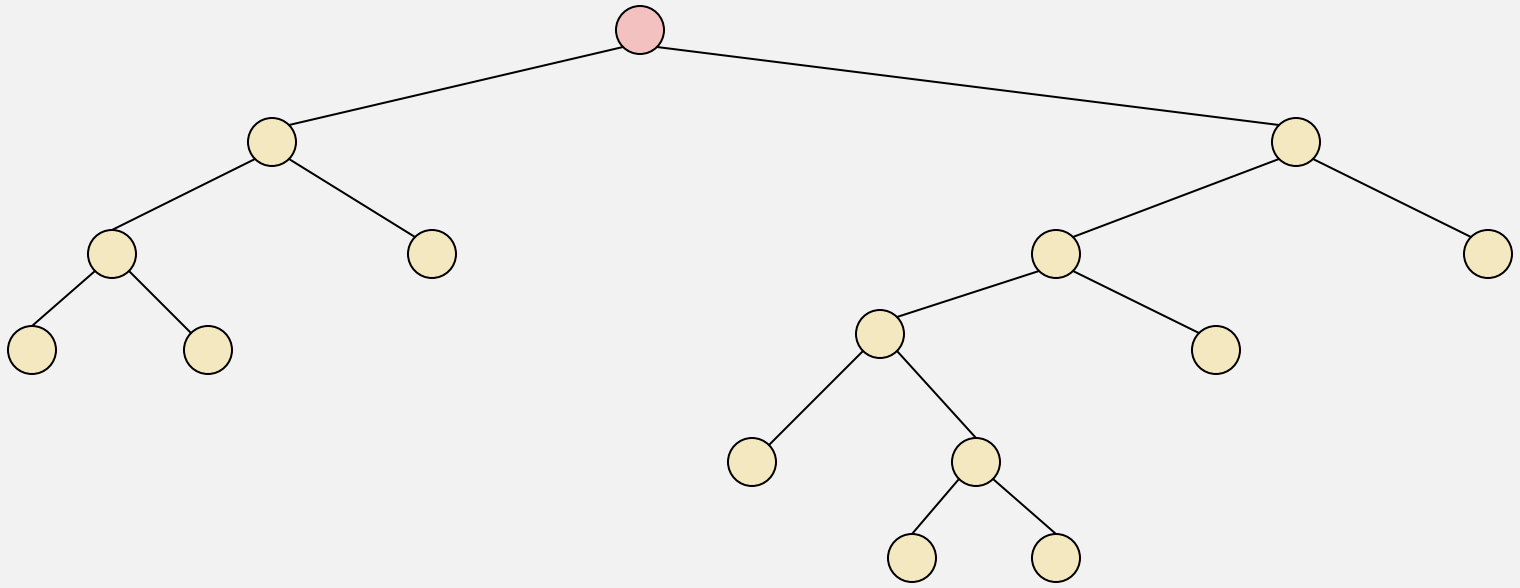
ενδοδιάταξη (inorder)



Διάσχιση Δυαδικού Δένδρου

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

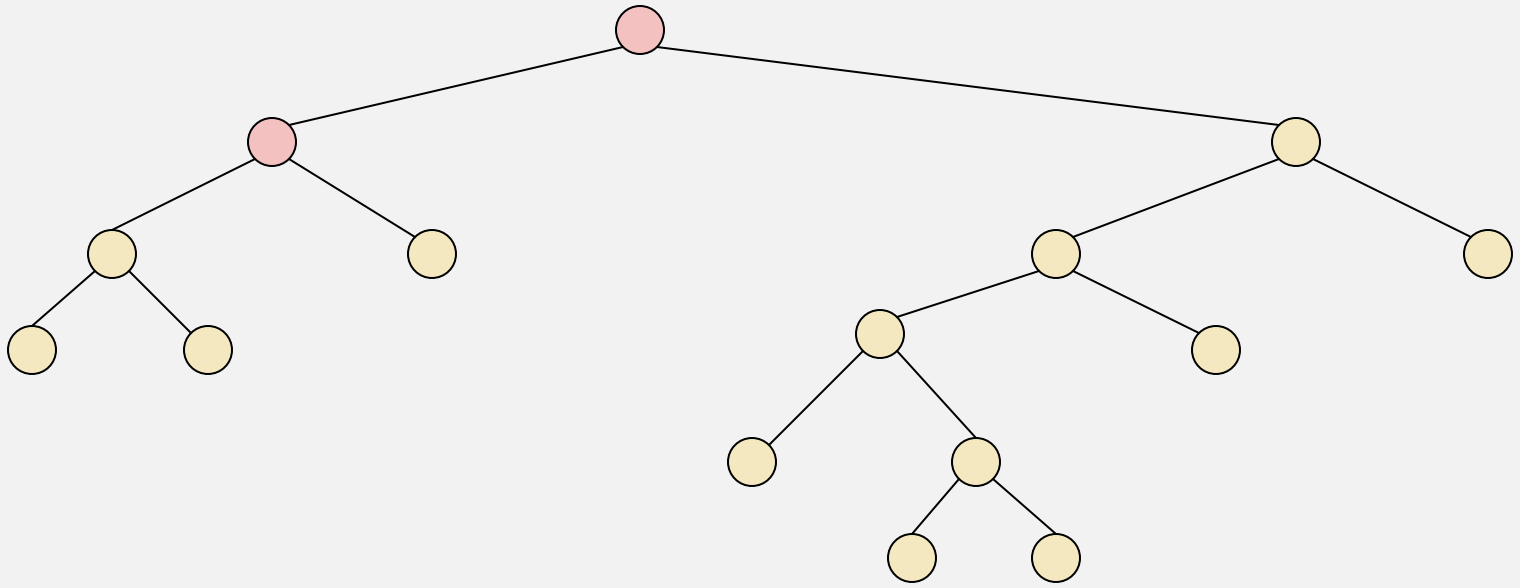
ενδοδιάταξη (inorder)



Διάσχιση Δυαδικού Δένδρου

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

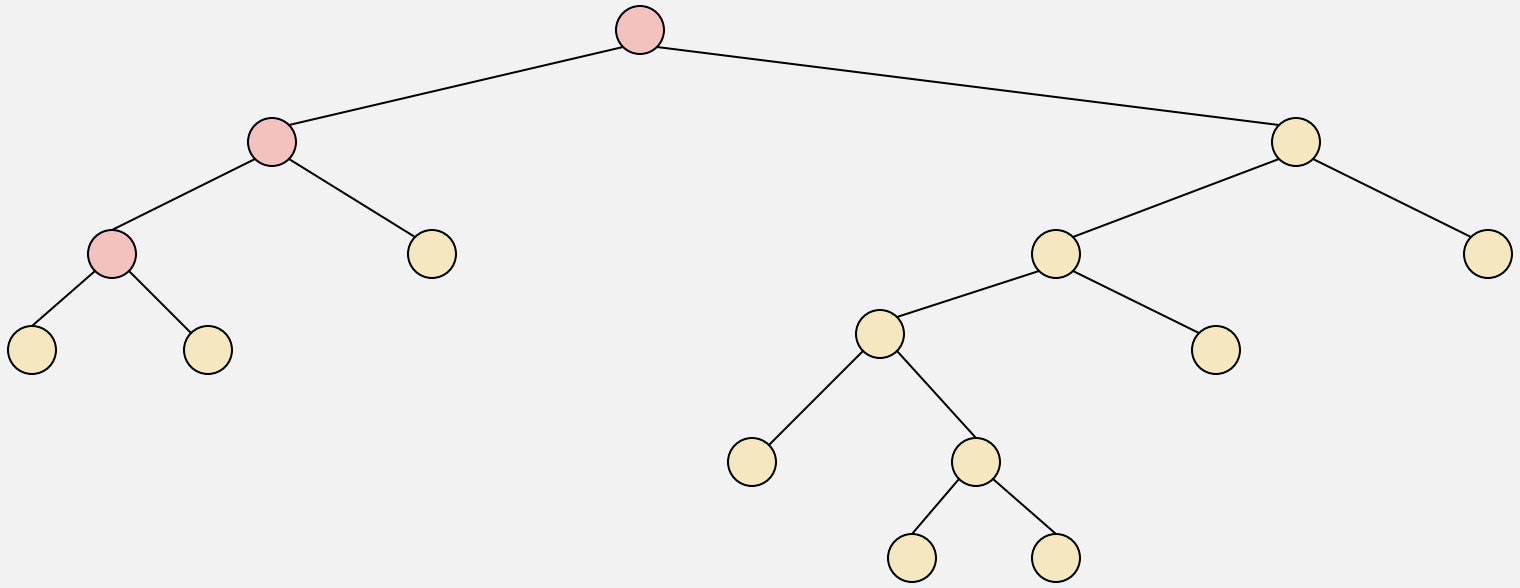
ενδοδιάταξη (inorder)



Διάσχιση Δυαδικού Δένδρου

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

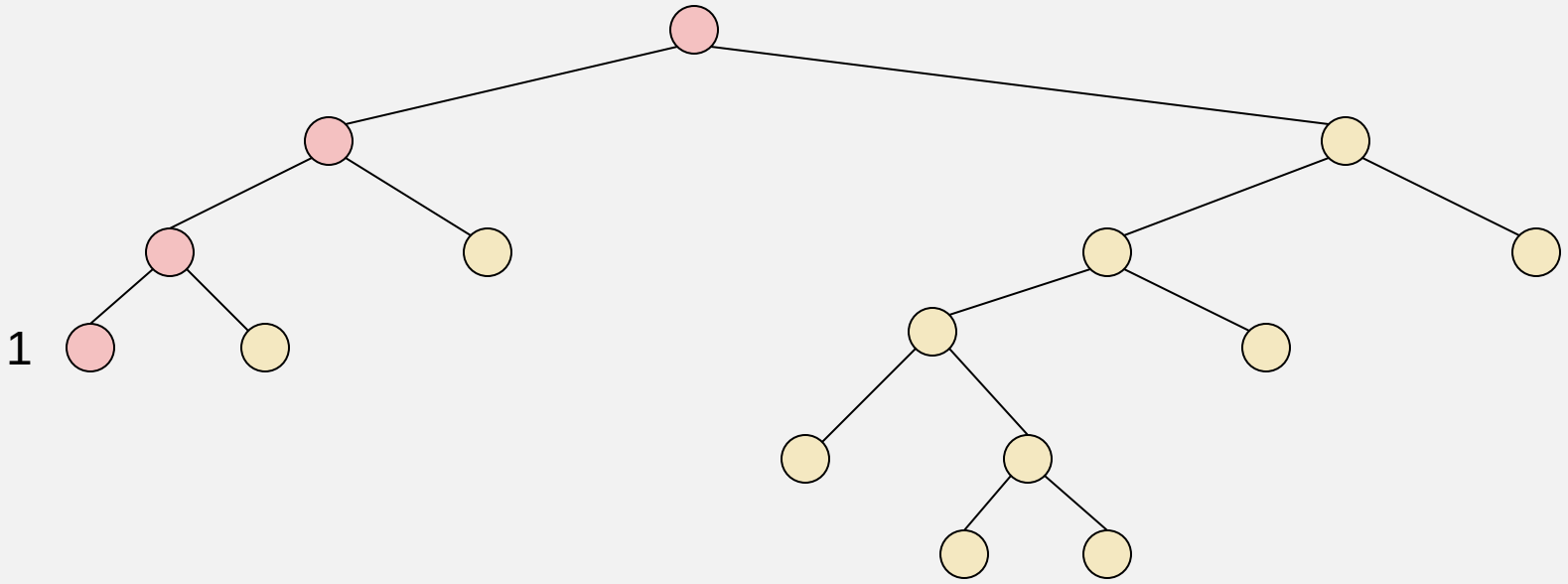
ενδοδιάταξη (inorder)



Διάσχιση Δυαδικού Δένδρου

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

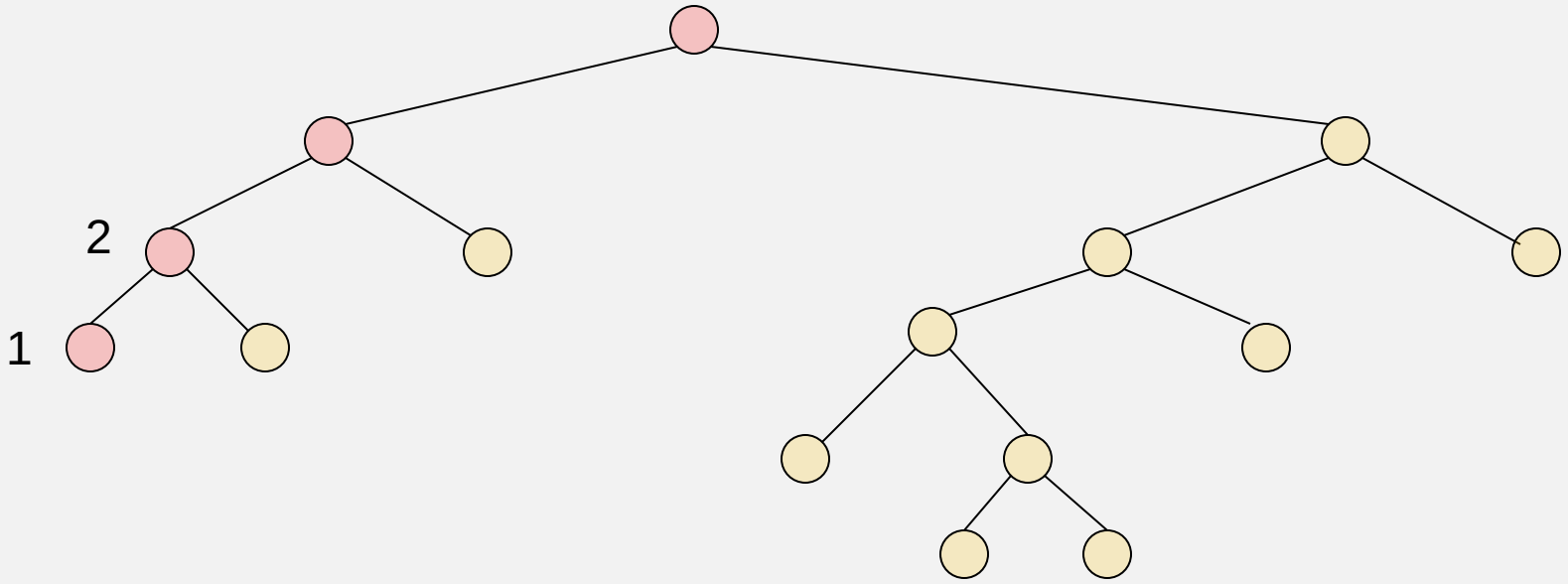
ενδοδιάταξη (inorder)



Διάσχιση Δυαδικού Δένδρου

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

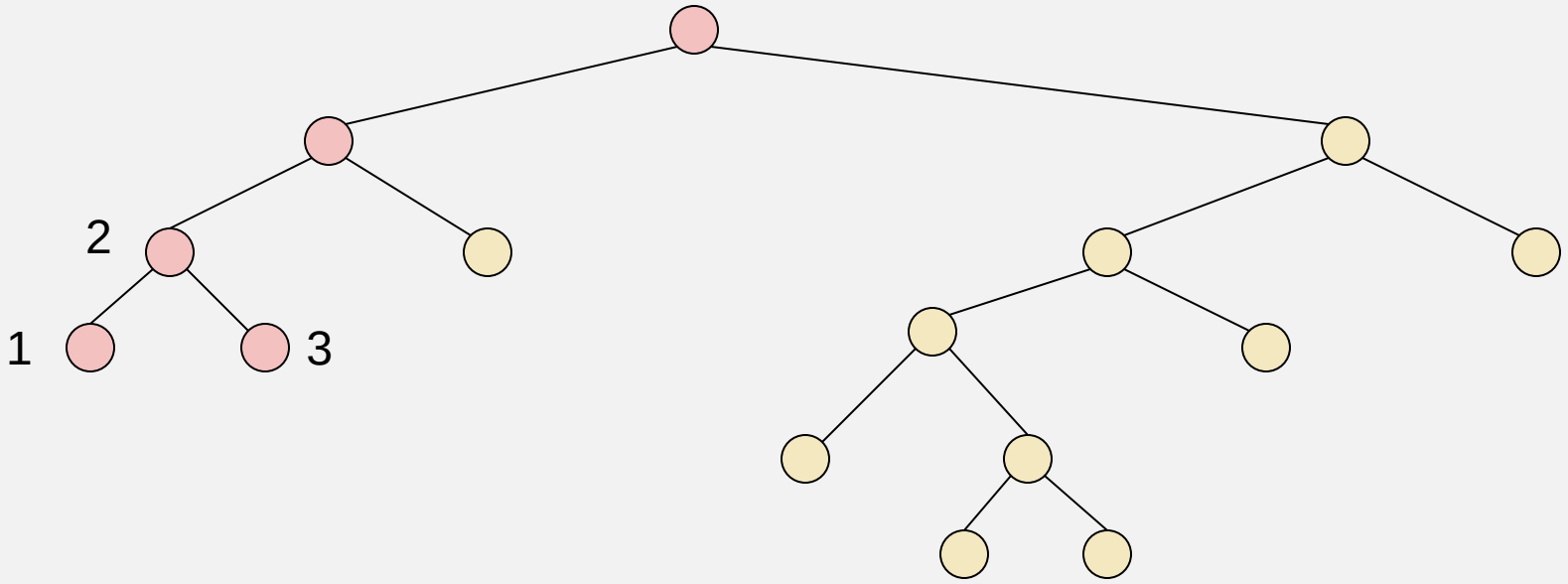
ενδοδιάταξη (inorder)



Διάσχιση Δυαδικού Δένδρου

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

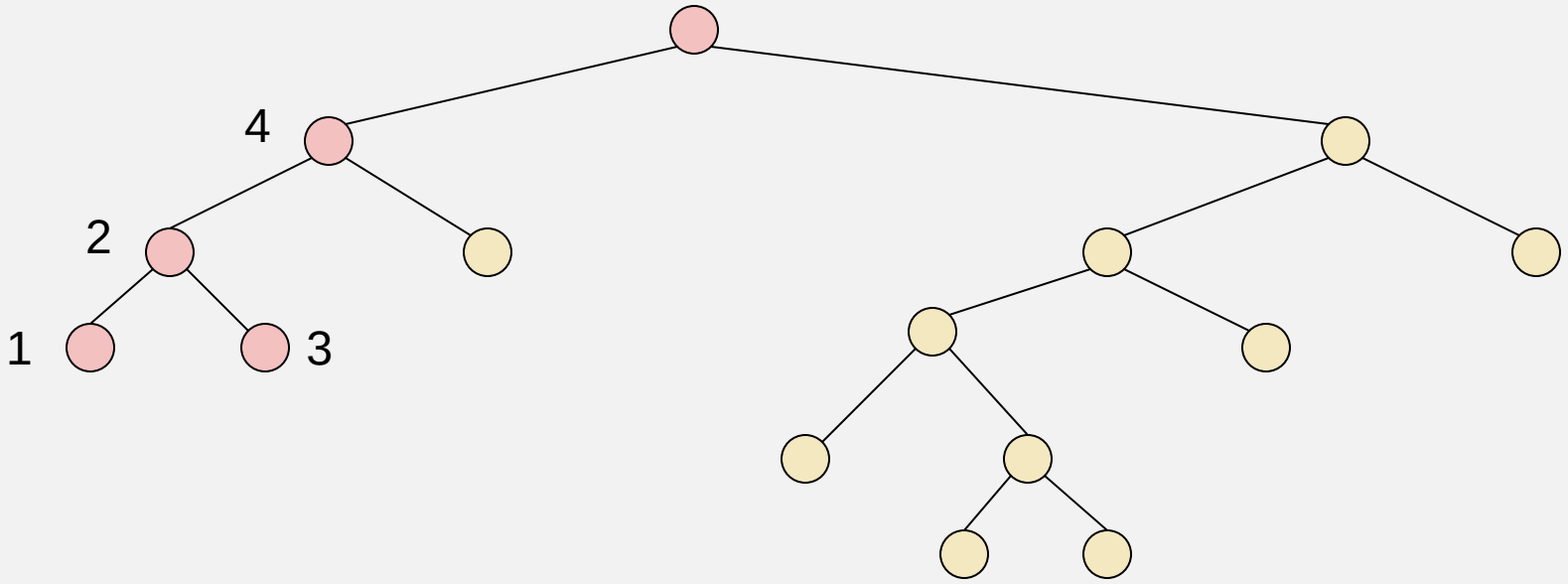
ενδοδιάταξη (inorder)



Διάσχιση Δυαδικού Δένδρου

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

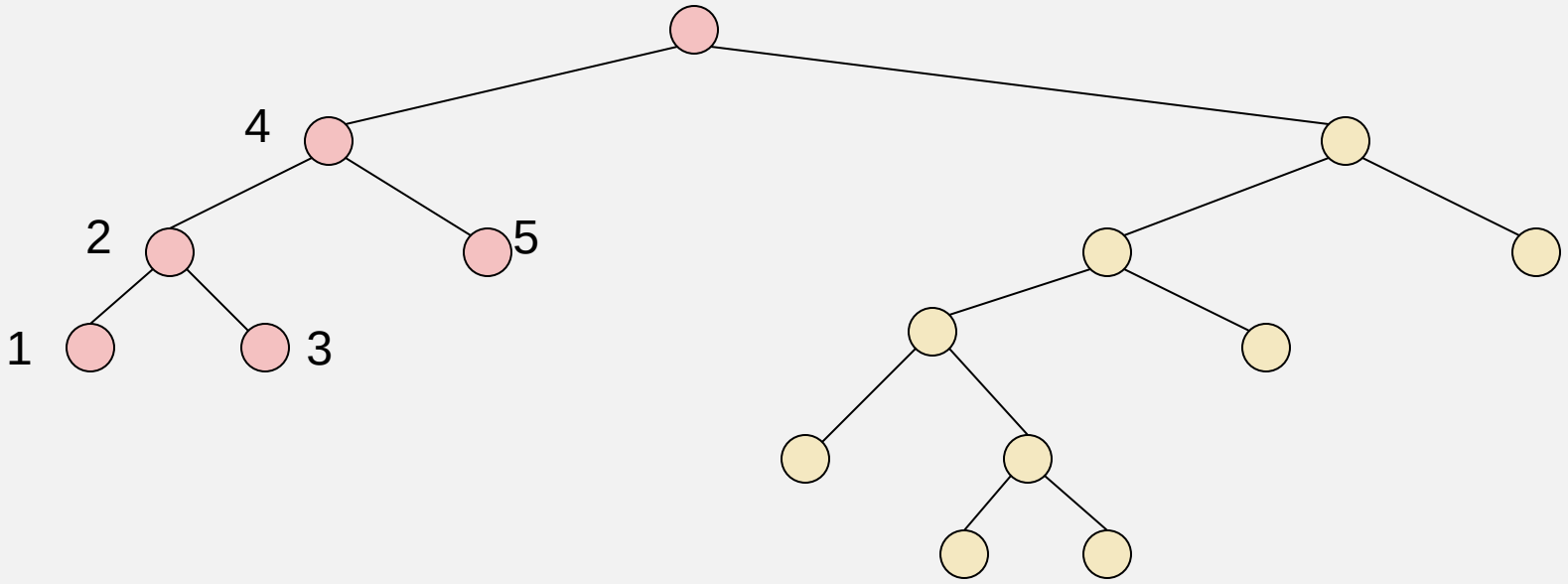
ενδοδιάταξη (inorder)



Διάσχιση Δυαδικού Δένδρου

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

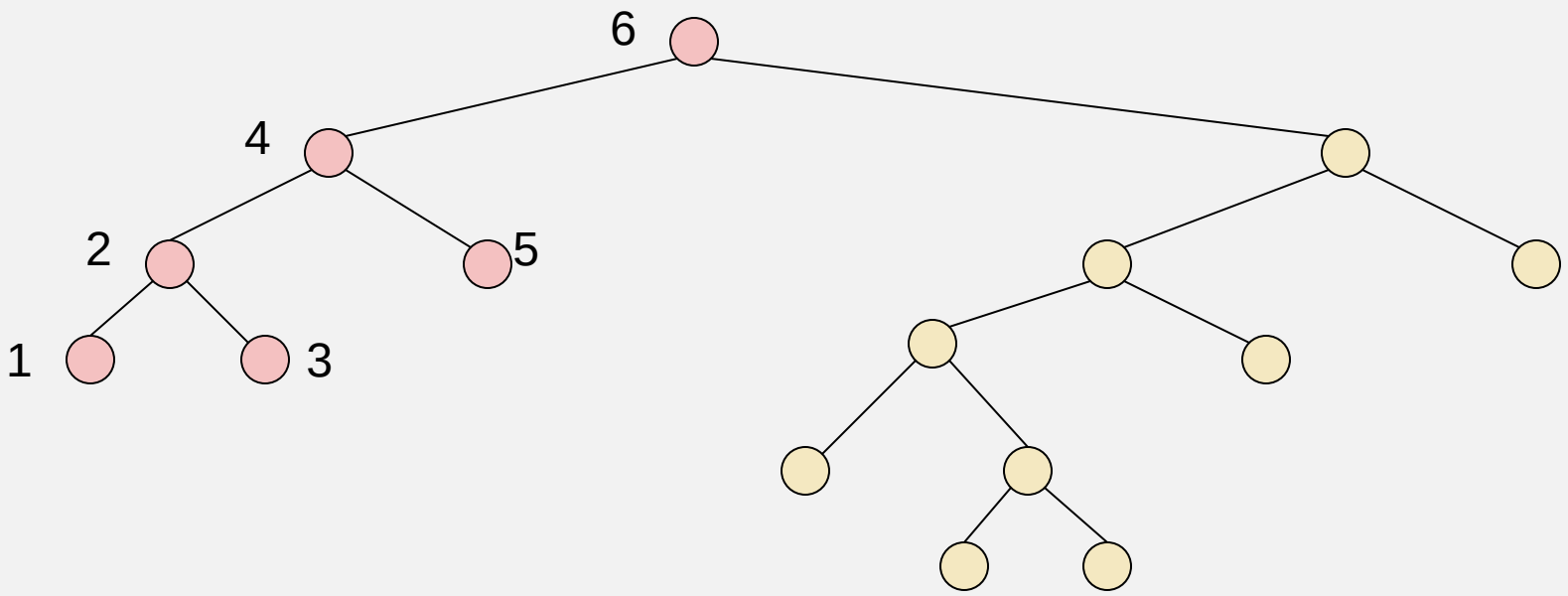
ενδοδιάταξη (inorder)



Διάσχιση Δυαδικού Δένδρου

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

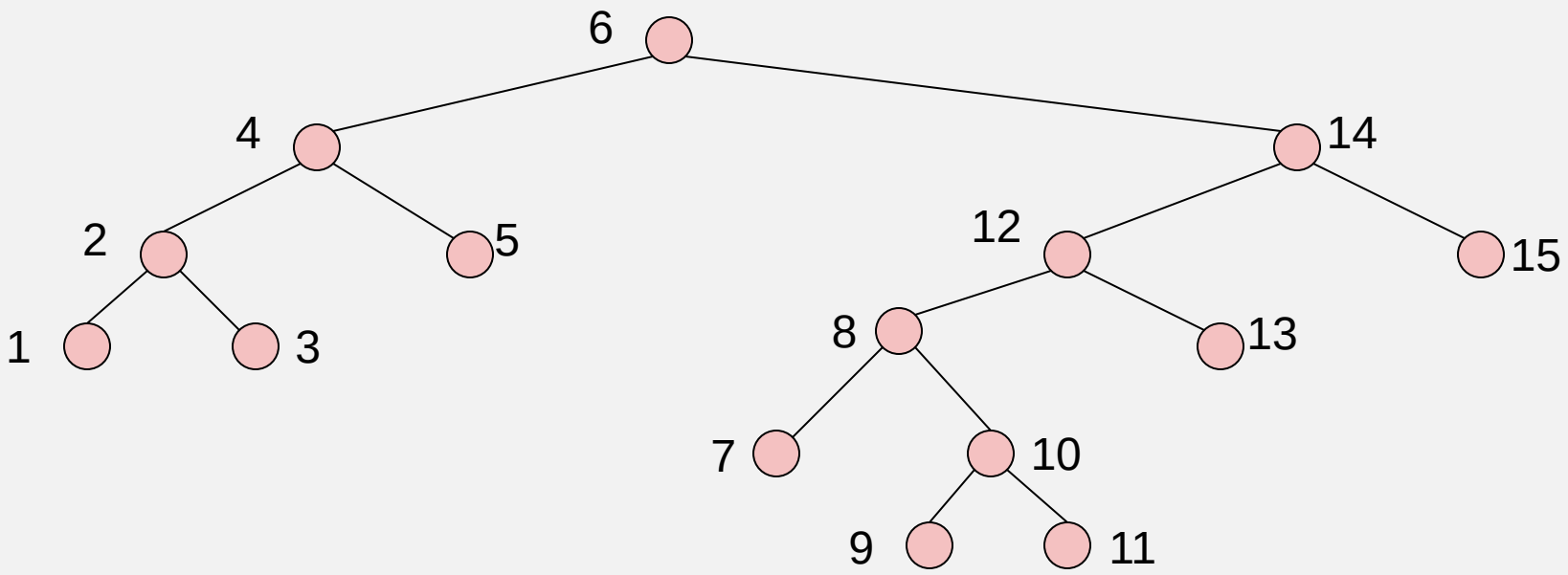
ενδοδιάταξη (inorder)



Διάσχιση Δυαδικού Δένδρου

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

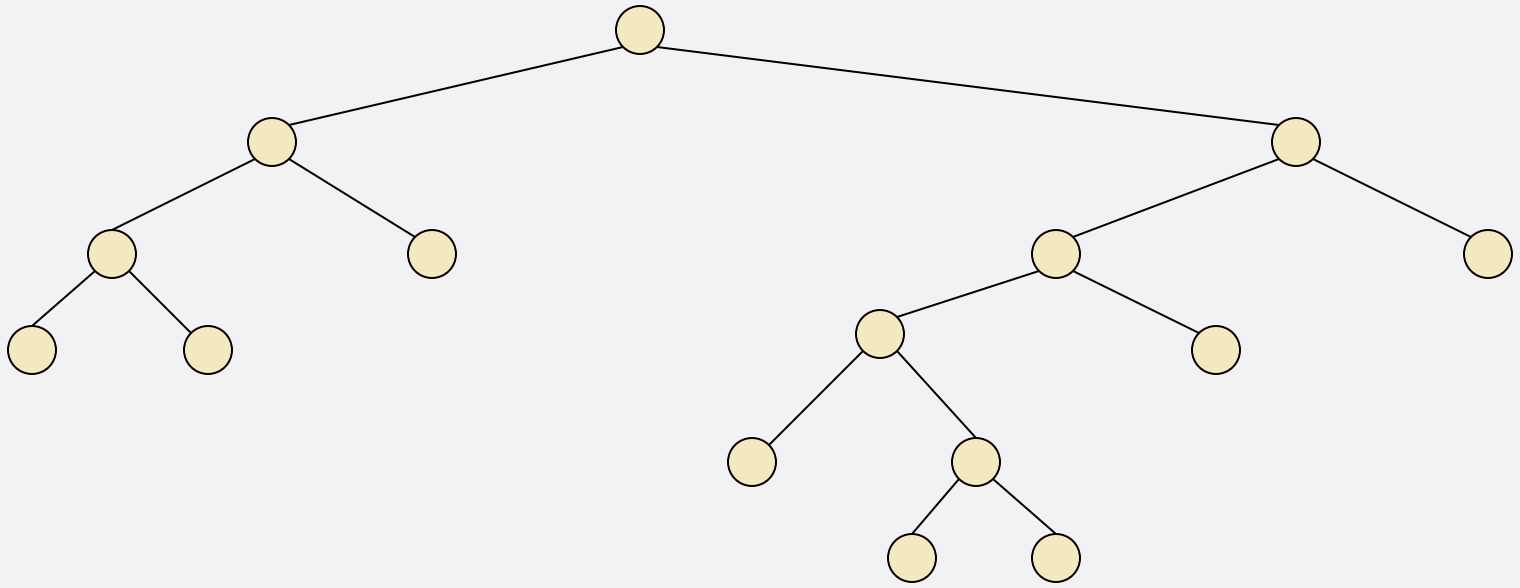
ενδοδιάταξη (inorder)



Διάσχιση Δυαδικού Δένδρου

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

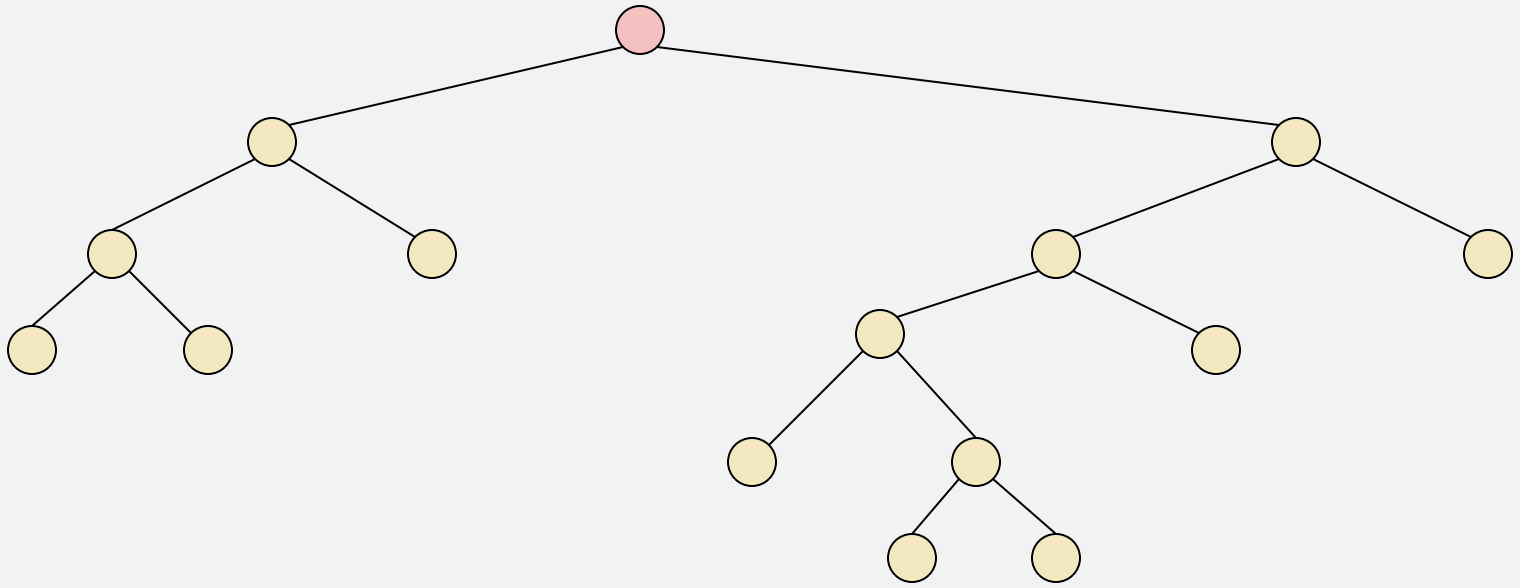
μεταδιάταξη (postorder)



Διάσχιση Δυαδικού Δένδρου

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

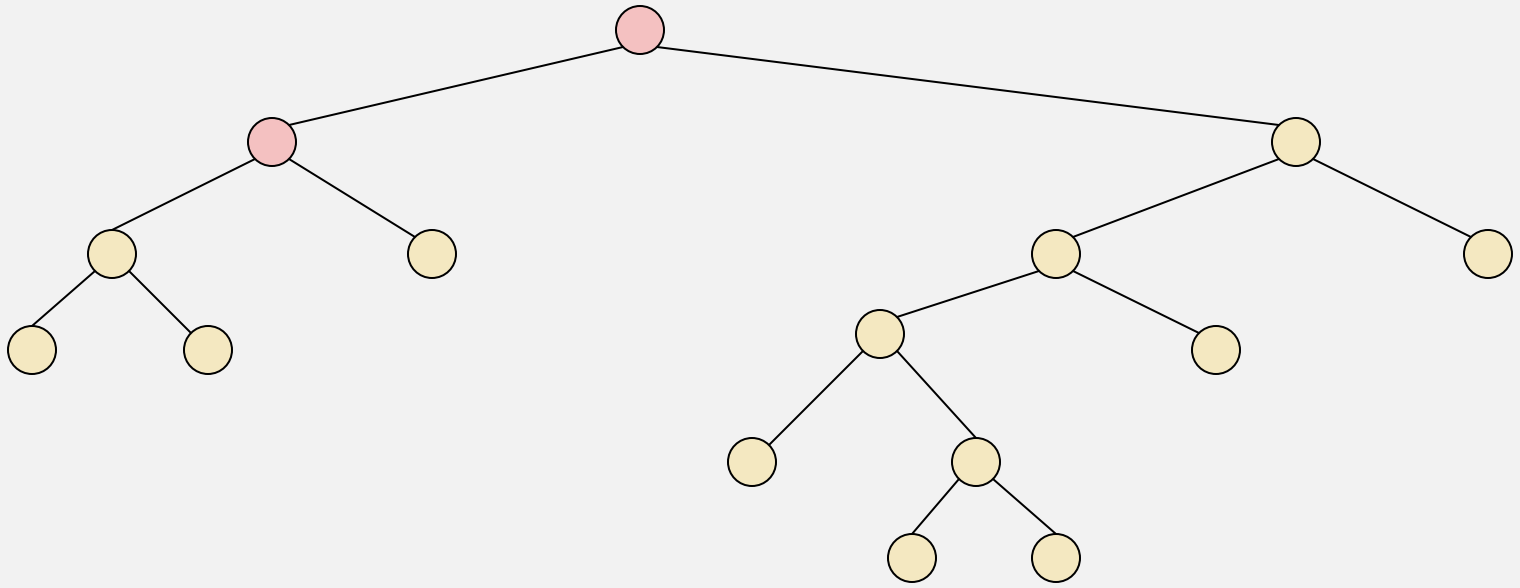
μεταδιάταξη (postorder)



Διάσχιση Δυαδικού Δένδρου

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

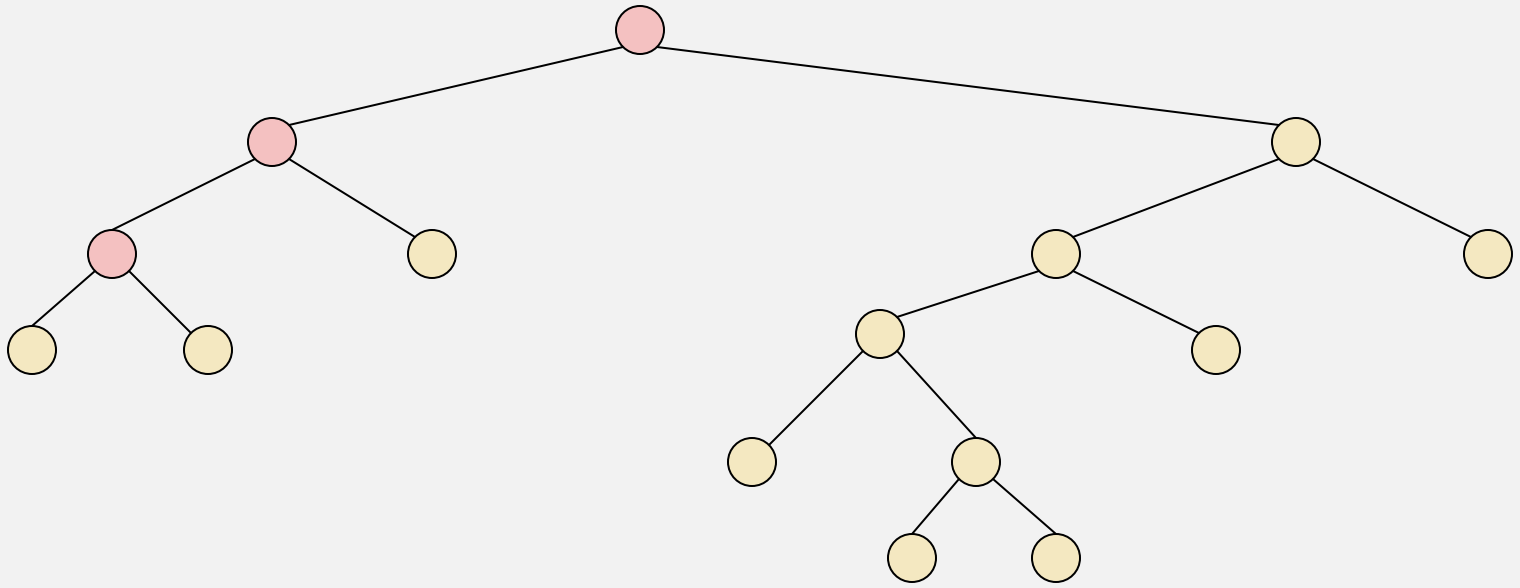
μεταδιάταξη (postorder)



Διάσχιση Δυαδικού Δένδρου

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

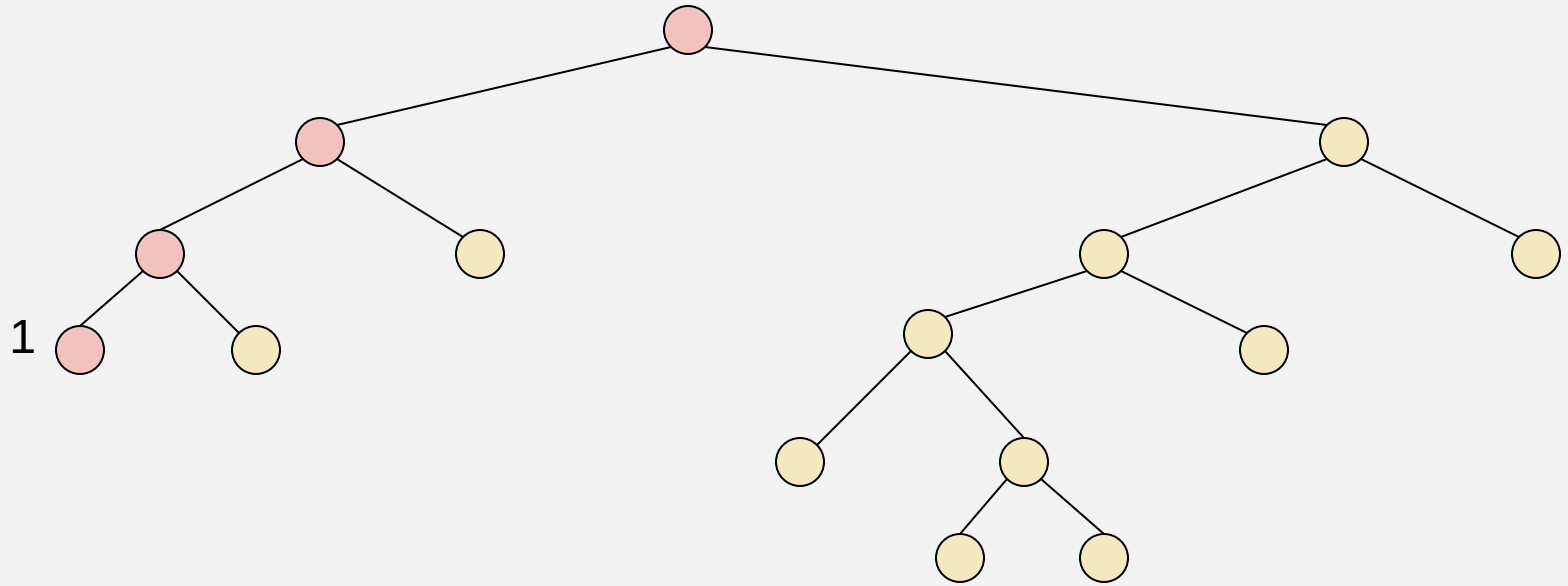
μεταδιάταξη (postorder)



Διάσχιση Δυαδικού Δένδρου

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

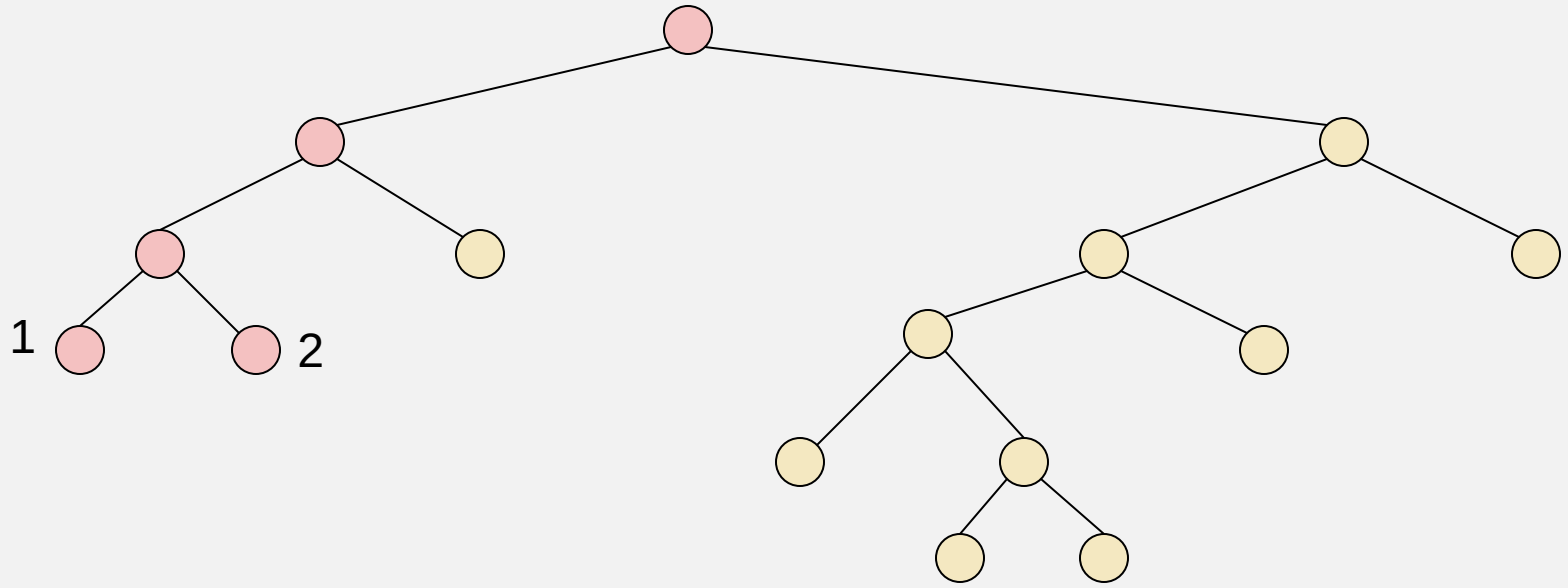
μεταδιάταξη (postorder)



Διάσχιση Δυαδικού Δένδρου

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

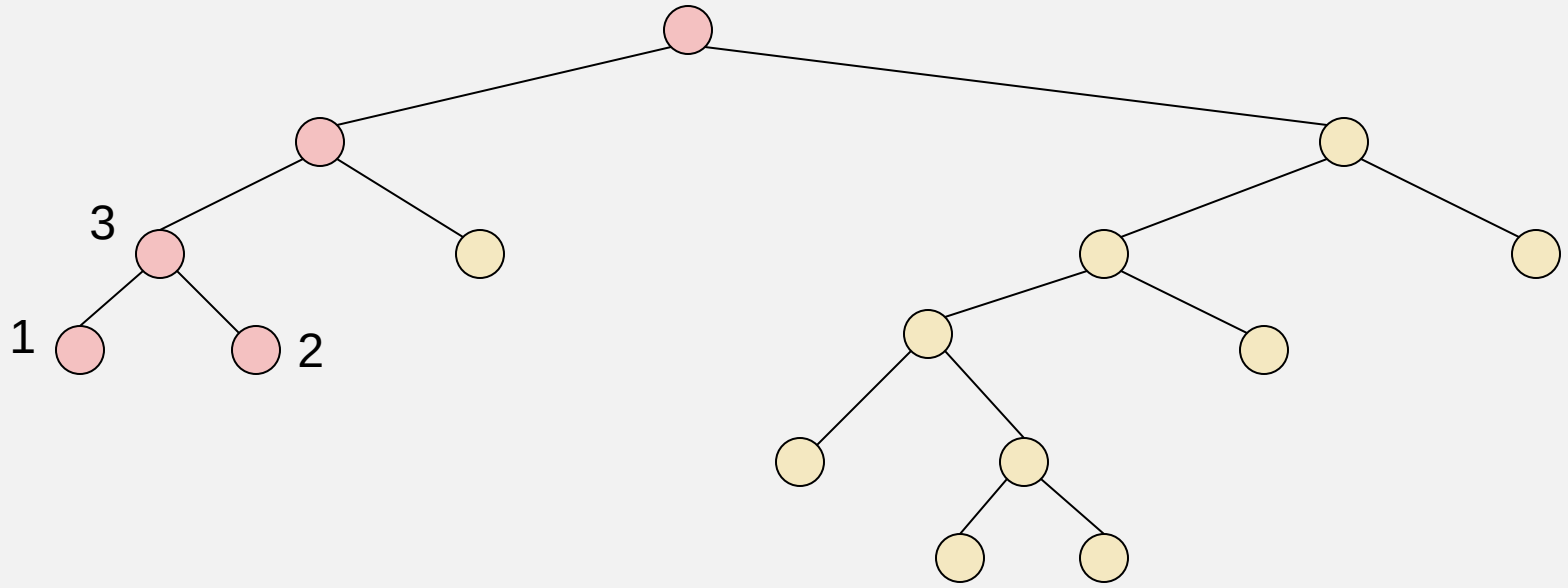
μεταδιάταξη (postorder)



Διάσχιση Δυαδικού Δένδρου

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

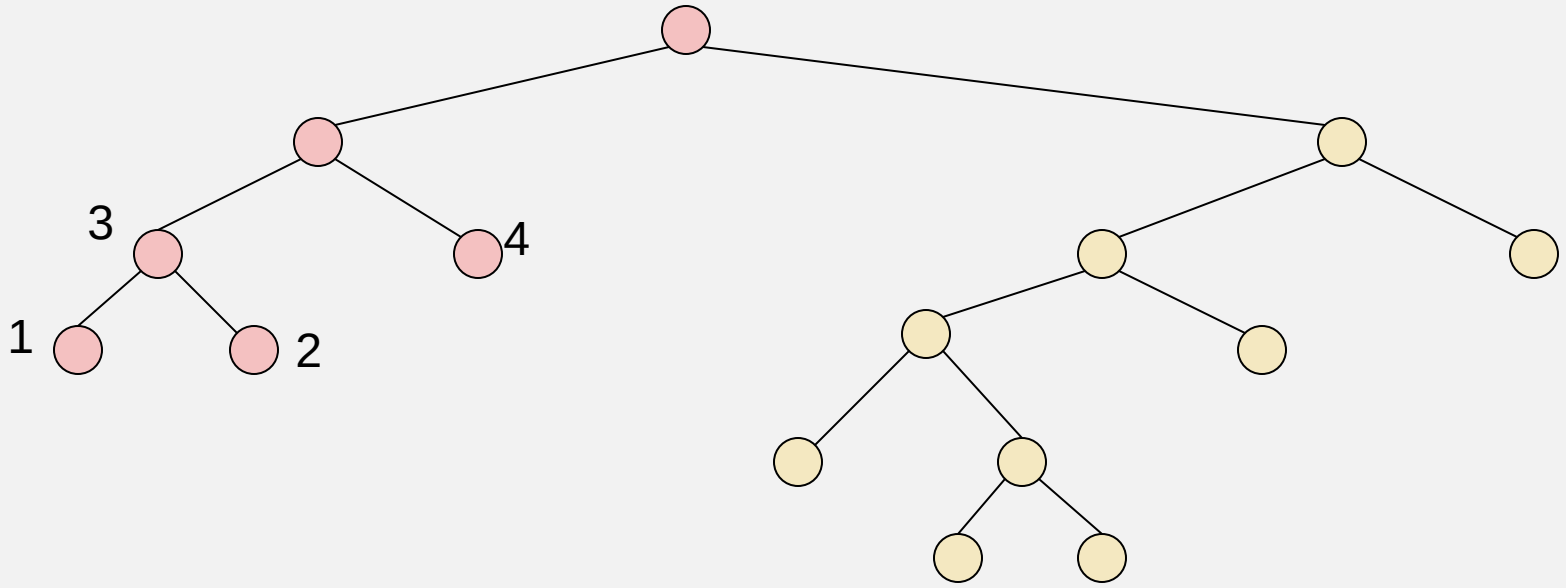
μεταδιάταξη (postorder)



Διάσχιση Δυαδικού Δένδρου

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

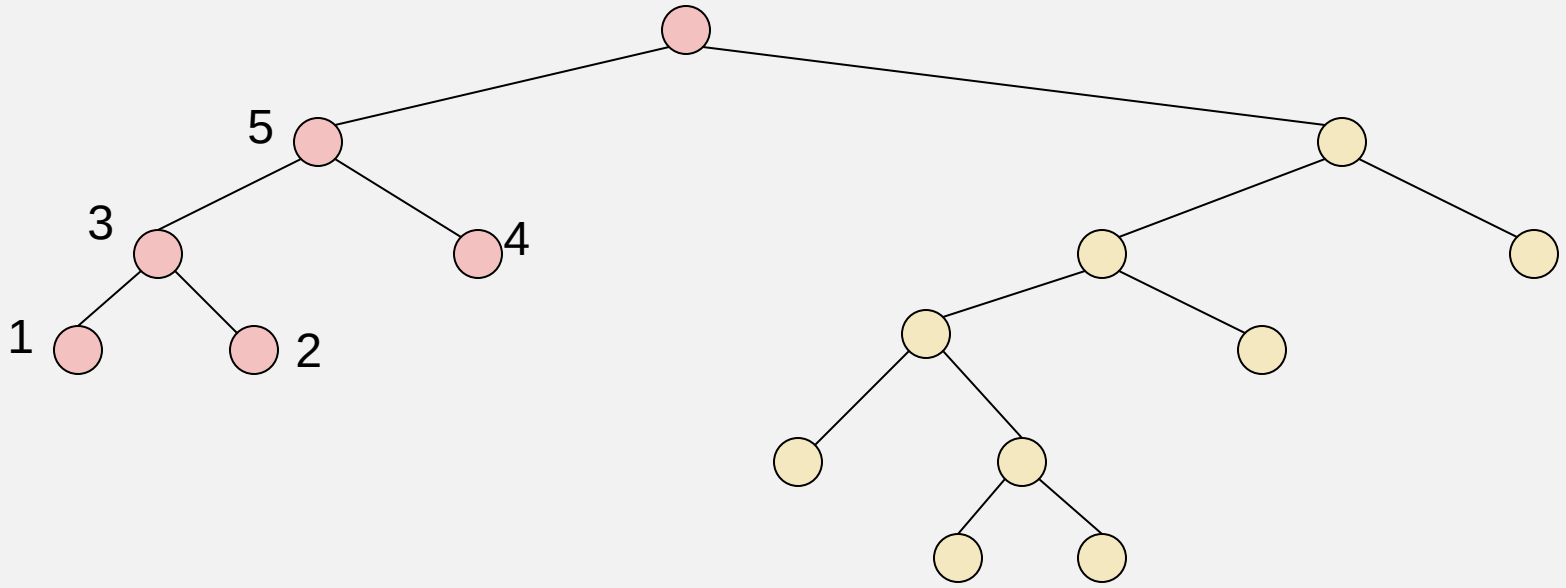
μεταδιάταξη (postorder)



Διάσχιση Δυαδικού Δένδρου

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

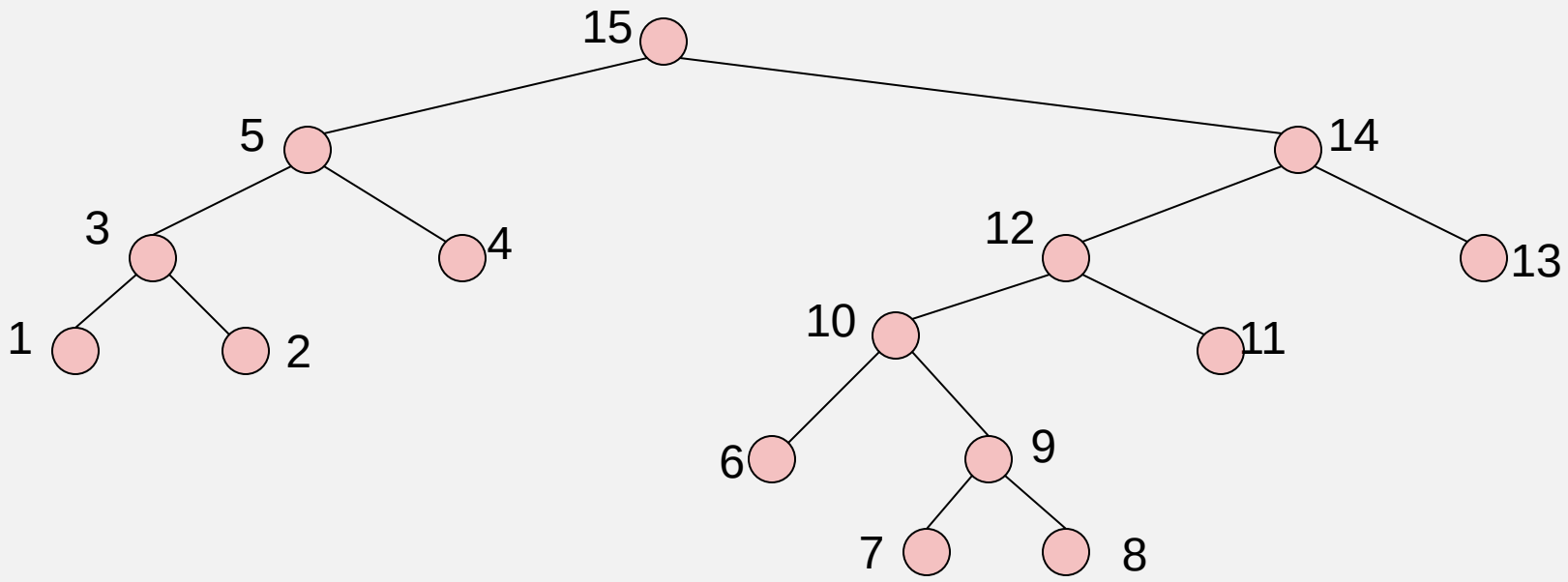
μεταδιάταξη (postorder)



Διάσχιση Δυαδικού Δένδρου

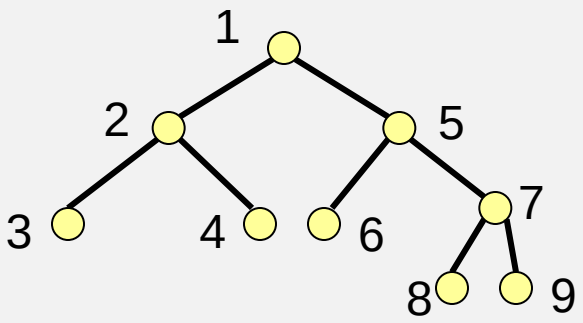
```
void traverse(Node h) {
        if (h == null) return;
        traverse(h.l);
        traverse(h.r);
        h.item.visit();
}
```

μεταδιάταξη (postorder)



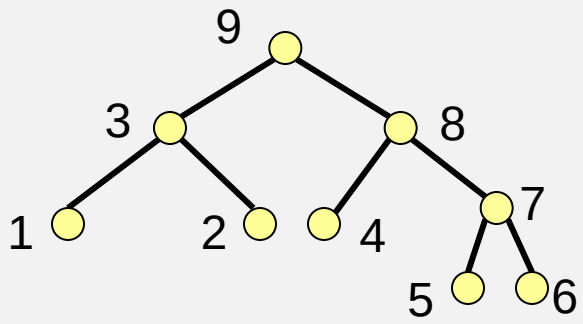
Διάσχιση Δυαδικού Δένδρου

προδιάταξη (preorder)



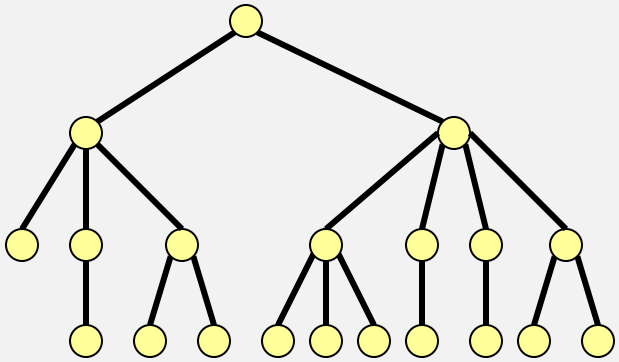
σειρά επεξεργασίας :
1. γονέας
2. αριστερό παιδί
3. δεξί παιδί

μεταδιάταξη (postorder)

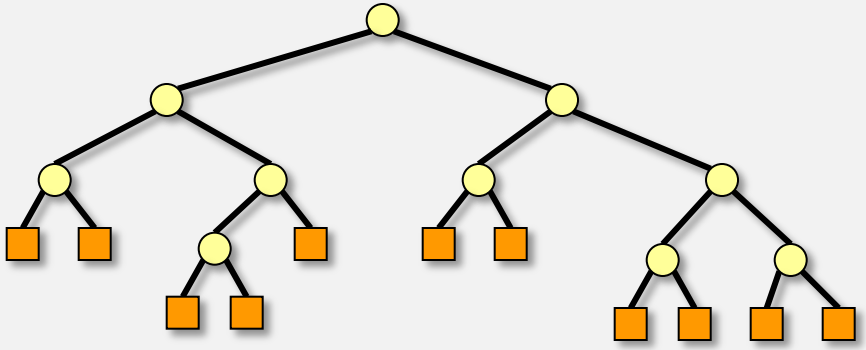


σειρά επεξεργασίας :
1. αριστερό παιδί
2. δεξί παιδί
3. γονέας

Η προδιάταξη και μεταδιάταξη μπορούν να εφαρμοστούν και σε δένδρα όπου κάθε εσωτερικός κόμβος έχει αυθαίρετο αριθμό παιδιών



Διάσχιση Δυαδικού Δένδρου



$N = 10$

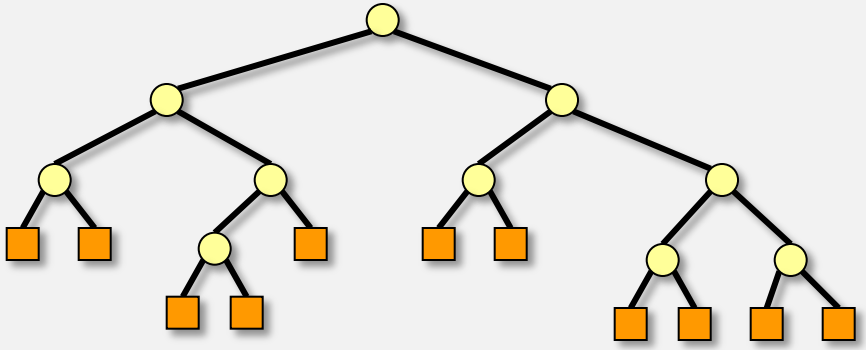
$$B = \left[\begin{array}{ccccccc} b_0 & b_1 & b_2 & \ldots & b_{19} & b_{20} \end{array} \right]$$

$$= \left[\begin{array}{ccccccccccccccccccccc} 1 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \end{array} \right]$$

Ένα δυαδικό δένδρο με N εσωτερικού κόμβους μπορεί να αναπαρασταθεί από μια ακολουθία B από $2N + 1$ δυαδικά ψηφία, που ικανοποιεί τις ακόλουθες συνθήκες:

- $N + 1$ ψηφιά είναι 0 και N ψηφία είναι 1
- Για κάθε θέση $i = 1, 2, \ldots, 2N + 1$, ο αριθμός των 1 που βρίσκονται πριν το b_i είναι μεγαλύτερος ή ίσος του αριθμού των 0 που βρίσκονται πριν το b_i

Διάσχιση Δυαδικού Δένδρου



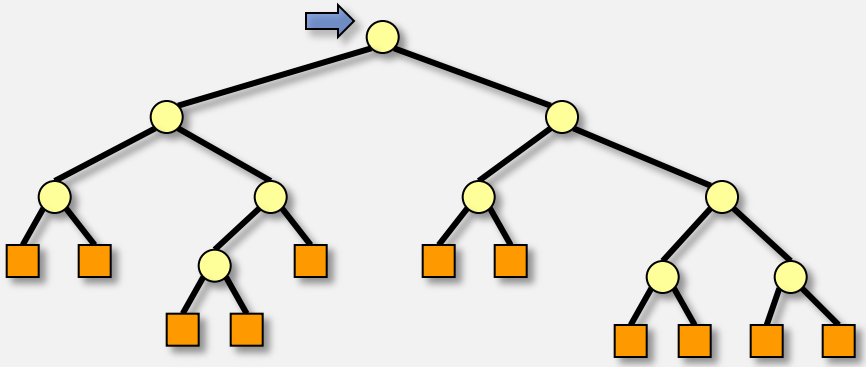
$N = 10$

$$B = \left[\begin{array}{ccccccc} b_0 & b_1 & b_2 & \ldots & b_{19} & b_{20} \end{array} \right]$$

$$= \left[\begin{array}{ccccccccccccccccccccc} 1 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \end{array} \right]$$

Εκτελούμε προδιατεταγμένη διάσχιση του δένδρου. Αν ο επόμενος κόμβος που συναντάμε είναι εσωτερικός τότε το επόμενο ψηφίο είναι 1, διαφορετικά, αν είναι εξωτερικός κόμβος, τότε το επόμενο ψηφίο είναι 0.

Διάσχιση Δυαδικού Δένδρου



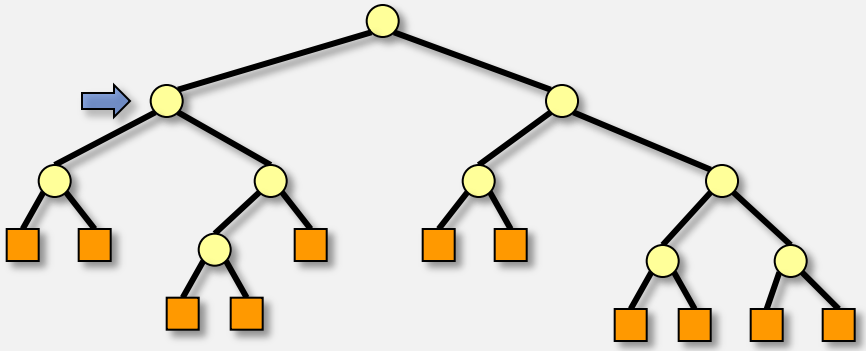
$N = 10$

$$B = \left[\begin{array}{ccccccccccccccccccccc} b_0 & b_1 & b_2 & \ldots & b_{19} & b_{20} \end{array} \right]$$

$$= \left[\begin{array}{ccccccccccccccccccccc} 1 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \end{array} \right]$$

Εκτελούμε προδιατεταγμένη διάσχιση του δένδρου. Αν ο επόμενος κόμβος που συναντάμε είναι εσωτερικός τότε το επόμενο ψηφίο είναι 1, διαφορετικά, αν είναι εξωτερικός κόμβος, τότε το επόμενο ψηφίο είναι 0.

Διάσχιση Δυαδικού Δένδρου



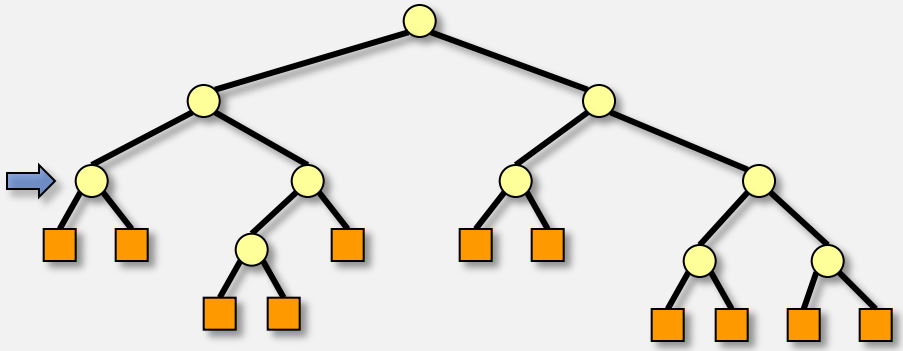
$N = 10$

$$B = \left[\begin{array}{cccccc} b_0 & b_1 & b_2 & \ldots & b_{19} & b_{20} \end{array} \right]$$

$$= \left[\begin{array}{cccccccccccccccccccccc} 1 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \end{array} \right]$$

Εκτελούμε προδιατεταγμένη διάσχιση του δένδρου. Αν ο επόμενος κόμβος που συναντάμε είναι εσωτερικός τότε το επόμενο ψηφίο είναι 1, διαφορετικά, αν είναι εξωτερικός κόμβος, τότε το επόμενο ψηφίο είναι 0.

Διάσχιση Δυαδικού Δένδρου



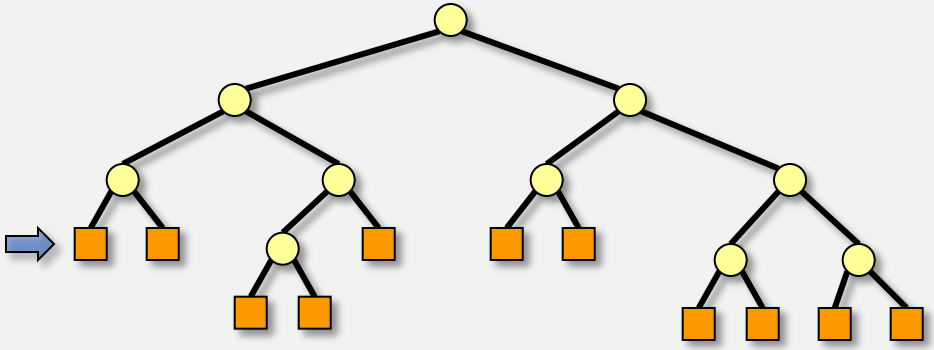
$N = 10$

$$B = \left[\begin{array}{ccccccc} b_0 & b_1 & b_2 & \ldots & b_{19} & b_{20} \end{array} \right]$$

$$= \left[\begin{array}{ccccccccccccccccccccc} 1 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \end{array} \right]$$

Εκτελούμε προδιατεταγμένη διάσχιση του δένδρου. Αν ο επόμενος κόμβος που συναντάμε είναι εσωτερικός τότε το επόμενο ψηφίο είναι 1, διαφορετικά, αν είναι εξωτερικός κόμβος, τότε το επόμενο ψηφίο είναι 0.

Διάσχιση Δυαδικού Δένδρου



$N = 10$

$$B = \left[\begin{array}{ccccccccccccccccccccc} b_0 & b_1 & b_2 & \ldots & b_{19} & b_{20} \end{array} \right]$$

$$= \left[\begin{array}{ccccccccccccccccccccc} 1 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \end{array} \right]$$

Εκτελούμε προδιατεταγμένη διάσχιση του δένδρου. Αν ο επόμενος κόμβος που συναντάμε είναι εσωτερικός τότε το επόμενο ψηφίο είναι 1, διαφορετικά, αν είναι εξωτερικός κόμβος, τότε το επόμενο ψηφίο είναι 0.

Διάσχιση Δυαδικού Δένδρου

Κατασκευή δυαδικού δένδρου από ακολουθία δυαδικών ψηφίων

Εξετάζουμε ένα-ένα τα ψηφία της ακολουθίας. Αν το επόμενο ψηφίο είναι 1 τότε δημιουργούμε νέο εσωτερικό κόμβο και κατασκευάζουμε αναδρομικά πρώτα το αριστερό και μετά το δεξί υποδένδρο. Διαφορετικά, αν το επόμενο ψηφίο είναι 0 τότε δημιουργούμε νέο εξωτερικό κόμβο.

$$B = [\ 1 \quad 1 \quad 1 \quad 0 \quad 0 \quad 1 \quad 1 \quad 0 \quad 0 \quad 0 \quad 1 \quad 1 \quad 0 \quad 0 \quad 1 \quad 1 \quad 0 \quad 0 \quad 1 \quad 0 \quad 0\ ]$$

Διάσχιση Δυαδικού Δένδρου

Κατασκευή δυαδικού δένδρου από ακολουθία δυαδικών ψηφίων

Εξετάζουμε ένα-ένα τα ψηφία της ακολουθίας. Αν το επόμενο ψηφίο είναι 1 τότε δημιουργούμε νέο εσωτερικό κόμβο και κατασκευάζουμε αναδρομικά πρώτα το αριστερό και μετά το δεξί υποδένδρο. Διαφορετικά, αν το επόμενο ψηφίο είναι 0 τότε δημιουργούμε νέο εξωτερικό κόμβο.

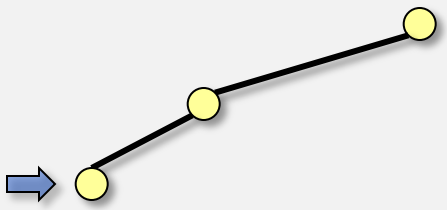
$$B = [\ 1 \quad 1 \quad 1 \quad 0 \quad 0 \quad 1 \quad 1 \quad 0 \quad 0 \quad 0 \quad 1 \quad 1 \quad 0 \quad 0 \quad 1 \quad 1 \quad 0 \quad 0 \quad 1 \quad 0 \quad 0\ ]$$

Διάσχιση Δυαδικού Δένδρου

Κατασκευή δυαδικού δένδρου από ακολουθία δυαδικών ψηφίων

Εξετάζουμε ένα-ένα τα ψηφία της ακολουθίας. Αν το επόμενο ψηφίο είναι 1 τότε δημιουργούμε νέο εσωτερικό κόμβο και κατασκευάζουμε αναδρομικά πρώτα το αριστερό και μετά το δεξί υποδένδρο. Διαφορετικά, αν το επόμενο ψηφίο είναι 0 τότε δημιουργούμε νέο εξωτερικό κόμβο.

$$B = [\ 1 \quad 1 \quad 1 \quad 0 \quad 0 \quad 1 \quad 1 \quad 0 \quad 0 \quad 0 \quad 1 \quad 1 \quad 0 \quad 0 \quad 1 \quad 1 \quad 0 \quad 0 \quad 1 \quad 0 \quad 0\ ]$$

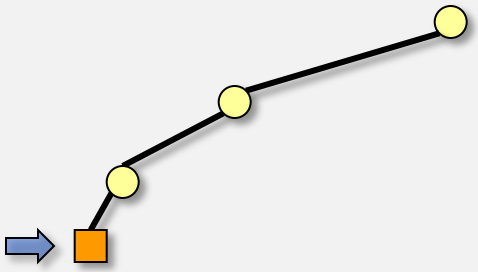


Διάσχιση Δυαδικού Δένδρου

Κατασκευή δυαδικού δένδρου από ακολουθία δυαδικών ψηφίων

Εξετάζουμε ένα-ένα τα ψηφία της ακολουθίας. Αν το επόμενο ψηφίο είναι 1 τότε δημιουργούμε νέο εσωτερικό κόμβο και κατασκευάζουμε αναδρομικά πρώτα το αριστερό και μετά το δεξί υποδένδρο. Διαφορετικά, αν το επόμενο ψηφίο είναι 0 τότε δημιουργούμε νέο εξωτερικό κόμβο.

$$B = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

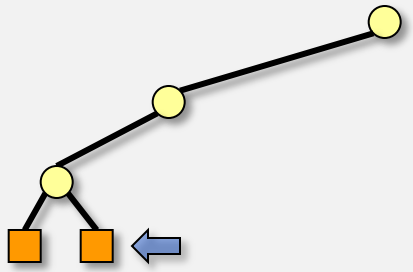


Διάσχιση Δυαδικού Δένδρου

Κατασκευή δυαδικού δένδρου από ακολουθία δυαδικών ψηφίων

Εξετάζουμε ένα-ένα τα ψηφία της ακολουθίας. Αν το επόμενο ψηφίο είναι 1 τότε δημιουργούμε νέο εσωτερικό κόμβο και κατασκευάζουμε αναδρομικά πρώτα το αριστερό και μετά το δεξί υποδένδρο. Διαφορετικά, αν το επόμενο ψηφίο είναι 0 τότε δημιουργούμε νέο εξωτερικό κόμβο.

$$B = [\ 1 \quad 1 \quad 1 \quad 0 \quad 0 \quad 1 \quad 1 \quad 0 \quad 0 \quad 0 \quad 1 \quad 1 \quad 0 \quad 0 \quad 1 \quad 1 \quad 0 \quad 0 \quad 1 \quad 0 \quad 0\ ]$$

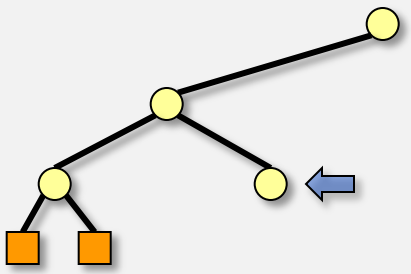


Διάσχιση Δυαδικού Δένδρου

Κατασκευή δυαδικού δένδρου από ακολουθία δυαδικών ψηφίων

Εξετάζουμε ένα-ένα τα ψηφία της ακολουθίας. Αν το επόμενο ψηφίο είναι 1 τότε δημιουργούμε νέο εσωτερικό κόμβο και κατασκευάζουμε αναδρομικά πρώτα το αριστερό και μετά το δεξί υποδένδρο. Διαφορετικά, αν το επόμενο ψηφίο είναι 0 τότε δημιουργούμε νέο εξωτερικό κόμβο.

$$B = [\ 1 \quad 1 \quad 1 \quad 0 \quad 0 \quad 1 \quad 1 \quad 0 \quad 0 \quad 0 \quad 1 \quad 1 \quad 0 \quad 0 \quad 1 \quad 1 \quad 0 \quad 0 \quad 1 \quad 0 \quad 0\ ]$$

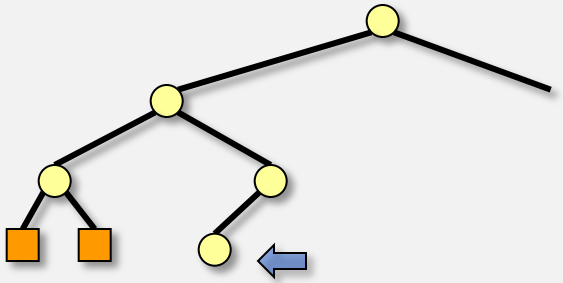


Διάσχιση Δυαδικού Δένδρου

Κατασκευή δυαδικού δένδρου από ακολουθία δυαδικών ψηφίων

Εξετάζουμε ένα-ένα τα ψηφία της ακολουθίας. Αν το επόμενο ψηφίο είναι 1 τότε δημιουργούμε νέο εσωτερικό κόμβο και κατασκευάζουμε αναδρομικά πρώτα το αριστερό και μετά το δεξί υποδένδρο. Διαφορετικά, αν το επόμενο ψηφίο είναι 0 τότε δημιουργούμε νέο εξωτερικό κόμβο.

$$B = [\ 1 \quad 1 \quad 1 \quad 0 \quad 0 \quad 1 \quad 1 \quad 0 \quad 0 \quad 0 \quad 1 \quad 1 \quad 0 \quad 0 \quad 1 \quad 1 \quad 0 \quad 0 \quad 1 \quad 0 \quad 0\ ]$$

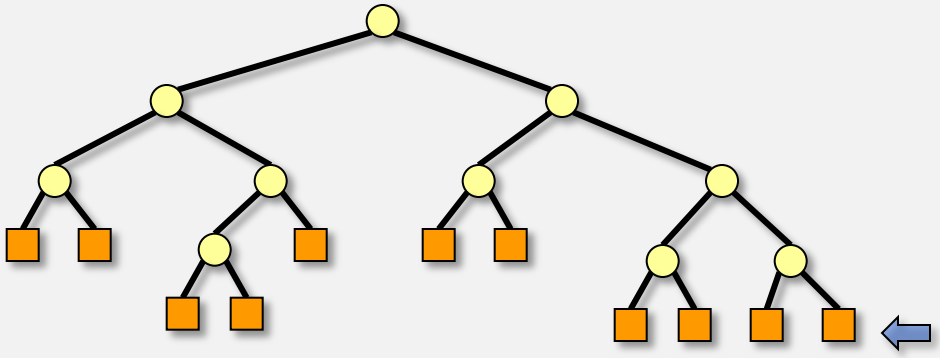


Διάσχιση Δυαδικού Δένδρου

Κατασκευή δυαδικού δένδρου από ακολουθία δυαδικών ψηφίων

Εξετάζουμε ένα-ένα τα ψηφία της ακολουθίας. Αν το επόμενο ψηφίο είναι 1 τότε δημιουργούμε νέο εσωτερικό κόμβο και κατασκευάζουμε αναδρομικά πρώτα το αριστερό και μετά το δεξί υποδένδρο. Διαφορετικά, αν το επόμενο ψηφίο είναι 0 τότε δημιουργούμε νέο εξωτερικό κόμβο.

$$B = [\ 1 \quad 1 \quad 1 \quad 0 \quad 0 \quad 1 \quad 1 \quad 0 \quad 0 \quad 0 \quad 1 \quad 1 \quad 0 \quad 0 \quad 1 \quad 1 \quad 0 \quad 0 \quad 1 \quad 0 \quad 0\ ]$$



Διάσχιση Δυαδικού Δένδρου

Κατασκευή δυαδικού δένδρου από ακολουθία δυαδικών ψηφίων

Εξετάζουμε ένα-ένα τα ψηφία της ακολουθίας. Αν το επόμενο ψηφίο είναι 1 τότε δημιουργούμε νέο εσωτερικό κόμβο και κατασκευάζουμε αναδρομικά πρώτα το αριστερό και μετά το δεξί υποδένδρο. Διαφορετικά, αν το επόμενο ψηφίο είναι 0 τότε δημιουργούμε νέο εξωτερικό κόμβο.

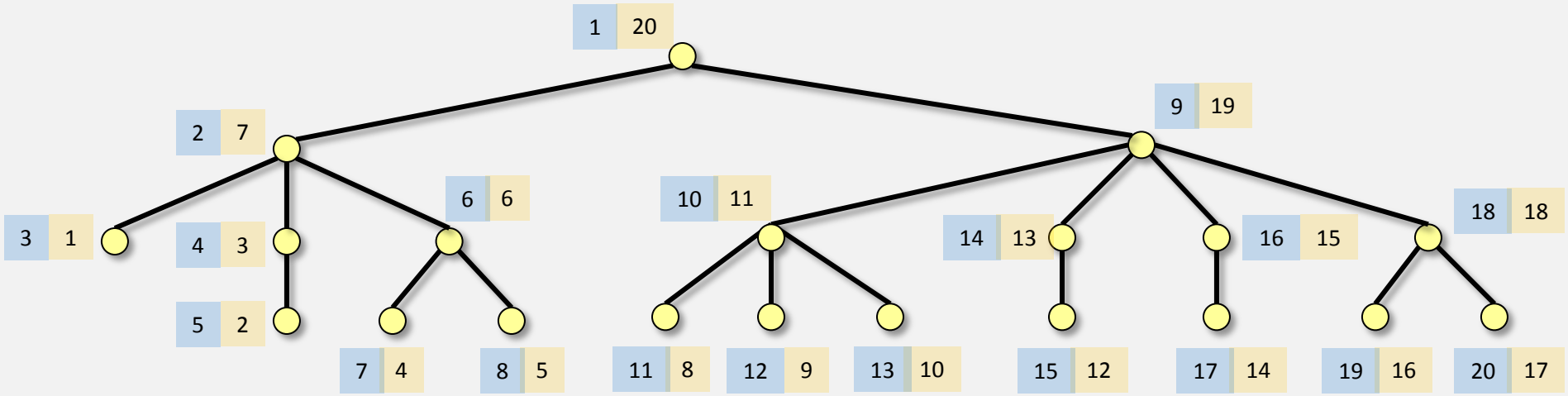
Κατασκευή με αναδρομή

```
class Node {
        Item item; Node l; Node r;
        Node(Item v, Node l, Node r) {
                this.item = v;
                this.l = l;
                this.r = r;
        }
}
```

```
int k; // τρέχουσα θέση
Node createBT(char[] B)
{
    if (k == B.length) return null;
    if (B[k++] == '0') return null;

    Item v = …
    Node x = new Node(v,null,null);
    x.l = createBT(B);
    x.r = createBT(B);
    return x;
}
```

Διάσχιση Δυαδικού Δένδρου

Η προδιάταξη και μεταδιάταξη μπορούν να εφαρμοστούν και σε δένδρα όπου κάθε εσωτερικός κόμβος έχει αυθαίρετο αριθμό παιδιών

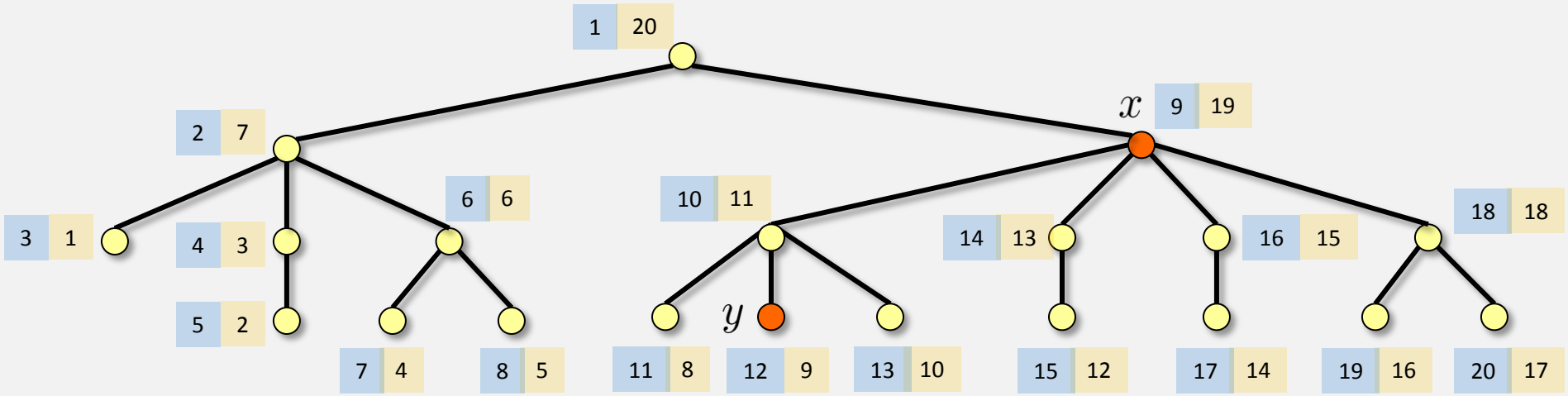


Προδιάταξη : Πρώτα ο γονέας, μετά τα παιδιά σε σειρά από αριστερά προς τα δεξιά

Μεταδιάταξη : Πρώτα τα παιδιά σε σειρά από αριστερά προς τα δεξιά, μετά ο γονέας

Διάσχιση Δυαδικού Δένδρου

Η προδιάταξη και μεταδιάταξη μπορούν να εφαρμοστούν και σε δένδρα όπου κάθε εσωτερικός κόμβος έχει αυθαίρετο αριθμό παιδιών



Προδιάταξη : Πρώτα ο γονέας, μετά τα παιδιά σε σειρά από αριστερά προς τα δεξιά

Μεταδιάταξη : Πρώτα τα παιδιά σε σειρά από αριστερά προς τα δεξιά, μετά ο γονέας

Ένας κόμβος y είναι απόγονος ενός κόμβου x αν και μόνο αν

$x \leq y$ στη σειρά προδιάταξης και $x \geq y$ στη σειρά μεταδιάταξης