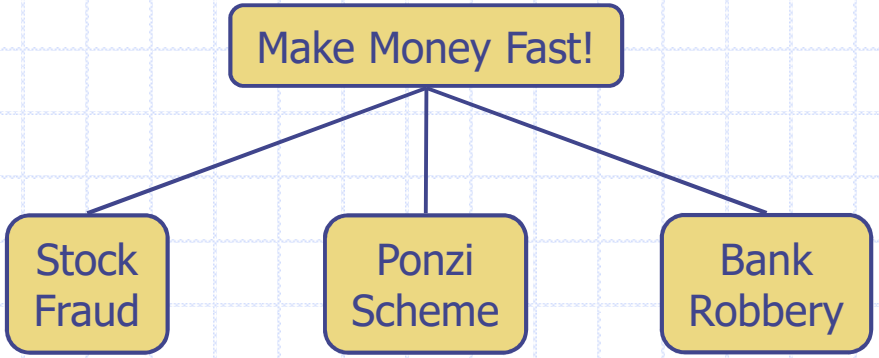
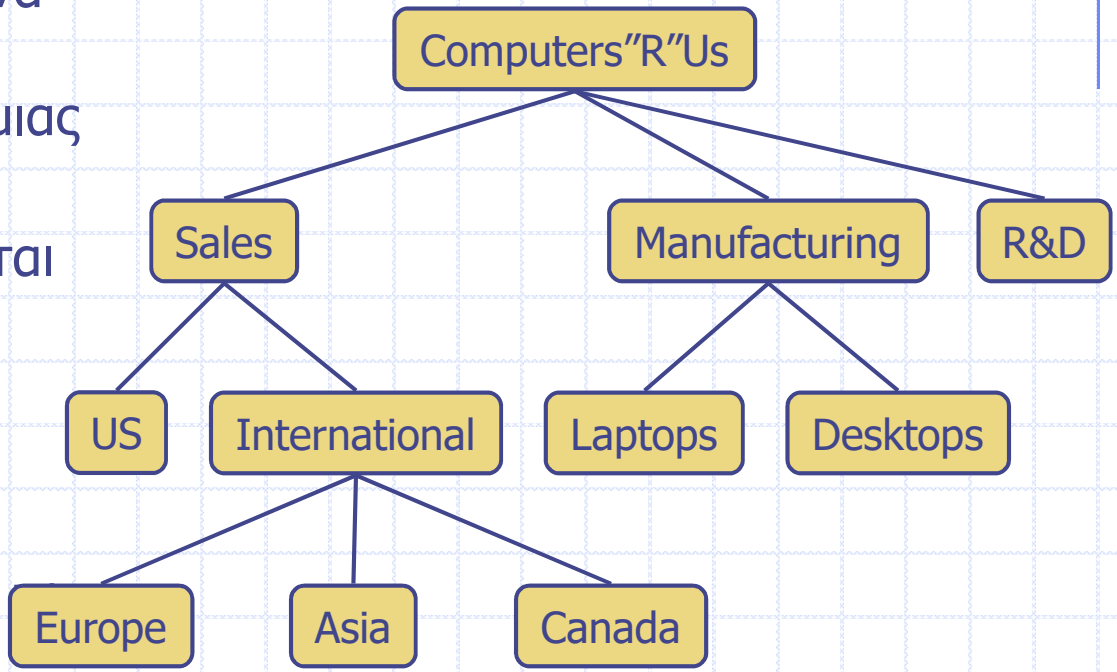
# Δένδρα



# Τι είναι δένδρο

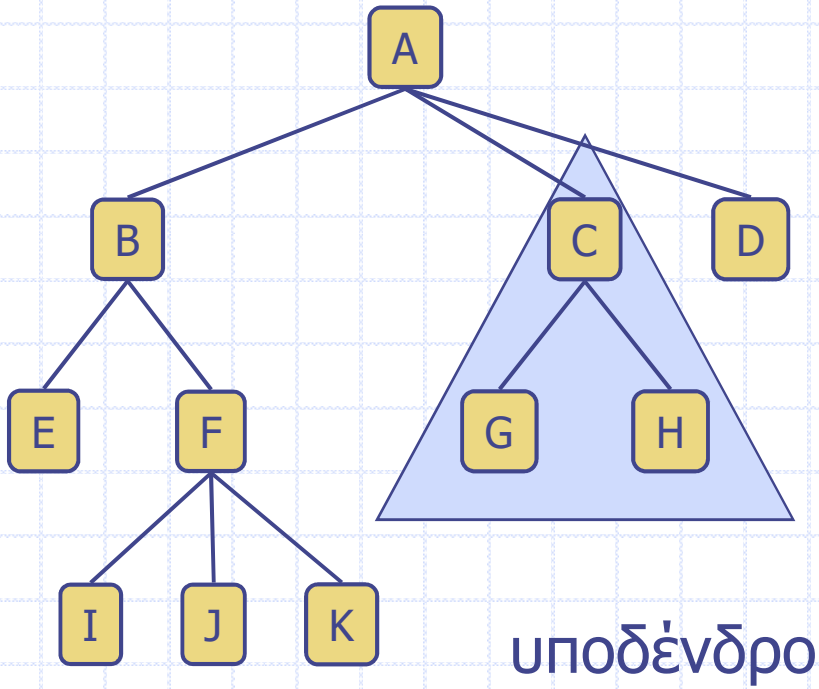
- Στην πληροφορική, ένα δένδρο είναι ένα αφηρημένο μοντέλο μιας ιεραρχικής δομής
- Ένα δένδρο αποτελείται από κόμβους με μια σχέση γονέα-παιδιού
- Εφαρμογές:
  - Οργανογράμματα
  - Συστήματα Αρχείων
  - Προγραμματιστικά περιβάλλοντα



# Ορολογία

- Ρίζα: κόμβος χωρίς γονέα (A)
- Εσωτερικός κόμβος: κόμβος με τουλάχιστον ένα παιδί (A, B, C, F)
- Εξωτερικός κόμβος (φύλο): κόμβος χωρίς παιδιά (E, I, J, K, G, H, D)
- Πρόγονοι ενός κόμβου: γονέας, παππούς, προπάππος, κλπ.
- Βάθος ενός κόμβου: το πλήθος των προγόνων
- Ύψος ενός δένδρου: μέγιστο βάθος οποιουδήποτε κόμβου(3)
- Απόγονος ενός κόμβου: παιδί, εγγόνι, δισέγγονο, κοκ.

- Υποδένδρο: δένδρο που αποτελείται από ένα κόμβο και τους απογόνους του




υποδένδρο

# Πλευρές και Διαδρομές σε ένα Δένδρο

- Μια ***πλευρά*** ενός δένδρου T είναι ένα ζεύγος κόμβων (u,v) τέτοιο ώστε το u είναι γονέας του v, ή αντίστροφα
- Μια ***διαδρομή (μονοπάτι)*** στο T είναι μια ακολουθία από κόμβους τέτοια ώστε κάθε δύο διαδοχικοί κόμβοι της ακολουθίας σχηματίζουν πλευρά.

# Διατεταγμένα Δένδρα

Ένα δένδρο είναι ***διατεταγμένο*** αν υπάρχει μια γραμμική διάταξη που ορίζεται για κάθε κόμβο: δηλαδή, μπορούμε να χαρακτηρίσουμε τα παιδιά ενός κόμβου σαν πρώτο, δεύτερο, τρίτο, κοκ. Μια τέτοια διάταξη συνήθως οπτικοποιείται παραθέτοντας τα αδέλφια από αριστερά προς τα δεξιά, σύμφωνα με την διάταξή τους. Τα διατεταγμένα δένδρα τυπικά δείχνουν τη γραμμική σειρά μεταξύ των αδελφών παραθέτοντας τα με τη σωστή σειρά.

# ΑΤΔ Δένδρο

- Χρησιμοποιούμε θέσεις στην αφαίρεση κόμβων
- Πρωτογενείς μέθοδοι:
  - integer size()
  - boolean isEmpty()
  - Iterator iterator()
  - Iterable positions()
- Μέθοδοι προσπέλασης:
  - position root()
  - position parent(p)
  - Iterable children(p)
- Μέθοδοι ερώτησης:
  - boolean isInternal(p)
  - boolean isExternal(p)
  - boolean isRoot(p)
- Μέθοδος ενημέρωσης :
  - element replace (p, o)
- Μπορεί να ορισθούν επιπλέον μέθοδοι ενημέρωσης από δομές δεδομένων που υλοποιούν ο ΑΤΔ δένδρο

# Βάθος Δένδρων

Έστω v ένας κόμβος ενός δένδρου T. Το ***βάθος*** του v είναι το πλήθος των προγόνων του v, χωρίς τον v.

Μπορεί να ορισθεί και αναδρομικά:

Αν v είναι η ρίζα, τότε το βάθος του v είναι 0.

Διαφορετικά, το βάθος του v ένα συν το βάθος του γονέα του v.

# Ύψος Δένδρου

Το ***ύψος*** ενός κόμβου v σε ένα δένδρο T ορίζεται αναδρομικά:

Αν το v είναι ένας εξωτερικός κόμβος, τότε το ύψος του v είναι 0.

Διαφορετικά, το ύψος του v είναι ένα συν το μέγιστο ύψος ενός παιδιού του v.

Το ***ύψος*** ενός μη κενού δένδρου T είναι το ύψος της ρίζας του T.

# Υπολογισμός Βάθους

```
Algorithm depth(T,v)
 if v is the root of T then
     return 0
else return 1+depth(T,w) όπου
     w ο γονέας του v
```

- Ο χρόνος τρεξίματος του αλγόριθμου depth(T,v) είναι $O(d_v)$, όπου $d_v$ δηλώνει το βάθος του κόμβου v στο δένδρο T, επειδή ο αλγόριθμος εκτελεί ένα σταθερού χρόνου αναδρομικό βήμα για κάθε πρόγονο του v. Επομένως ο αλγόριθμος depth(T,v) τρέχει χρόνο O(n) στη χειρότερη περίπτωση, όπου n είναι το συνολικό πλήθος των κόμβων του T, αφού ένας κόμβος του T μπορεί να έχει στη χειρότερη περίπτωση βάθος n-1.

# Υπολογισμός ύψους

```
Algorithm height1(T):
h←0
For each vertex v in T do
If v is an external node in T
    then h←max(h, depth(T,v))
Return h

Algorithm height2(T,v)
If v is an external node in T
    then return 0
else h ←0
for each child w of v in T do
 h←max(h, height2(T,w))
 return 1+h
```

- Αφού ο height1 καλεί τον αλγόριθμο depth(v) σε κάθε εξωτερικό κόμβο v του T, ο χρόνος τρεξίματος του height1 δίνεται από το $O(n+\Sigma_v(1+ d_v))$, όπου n είναι το πλήθος των κόμβων του T, $d_v$ είναι το βάθος του κόμβου v, και E είναι το σύνολο των εξωτερικών κόμβων του T.
- Ο αλγόριθμος height2 απαιτεί χρόνο $O(1+ c_v)$ σε κάθε κόμβο v, και ο χρόνος τρεξίματός της είναι $O(\Sigma_v(1+ c_v))$. ($c_v$ το πλήθος των παιδιών του κόμβου v)

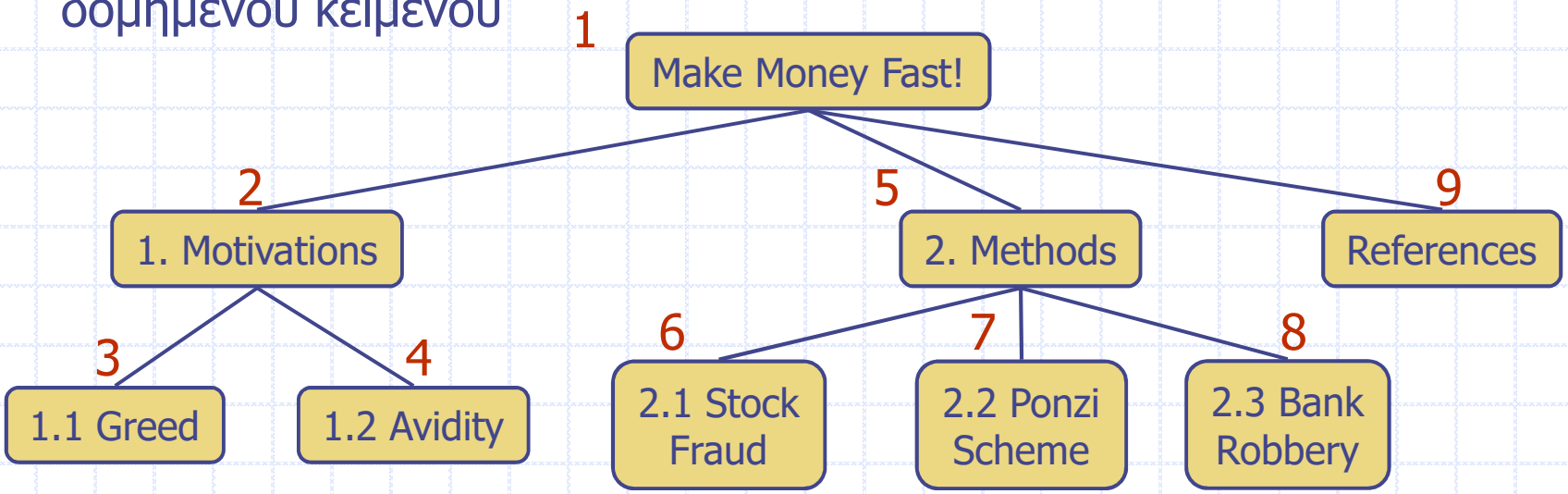
**Πρόταση**: *Έστω Τ ένα δένδρο με n κόμβους, και έστω $c_v$ το πλήθος των παιδιών του κόμβου v του Τ. Τότε αθροίζοντας στους κόμβους του Τ, $\Sigma_v c_v = n-1$.*

Επομένως ένα δένδρο με n κόμβους έχει n-1 πλευρές.

# Προδιατεταγμένη Σάρωση

- Μια σάρωση επισκέπτεται τους κόμβους ενός δένδρου κατά συστηματικό τρόπο
- Σε μια προδιατεταγμένη σάρωση, η επίσκεψη σε ένα κόμβο προηγείται αυτής των απογόνων του
- Εφαρμογή: εκτύπωση ενός δομημένου κειμένου

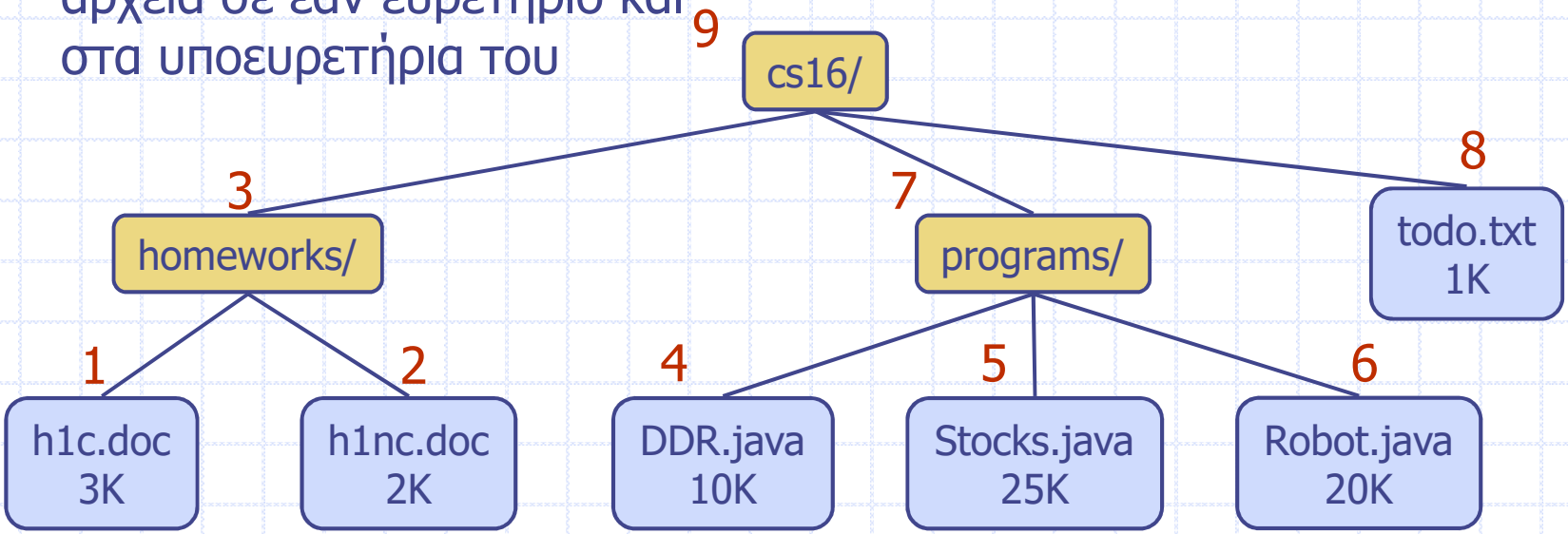
**Algorithm** *preOrder*(*v*)
  *visit*(*v*)
  **for each** child *w* of *v*
    *preorder* (*w*)



# Μεταδιατεταγμένη Σάρωση

- Στη μεταδιατεταγμένη σάρωση η επίσκεψη σε ένα κόμβο γίνεται αφού επισκεφθούμε τους απογόνους του
- Εφαρμογή: υπολογισμός του χώρου που καταλαμβάνουν αρχεία σε εάν ευρετήριο και στα υποευρετήρια του

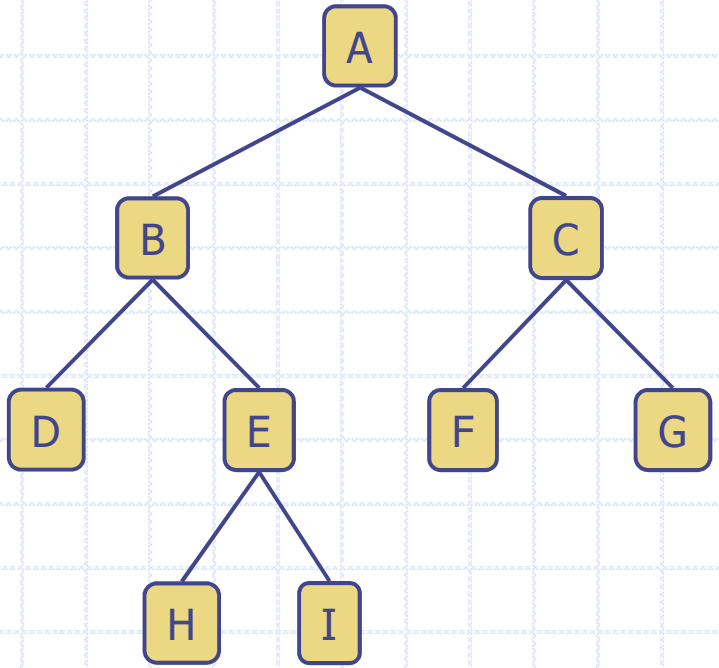
**Algorithm** *postOrder*(*v*)
  **for each** child *w* of *v*
    *postOrder* (*w*)
  *visit*(*v*)



# Δυαδικά δένδρα

- Δυαδικό είναι ένα δένδρο με τις παρακάτω ιδιότητες:
  - Κάθε εσωτερικός κόμβος έχει το πολύ δύο παιδιά (ακριβώς δύο για κανονικά δυαδικά δένδρα)
  - Τα παιδιά ενός κόμβου είναι ένα διατεταγμένο ζεύγος
- Ονομάζουμε τα παιδιά ενός εσωτερικού κόμβου αριστερό παιδί και δεξιό παιδί
- Εναλλακτικός αναδρομικός ορισμός: ένα δυαδικό δένδρο είναι είτε
  - ένα δένδρο που αποτελείται από ένα κόμβο, ή
  - ένα δένδρο που η ρίζα του έχει ένα διατεταγμένο ζεύγος παιδιών, που το καθένα είναι ένα δυαδικό δένδρο

- Εφαρμογές:
  - αριθμητικές εκφράσεις
  - διαδικασίες αποφάσεων
  - αναζήτηση

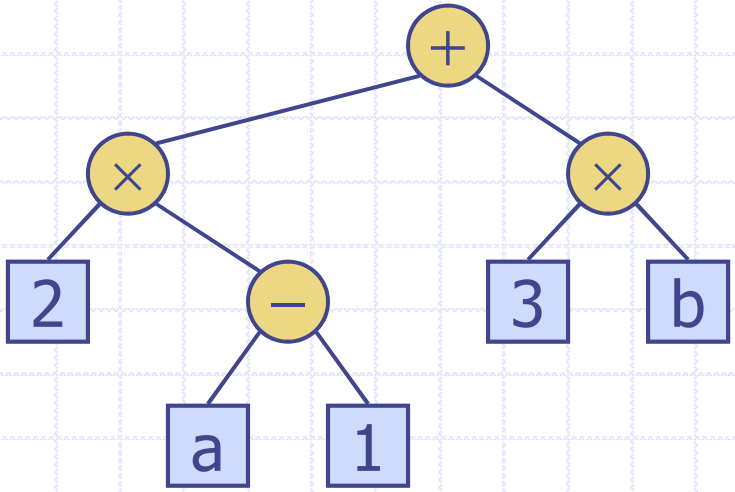


# Κανονικό ή πλήρες δυαδικό δένδρο

- Ένα δυαδικό δένδρο λέγεται κανονικό (στη βιβλιογραφία αναφέρεται και σαν πλήρες) αν κάθε κόμβος έχει μηδέν ή δύο παιδιά. Δηλαδή κάθε εσωτερικός κόμβος έχει δυο παιδιά.

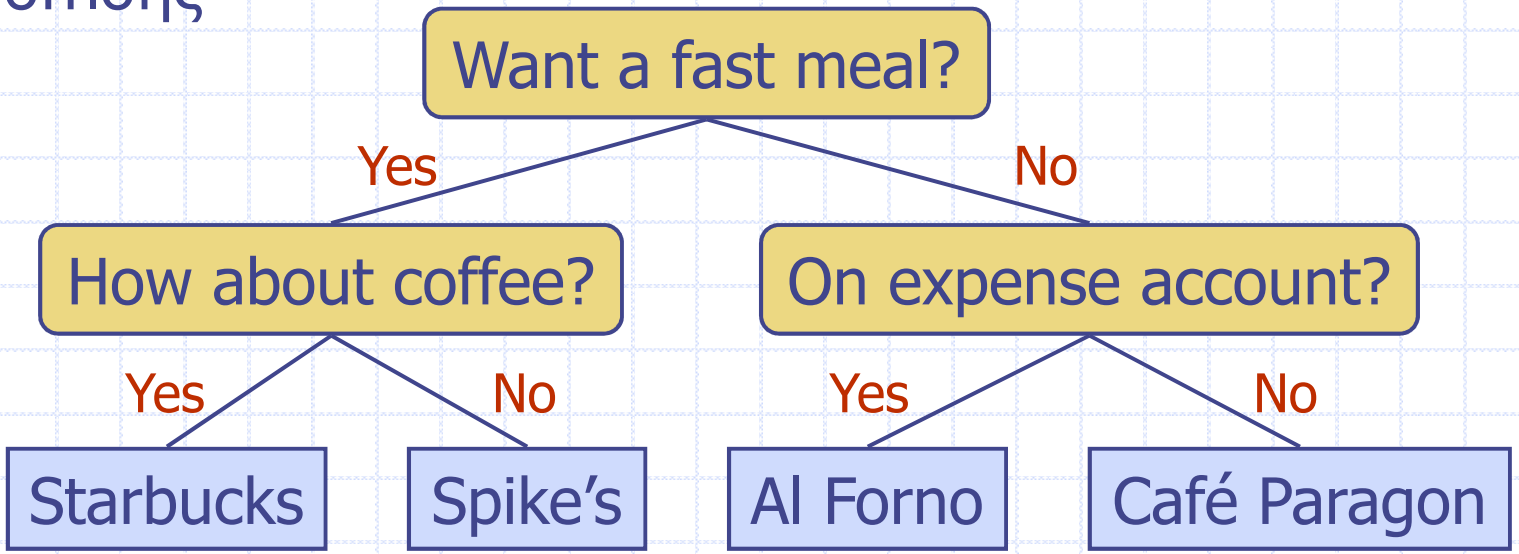
# Δένδρο αριθμητικής έκφρασης

- Δυαδικό δένδρο που σχετίζεται με μια αριθμητική έκφραση
  - εσωτερικοί κόμβοι: τελεστές
  - εξωτερικοί κόμβοι: μεταβλητές
- Παράδειγμα: αριθμητικά έκφραση δένδρου για την παράσταση $(2 \times (a - 1) + (3 \times b))$



# Δένδρα αποφάσεων

- Δυαδικό δένδρο που σχετίζεται με διαδικασία αποφάσεων
  - εσωτερικοί κόμβοι: ερωτήσεις με απάντηση ναι/όχι
- Παράδειγμα: απόφαση επιλογής καταστήματος σίτισης



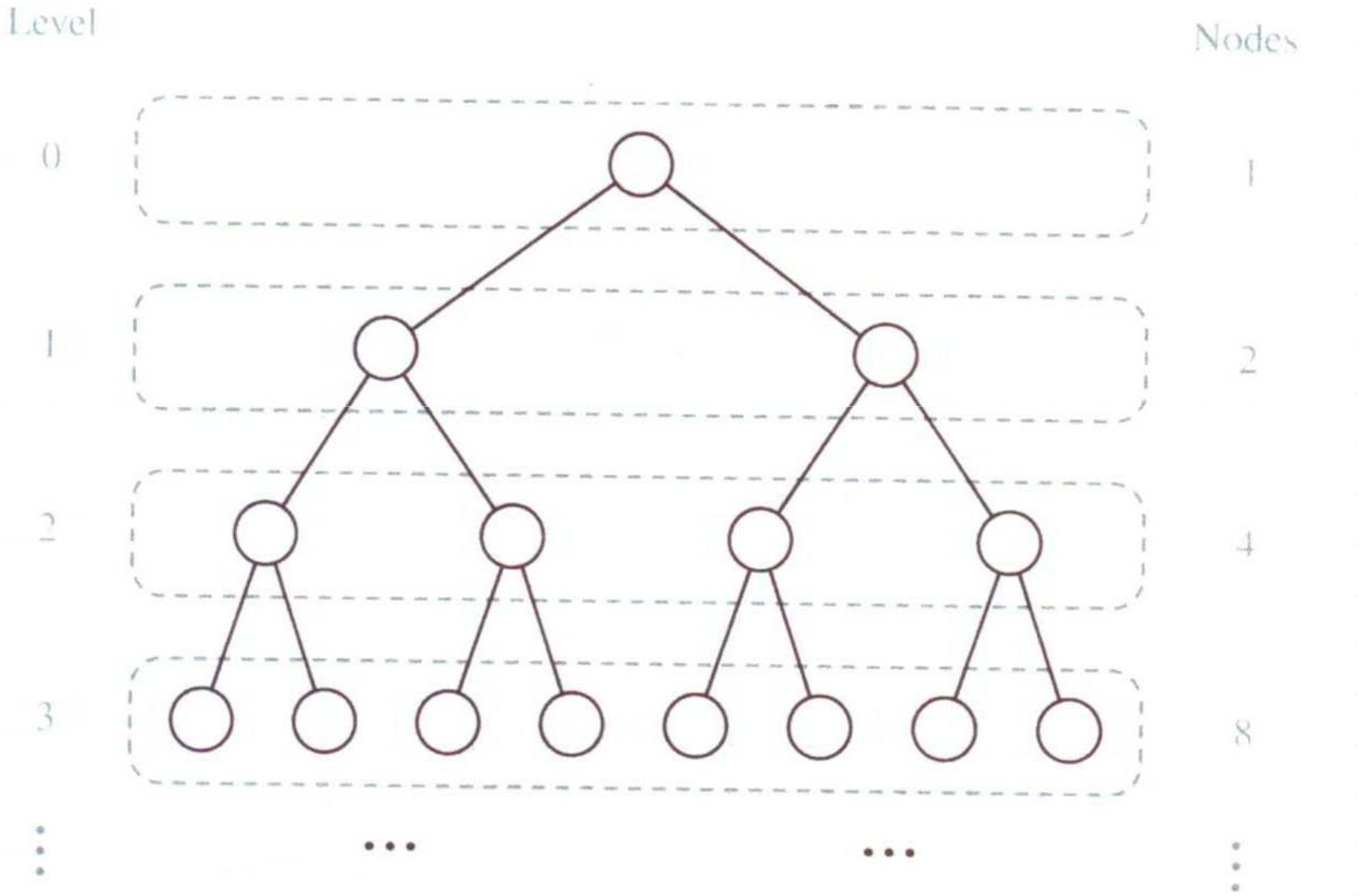
# Αναδρομικός Ορισμός των Δυαδικών Δένδρων

- Ένα δυαδικό δένδρο είναι είτε κενό ή αποτελείται από:
  - Ένα κόμβο r, που ονομάζεται ρίζα του T και αποθηκεύει ένα στοιχείο
  - Ένα δυαδικό δένδρο, που ονομάζεται το αριστερό υποδένδρο του T
  - Ένα δυαδικό δένδρο, που ονομάζεται το δεξιό υποδένδρο του T

# Ο ΑΤΔ Δυαδικό δένδρο

- left(v): επιστρέφει το αριστερό παιδί του v· συνθήκη λάθους αν ο v δεν έχει αριστερό παιδί
- right(v): επιστρέφει το δεξιό παιδί του v· συνθήκη λάθους αν ο v δεν έχει δεξιό παιδί
- hasLeft(v): έλεγχος αν ο v έχει αριστερό παιδί
- hasRight(v): έλεγχος αν ο v έχει δεξιό παιδί

# Ιδιότητες δυαδικων δένδρων



# Ιδιότητες των Δυαδικών Δένδρων

- Συμβολισμοί
  - n πλήθος κόμβων
  - e πλήθος εξωτερικών κόμβων
  - i πλήθος εσωτερικών κόμβων
  - h ύψος του δένδρου

- Ιδιότητες
  - $h+1 \leq n \leq 2^{h+1}-1$
  - $1 \leq e \leq 2^h$
  - $h \leq i \leq 2^h-1$
  - $\log(n+1)-1 \leq h \leq n-1$

- Ιδιότητες κανονικού
  - $2h+1 \leq n \leq 2^{h+1}-1$
  - $h+1 \leq e \leq 2^h$
  - $h \leq i \leq 2^h-1$
  - $\log(n+1)-1 \leq h \leq (n-1)/2$

**Πρόταση**

Για ένα μη κενό δυαδικό δέντρο Τ, αν $n_0$ είναι το πλήθος των τερματικών (εξωτερικών) κόμβων και $n_2$ είναι το πλήθος των εσωτερικών κόμβων βαθμού 2, τότε $n_0 = n_2 + 1$.

**Απόδειξη**

$$n = n_0 + n_1 + n_2 \quad (1)$$

Επίσης $n = B + 1$ όπου B είναι το πλήθος των ακμών.

Αλλά επίσης

$$B = n_1 + 2n_2$$

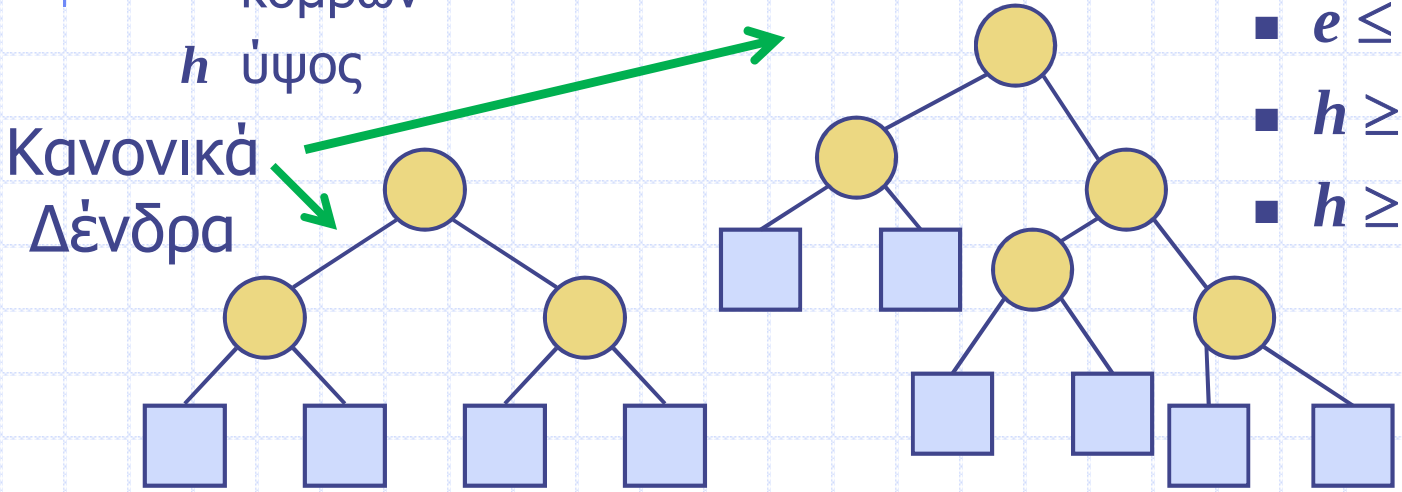
άρα $n = 1 + n_1 + 2n_2$ (2)

Από τις (1) και (2) έχουμε :

$$n_0 = n_2 + 1.$$

# Ιδιότητες των κανονικών δυαδικών δένδρων

- Συμβολισμός
  - $n$ πλήθος κόμβων
  - $e$ πλήθος εξωτερικών κόμβων
  - $i$ πλήθος εσωτερικών κόμβων
  - $h$ ύψος




- Ιδιότητες:
  - $e = i + 1$
  - $n = 2e - 1$
  - $h \leq i$
  - $h \leq (n - 1)/2$
  - $e \leq 2^h$
  - $h \geq \log_2 e$
  - $h \geq \log_2 (n + 1) - 1$

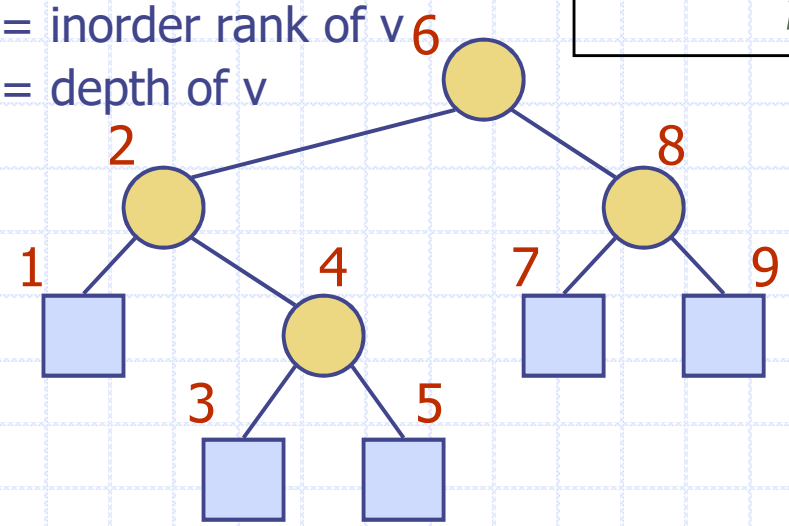
# ΑΤΔ Δυαδικό Δένδρο

- Ο ΑΤΔ BinaryTree επεκτείνει τον ΑΤΔ Tree ADT, δηλ., κληρονομεί όλες τις μεθόδους του ΑΤΔ Tree
- Επιπλέον Μέθοδοι:
  - position left(p)
  - position right(p)
  - boolean hasLeft(p)
  - boolean hasRight(p)
- Μπορούν να ορισθούν μέθοδοι τροποποίησης από τις δομές δεδομένων που υλοποιούν τον ΑΤΔ BinaryTree

# Ενδοδιατεταγμένη Σάρωση

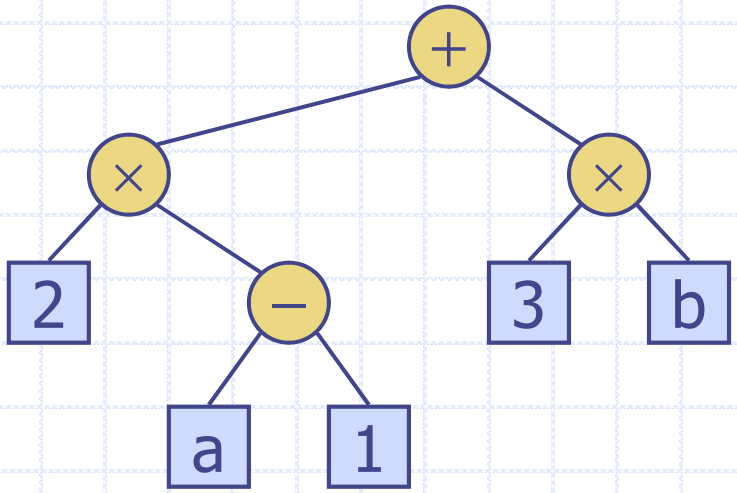
- Στην ενδοδιατεταγμένη σάρωση η επίσκεψη σε ένα κόμβο μετά την επίσκεψη στο αριστερό υποδένδρο του και πριν από την επίσκεψη στο δεξιό υποδένδρο του
- Εφαρμογή: σχεδιασμός ενός δυαδικού δένδρου
  - x(v) = inorder rank of v
  - y(v) = depth of v

**Algorithm** ***inOrder(v)***
  **if** ***hasLeft*** **(*v*)**
    ***inOrder*** **(*left*** **(*v*))**
  ***visit(v)***
  **if** ***hasRight*** **(*v*)**
    ***inOrder*** **(*right*** **(*v*))**



# Εκτύπωση Αριθμητικών Εκφράσεων

- Εξειδίκευση μιας ενδοδιατεταγμένης σάρωσης
  - Τύπωσε μεταβλητή ή τελεστή όταν επισκέπτεσε έναν κόμβο
  - print "(" before traversing left subtree
  - print ")" after traversing right subtree

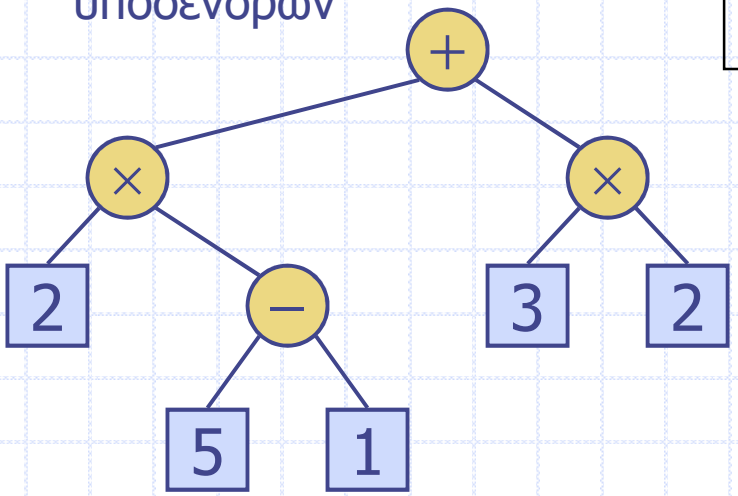



**Algorithm** *printExpression*(*v*)
  **if** *hasLeft* (*v*)
    *print*("(")
    *inOrder* (*left*(*v*))
  *print*(*v.element* ())
  **if** *hasRight* (*v*)
    *inOrder* (*right*(*v*))
    *print* (")")

$((2 \times (a - 1)) + (3 \times b))$

# Υπολογισμός Αριθμητικών εκφράσεων

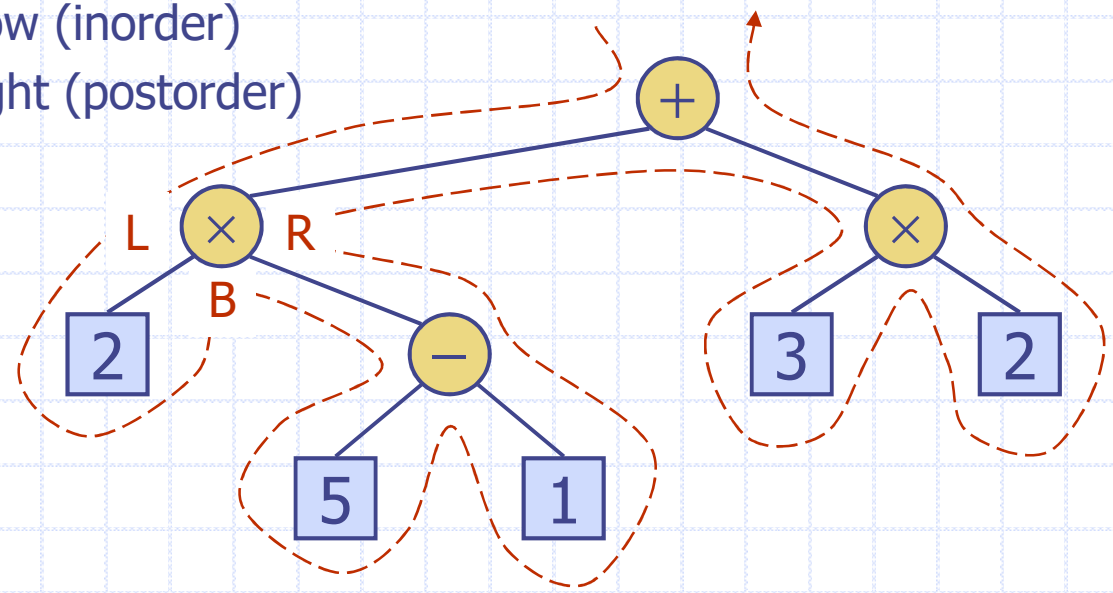
- Εξειδίκευση μιας μεταδιατεταγμένης σάρωσης
  - αναδρομική μέθοδος που επιστρέφει την τιμή ενός υποδένδρου
  - Ωστην επίσκεψη σε ένα εσωτερικο κόμβο, συνδυάζονται οι τιμές των υποδένδρων




**Algorithm** *evalExpr*(*v*)
  **if** *isExternal* (*v*)
    **return** *v.element* ()
  **else**
    $x \leftarrow$ *evalExpr*(*leftChild* (*v*))
    $y \leftarrow$ *evalExpr*(*rightChild* (*v*))
    $\lozenge \leftarrow$ operator stored at *v*
    **return** $x \lozenge y$

# Σάρωση Euler

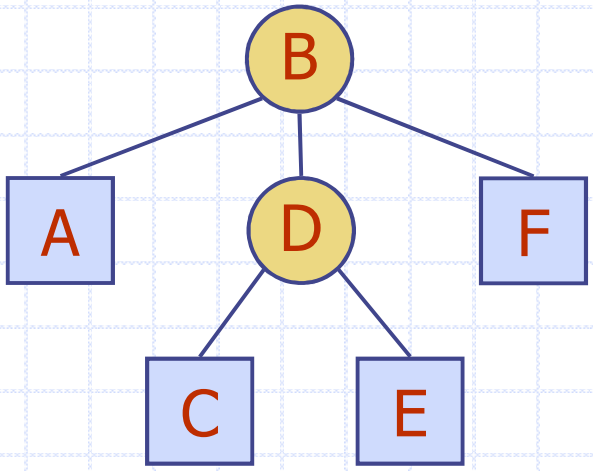
- Πρωτογενής σάρωση ενός δυαδικού δένδρου
- Περιλαμβάνει ειδικές περιπτώσεις τις σαρώσεις προδιατεταγμένη, μεταδιατεταγμένη και ενδοδιατεταγμένη
- Διάσχυση γύρω του δένδρου και επίσκεψη κάθε κόμβου τρεις φορές:
  - on the left (preorder)
  - from below (inorder)
  - on the right (postorder)

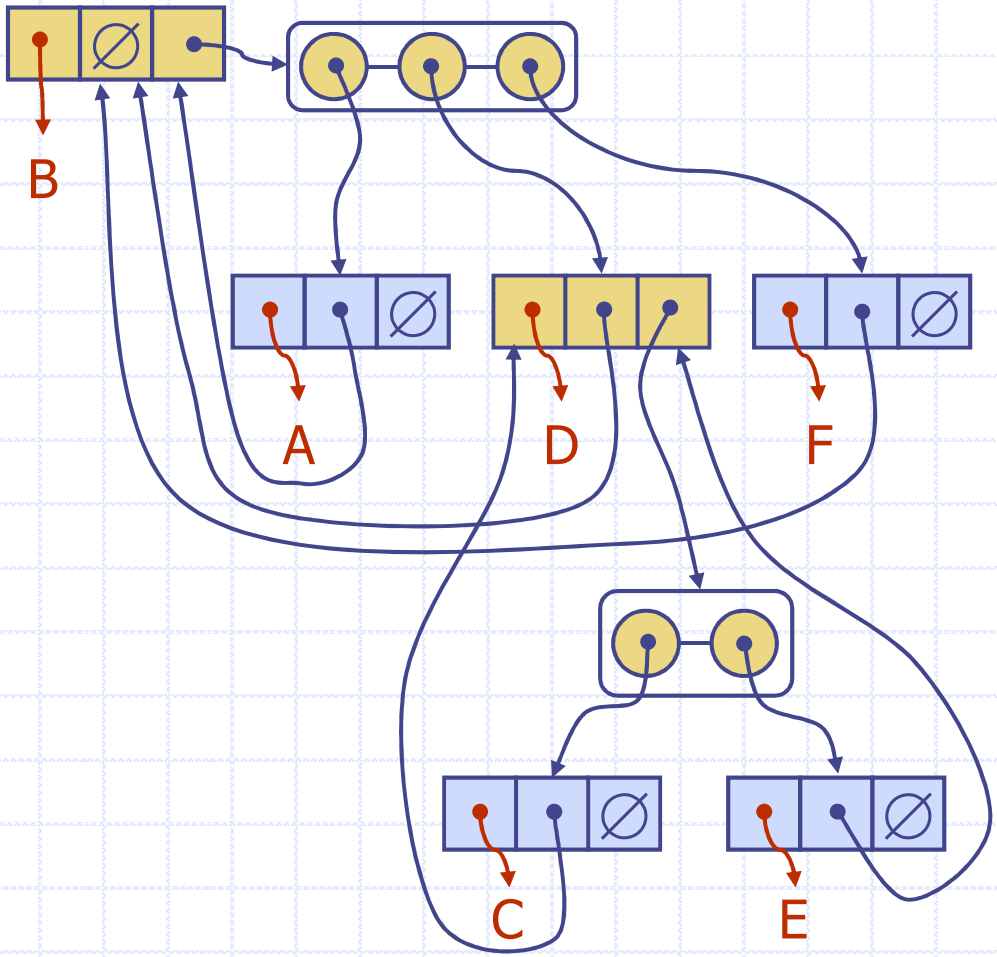


Άτυπα η σάρωση Euler ενός δένδρου T μπορεί να ορισθεί σαν ένας "περίπατος" γύρω από το T, όπου ξεκινάμε από τη ρίζα πηγαίνοντας στο αριστερό παιδί, θεωρώντας τις πλευρές σαν "τοίχους" που είναι πάντα στα αριστερά μας.

# Συνδεδεμένη δομή για δένδρα

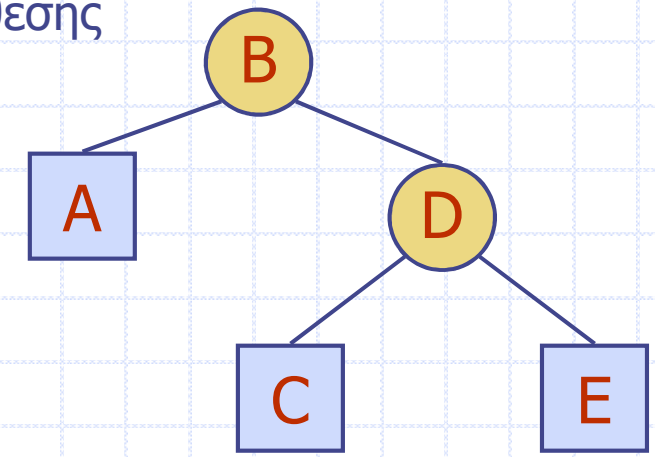
- Ένας κόμβος παριστάνεται από ένα αντικείμενο που αποθηκεύει
  - Δεδομένα
  - Τον κόμβο γονέα
  - Ακολουθία από κόμβους παιδιά
- Τα αντικείμενα κόμβοι υλοποιούν τον ΑΤΔ θέσης

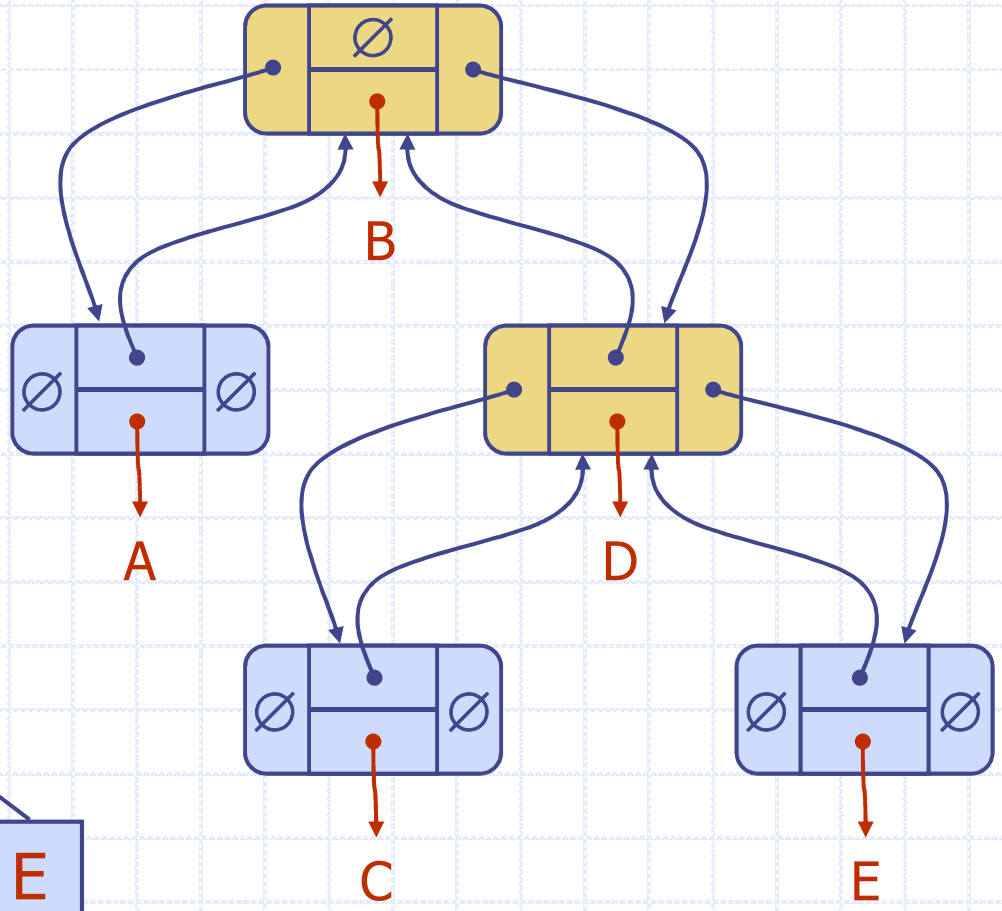


# Συνδεδεμένη δομή για δυαδικά δένδρα

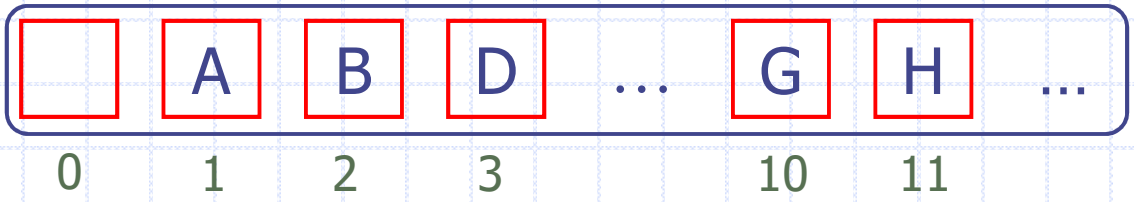
- Ένας κόμβος παριστάνεται από ένα αντικείμενο που αποθηκεύει
  - Δεδομένα
  - Κόμβο γονέα
  - Κόμβο αριστερό παιδί
  - Κόμβο δεξιό παιδί
- Τα αντικείμενα κόμβοι υλοποιούν τον ΑΤΔ θέσης

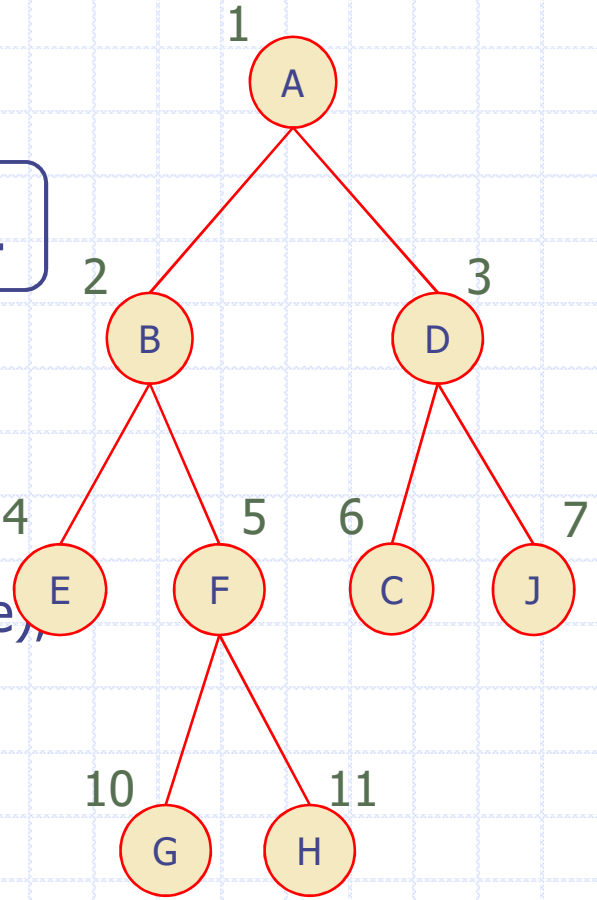


# Αναπαράσταση Δυαδικών δένδρων που βασίζεται σε πίνακες

- Οι κόμβοι αποθηκεύονται σε ένα πίνακα A




- Ο κόμβος v αποθηκεύεται στο A[rank(v)]
  - rank(root) = 1
  - αν node είναι το αριστερό παιδί του parent(node), rank(node) = 2 · rank(parent(node))
  - αν node είναι το δεξί παιδί του parent(node), rank(node) = 2 · rank(parent(node)) + 1

# Μέθοδος Προτύπου Περιγράμματος

- Γενικός αλγόριθμος
- Υλοποιείται από την αφηρημένη Java κλάση
- Οι μέθοδοι Visit ορίζονται ξανά από τις υποκλάσεις
- Μέθοδος Περιγράμματος eulerTour
  - Καλείται αναδρομικά στα αριστερά και δεξιά παιδιά
  - Ένα αντικείμενο TourResult με πεδία f left, right και out καταγράφει την έξοδο των αναδρομικών κλήσεων της eulerTour

```
public abstract class EulerTour <E, R> {
    protected BinaryTree<E> tree;
    public abstact R execute(BinaryTree<E> T);
    protected void init(BinaryTree<E> T) { tree = T; }
    protected R eulerTour(Position<E> v) {
        TourResult<R> r = new TourResult<R>();
        visitLeft(v, r);
        if (tree.hasLeft(p))
            { r.left=eulerTour(tree.left(v)); }
        visitBelow(v, r);
        if (tree.hasRight(p))
            { r.right=eulerTour(tree.right(v)); }
        return r.out;
    }
    protected void visitLeft(Position<E> v, TourResult<R> r) {}
    protected void visitBelow(Position<E> v, TourResult<R> r) {}
    protected void visitRight(Position<E> v, TourResult<R> r) {}
}
```

# Εξειδικεύσεις της περιοδείας Euler

- Εξειδίκευση της κλάσης EulerTour για υπολογισμό αριθμητικών εκφράσεων
- Υποθέσεις
  - Οι κόμβοι αποθηκεύουν αντικείμενα ExpressionTerm με την μέθοδο getValue
  - ExpressionVariable αντικείμενα σαν εξωτερικοί κόμβοι
  - ExpressionOperator αντικείμενα στους εσωτερικούς κόμβους με την μέθοδο setOperands(Integer, Integer)

```
public class EvaluateExpressionTour
      extends EulerTour<ExpressionTerm, Integer> {
   public Integer execute
         (BinaryTree<ExpressionTerm> T) {
      init(T);
      return eulerTour(tree.root());
   }
   protected void visitRight
         (Position<ExpressionTerm> v,
          TourResult<Integer> r) {
      ExpressionTerm term = v.element();
      if (tree.isInternal(v)) {
         ExpressionOperator op = (ExpressionOperator) term;
         op.setOperands(r.left, r.right); }
      r.out = term.getValue();
   }
}
```

- Δείξτε ότι σε ένα κανονικό δυαδικό δένδρο αν $n_E$ είναι το πλήθος των εξωτερικών κόμβων και $n_I$ το πλήθος των εσωτερικών κόμβων τότε ισχύει:

$$n_E = n_I + 1$$

Απαντήστε τις παρακάτω ερωτήσεις.

- Ποιό είναι το ελάχιστο πλήθος εξωτερικών κόμβων ενός κανονικού δυαδικού δένδρου με ύψος h; Αιτιολογήστε την απάντησή σας.
- Ποιό είναι το μέγιστο πλήθος εξωτερικών κόμβων ενός κανονικού δυαδικού δένδρου με ύψος h; Αιτιολογήστε την απάντησή σας.
- Έστω T ένα κανονικό δυαδικό δένδρο ύψους h με n κόμβους. Δείξτε ότι

$$\log(n+1) - 1 \leq h \leq (n-1)/2$$

- Για ποιές τιμές των n και h μπορούμε να πιάσουμε ισότητα για τα παραπάνω άνω και κάτω όρια του h;

- Έστω Τ ένα (πιθανόν μη κανονικό) δυαδικό δένδρο με n κόμβους, και έστω D το άθροισμα του βάθους όλων των εξωτερικών κόμβων του Τ. Περιγράψτε ένα σχήμα του Τ ώστε το D να είναι $\Omega(n^2)$. Ένα τέτοιο δένδρο θα ήταν η χειρότερη περίπτωση για τον ασυμπτωτικό χρόνο τρεξίματος του Αλγορίθμου height1

- Έστω Τ ένα διατεταγμένο δένδρο με περισσότερους από έναν κόμβους. Είναι δυνατόν η προδιατεταγμένη διάσχιση του Τ να επισκέπτεται τους κόμβους με την ίδια σειρά με την μεταδιατεταγμένη διάσχιση του Τ; Αν ναι, δώστε ένα παράδειγμα, διαφορετικά αιτιολογήστε γιατί δεν μπορεί να συμβεί αυτό. Παρομοίως, είναι δυνατόν η προδιατεταγμένη διάσχιση του Τ να επισκέπτεται τους κόμβους με την αντίστροφη σειρά με την μεταδιατεταγμένη διάσχιση του Τ; Αν ναι, δώστε ένα παράδειγμα, διαφορετικά αιτιολογήστε γιατί δεν μπορεί να συμβεί αυτό.

- Απαντήστε την προηγούμενη ερώτηση για την περίπτωση ενός κανονικού δυαδικού δένδρου με περισσότερους από έναν κόμβους.

Έστω Τ ένα δένδρο με n κόμβους. Ορίζουμε τον **ελάχιστο κοινό πρόγονο** (LCA) μεταξύ δύο κόμβων v και w τον ελάχιστο κόμβο στο Τ που έχει απογόνους και τον v και τον w (επιτρέποντας σε ένα κόμβο να είναι απόγονος του εαυτού του). Όταν δίδονται δύο κόμβοι v και w, περιγράψτε έναν αποδοτικό αλγόριθμο για την εύρεση του LCA των v και w. Ποιος ο χρόνος τρεξίματος του αλγορίθμου σας;

Algorithm LCA (Node v, Node w)

int $v_{depth}$; int $w_{depth}$;

while $v_{depth} > w_{depth}$ do

v=v.parent;

while $w_{depth} > v_{depth}$ do

w=w.parent;

while v≠w do

v=v.parent;

w=w.parent;

return v;