Δομές Δεδομένων

Μέρος 5ο: Δένδρα

Εξάμηνο Σπουδών: 6ο

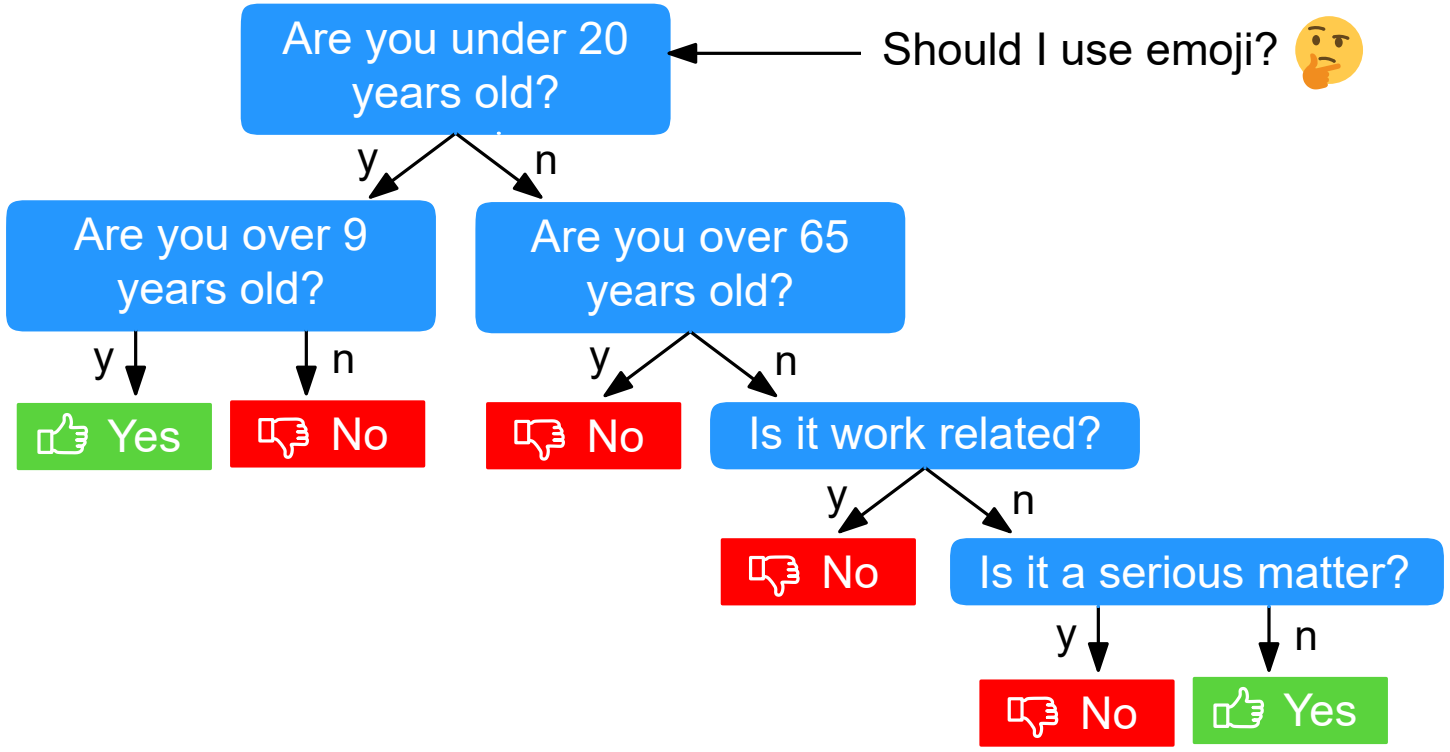
Κωδικός Μαθήματος: 681

Τμήμα Μαθηματικών
Πανεπιστήμιο Ιωαννίνων

Μιχάλης Α. Μπέκος
bekos@uoi.gr

Δένδρα
Trees

- Στην επιστήμη των υπολογιστών, ένα δένδρο είναι ένα αφηρημένο μοντέλο μιας ιεραρχικής δομής
- Ένα δένδρο αποτελείται από κόμβους με σχέση γονέα-παιδιού



- Εφαρμογές:
  - Δένδρα αποφάσεων
    Decision trees
  - Οργανογράμματα
    Organization charts
  - Γενεαλογικά δένδρα
    Genealogy trees
  - Συστήματα αρχείων
    File systems

Δένδρα: Ορολογία

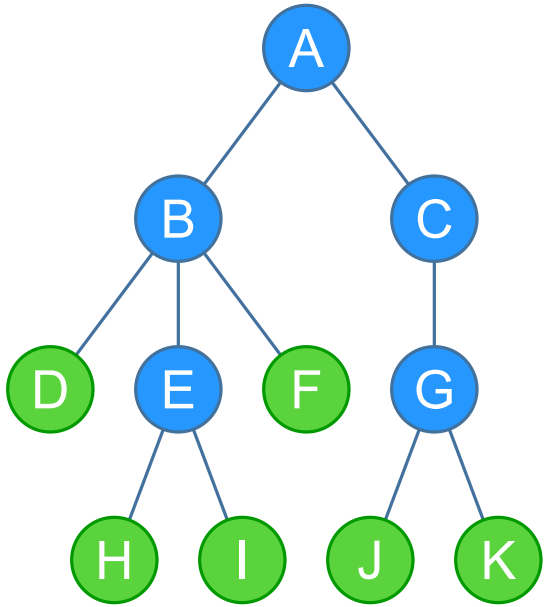
Trees terminology

- Ρίζα: Ο κόμβος του δένδρου χωρίς γονέα.
  root
  βλ. κόμβο A στο παράδειγμα

- Εσωτερικός κόμβος: Κάθε κόμβος με τουλάχιστον ένα παιδί.
  internal node
  βλ. κόμβους A, B, C, E και G στο παράδειγμα

- Εξωτερικός κόμβος ή φύλλο: Κάθε κόμβος χωρίς παιδιά.
  External node or leaf
  βλ. κόμβους D, F, H, I, και K στο παράδειγμα

- Πρόγονος κόμβου: Ο γονέας του κόμβου ή ένας πρόγονος του γονέα.
  Ancestors of a node
  π.χ. ο κόμβος B είναι πρόγονος του H στο παράδειγμα

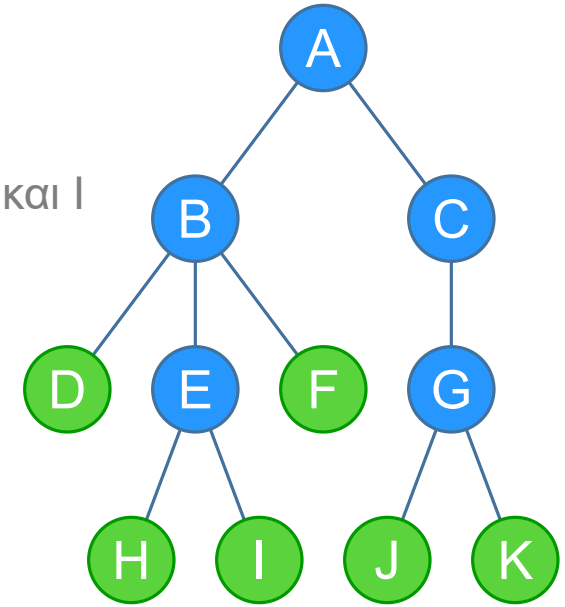


Δένδρα: Ορολογία

Trees terminology

- **Απόγονος κόμβου:** Ένα παιδί του κόμβου ή ένας απόγονος κάποιου παιδιού.
  Descendant of a node
  π.χ. ο κόμβος H είναι πρόγονος του B στο παράδειγμα

- **Υποδένδρο:** Αποτελείται από ένα κόμβο και τους απογόνους του.
  Subtree
  π.χ. το υποδένδρο του κόμβου E αποτελείται από τους κόμβους E, H και I



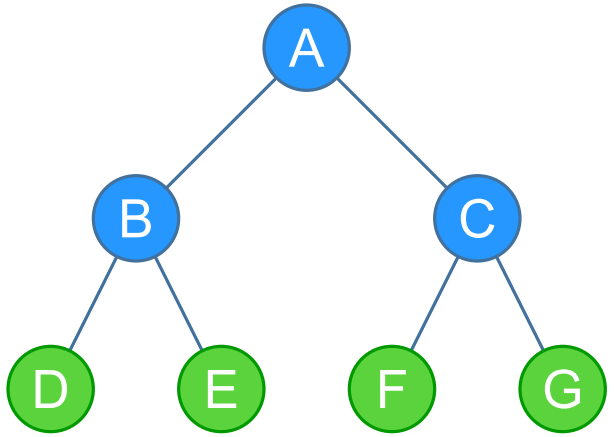
- **Βάθος κόμβου:** Ισούται με το πλήθος των προγόνων του κόμβου.
  Depth of a node
  π.χ. το βάθος του κόμβου H είναι 3 στο παράδειγμα

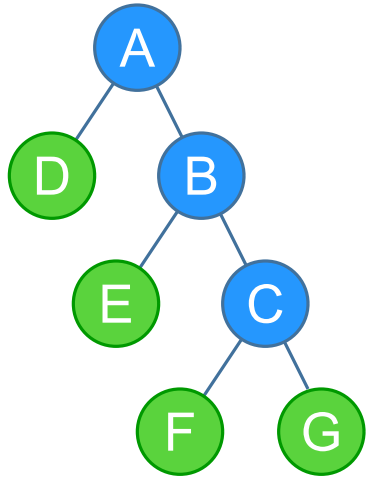
- **Ύψος δένδρου:** Ισούται με το μέγιστο βάθος ενός κόμβου στο δένδρο.
  Depth of a node
  το βάθος του δένδρου είναι 3 στο παράδειγμα

Δυαδικά Δένδρα

Binary trees

- Κάθε εσωτερικός κόμβος έχει το πολύ δύο παιδιά





- Τα παιδιά κάθε κόμβου είναι διατεταγμένα:
  - Το πρώτο αναφέρεται ως αριστερό
  - Το δεύτερο αναφέρεται ως δεξιό

- Σημειογραφία:
  - n ← πλήθος κόμβων
  - i ← πλήθος εσωτερικών κόμβων
  - e ← πλήθος εξωτερικών κόμβων
  - m ← πλήθος ακμών
  - h ← ύψος του δένδρου

- Ιδιότητες κανονικών δυαδικών δένδρων:
  (κάθε κόμβος έχει 0 ή 2 παιδιά)
  - $m = n - 1$
  - $e = i + 1$
  - $n = 2e - 1$
  - $h \leq i$
  - $h \leq \frac{n-1}{2}$
  - $e \leq 2^h$
  - $h \geq \log e$
  - $h \geq \log(n+1) - 1$

ΑΤΔ: Δένδρο

ADT: Tree

- Ο ΑΤΔ στοιχείο μοντελοποιεί μια θέση σε ένα δένδρο.

`Entry parent()`	επιστρέφει τον γονέα του στοιχείου	access
`list<Entry> children()`	επιστρέφει μια λίστα με τα παιδιά του στοιχείου	access
`bool isRoot()`	επιστρέφει `true`, αν το στοιχείο είναι η ρίζα	queries
`bool isLeaf()`	επιστρέφει `true`, αν το στοιχείο είναι φύλλο	queries

- Ο ΑΤΔ δένδρο μοντελοποιεί μια ιεραρχική δομή υποστηρίζοντας τις ακόλουθες λειτουργίες.

`empty()`	ελέγχει αν η λίστα είναι κενή	capacity
`size()`	επιστρέφει το πλήθος των στοιχείων της λίστας	capacity
`Entry root()`	επιστρέφει το στοιχείο της ρίζας του δένδρου	access
`list<Entry> entries()`	επιστρέφει μια λίστα με τα στοιχεία του δένδρου	access

Βασικοί Υπολογισμοί

- Αλγόριθμος για τον υπολογισμό του βάθος ενός κόμβου:

  Είσοδος: Ένα δένδρο T και ένας κόμβος v.
  Εξοδος : Το βάθος του κόμβου v στο T.

```
1 Algorithm depth(T, v)
2     if v.isRoot() then
3         return 0
4     else
5         return 1 + depth(T, v.parent());
```

  - Πολυπλοκότητα: Γραμμική ως προς το βάθος του κόμβου

Βασικοί Υπολογισμοί

- Αλγόριθμος για τον υπολογισμό του ύψους του δένδρου:

  Είσοδος: Ένα δένδρο T και ένας κόμβος v.
  Εξοδος : Το ύψος του υποδένδρου του T με ρίζα v.

```
1 Algorithm height(T, v)
2     if v.isLeaf() then return 0;
3     else
4         h ← 0;
5         foreach child c in v.children() do
6             h = max{h, height(T,c)}
7         return 1 + h;
```

  - Αρχική κλήση: `height(T, T.root())`
  - Πολυπλοκότητα: $\mathcal{O}(n)$

Διελεύσεις Δένδρων

Tree traversals

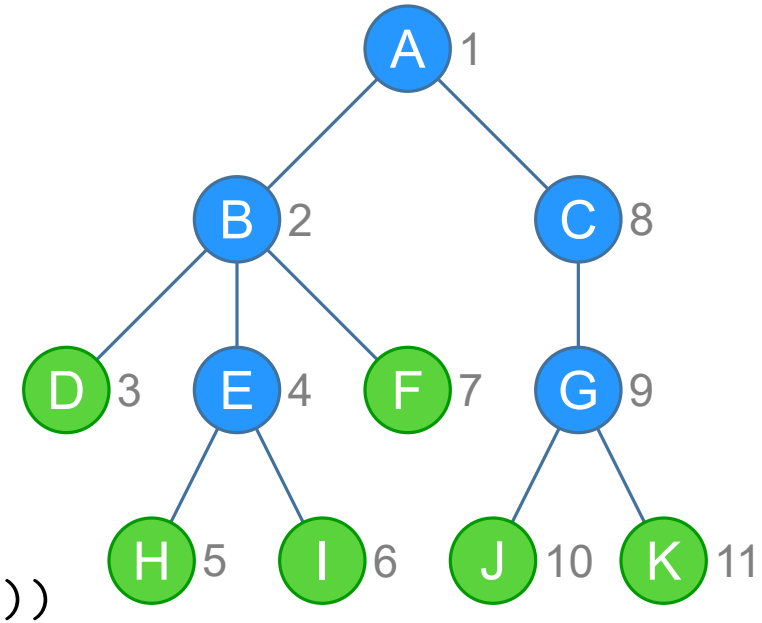
- Διέλευση δένδρου: Μια διαδικασία επίσκεψης/επεξεργασίας όλων των κόμβων ενός δένδρου με συστηματικό τρόπο
  - Προδιάταξη
    Preorder traversal
  - Μεταδιάταξη
    Postorder traversal
  - Εσωδιάταξη
    Inorder traversal
  - Διάταξη κατά επίπεδα
    Level-order traversal

Προδιάταξη Δένδρου

Preorder tree traversal

- Προδιάταξη δένδρου: Κάθε κόμβος προηγείται των απογόνων του
  (σ.σ. δίνεται προτεραιότητα στον γονέα έναντι των παιδιών)

```
1 Algorithm PreOrder(v)
2 |  if v == null then return;
3 |  visit(v);
4 |  foreach child x of v do
5 |  |  PreOrder(x);
```



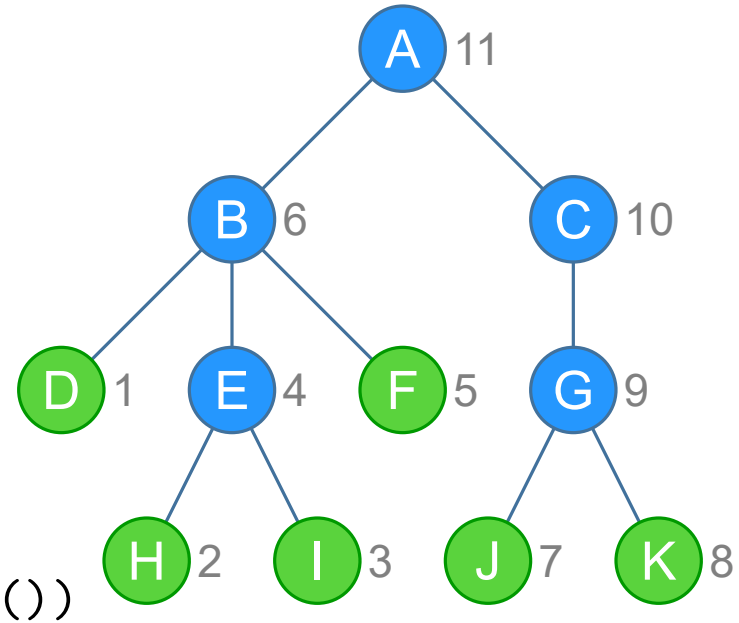
- Αρχική κλήση: `preOrder(T.root())`
- Πολυπλοκότητα: $\mathcal{O}(n)$

Μεταδιάταξη Δένδρου

Postorder tree traversal

- Μεταδιάταξη δένδρου: Κάθε κόμβος έπεται των απογόνων του
  (σ.σ. δίνεται προτεραιότητα στα παιδιά έναντι του γονέα)

```
1 Algorithm PostOrder(v)
2     if v == null then return;
3     foreach child x of v do
4         PostOrder(x);
5     visit(v);
```



- Αρχική κλήση: `postOrder(T.root())`
- Πολυπλοκότητα: $\mathcal{O}(n)$

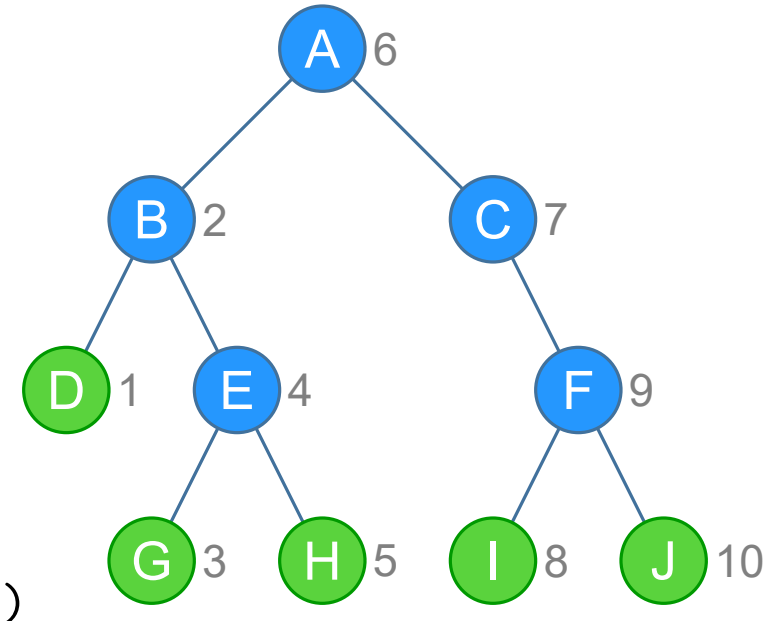
Εσωδιάταξη Δένδρου

Inorder tree traversal

- Εσωδιάταξη δένδρου: Εφαρμόζεται μόνο σε δυαδικά δένδρα:

  Κάθε κόμβος έπεται (προηγείται) του αριστερού (δεξιού) του παιδιού

```
1 Algorithm inOrder(v)
2 |   if v == null then return;
3 |   inOrder(v.leftChild());
4 |   visit(v);
5 |   inOrder(v.rightChild());
```



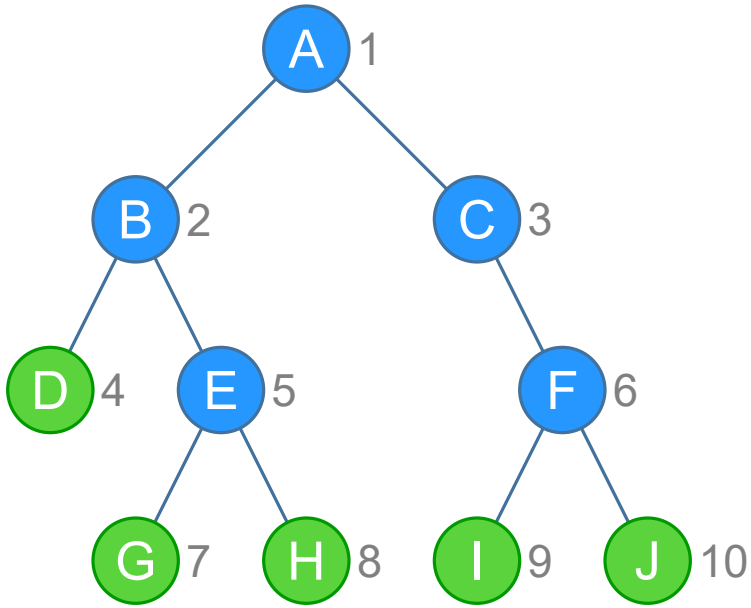
  - Αρχική κλήση: `inOrder(T.root())`
  - Πολυπλοκότητα: $\mathcal{O}(n)$

Διάταξη κατά Επίπεδα

Level-order tree traversal

- Διάταξη κατά επίπεδα: Επισκέπτεται τους κόμβους ανά επίπεδα, από “αριστερά προς δεξιά”

```
1  Algorithm levelOrder()
2  |  Queue *q = new Queue();
3  |  q.enqueue(T.root());
4  |  while !q.empty() do
5  |  |  w = q.dequeue();
6  |  |  visit(w);
7  |  |  foreach child x of w do
8  |  |  |  q.enqueue(x);
```

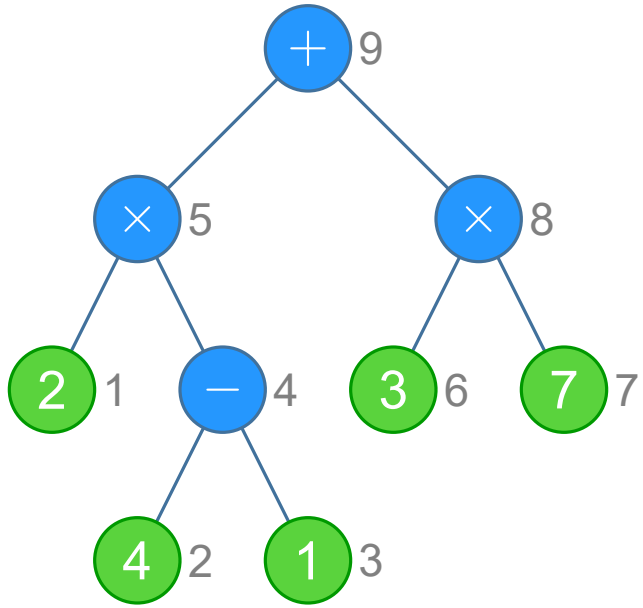


Πολυπλοκότητα: $\mathcal{O}(n)$

Δέντρα Αριθμητικών Εκφράσεων

Arithmetic expression trees

- Δυαδικά δένδρα που σχετίζονται με αριθμητικές εκφράσεις
  - Εσωτερικοί κόμβοι: Τελεστές
  - Εξωτερικοί κόμβοι: Τελεστέοι



- Παράδειγμα: $(2 \times (4 - 1) + (3 \times 7))$

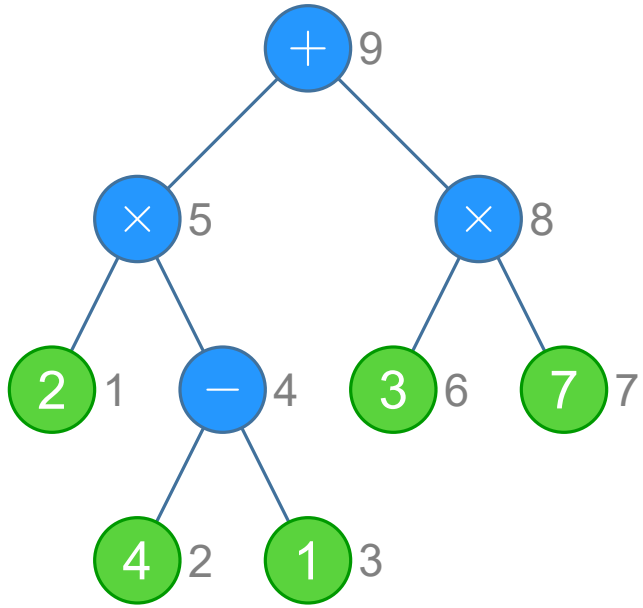
- Ερωτήσεις: Δοθέντος ενός δένδρου αριθμητικής έκφρασης,
  - πώς θα εκτυπώνατε την έκφραση στην οποία αντιστοιχεί; $\longrightarrow$ $((2 \times (4 - 1) + (3 \times 7))$
  - πώς θα αποτιμούσατε την έκφραση στην οποία αντιστοιχεί; $\longrightarrow$ $= 27$

Εκτύπωση αριθμητικών εκφράσεων

Print arithmetic expressions

- Εξειδίκευση της εσωδιάταξης:
  - εκτύπωσε τον τελεστή/τελεστέο όταν επισκέπτεσαι τον κόμβο
  - εκτύπωσε “(” πριν την διάσχιση του αριστερού υποδέντρου
  - εκτύπωσε “)” μετά την διάσχιση του δεξιού υποδέντρου.

```
1 Algorithm printExpression(v)
2 |   if v == null then return;
3 |   if !v.isLeaf() then print "(";
4 |   printExpression(v.leftChild());
5 |   print v.element() ;
6 |   printExpression(v.rightChild());
7 |   if !v.isLeaf() then print ")";
```



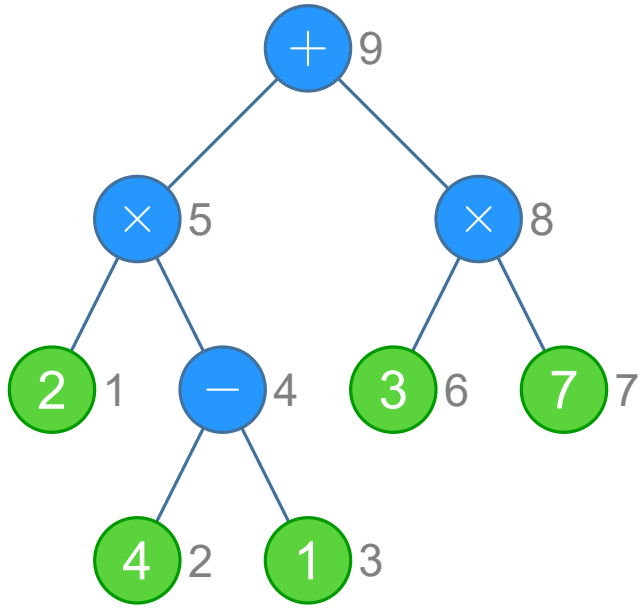
$$((2 \times (4 - 1) + (3 \times 7))$$

Αποτίμηση αριθμητικών εκφράσεων

Evaluate arithmetic expressions

- Εξειδίκευση της μεταδιάταξης:
  - μια αναδρομική κλήση επιστρέφει την τιμή ενός υποδέντρου.
  - κατά την επίσκεψη ενός εσωτερικού κόμβου, συνδυάζονται οι τιμές των υποδέντρων.

```
1 Algorithm evaluateExpression(v)
2 |   if v.isLeaf() then return v.element() ;
3 |   x ← evaluateExpression(v.leftChild());
4 |   y ← evaluateExpression(v.rightChild());
5 |   ◇ ← v.element();
6 |   return x ◇ y;
```



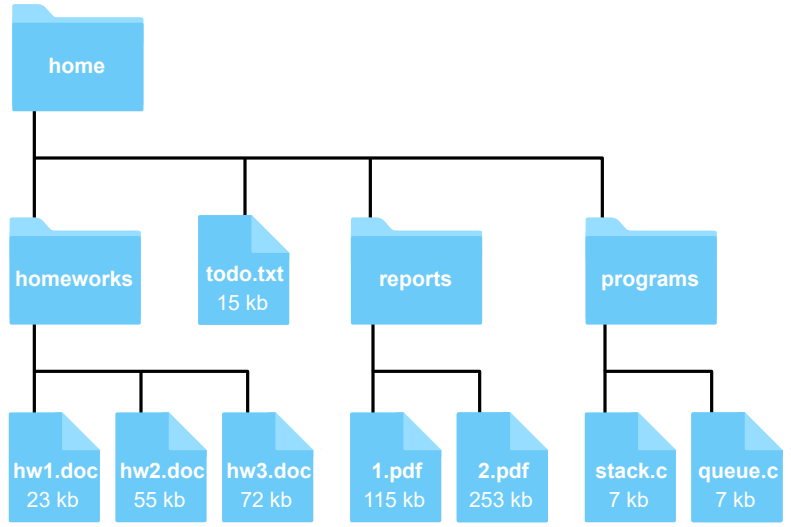
Αποτίμηση: 27

Συνολικό Μεγέθος Αρχείων Καταλόγου

Space used by files in a directory and its subdirectories

- Εξειδίκευση της μεταδιάταξης:
  - μια αναδρομική κλήση επιστρέφει το μέγεθος ενός υποκαταλόγου.
  - για τον υπολογισμό του μεγέθους ενός καταλόγου, συνδυάζονται τα μεγέθη των υποκαταλόγων και των αρχείων του.

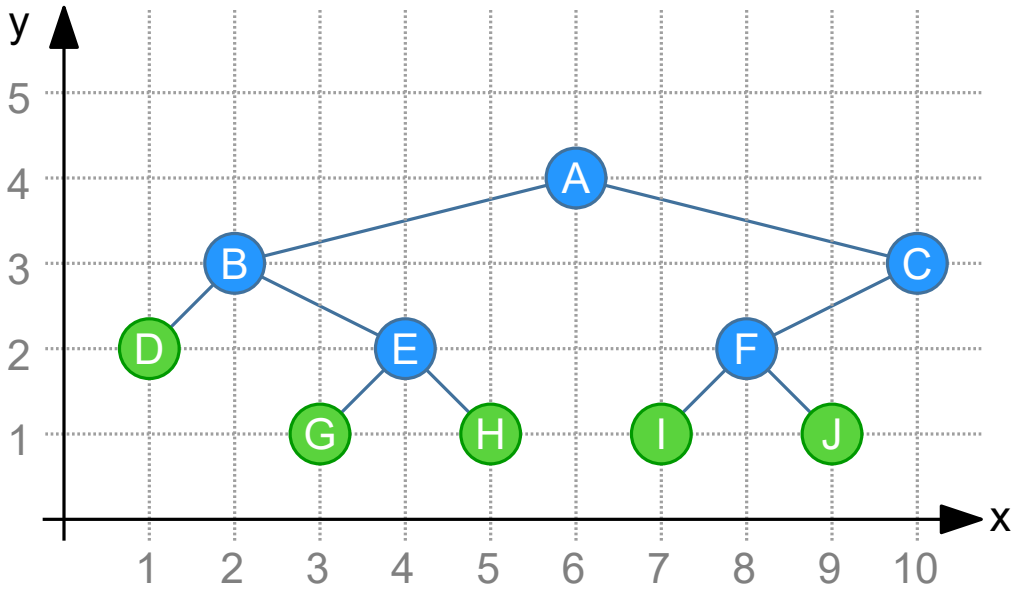
```
1 Algorithm computeSize(f)
2     if f.isFile() then
3         return f.size();
4     else
5         size ← 0;
6         foreach file or directory e in f do
7             size += computeSize(e);
8         return size;
```



Απεικονίσεις Δυαδικών Δένδρων

Binary tree drawings

- Μπορούμε να κατασκευάσουμε μια απεικόνιση ενός δυαδικού δένδρου ως εξής:
  - `x(v)` = `inOrder(v)`
  - `y(v)` = `height(T) + 1 - depth(v)`

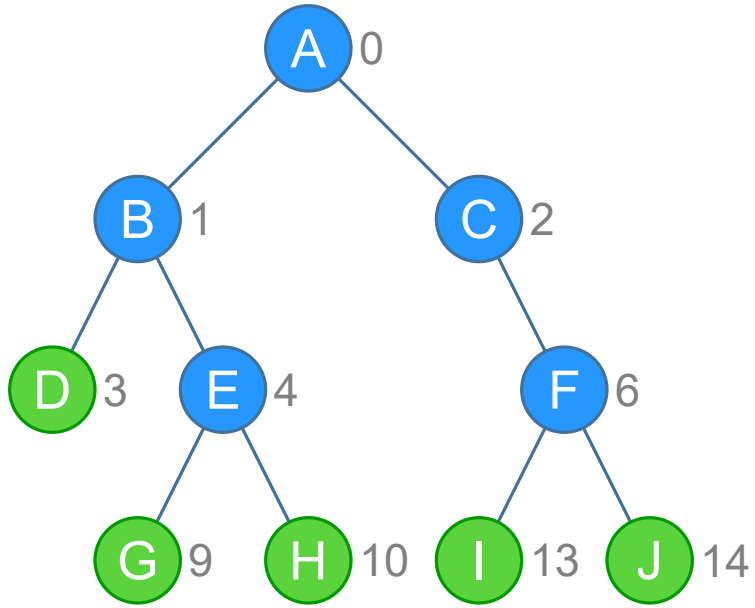


- Ιδιότητες:
  - Κάθε κόμβος απεικονίζεται σε ένα σημείο ενός ακέραιου πλέγματος
  - Το ακέραιο πλέγμα έχει μέγεθος $\mathcal{O}(n) \times \mathcal{O}(n)$
  - Δεν υπάρχουν διασταυρώσεις μεταξύ ακμών του δένδρου

Διανυσματική Υλοποίηση ενός Δυαδικού Δένδρου

Array-Based Representation of Binary Trees

- Το στοιχείο κάθε κόμβου αποθηκεύεται σε ένα διάνυσμα A

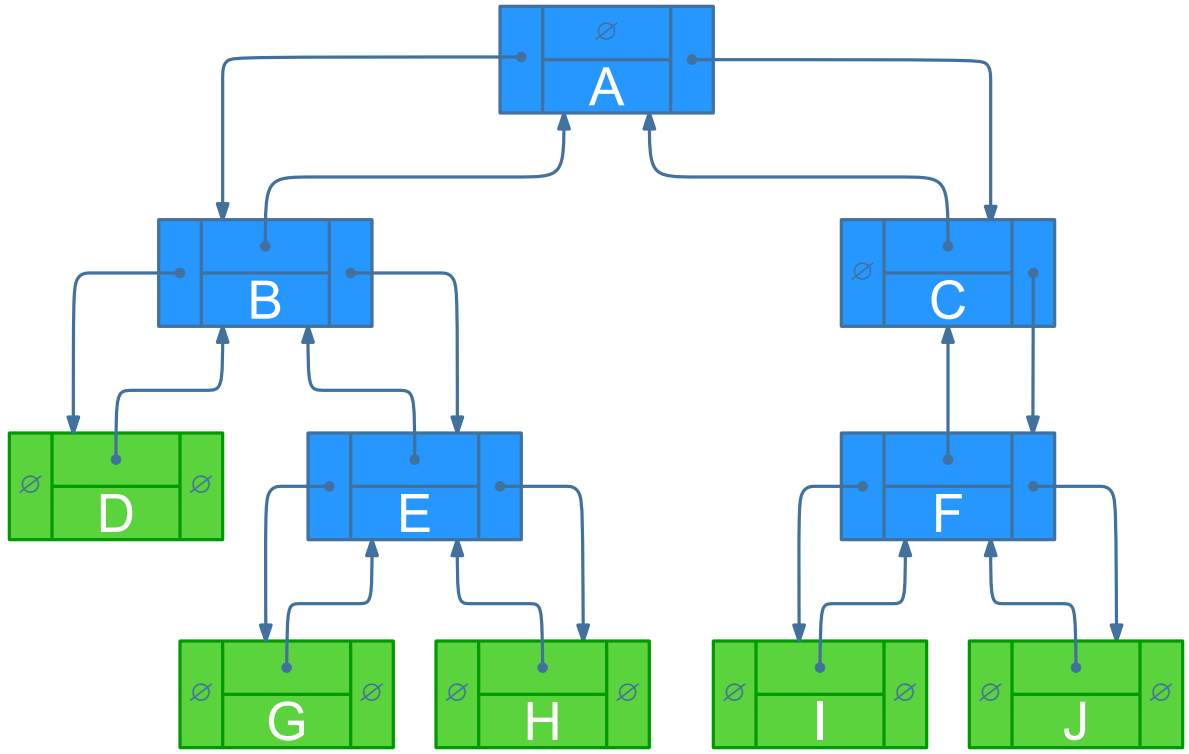


- Το στοιχείο του κόμβου v αποθηκεύεται στη θέση `rank(v)` του διανύσματος A, όπου:

$$\texttt{rank(v)} = \begin{cases} 0, & \text{αν ο κόμβος } v \text{ είναι η ρίζα} \\ 2\,\texttt{rank(v.parent())+1}, & \text{αν ο κόμβος } v \text{ είναι αριστερό παιδί} \\ 2\,\texttt{rank(v.parent())+2}, & \text{αν ο κόμβος } v \text{ είναι δεξί παιδί} \end{cases}$$

Υλοποίηση ενός Δυαδικού Δένδρου ως Συνδεδεμένη Δομή

- Ένας κόμβος αποθηκεύει:
  - ένα στοιχείο
  - μια σύνδεση στον κόμβο γονέα
  - μια σύνδεση στον κόμβο αριστερό-παιδί
  - μια σύνδεση στον κόμβο δεξι-παιδί
- Ένα δένδρο αποθηκεύει:
  - μια αναφορά στον κόμβο-ρίζα



- Σημείωση: ένας κόμβος ενός γενικού (σ.σ., μη-δυαδικού) δένδρου αποθηκεύει μια λίστα κόμβων (σ.σ., των παιδιών του) αντί των δύο συνδέσεων στους κόμβους αριστερό- και δεξιό-παιδί

Επιπλέον υλικό

- Ενότητα 2.3:
  Michael T. Goodrich, Roberto Tamassia, Αλγόριθμοι σχεδίαση και εφαρμογές
  ISBN: 9789605126971, εκδόσεις Γκιούρδα, 2016.