Les Arbres

Sandrine Vial
sandrine.vial@uvsq.fr

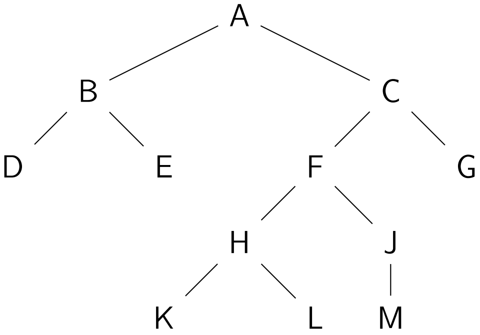
Novembre 2020

Les arbres binaires

1. Etude d'une classe particulière d'arbres
2. Propriétés
3. Algorithmes

Les arbres binaires

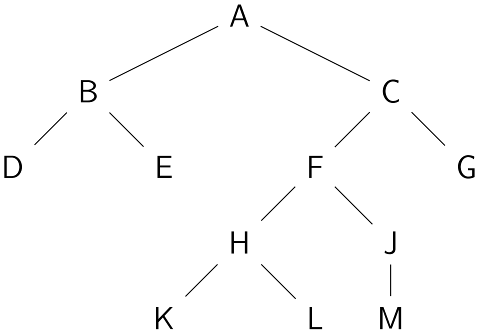
Tous les nœuds d'un arbre binaire ont $0, 1$ ou 2 enfants.



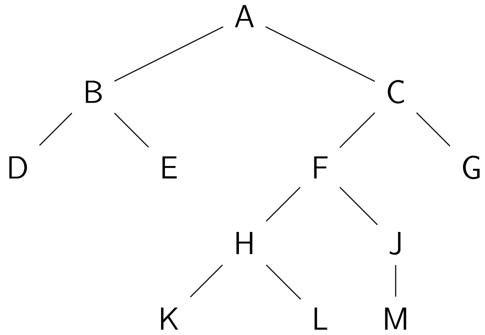
Les arbres binaires : Définitions

- **Enfant gauche** de n = racine du sous-arbre gauche de n.
- **Enfant droit** de n = racine du sous-arbre droit de n.
- **Bord gauche** de l'arbre = le chemin depuis la racine en ne suivant que des fils gauche.
- **Bord droit** de l'arbre = le chemin depuis la racine en ne suivant que des fils droits.

Exemple : arbre binaire



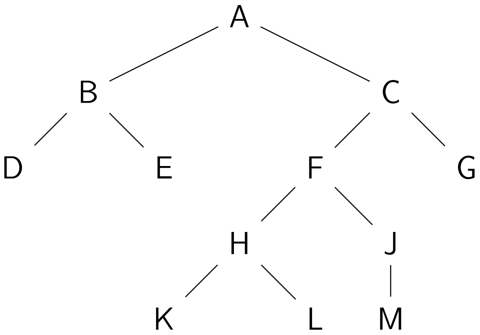
Exemple : arbre binaire



Taille de l'arbre :
Longueur de cheminement externe :
Hauteur moyenne :

Nombre de feuilles :
Longueur de cheminement :
Hauteur moyenne externe :

Exemple : arbre binaire



Taille de l'arbre : 12
Longueur de cheminement externe : 18
Hauteur moyenne : 2.33

Nombre de feuilles :6
Longueur de cheminement : 28
Hauteur moyenne externe : 3

Quelques arbres binaires particuliers

1. Arbre binaire *filiforme*
2. Arbre binaire *complet*
   - 1 nœud à la hauteur 0
   - 2 nœuds à la hauteur 1
   - 4 nœuds à la hauteur 2
   - ...
   - 2^h nœuds à la hauteur h.
   - Nombre total de nœuds d'un arbre de hauteur h :

$$2^0 + 2^1 + 2^2 + \cdots + 2^h = 2^{h+1} - 1$$

Quelques arbres binaires particuliers

1. Arbre binaire *parfait* :
   - Tous les niveaux sont remplis sauf le dernier.
   - Les feuilles sont le plus à gauche possible.
2. Arbre binaire *localement complet* : chaque nœud a 0 ou 2 fils.

Propriétés sur les arbres (1)

Lemme

$$h(T) \leq taille(T) - 1$$

Idée de Preuve

Egalité obtenue pour un arbre filiforme.

Propriétés sur les arbres (2)

Lemme

Pour tout arbre binaire T de taille n et de hauteur h on a :

$$\lfloor \log_2 n \rfloor \leq h \leq n - 1$$

Idée de Preuve

- Arbre filiforme : arbre de hauteur h ayant le plus petit nombre de nœuds : $n = h + 1$ (seconde inégalité).
- Arbre complet : arbre de hauteur h ayant le plus grand nombre de nœuds : $n = 2^{h+1} - 1$ (première inégalité).

Propriétés sur les arbres

Corollaire

Tout arbre binaire non vide T ayant f feuilles a une hauteur $h(T)$ supérieure ou égale à $\lceil \log_2 f \rceil$.

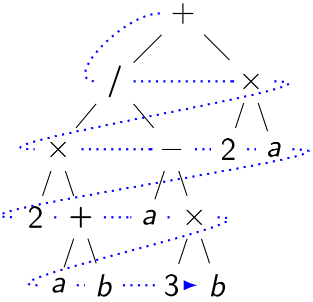
Lemme

Un arbre binaire localement complet ayant n nœuds internes a $(n+1)$ feuilles.

Exploration

- Pas aussi simple que dans le cas des listes.
- Pas d'ordre naturel.
- Deux types de parcours :
  - En largeur d'abord
  - En profondeur d'abord

Parcours en largeur d'abord



Ordre d'évaluation des nœuds :

$$+/\times\times-2\ a\ 2+a\times a\ b\ 3\ b$$

Type de données

Mise en œuvre chaînée

```
Enregistrement Nœud {
   Val      : entier;
   Gauche : ↑ Nœud;
   Droit : ↑ Nœud;
}
```

Parcours en largeur d'abord

Algorithme 1 Parcours en largeur d'un arbre binaire

```
ParcoursEnLargeur(r : Nœud)
▷ Entrée : r (la racine d'un arbre)
▷ Sortie : traitement de tous les nœuds de l'arbre enraciné en r
▷ Variables locales :
    ce_niveau, niveau_inférieur : File ;
    o : Nœud ;
Debut
    ce_niveau ← { r } ;
    tant que (ce_niveau est non vide) faire
        niveau_inférieur = { } ;
        pour chaque nœud o de ce_niveau faire
            traiter o ;
            niveau_inferieur ← niveau_inférieur ∪ enfants de o.
        fin pour
        ce_niveau ← niveau_inférieur ;
    fin tant que
Fin
```

Parcours en profondeur d'abord

Ordre d'évaluation des nœuds : ça dépend ....

Parcours en profondeur d'abord

- Parcours infixe :

$$2 \times a + b/a - 3 \times b + 2 \times a$$

- Parcours préfixe :

$$+/ \times 2 + a\ b - a \times 3\ b \times 2\ a$$

- Parcours postfixe

$$2\ a\ b + \times a\ 3\ b \times -/2\ a \times +$$

Parcours en profondeur d'abord

Algorithme 2 Parcours en profondeur d'un arbre binaire

```
ParcoursEnProfondeur(r : Nœud)
▷ Entrée : r (la racine d'un arbre)
▷ Sortie : traitement de tous les nœuds de l'arbre enraciné en r
 Debut
     si r = ∅
         traitement de l'arbre vide
     sinon
         traitement_prefixe(r) ;
         ParcoursEnProfondeur(r.Gauche) ;
         traitement_infixe(r) ;
         ParcoursEnProfondeur(r.Droit) ;
         traitement_postfixe(r) ;
     fin si
 Fin
```