Les arbres

Numérique et Sciences Informatiques - Lycée P. Méchain

---

1 Notion d'arbre

Un arbre est une structure de données utilisée lorsque certains éléments à représenter sont liés de façon hiérarchique.

— arbre généalogique
— rencontres d'un tournoi sportif
— relations hiérarchiques dans une organisation

Les arbres en informatique

Les arbres sont très utilisés en informatique, d'une part parce que les informations sont souvent hiérarchisées, et peuvent être représentées naturellement sous une forme arborescente, et d'autre part, parce que les structures de données arborescentes permettent de stocker des données volumineuses de manière efficace.

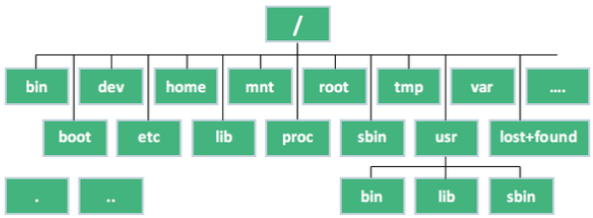

FIGURE 1 – Système de fichiers sous GNU/Linux

Exemples d'utilisation en informatique :

— systèmes de fichiers
— pages web
— arbres de décisions

2 Le vocabulaire des arbres

Un arbre est constitué d'une racine, de nœuds, de feuilles et de branches qui relient les éléments précédents entre eux.
Les arbres informatiques ont une particularité, la racine est en haut.

— **Nœud** : les éléments de l'arbre.
— **Racine** (ou nœud racine) : c'est le nœud de *départ* de l'arbre.
— **Feuille** : tout nœud n'ayant pas de successeur.
— **Nœud interne** : nœud ni feuille ni racine.
— **Branche** : relation entre deux nœuds.
— **Père** d'un nœud : le nœud précédent.Chaque nœud a un père et un seul, sauf la racine.
— **Arité** d'un nœud : nombre de fils.
— **Frère** d'un nœud : un nœud ayant le même père.
— On peut aussi utiliser les termes **ascendant** et **descendant**.

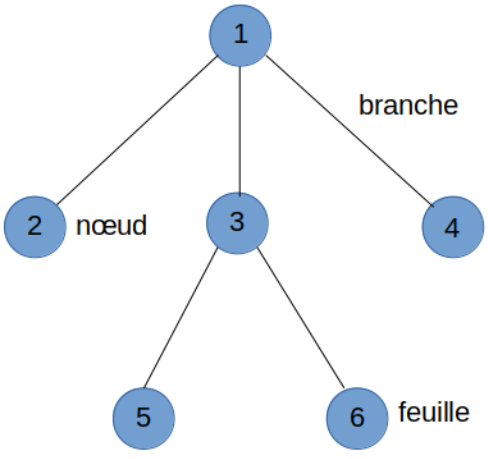

FIGURE 2 – Exemple d'arbre

Les nœuds sont généralement **étiquetés** par des valeurs mais il est également possible d'étiqueter les branches. On parle alors d'**arbre pondéré**.

— **Taille** de l'arbre : nombre de nœuds.
— **Profondeur** d'un nœud : nombre d'ascendants du nœud ou nombre de branches pour atteindre le nœud.
— **Hauteur** de l'arbre : La hauteur d'un arbre est la profondeur de la feuille la plus profonde. On rencontre aussi parfois la définition suivante la hauteur d'un arbre est égale à *1 + la profondeur du nœud le plus profond.*

Exemple :

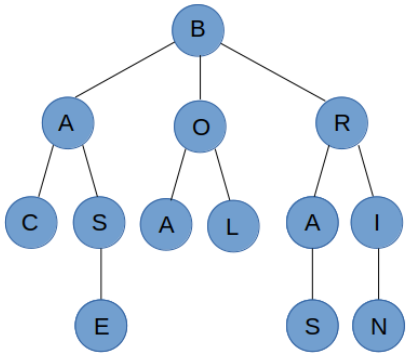

FIGURE 3 – Arbre lexicographique utilisé pour stocker des mots. Les préfixes communs à plusieurs mots apparaissent une seule fois dans l'arbre

Pour l'arbre de la figure 3 :
C a une profondeur de 2, **N** a une profondeur de 3.
La **hauteur** de l'arbre est 3 selon la première définition et 4 selon la seconde définition.

3 Les arbres binaires

Un arbre binaire (**AB**) est un arbre dont tous les nœuds sont d'arité au maximum 2. Ce sont les arbres qui seront étudiés en NSI. Selon Donald Knuth (The Art of Computer Programming Vol 1) un arbre binaire n'est pas un cas particulier d'arbre mais un autre concept. Il a entre autres montré que les arbres binaires et les arbres quelconques doivent être traités de façon différente, les arbres binaires ayant des propriétés spécifiques que n'ont pas les arbres généraux.

Un arbre binaire est soit vide, soit constitué d'une racine et de sous-arbres. On peut pour chaque sous arbre appliquer de manière récursive la même définition.

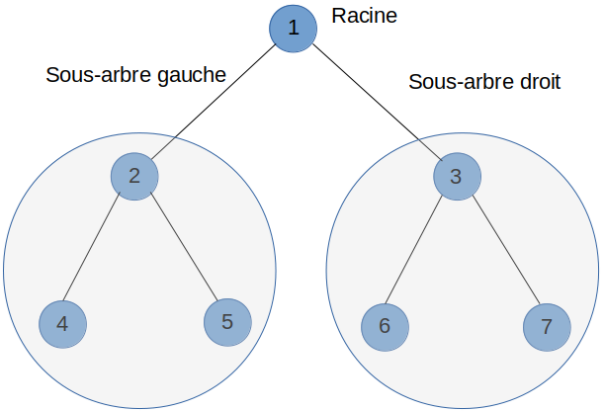

FIGURE 4 – Sous-arbres

Un **arbre binaire** (**AB**) est constitué d'une racine et de 2 sous-arbres : un **sous arbre gauche** et **un sous arbre droit**.

Quelques définitions sur les arbres binaires

- Un arbre binaire est **parfait** si tous les niveaux sont remplis. Toutes les feuilles sont donc à la même profondeur de la racine.
- Un arbre binaire est **strict** si chaque nœud a 0 ou 2 fils.
- Un arbre binaire est **presque complet** si tous les niveaux sont remplis sauf le dernier dans lequel les feuilles sont alignées à gauche.
- Un arbre binaire est **dégénéré** si tout nœud n'a qu'un fils.

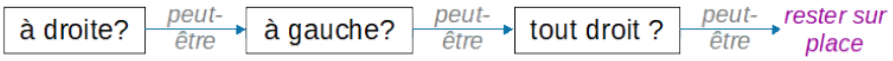

FIGURE 5 – Exemple d'arbre dégénéré

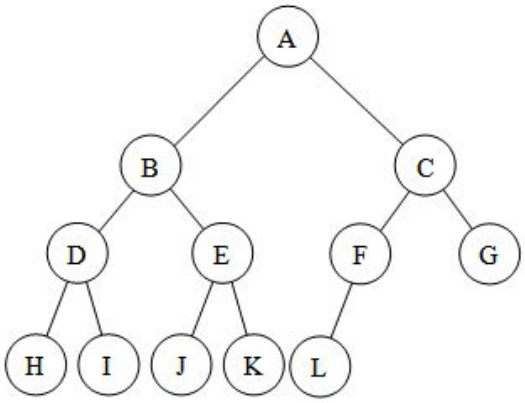

FIGURE 6 – Exemple d'arbre presque complet

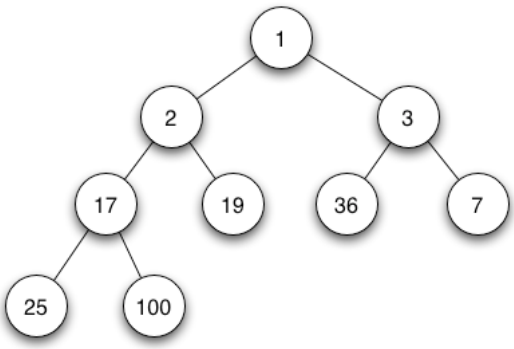

FIGURE 7 – Exemple d'arbre strict

4 Parcours en largeur d'un arbre

On parcourt par distance croissante à partir de la racine. Dans l'exemple ci-dessous cela revient à traiter les nœuds dans l'ordre : **a, b, c, d, e, f, g, i, j, k, m, t**

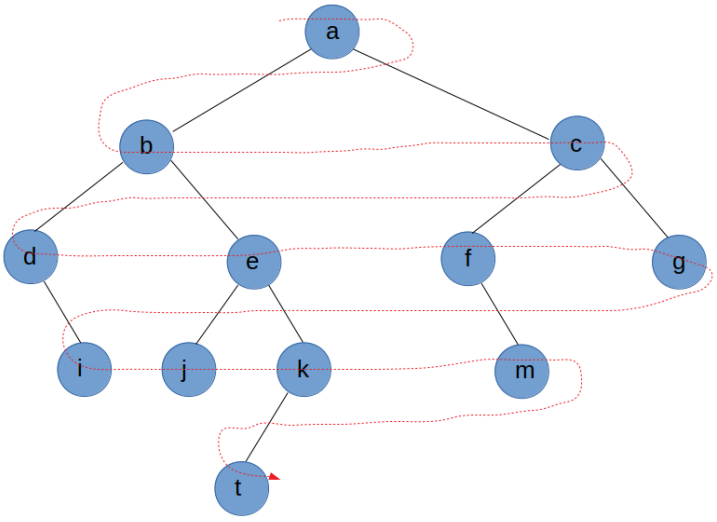

FIGURE 8 – Principe du parcours en largeur

5 Parcours en profondeur d'un arbre

Pour parcourir un arbre en profondeur on suit le parcours en pointillés dans l'ordre des numéros.

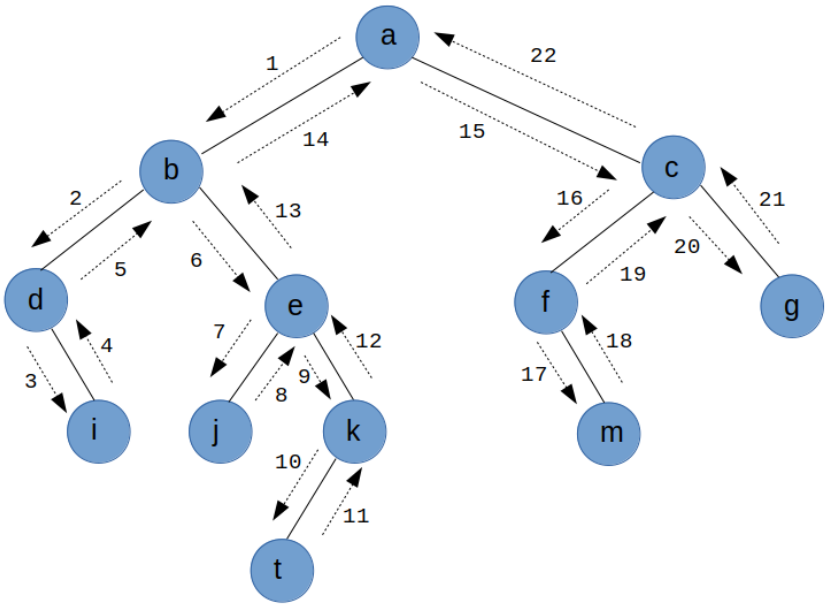

FIGURE 9 – Principe du parcours en profondeur

A partir de ce parcours en profondeur, 3 possibilités sont offertes :

— traiter les nœuds selon un ordre préfixe (parcours préfixe).

— traiter les nœuds selon un ordre postfixe (parcours postfixe).

— traiter les nœuds selon un ordre infixe (parcours infixe).

Afin de faciliter la découverte de l'ordre dans lequel les nœuds sont traités dans un parcours en profondeur on peut utiliser le chemin en pointillés ci-dessous qui considère l'arbre comme strict (chaque nœud a 0 ou 2 fils). Les feuilles vides ne sont pas traités mais utiles pour définir l'ordre des nœuds dans un parcours en profondeur infixe.

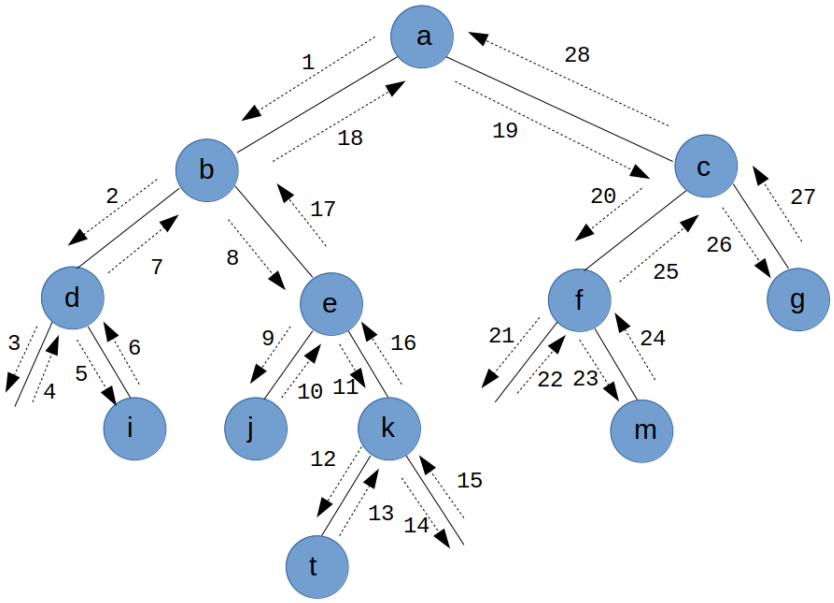

FIGURE 10 – Principe du parcours en profondeur sur un arbre strict

Lorsqu'on arrive sur une feuille non vide on prend en compte le nœud et on remonte. Si la feuille est vide, on remonte. Pour la racine et les nœuds internes on applique les règles ci-dessous (à partir du tracé de la figure 10) :

— **parcours préfixe** : on prend en compte un nœud la première fois qu'on le rencontre dans le parcours.
Pour l'arbre figure 10 cela revient à traiter les nœuds dans l'ordre : **a, b, d, i, e, j, k, t, c, f, m, g**

— **parcours postfixe** : on prend en compte un nœud la dernière fois qu'on le rencontre.
Pour l'arbre figure 10 cela revient à traiter les nœuds dans l'ordre : **i, d, j, t, k, e, b, m, f, g, c, a**

— **parcours infixe** : on prend en compte un nœud au retour du fils gauche donc la deuxième fois qu'on le rencontre.
Pour l'arbre figure 10 cela revient à traiter les nœuds dans l'ordre : **d, i, b, j, e, t, k, a, f, m, c, g**

Algorithme du parcours en profondeur préfixe

```
funct parcours_prefixe(arbre)
  if arbre n'est pas vide
    then afficher_racine(arbre)
         parcours_prefixe(sous-arbre gauche)
         parcours_prefixe(sous-arbre droit)
  fi
.
```

Algorithme du parcours en profondeur postfixe

```
funct parcours_postfixe(arbre)
  if arbre n'est pas vide
    then
         parcours_postfixe(sous-arbre gauche)
         parcours_postfixe(sous-arbre droit)
         afficher_racine(arbre)
  fi
.
```

Algorithme du parcours en profondeur infixe

```
funct parcours_infixe(arbre)
  if arbre n'est pas vide
    then
         parcours_infixe(sous-arbre gauche)
         afficher_racine(arbre)
         parcours_infixe(sous-arbre droit)
  fi
.
```

6 Quelques calculs avec les arbres

Le nombre d'arbres différents à m branches est le nombre de Catalan : $\frac{(2m)!}{m!(m+1)!}$

Dans un AB (arbre binaire) de hauteur h et de taille n :
$h+1 \leq n \leq 2^{h+1}-1$ si la hauteur est la profondeur de la feuille la plus profonde.
$h \leq n \leq 2^h - 1$ si la hauteur est 1 + profondeur de la feuille la plus profonde.