Algorithmique & programmation (2)

Chapitre 3

Arbres

Table des matières

Sorbonne Nouvelle

24 mars 2021

3.2 Définition et terminologie

3.2.1 Introduction

— Résultats d'un tournoi de badminton (arbre binaire) (fig. 3.1)
— Pédigree d'un cheval de course (ou arbre généalogique ascendant) (arbre binaire)
— Arbre généalogique descendant (non binaire)
— Arbre syntaxique (ln)
— Représentation syntaxique d'une expression (arithmétique, logique, rationnelle...)
— Trace des appels d'une fonction récursive
— Stockage des fichiers dans un système
— Structure des questions dans un système de diagnostic
— Structure de données associée à un tri (maximier)

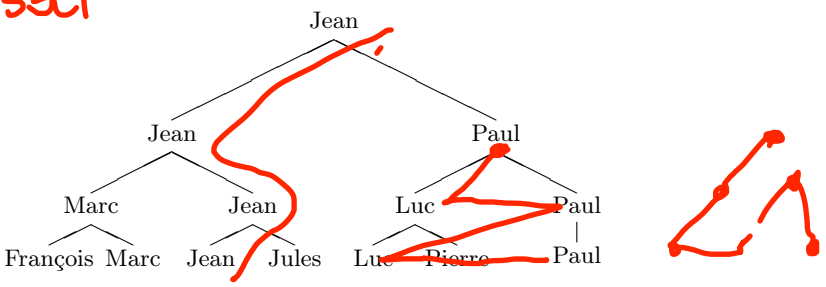


FIGURE 3.1 – Arbre d'un tournoi de badminton

La richesse de l'arbre par rapport aux structures linéaires est que le ***parcours*** est non trivial : il y a de multiples façons de « se promener » dans l'arbre, c'est-à-dire de traiter successivement les éléments de l'arbre, en étant *guidé* par la structure.
L'arbre peut donc être vu comme une **structure de données**, comme la liste, mais aussi comme une **structure de contrôle** complexe, qui détermine, par son organisation, l'ordre de certaines opérations.
Une propriété intrinsèque de la structure d'arbre est la récursivité, et les définitions des caractéristiques des arbres, aussi bien que les algorithmes qui les manipulent, s'écrivent très naturellement de manière récursive.
Il faut noter que les arbres sont en fait des cas particuliers de graphes, et on pourrait donc les étudier en tant que tels (une partie des notions introduites dans ce chapitre sont pertinentes pour les graphes en général). Mais il y a tant de spécificités pour ces cas particuliers, qui donnent lieu à des algos beaucoup moins complexes, que cela justifie de traiter les arbres séparément.

3.2.2 Définitions

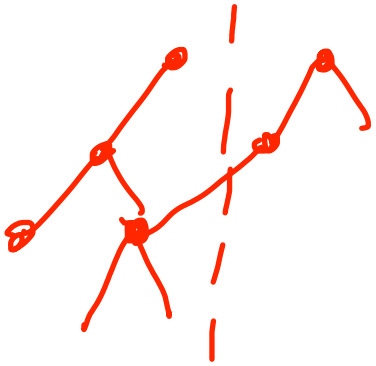
Déf. 1 (Arbre (récursif))

Un arbre $\mathcal{A} = \langle o, A_1 \dots A_p \rangle$
est la donnée d'une **racine** o et d'une liste finie d'arbres disjoints $A_1 \dots A_p$.

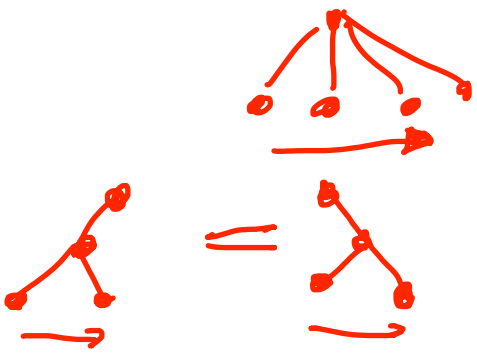
(liste vide)

Déf. 2 (Forêt)

Une **forêt** est une liste finie d'arbres disjoints.
Une **forêt partagée** est une liste finie d'arbres non disjoints.



— Arbre = Racine + Forêt
— Arbre minimal = 1 racine
— Les fils sont ordonnés,
— mais pas de façon absolue

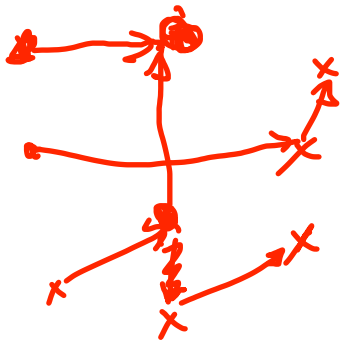


Déf. 3 (Arborescence)

Soit S un ensemble de *sommets*.
Soit $r \in S$ un sommet distingué (*racine*)
On définit un arbre A comme la donnée de $\langle S, r, A \rangle$ avec
$A \subset S \times S$, et $\forall s \in S, s \neq r, \exists! p \in S \ / \ (p, s) \in A$
(En mots : tout sommet, sauf la racine, a un père et un seul)

Déf. 4 (Arbre (planaire) (non récursif))

Un **arbre planaire** est une arborescence munie d'un ordre (total) sur tous les arcs sortants de chaque sommet.



Terminologie

graphe

arbre graphe connexe sans cycle

arborescence arbre orienté tel qu'il existe un chemin de la racine vers tous les autres sommets.

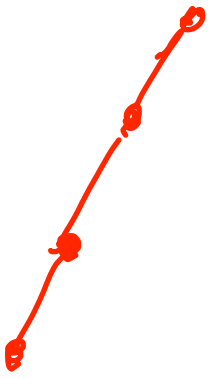
arbre planaire arborescence avec un ordre sur les arcs sortants de tous les sommets

arbre binaire arbre planaire dont le degré sortant est inférieur ou égal à deux

arbre binaire complet arbre binaire dont les sommets sont soit deux fils soit zéro

arbre binaire parfait arbre binaire complet dont toutes les feuilles sont à la même distance de la racine

- arc (orienté) : relation père/fils
- feuille
- nœud (sommet interne/branchant)
- frère
- chemin
- branche
- hauteur de l'arbre (nombre d'arcs)
- profondeur d'un nœud (nombre d'arcs)



3.3 Représentation

3.3.1 Listes de primitives

A :

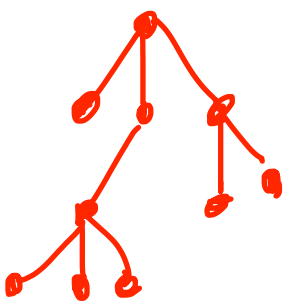
racine : noeud
liste-fils : liste d'arbres

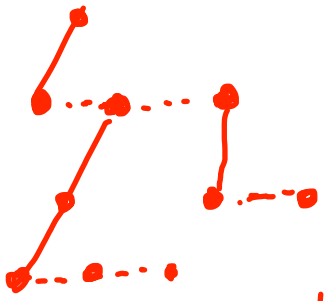
---

racine : Arbre → noeud
liste-fils : noeud → liste d'arbres

---

racine : Arbre → noeud
premier-fils : noeud → arbre
frère : noeud → arbre



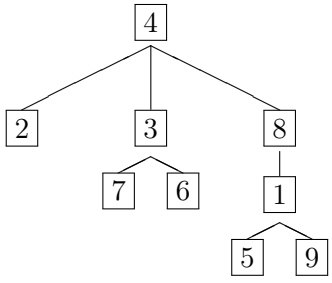


"binarisation"

3.4 Parcours

3.4.1 Notion (générale) de parcours

FIGURE 3.2 – Illustration des différents ordres de parcours possibles



1 ***Aléatoire*** (par rapport à la structure)
1, 2, 3, 4, 5...
2 ***Profondeur*** (Trémaux)
(*depth first*) (pile) Lifo
4, 2, 3, 7, 6, 8, 1, 5, 9
3 ***Largeur*** (hiérarchique, militaire)
(*width first*) (file) Fifo
4, 2, 3, 8, 7, 6, 1, 5, 9

4 2 3 7 6 8 1 5 9 1e rencontre
2 7 6 3 5 9 1 8 4 dernière rencontre

3.4.2 Parcours en profondeur

Version récursive

```
procedure parcours(X) ;
begin
  [1]
  pour tout fils Y de X faire
    parcours(Y) ;
  [2]
end
```

```
void parcours(X) ;
{
  [1]
   l = liste-fils(X) ;
   if (len(l) == 0)
      [2]
      return ;
   for i in range(len(l)) {
      parcours(element(l,i)) ;
      [3]
   }
   [4]
}
```

FIGURE 3.3 – Parcours en profondeur, récursif