DUT 2 Informatique – S3
M3103 – Algorithmique avancée

CM 03 – Construction de structures arborescentes

Équipe pédagogique : Camille Kurtz, Jacques Alès-Bianchetti, Simon Fuhlhaber

prénom.nom@parisdescartes.fr

18 août 2016

U S PC
Université Sorbonne Paris Cité

UNIVERSITÉ PARIS DESCARTES
INSTITUT UNIVERSITAIRE DE TECHNOLOGIE

Plan

Plan

Introduction

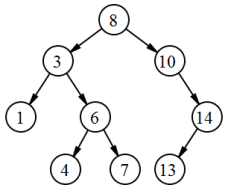
Définition

Qu'est ce qu'un arbre ?

- Il s'agit d'une **structure de données** organisée de manière arborescente
- On peut considérer un arbre comme une **généralisation d'une liste** car les listes peuvent être représentées par des arbres

A quoi cela peut servir ?

1. organiser et ranger des données
2. La complexité des algorithmes d'insertion, de suppression ou de recherche est généralement plus faible que dans le cas des listes



$\Longrightarrow$ La structure d'arbre est **très utilisée en informatique**

Introduction

Donald Knuth (naissance 1938)
Professeur émérite à Stanford

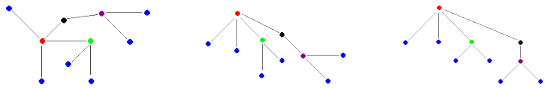
Importance des arbres en Informatique

Donald Knuth, auteur de l'ouvrage de référence sur l'algorithmique, en plusieurs volumes, intitulé « The Art of Computer Programming », considère les arbres **comme la structure la plus fondamentale de l'informatique**

Notions de graphes

Un arbre est un graphe

- Les mathématiciens voient les arbres comme des cas particuliers de graphes non orientés, fortement connexes et acycliques
- Un arbre est donc un couple $A = (V, E)$, où V est un ensemble de noeuds , et $E \subset V \times V$ est un ensemble d'arrêtes , chaque arête $e \in E$ reliant deux nœuds $x_1, x_2 \in V$



(a) (b) (c)

3 représentations graphiques de la même structure d'arbre

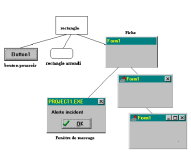
- (a) tous les sommets ont une disposition équivalente
- (b) et (c) le sommet « rouge » se distingue des autres
- Lorsqu'un sommet est distingué par rapport aux autres, on le dénomme . . . et la même structure d'arbre s'appelle une . . .

Plan

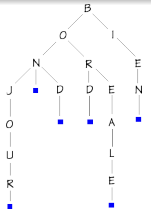
Familles d'arbres

Classement des arbres et utilisations

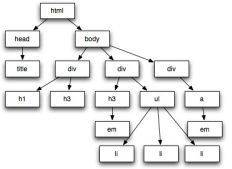
- **Arbres réguliers** : arbre binaire équilibré (toujours 2 branches)
- **Arbres irréguliers** : arbres déséquilibrés



Interface graphique et héritage



Arbre lexicographique



Fichier HTML

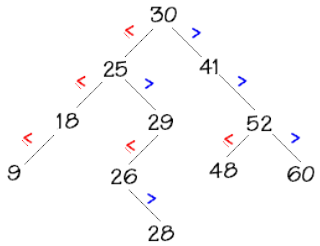
Remarques

1. Les **arbres binaires** sont les plus utilisés en informatique
2. Il est toujours possible de « binariser » un arbre non binaire, ce qui nous permettra ici de **n'étudier que les structures d'arbres binaires**

Familles d'arbres

Arbre binaire de recherche (ABR)

- Les **arbres binaires de recherche (ABR)** sont des arbres dont les nœuds sont de degré 2 et qui sont tels que pour chaque nœud la valeur de son fils gauche lui est inférieure ou égale , la valeur de son fils droit . lui est strictement supérieure
- Un tel arbre peut être utilisé pour maintenir triés des ensembles de valeurs (Java.Util.TreeSet)



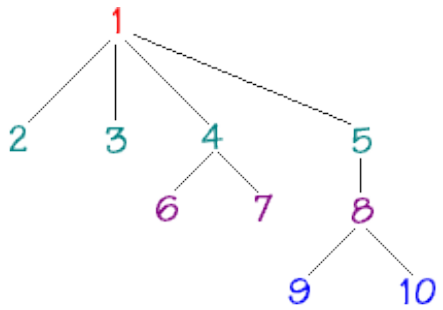
ABR ayant comme racine 30 et stockant des entiers

Plan

Notations

Étiquette

- Un arbre A dont tous les nœuds sont nommés est dit ... etiquetté
- **L'étiquette** (ou nom du sommet) représente ... la valeur du noeuds ou bien l'info associée



Exemple d'un arbre A étiqueté avec des entiers entre 1 et 10

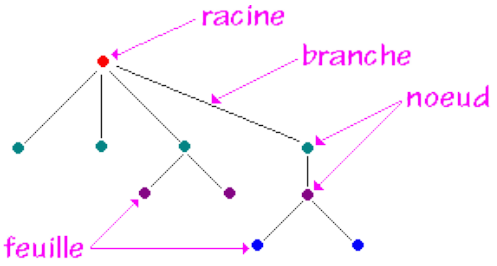
Notations

Racine, nœud, branche, feuille

Un arbre A (non-vide) est toujours composé de :

- 1 **racine**
- $n > 0$ **nœuds** (dont la racine)
- $n - 1$ **branches**

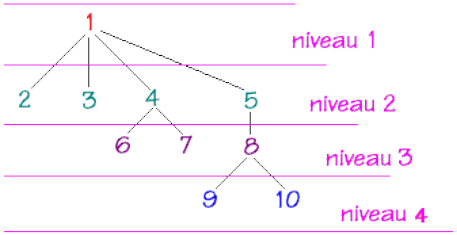
Les nœuds terminaux sont nommés fueilles



Notations

Profondeur ou niveau d'un nœud

- La **profondeur** (ou niveau) d'un nœud x est égale au nombre de noeuds à partir de la racine pour aller au noeud x
- On notera $p(x)$ la **fonction profondeur** d'un nœud x



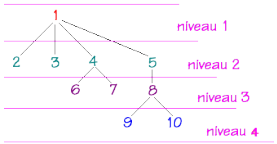
Exemple

- Pour atteindre le nœud 9, il faut parcourir le lien 1–5, puis 5–8, puis 8–9 soient 4 nœuds donc 9 est de profondeur égale à 4, soit $h(9) = 4$
- Par définition $p(racine) = 1$ (pour tout arbre non vide)

Notations

Chemin d'un nœud

On appelle **chemin du nœud** x la suite des noeuds pour aller de la racine au neoud x



Exemples

- Chemin du nœud 10 = <1,5,8,10>
- Chemin du nœud 9 = <1,5,8,9>
- Chemin du nœud 1 = <1>

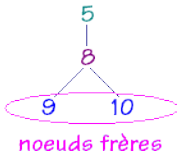
Remarquons que la profondeur p d'un nœud x est égale au nombre de nœuds dans le chemin

$$p(x) = \text{Cardinal(Chemin(x))}$$

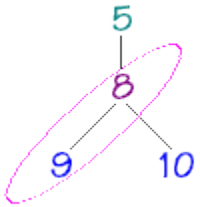
Notations

Nœuds frères, parents, enfants, ancêtres

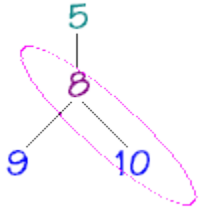
Le vocabulaire de lien entre nœuds de niveaux différents et reliés entres eux est emprunté à la **généalogie**



9 et 10 sont ..frères



9 est ..enfant de 8
8 est ..parent de 9



10 est ..enfant de 8
8 est ..parent de 10

5 est ..parent de 8 et 9

. ancêtre de 9 et 10

Ascendance et descendance

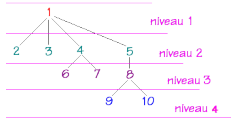
On parle aussi d'**ascendant**, de **descendant** ou de **fils** pour évoquer des relations entres les nœuds d'un même arbre reliés entre eux

Notations

Calcul récursif de la profondeur

On peut définir **récursivement** la profondeur p d'un nœud x via celle de son père :

$$p(x) = \ldots \begin{array}{l} 1 \text{ si x est la racine} \\ 1+ \text{p(parent(x)) sinon} \end{array}$$



Exemple

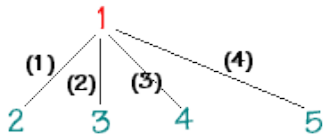
Calculons récursivement la profondeur du nœud 9, notée $p(9)$:

$$p(9) = 1 + p(8)$$
$$p(8) = 1 + p(5)$$
$$p(5) = 1 + p(1)$$
$$p(1) = 1$$
$$\Rightarrow p(5) = 2 \Rightarrow p(8) = 3 \Rightarrow p(9) = 4$$

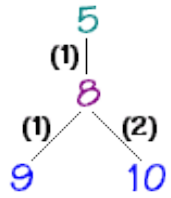
Notations

Degré d'un nœud

Par définition le **degré** d'un nœud est égal au nombre de ses descendants



Le nœud 1 est de **degré** 4, car il a 4 enfants



Le nœud 5 a un enfant donc son **degré** est 1
Le nœud 8 est de **degré** 2 car il a 2 enfants

Remarque

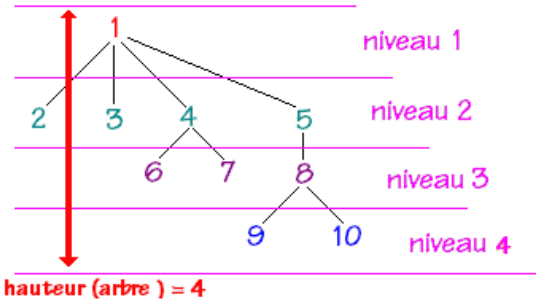
Remarquons que lorsqu'un arbre a tous ses nœuds de degré 1 (sauf les feuilles), on le nomme **arbre dégénéré** et que c'est en fait une liste

Notations

Hauteur d'un arbre

La **hauteur** d'un arbre est le nombre de noeuds du chemin le plus long . La hauteur h d'un arbre correspond donc à la profondeur maximum de ses nœuds :

$$h(A) = \max \{ p(x) \}$$

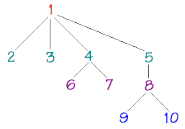


Notations

Degré d'un arbre

Le **degré d'un arbre** est égal au plus grand degré :

$$d(A) = \max \{ d(x) \}$$



Exemples

- $d(1) = 4$ et $d(2) = 0$
- $d(3) = 0$ et $d(4) = 2$
- $d(5) = 1$ et $d(6) = 0$
- $d(7) = 0$ et $d(8) = 2$
- $d(9) = 0$ et $d(10) = 0$

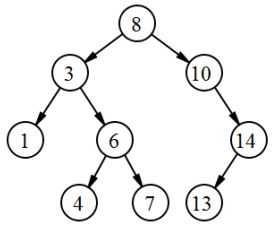
La valeur maximale est 4, donc 4.

Notations

Taille d'un arbre

On appelle **taille d'un arbre** le nombre total de nœuds de cet arbre :

$$taille(x) = 1 + \sum_{i=1}^{degré(a)} \text{Taille}(fils_i(x))$$



Exemple

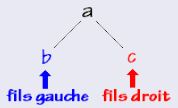
Cet arbre a pour taille 9 (car il a 9 nœuds)

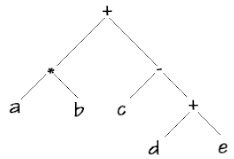
Plan

Arbres binaires

Notions préliminaires

- Un arbre binaire est de degré 2 (noeuds de degré 2 ou plus)
- Les descendants d'un nœud sont lus **de gauche à droite** et sont appelés respectivement fils gauche et fils droit de ce nœud
- Nous noterons $< a, b, c >$ par convention l'arbre binaire dessiné ci-dessous





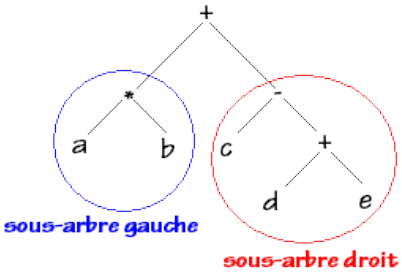
Exemple : l'arbre abstrait de l'expression $a * b + c - (d + e)$ est un arbre binaire

Arbres binaires

Exemple

Les sous-arbres gauche et droit de l'arbre A :

- $filsG(A) =< *, a, b >$
- $filsD(A) =< -, c, < +, d, e >>$



Type Abstrait de Données (TAD) : ArbreBin

Utilise : utilise : T_0 (information du nœud), Nœud, Booleens

Opérations :

- Æ (arbre vide) : $\varnothing \rightarrow$ ArbreBin
- Racine : ArbreBin $\rightarrow$ Nœud
- filsG : ArbreBin $\rightarrow$ ArbreBin
- filsD : ArbreBin $\rightarrow$ ArbreBin
- Constr : Nœud $\times$ ArbreBin $\times$ ArbreBin $\rightarrow$ ArbreBin
- Est_Vide : ArbreBin $\rightarrow$ Booleen
- Info : Nœud $\rightarrow T_0$

Préconditions :

- Racine(Arb) `def_ssi` Arb ! = Æ
- filsG(Arb) `def_ssi` Arb != Æ
- filsD(Arb) `def_ssi` Arb != Æ

Axiomes : rac $\xrightarrow{\text{a pour type}}$ Nœud, fg $\xrightarrow{\text{a pour type}}$ ArbreBin, fd $\xrightarrow{\text{a pour type}}$ ArbreBin

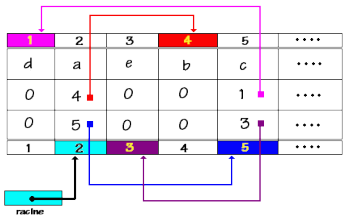
- Racine(Constr(rac,fg,fd)) = rac
- filsG(Constr(rac,fg,fd)) = fg
- filsD(Constr(rac,fg,fd)) = fd
- Info(rac) = T_0

Implantation des arbres binaires

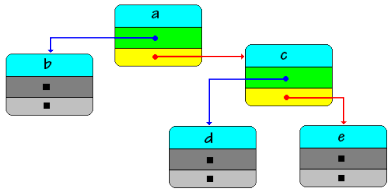
Exemples et implémentations d'arbres binaires étiquetés

On peut représenter un arbre binaire étiqueté **selon 2 spécifications différentes** :

1. Une implantation fondée sur une . . .
   , nécessitant de connaître au préalable le nombre maximal de nœuds de l'arbre (ou encore sa taille)
2. Une implantation fondée sur une . . .
   implémentée soit par des pointeurs (variables dynamiques), soit par des références (objets)



(1) Implantation par tableau



(2) Implantation par chaînage

Implantation par tableaux

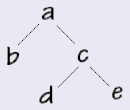
Spécification

- Un nœud est **une structure statique** contenant 3 éléments :
  1. l'info du noeud . . .
  2. fils gauche
  3. file droit . . .
- Pour un arbre binaire de taille n, **chaque nœud de l'arbre binaire est stocké dans une cellule d'un tableau** de dimension 1 à n cellules
- Un nœud est repéré dans le tableau par **l'indice de la cellule le contenant**
- Le champ **fils gauche** du nœud sera l'indice de la cellule contenant **le descendant gauche**, et le champ **fils droit** vaudra l'indice de la cellule contenant **le descendant droit**

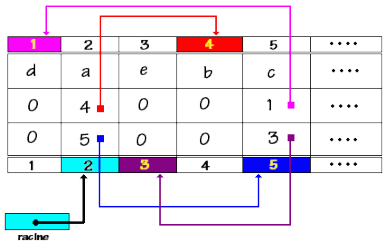
Implantation par tableaux

Exemple

Selon l'implantation choisie, par hypothèse de départ, la racine $< a, vers\ b, vers\ c >$ est contenue dans la cellule d'indice 2 du tableau, les autres nœuds sont supposés être rangés dans les cellules 1, 3, 4, 5 :



$$
\begin{aligned}
racine = \quad & table[2] \\
table[1] = \quad & < d, 0, 0 > \\
table[2] = \quad & < a, 4, 5 > \\
table[3] = \quad & < e, 0, 0 > \\
table[4] = \quad & < b, 0, 0 > \\
table[5] = \quad & < c, 1, 3 >
\end{aligned}
$$



Explications

- $table[2] =< a, 4, 5 >$ signifie que le fils gauche de ce nœud est dans table[4] et son fils droit dans table[5]
- $table[5] =< c, 1, 3 >$ signifie que le fils gauche de ce nœud est dans table[1] et son fils droit dans table[3]
- $table[1] =< d, 0, 0 >$ signifie que ce nœud est une feuille

Implantation par tableaux en Java

```
public class ArbreBin{
    int tailleArbre = n; // n est le nombre de noeuds
    int indRacine = -1; //Indice de la racine dans le tableau

    // Le tableau codant un arbre
    Integer [][] tableau = new Integer [3][n];
    ...
}
```

Explications

- Lorsque indRacine = -1 on dit que l'arbre est vide
- **L'accès à la racine** de l'arbre s'effectue ainsi : tableau[indRacine]
- **L'accès à l'info de la racine** de l'arbre : tableau[indRacine][2]
- L'accès au fils gauche de la racine de l'arbre s'effectue ainsi : tableau[tableau[indRacine][0]]
- **L'insertion ou la suppression d'un nœud s'effectue directement dans une cellule du tableau.** Il faut donc posséder une structure (*e.g.*, liste) permettant de connaître les cellules libres ou de ranger une cellule nouvellement libérée
- L'insertion se fera dans la première cellule libre
- La suppression rajoutera une nouvelle cellule dans l'espace libre

Implantation des arbres binaires par chaînage

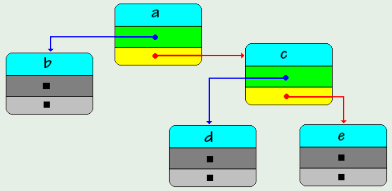
Spécification

Le nœud reste une structure statique contenant 3 éléments dont 2 dynamiques :

1. l'info du noeud
2. référence vers le fils gauche
3. référence vers le fils droit

Exemple

Selon l'implantation choisie, par hypothèse de départ, la référence vers la racine pointe vers la structure statique (le nœud) $< a, ref\ vers\ b, ref\ vers\ c >$:



Explications :

- ref racine $\leftarrow < a, ref\ vers\ b, ref\ vers\ c >$
- ref vers b $\leftarrow < b, null, null >$
- ref vers c $\leftarrow < a, ref\ vers\ d, ref\ vers\ e >$
- ref vers d $\leftarrow < d, null, null >$
- ref vers e $\leftarrow < e, null, null >$

Implantation des arbres binaires par chaînage en Java

```
public class Arbre {
    private int valeur; // Info. du Noeud
    // Chainage vers fils
    private Arbre gauche=null, droit=null;

    // CONSTRUCTEURS
    public Arbre(int x) {
        this.valeur = x;
    }
    public Arbre(int x, Arbre g, Arbre d) {
        this.valeur = x;
        this.gauche = g;
        this.droit = d;
    }
    // ACCESSEURS
    public int getValeur() {
        return this.valeur;
    }
    public Arbre getSousArbreGauche() {
        return this.gauche;
    }
    public Arbre getSousArbreDroit() {
        return this.droit;
    }
    // LE MAIN POUR TESTER
    public static void main(String[] arg) {
        Arbre b = new Arbre(2,new Arbre(1),new Arbre(4,new Arbre(3),new Arbre(5)));
        Arbre c = new Arbre(10,new Arbre(8),new Arbre(12));
        Arbre racine = new Arbre(6,b,c);
        ...
     }
}
```

Explications

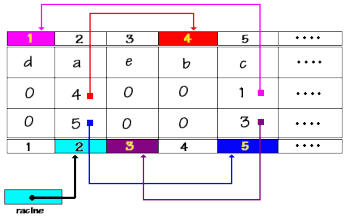
- L'accès à l'info de la racine de l'arbre s'effectue ainsi : racine.getValeur()
- L'accès au fils gauche de la racine : racine.getSousArbreGauche()
- L'accès au droit de la racine : racine.getSousArbreDroit()

Implantation des arbres binaires

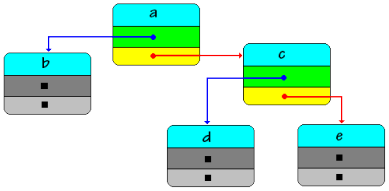
Complexité spatiale et temporelle liée aux 2 implémentations

Soit un ensemble de n données à stocker dans un arbre :

- **Complexité spatiale** : 3n = Theta(n)
- **Complexité temporelle** : Accèder à un noeud, supprimer un noeud



(1) Implantation par tableau



(2) Implantation par chaînage

Plan

Fonctions élémentaires sur les arbres

```
boolean estFeuille(){ //Test si feuille
    if (this.getFilsG() == null && this.getFilsD() == null){
        return true;
    }else{ return false; }
}
boolean estUnNoeud(){ //Test si noeud
    return !estFeuille();
}
static int hauteur(Arbre a){ // Hauteur de l'arbre
    if (a == null)
        return 0;
    else
        return (1.+ Math.max(hauteur(a.getFilsG()), hauteur(a.getFilsD())));

}
static int nbFeuilles(Arbre a){ // Nombre de feuilles de l'arbre
    if (a == null) return 0;
    else if (a.estFeuille()) return 1;
    else return nbFeuilles(a.getFilsG()) + nbFeuilles(a.getFilsD());
}
```


Références

Bibliographie

Des éléments de ce cours sont empruntés de [Cormen(2011), Passat(2007)]

T. Cormen.
Introduction à l'algorithmique.
Dunod, 2011.

N. Passat.
CM 4 – Arbres.
Module Structures de données et algorithmes fondamentaux, L3 mention Sciences du Vivant, Université Louis Pasteur, 2007.