Les arbres

DUT Informatique – Deuxième année
Université du Havre

G. Simon/B. Mermet – octobre 2007

Introduction

- Structure de données récursive
  - (Pile)
  - Liste
- Utilisée pour représenter des hiérarchies
  - Classification des plantes, animaux
  - Généalogie
  - Représentation d'expressions arithmétiques
  - Dans les SE : arborescence de fichiers
  - En JAVA : hiérarchie d'héritage
- Complexité des algos de maj généralement moindre que pour les listes (arbres équilibrés)

Objectifs du cours

- Comprendre les arbres
  - Définition
  - Propriétés
  - Opérations
- Savoir implanter un arbre en JAVA

Plan du cours

- Notions générales sur les arbres
  - Définition, vocabulaire associé
  - Propriétés
  - Exemples d'utilisation des arbres
  - Parcours d'arbres
- Cas particulier des arbres binaires
  - Type abstrait
  - Implantation en JAVA (structure, parcours)
  - Arbres binaires de recherche

Définition informelle et vocabulaire

- Spécification :
  - Informelle : ensemble d'éléments appelés noeuds liés par une relation de parenté établissant une hiérarchie entre les noeuds
  - Représentation graphique
- Terminologie associée
  - Liens de parenté : père, fils, frère
  - Branche
  - Noeud, noeud racine, noeud feuille
  - Etiquette d'un noeud
  - Sous-arbre

Définition formelle et récursive

- En maths : cas particulier de graphe non orienté, connexe et acyclique.
- Définition formelle :
  - Un noeud N unique est un arbre dont N est la racine
  - Si N est un noeud et A_1,A_2,...,A_k sont des arbres dont les racines sont N_1,N_2,..., N_k alors la structure telle que N est le noeud père de N_1,N_2,..., N_k est aussi un arbre.
  - Un arbre sans noeud est un arbre vide
- Forêt : ensemble d'arbres

Propriétés d'un noeud

- Chemin d'un noeud X : c'est la suite de noeuds $N_1, N_2, \ldots, X$ de l'arbre tels que N_i est le père de N_{i+1} et N_1 est la racine de l'arbre.
- Longueur d'un chemin : nombre de liens de parenté dans le chemin (nombre de noeuds du chemin - 1)
- Profondeur d'un noeud X :
  - Longueur du chemin de X
  - Définition récursive :
    - profondeur(racine) = 0;
    - profondeur(X) = 1 + profondeur(pere(X))
- Degré d'un noeud X : nombre de fils de X

Propriétés d'un arbre

- Taille d'un arbre :
  - Nombre de noeuds de l'arbre
  - Définition récursive
    - taille(arbre) = 1 + somme (taille(Sa)) pour tout Sa sous-arbre de la racine de l'arbre.
- Branche :
  - Chemin d'une feuille
  - => Autant de branches que de feuilles
- Hauteur d'un arbre :
  - La plus grande profondeur de l'ensemble des noeuds de l'arbre
  - La longueur de la branche la plus longue
  - Hauteur d'un arbre vide = -1

Propriétés d'un arbre (2)
Degré d'un arbre

- Arbre de degré n : arbre dont tous les noeuds sont au maximum de degré N
  - Cas particulier 1 : arbre binaire (arbre de degré 2)
  - Cas particulier 2 : arbres dégénérés
    - Arbres de degré 1
    - Ce sont en fait des listes
  - Attention : en général, un arbre dit n-aire est un arbre dont le degré n'est pas connu.
- Niveau d'un arbre :
  - ensemble des noeuds de même profondeur
  - Valeur du niveau = valeur de la profondeur des noeuds

Exemples d'utilisation des arbres

- Arbres abstraits :
  - Mettre en évidence la structure d'une expression
  - Utilisé en analyse syntaxique pour la compilation
  - Cas particulier : les expressions arithmétiques
- Arbre lexicographique :
  - classement de mots par ordre alphabétique
  - Exemple : bonjour, bord, bond, boreale, bien
- Hiérarchie d'héritage en JAVA : c'est un arbre à cause de l'absence d'héritage multiple.

Parcours d'arbre

- Opération de base sur les arbres
- Définition
  - Appliquer un même traitement à tous les noeuds d'un arbre
  - Cas particulier de traitement : affichage
- Type de parcours : ordre particulier de visite de noeuds de l'arbre.
  - Parcours en profondeur : préfixe, postfixe, infixe
  - Parcours en largeur

Types de parcours d'arbres

- En largeur : parcours par niveau de l'arbre et de gauche à droite à l'intérieur de chaque niveau
- En profondeur :
  - Exploration récursive par sous-arbres des différents noeuds de l'arbre
  - Selon l'ordre de parcours récursif des sous-arbres, on parle de :
    - Parcours en profondeur de gauche à droite
    - Parcours en profondeur de droite à gauche
  - Selon le moment du traitement d'un noeud père par rapport à ses fils on parle de parcours :
    - Préfixe : le père avant ses fils
    - Postfixe : le père après ses fils
    - Infixe : le père entre chacun de ses fils

Arbres binaires

- Arbres de degré 2
- Vocabulaire spécifique associé :
  - Sous-arbre gauche / droit (sag, sad)
  - Fils gauche / droit
- Spécialisation du calcul récursif de la taille d'un arbre binaire :
  - Taille (arbre) = 1+taille(sag(racine))+taille(sad(racine))
- Un arbre non binaire peut toujours être représenté par un arbre binaire

Arbres binaires : Type Abstrait de Données

TAD ArbreBin
Utilise T, Noeud, Booléens
Opérations
- $\emptyset$: $\rightarrow$ ArbreBin
- racine : ArbreBin $\rightarrow$ Noeud
- sag : ArbreBin $\rightarrow$ ArbreBin
- sad : ArbreBin $\rightarrow$ ArbreBin
- cons : Noeud $\times$ ArbreBin $\times$ ArbreBin $\rightarrow$ ArbreBin
- estVide : ArbreBin $\rightarrow$ Booléen
- info : Noeud $\rightarrow$ T

Préconditions
- racine(A) valide si A $\neq \emptyset$
- sag(A) valide si A $\neq \emptyset$
- sad(A) valide si A $\neq \emptyset$

Axiomes
- $\forall$ rac $\in$ Noeud, $\forall$ sg $\in$ ArbreBin, $\forall$ sd $\in$ ArbreBin
- racine(cons(rac,sg,sd)) = rac
- sag(cons(rac,sg,sd)) = sg
- sad(cons(rac,sg,sd)) = sd
- info(rac) $\in$ T

Fin TAD ArbreBin

Algorithme générique de parcours en profondeur GD d'un arbre binaire

Soit arb ∈ ArbreBin
Soit traiter1, traiter2 et traiter3 des fonctions effectuant une opération sur des données de type T

```
parcourir(arb)
        traiter1(info(racine(arb)));
        si sag ≠ ∅ alors parcourir(sag(arb));
        traiter2(info(racine(arb)));
        si sad ≠ ∅ alors parcourir(sad(arb));
        traiter3(info(racine(arb)));
fin parcourir
```

Algorithme générique d'un parcours en profondeur GD préfixe

Soit arb ∈ ArbreBin
Soit traiter1 unefonction effectuant une opération sur des données de type T

```
parcourir(arb)
        traiter1(info(racine(arb)));
        si sag ≠ ∅ alors parcourir(sag(arb));
        si sad ≠ ∅ alors parcourir(sad(arb));
fin parcourir
```

Algorithme générique d'un parcours en profondeur GD postfixe

Soit arb ∈ ArbreBin
Soit traiter3 une fonction effectuant une opération sur des données de type T

```
parcourir(arb)
        si sag ≠ ∅ alors parcourir(sag(arb));
        si sad ≠ ∅ alors parcourir(sad(arb));
        traiter3(info(racine(arb)));
fin parcourir
```

Algorithme générique d'un parcours en profondeur GD infixe

Soit arb ∈ ArbreBin
Soit traiter2 une fonction effectuant une opération sur des données de type T

```
parcourir(arb)
        si sag ≠ ∅ alors parcourir(sag(arb));
        traiter2(info(racine(arb)));
        si sad ≠ ∅ alors parcourir(sad(arb));
fin parcourir
```

Algorithme générique de parcours en largeur d'un arbre binaire

Soit arb ∈ ArbreBin et prem ∈ ArbreBin
Soit fifo, une file d'attente

```
parcourirLargeur(arb)
        ajouter arb dans fifo
        tant que fifo non vide faire
                prem = prochain élément de fifo
                traiter(racine(prem))
                si sag ≠ ∅ alors ajouter sag(prem) dans fifo
                si sad ≠ ∅ alors ajouter sad(prem) dans fifo
        fin tant que
fin parcourirLargeur
```

Implantations d'un arbre binaire

- Quand espace mémoire critique, possibilité d'implanter avec un tableau simple
- Implantation en JAVA
  - Une classe générique Arbre<T>
    - Un Noeud<T>
    - Un sous-arbre gauche de type Arbre<T>
    - Un sous-arbre droit de type Arbre<T>
  - Une classe générique Noeud<T>
    - Un identifiant de noeud
    - Une valeur de noeud de type T (cf T dans le TAD)

Arbres binaires de recherche

- Arbres binaires ordonnés
  - À rapprocher d'un tableau ou d'une liste triée
  - Offre des performances de maj et de recherche supérieures à celles que l'on obtient pour un tableau où une liste triée.
- Définition : un arbre binaire de recherche est un arbre binaire tel que :
  - Les valeurs des noeuds du sous-arbre gauche de tout noeud X de l'arbre sont inférieures à X
  - Les valeurs des noeuds du sous-arbre droit de tout noeud X de l'arbre sont supérieures à X

Recherche d'un élément dans un arbre binaire de recherche

Soit arb ∈ ArbreBin
Le résultat est le sous-arbre dont l'élément recherché est la racine

```
ArbreBin recherche (T val, ArbreBin arb)
        si arb = ∅ alors resultat = ∅;
        sinon
                si (val < info(racine(arb))) alors
                        resultat = recherche(val,sag(arb))
                sinon si (val > info(racine(arb)))
                        alors resultat = recherche(val,sad(arb))
                        sinon resultat = arb
        retourne resultat;
fin ajout
```

Ajout d'un élément dans un arbre binaire de recherche

Soit arb ∈ ArbreBin et nouv ∈ Noeud
On suppose que l'insertion génère un nouvel arbre

```
ArbreBin ajout (T val, ArbreBin arb)
        nouv = nouveau Noeud de valeur val
        si arb = ∅ alors resultat = cons(nouv,∅,∅);
        sinon
                si (val < info(racine(arb))) alors
                                si (sag(arb) = ∅) alors insérer nouv comme
                                                        sous-arbre gauche de arb
                                sinon ajout(val,sag(arb))
                sinon si (sad(arb) = ∅) alors insérer nouv comme
                                                        sous-arbre droit de arb
                                sinon ajout(val,sad(arb))
fin ajout
```

Suppression d'un élément dans un arbre binaire de recherche

- 3 étapes
  - Trouver l'élément dans l'arbre
  - Supprimer le noeud correspondant
  - Réorganiser l'arbre résultant

Suppression d'un élément dans un arbre binaire de recherche (2)

Soit arb ∈ ArbreBin

Arbre binaire parfait
Arbre binaire complet