TD 11 – arbres n-aires

Structures de données (IF 122)

Arbres binaires stricts

On définit un arbre binaire *strict* (non étiqueté) comme étant soit une feuille, soit composé d'une racine et de deux fils qui sont eux-mêmes des arbres binaires stricts.

Exercice 1 (TD) — *Un peu de math.*

1. Démontrer que tout arbre binaire strict a un nombre impair de nœuds .
2. Quels est, en fonction de n, le nombre d'arêtes d'un arbre binaire à n nœuds ?
3. Même question pour le nombre de feuilles ?

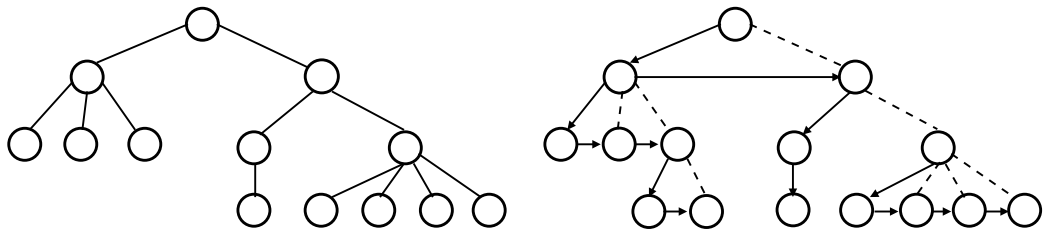
Arbres n-aires

On a travaillé jusqu'à présent avec des arbres binaires et binaires de recherches dans lesquels chaque nœud avait au plus 2 sous-arbres. Nous allons maintenant généraliser la structure pour prendre en compte la possibilité pour les nœuds d'avoir un nombre quelconque de sous-arbres. On appelle de tels arbres des arbres n-aires.

Pour les représenter, on va utiliser le principe suivant :

- Chaque nœud comporte un pointeur vers son premier fils.
- Chaque nœud comporte un pointeur vers son frère cadet.

Comme sur la figure suivante :



Arbre voulu — Implémentation correspondante

Exercice 2 (TD/TP) — *Modélisation* Écrire la classe java correspondante, les constructeurs naturels et les primitives associées.

Exercice 3 (TD/TP) — *Nombre de fils*

1. Écrire une méthode renvoyant le nombre de frères plus jeunes que possède un nœud donné (c'est-à-dire situés à sa droite).
2. Écrire une méthode renvoyant le nombre de fils que possède un nœud donné.

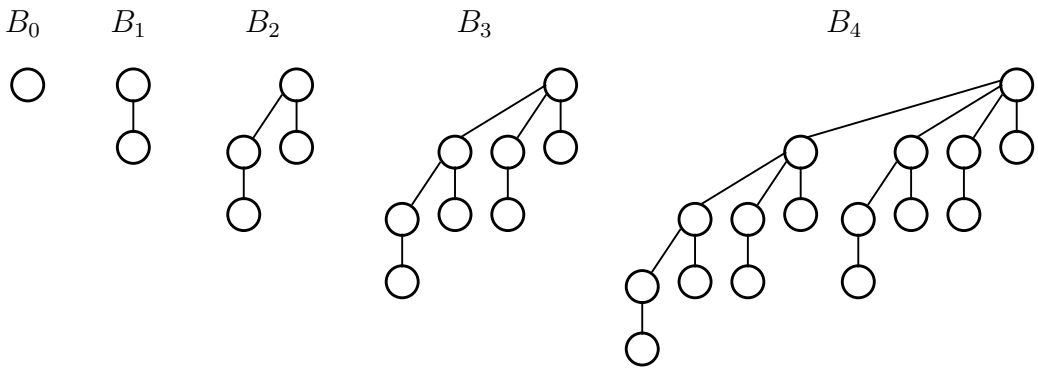
Exercice 4 (TP) — *Hauteur* Écrivez la méthode renvoyant la hauteur d'un arbre.

Exercice 5 (TD) — *Autre modélisation* Imaginez une autre implémentation des arbres n-aires, cette fois en utilisant la structure de liste pour représenter les fils d'un nœud .

Arbres binomiaux

Les arbres binomiaux sont un cas particulier d'arbres n-aire. Il s'agit d'une famille d'arbre B_k définie inductivement par :

- B_0 est un arbre constitué d'un seul nœud .
- B_k est un arbre constitué de deux arbres binomiaux B_{k-1} reliés entre eux. La racine de l'un est fils de la racine de l'autre.



Exercice 6 (TD) — *Propriétés*

1. Combien de fils possède la racine d'un arbre B_k ? Démontrez le.
2. Quelle est la hauteur d'un arbre B_k ? Démontrez le.
3. Montrez qu'un arbre B_k a exactement 2^k nœuds .
4. Montrez qu'il y a exactement C_k^i nœuds à la profondeur i d'un arbre B_k.
   rappel : Les coefficents binomiaux sont donnés par la formule :

$$C_n^p = \begin{cases} 1 & \text{si } n = 0 \text{ ou } p = 0, \\ C_{n-1}^p + C_{n-1}^{p-1} & \text{sinon} \end{cases}$$

   Et une autre propriété des C_n^p utile ici : $C_n^p = C_n^{n-p}$

Exercice 7 (TD/TP) — *Implémentation* Adaptez l'implémentation des arbres n-aires faite précédemment pour manipuler des arbres binomiaux. Pour le constructeur, on s'intéresse ici simplement à la création d'un arbre de type B_0.

Exercice 8 (TD/TP) — *Création d'un arbre B_k* Écrivez un constructeur prenant deux arbres B_k et renvoyant un arbre B_{k+1} (la racine de cet arbre sera celle du deuxième arbre B_k passé en paramètre).

▶ **Exercice 1**

1. Par induction sur la structure de l'arbre
2. aretes = noeuds - 1
3. noeuds = 2 × feuilles - 1

▶ **Exercice 6**

Toute les démos se font par récurrence sur k en utilisant le fait que B_{k+1} est former de deux B_k.

1. k
2. k (si on considère que les arbres B_0 sont de hauteur 0
3. $2^k \times 2^k = 2^{k+1}$
4. Le seul qui soit un peu vicieux. Le cas de B_0 est trivial. Pour le pas de récurrence $(k-1) \to k$, étude par cas en fonction de i :
   - $i < k$ et alors le nombre de nœuds de profondeur i est la somme du nombre de nœuds de profondeur i dans B_{k-1} ajouté au nombre de nœuds de profondeur $i-1$ dans l'autre B_{k-1}. Ce qui donne par récurrence $C_{k-1}^i + C_{k-1}^{i-1}$ c'est à dire C_k^i.
   - $i = k$ est alors dans l'un des B_{k-1}, il n'y a pas de nœuds de profondeur k. Donc c'est à chercher seulement dans l'autre, ce qui donne $C_{k-1}^{k-1} = C_{k-1}^0 = 1 = C_k^0 = C_k^k$.