LYCÉE LECONTE DE LISLE

Arbres et tas binomiaux

Vincent Picard

Arbres binomiaux

Attention à ne pas confondre avec les arbres binaires.

Un **arbre binomial** d'ordre k est :

- pour $k = 0$: une feuille (ou nœud externe)
- pour $k > 0$: un nœud possédant k fils qui sont des arbres binomiaux d'ordres $k-1, k-2, \ldots, 0$ (dans cet ordre)

Arbres binomiaux : questions

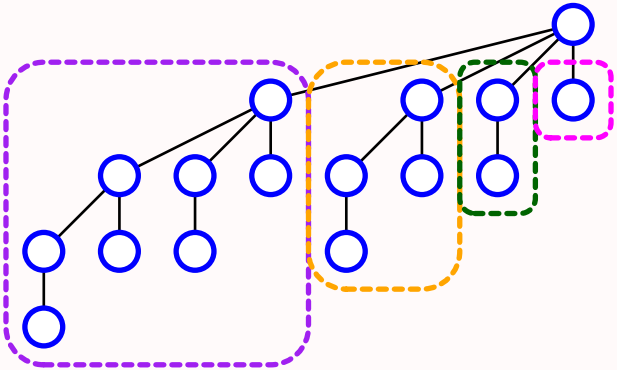
1. Dessiner les arbres binomiaux d'ordres 1, 2, 3, 4.

2. Montrer qu'un arbre binomial d'ordre k est de hauteur k et possède 2^k nœuds.

3. Montrer qu'un arbre binomial d'ordre k possède $\binom{k}{p}$ nœuds à profondeur p. **Indication :** remarquer qu'un arbre binomial d'ordre k s'obtient en ajoutant à un arbre binomial d'ordre $k-1$ un fils gauche qui est aussi un arbre binomial d'ordre $k-1$.

Cette dernière propriété justifie le nom d'arbre binomial.

Un arbre binomial



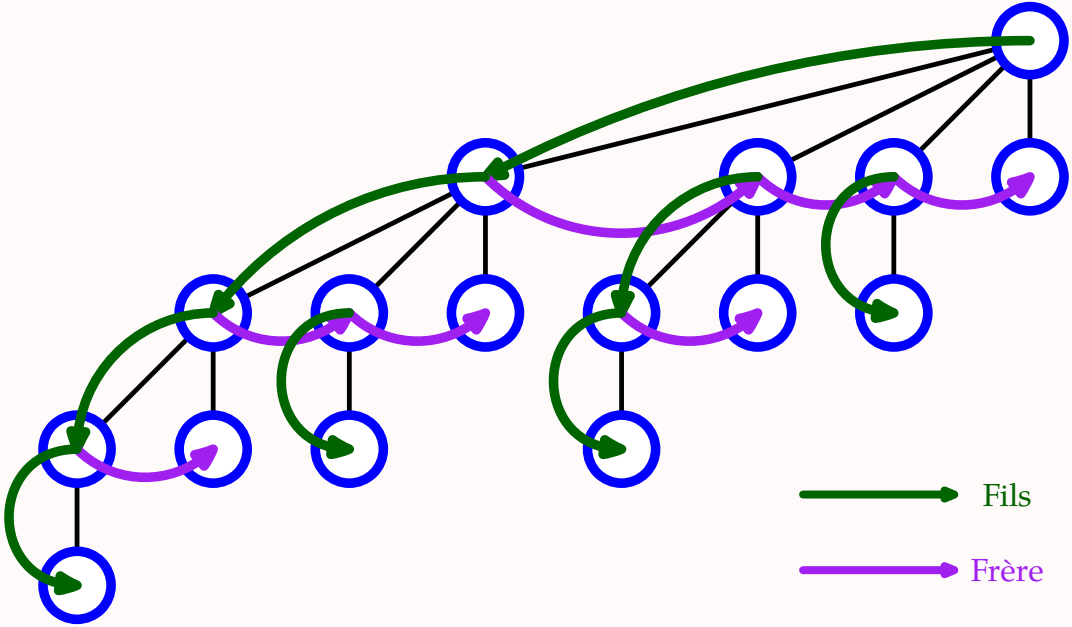
Un arbre binomial d'ordre 4 ($n = 16$)

Arbre binomiaux : implémentation en C

On code les nœuds d'un arbre binomial avec la structure :

```
struct noeud_s {
    int val; /* Etiquette du noeud */
    int ordre; /* Ordre du sous-arbre */
    /* Lien vers son fils de plus grand ordre */
    struct noeud_s *fils; // NULL si aucun fils
    /* Lien vers son petit frère (ordre juste en dessous) */
    struct noeud_s *frere; // NULL si pas de frère
};
typedef struct noeud_s Noeud;
```

Un arbre binomial en C : avec les pointeurs



Arbres binomiaux : exercices en C

Écrire les fonctions C suivantes :

1. `int hauteur(Noeud *arbre)` : calcule la hauteur d'un arbre binomial (sans utiliser son ordre)

2. `int taille(Noeud *arbre)` : calcule la taille d'un arbre binomial (sans utiliser son ordre, ni la question précédente)

3. `bool est_tasmin(Noeud *arbre)` : vérifie si un arbre binomial vérifie la propriété de tas min

Arbres binomiaux : fusion

Les arbres binomiaux possèdent la bonne propriété de pouvoir être facilement fusionnés :

Soit A_1 et A_2 deux arbres binomiaux d'ordre k.

1. On ajoute A_2 comme premier fils (le plus à gauche) de A_1, quel est le résultat ?
2. Implémenter en C la fonction :

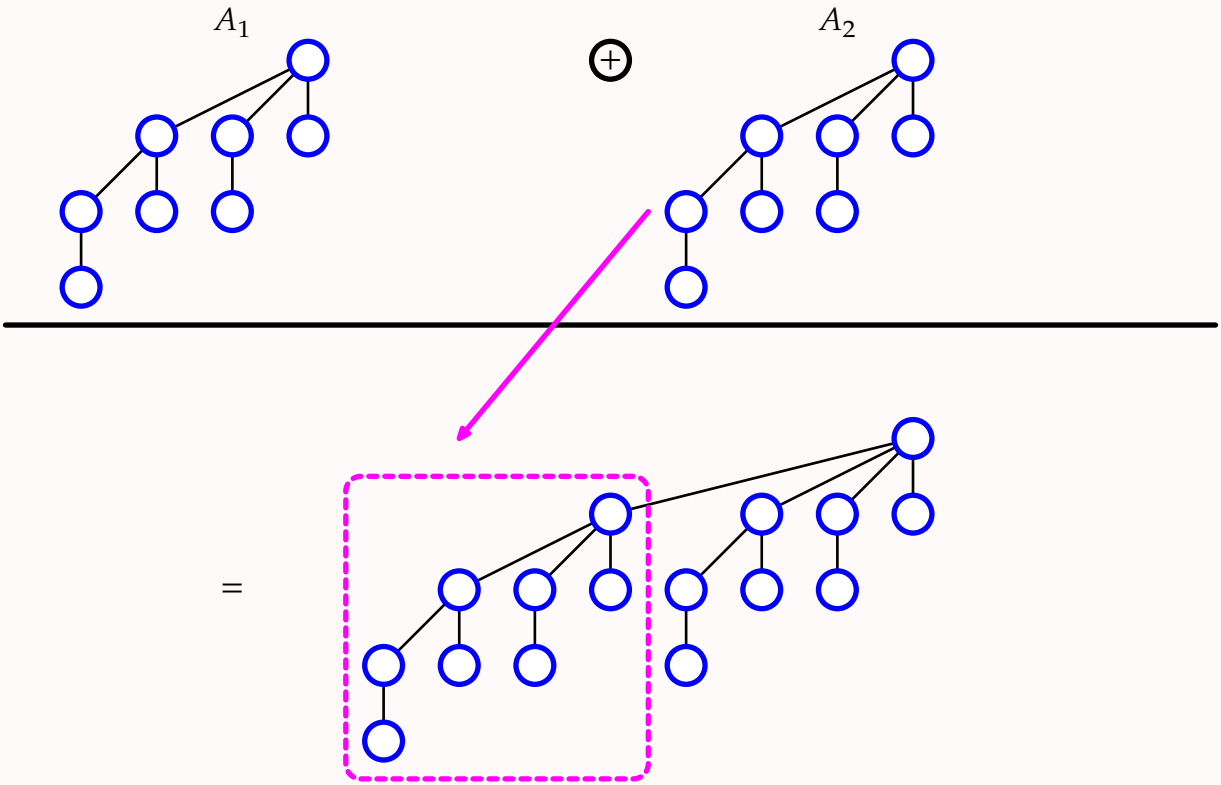
   ```
   void fusion(Noeud *a1, Noeud *a2)
   ```

   qui réalise cette opération en modifiant seulement les pointeurs dans les deux arbres
3. Quelle est sa complexité ?
4. Implémenter la fonction

   ```
   Noeud* fusion_tas(Noeud *a1, Noeud *a2)
   ```

   qui réalise la même chose que `fusion` mais qui suppose que `a1` et `a2` vérifient la propriété de tas min. Le résultat doit aussi vérifier la propriété de tas min, on retourne la racine.

La fusion illustrée



To be concluded

Dans le prochain épisode :

on réalise un tas min avec une forêt d'arbres binomiaux...