Licence M/I

Algorithmique: TD 5

24 octobre 2005

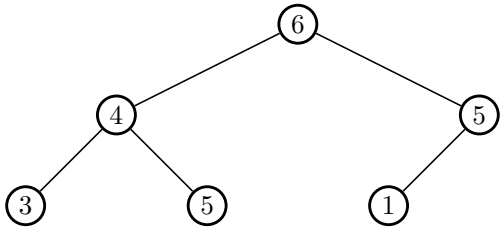
Exercice 1: Proposer un implémentation de la structure de données *file de priorité* ayant l'interface suivante :

```
interface PriorityFile{

    boolean isEmpty(); // teste si la pile est vide
    Comparable peek(); // retourne la valeur de la priorite maximale
    Object pop();      // retourne l'objet de priorite maximale
    Ticket push(Object k, Comparable p);
                       /** insere l'objet k de priorite p
                         * et retourne un Ticket correspondant */
    Ticket[] initFile(Object[] tk, Comparable[] tp);
                       /** initialise la file de priorite avec
                         * les objets tk[i] de priorite tp[i]
                         * retourne un tableau de Ticket */
    void update(Ticket a,Comparable p);
                       // Met a jour la priorite de l'objet de Ticket a
    String toString(); // convertit la file en chaine de caracteres
}
```

Exercice 2: Tas de Fibonacci. Un tas est un arbre binaire complet parfaitement équilibré, ce qui signifie que si l'on regarde toutes les feuilles de la gauche vers la droite, les premières sont toutes à une certaine profondeur p et les suivantes à la profondeur $p-1$.
a) Dessiner le tas de taille 10.

En fait, chaque nœud d'un tas contient un nombre ; étant donné la forme très particulière d'un tas, on peut décrire le décrire entièrement en donnant la liste des valeurs contenues dans ses nœuds profondeur par profondeur.
b) Dessiner le tas $[7; 3; 9; 5; 6; 4; 6; 3; 7]$.
c) Donner un tableau correspondant au tas suivant :



Un tas-max est un tas tel que la valeur de tout nœud est supérieure à la valeur de ses fils éventuels.
d) Dans un tas-max, où se trouve la valeur maximale ?

Pour organiser un tas-max, on dispose de deux opérations de “percolation”, l'une de haut en bas (top-down), l'autre de bas en haut (bottom-up). Lors de la percolation top-down, on part d'un nœud et, s'il est plus petit qu'un de ses fils, on échange le plus grand des fils avec le nœud. On continue jusquà ce que les fils du nœud soient plus petits ou que l'on arrive dans une feuille. Lors de la percolation bottom-up, on part d'un nœud et, s'il est plus grand que son pere, on echange, et on recommence tant que le père est plus petit. Pour transformer un tas en tas-max (*tamiser*), on agit récursivement : on transforme les sous-arbres gauche et droit en tas-max, puis on fait une percolation top-down de la racine.
e) Tamiser le tas [1 ;2 ;3 ;4 ;5 ;6] ; quelle est la complexité de cette opération ?
f) Comment insérer ou supprimer une valeur dans un tas ? Créer un tas-max en insérant successivement les valeurs 1,2,3,4,5 et 6.

g) Proposer une implémentation pour les tas.
h) Proposer un algorithme de tri par tas. Quel est sa complexité ?
i) Supposons que l'on implémente une file de priorité ‘ l'aide d'un tas ; quelle serait la complexité des opérations sur la file de priorité ?
j) On veut traiter une (longue) liste d'objets et conserver parmi ceux-ci les k objets de priorité maximale. Proposer un algorithme.