TD 5

Les tas, les files et les *pâtons*

Exercice 1 (Tas, septembre 2007)**.**
Dans un tas max où se trouve l'élément maximum ? Où peut-on trouver l'élément minimum ? (Facile)

1 pt
6 min

Exercice 2 (Insertion / suppression tas, septembre 2007)**.**
Former un tas max en insérant successivement et dans cet ordre les éléments : $10, 7, 3, 9, 11, 5, 6, 4, 8$ (répondre en représentant le tas obtenu). Supprimer l'élément maximum (répondre en représentant le nouveau tas).

1,5 pt
9 min

Exercice 3 (Insertion / suppression)**.**
On se donne le tas max (ou maximier) de la figure 1. En utilisant les algorithmes vus en cours :

1. Insérer un élément de clé 17 dans ce tas. Répondre en représentant le nouveau tas.

0.5 pt
4 min

2. En repartant du tas initial, retirer l'élément de clé maximale. Répondre en représentant le nouveau tas.

0.5 pt
4 min

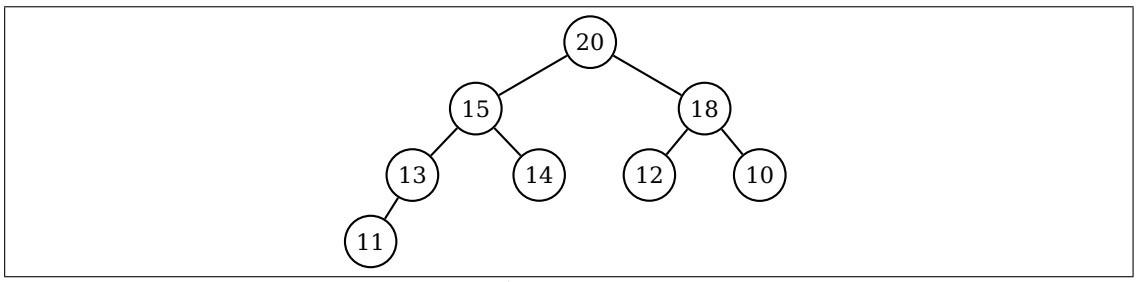

Figure 1: Tas

Exercice 4 (Indexation)**.**
Dans le cours nous avons vu qu'un tas max est un arbre quasi-parfait ayant de plus la propriété de dominance des tas : le parent est toujours plus grand que ses fils. Nous avons également vu qu'un arbre quasi-parfait est efficacement représenté par un tableau en mettant la racine à la première case.

1. Représenter sous forme d'arbre quasi-parfait le tableau suivant :

20	19	10	7	15	12	3	6	5	1

   Est-ce un tas ?

pour mémoriser un tas d'entiers, on se donne un tableau d'entiers `tab` suffisamment grand (on suppose que l'allocation mémoire initiale du tableau est suffisante pour l'usage qui en sera fait) et un entier `taille` qui correspondra au nombre d'éléments du tas actuellement stockés dans le tableau.

Un nœud du tas sera représenté par un indice du tableau. Étant donné un nœud i nous voulons pouvoir obtenir son parent à l'indice `parent(i)`, son fils gauche à l'indice `gauche(i)` et son fils droit à l'indice `droite(i)`.

2. En vous aidant de l'exemple, proposer des fonctions pour le calcul de ces indices.

Pour l'insertion d'un élément et la suppression de l'élément maximum nous avons vu en cours qu'il faut faire appel respectivement à deux procédures `maintien_haut()` et `maintien_bas()`. Elles prenent en argument le tas (le tableau et la taille) et l'indice à partir duquel effectuer le maintien.

3. Écrire ces procédures `maintien_haut()` et `maintien_bas()` de manière récursive.
4. Écrire une fonction `estvide()` qui prend le tas (tableau et taille) en argument et renvoie vrai (1) si le tas est vide et faux (0) sinon.
5. Écrire la fonction `maximum()` qui renvoie l'élément (ici un entier) à la racine du tas.
6. Écrire la procédure `retirer_maximum()` qui retire l'élément maximum du tas (et reforme un tas) ainsi que la procédure `inserer()` qui insére un élément (un entier `e` passé en argument) dans le tas.
7. Proposer un algorithme de tri utilisant les tas (on triera un tableau t de taille n passé en argument en utilisant un tas stocké dans un tableau de $N > n$ cases).

Exercice 5 (Tri par tas)**.**
Simuler étape par étape l'exécution d'un tri par tas sur le tableau $12, 15, 26, 30, 10, 80, 29, 31, 23, 45$ en représentant les tas successifs obtenus :

1. en supposant que le tri consiste en (i) la formation du tas par ajouts successifs des éléments du tableau dans le tas (ii) puis à leurs retraits.
2. Puis en supposant que pour former le tas, on utilise plutôt l'algorithme qui consiste en rétablir la propriété de domination, à partir de l'arbre quasi-parfait correspondant au tableau, par application de `maintienBas` sur chacun des nœuds internes, successivement, par indices décroissants (du nœud interne d'indice `parent(N-1)` à la racine, d'indice 0).

Exercice 6 (Septembre 2007)**.**
On se donne un tableau T de p piles d'entiers. Autrement dit les éléments de T sont des piles et chaque pile contient des entiers. On suppose que T est tel que :

4 pt
24 min

- Aucune pile n'est vide et il y a en tout N entiers répartis dans les piles (donc $p \leq N$)
- les sommets des piles vont décroissants :

$$\mathsf{Sommet}(T[0]) \geq \mathsf{Sommet}(T[1]) \geq \ldots \geq \mathsf{Sommet}(T[p-1])$$

- Dans chaque pile les entiers sont ordonnés du plus grand (au sommet) au plus petit.

On souhaite réaliser l'interclassement des éléments des p piles de manière à obtenir un nouveau tableau contenant les N éléments dans l'ordre croissant.

Pouvez vous donner un algorithme en $O(N \log N)$ comparaisons pour ce problème ? (Justifier) Vous pouvez utiliser des algorithmes vus en cours et les résultats de complexité sur ces algorithmes.

Indication : le tableau T est un tas max dont les éléments sont des piles et où les clés sont les entiers au sommet des piles.

Corrigé

Correction de l'exercice 1.

Du fait de la propriété de domination, la racine contient l'élément maximum. De même, il est impossible que l'élément minimum soit ailleurs que dans l'une des feuilles (sinon ses descendants seraient plus petits) mais cela peut être n'importe quelle feuille.

Correction de l'exercice 2.

Le tas obtenu par insertion successives est donné figure 2, et le tas obtenu en supprimant l'élément maximum (11) est donné figure 3.

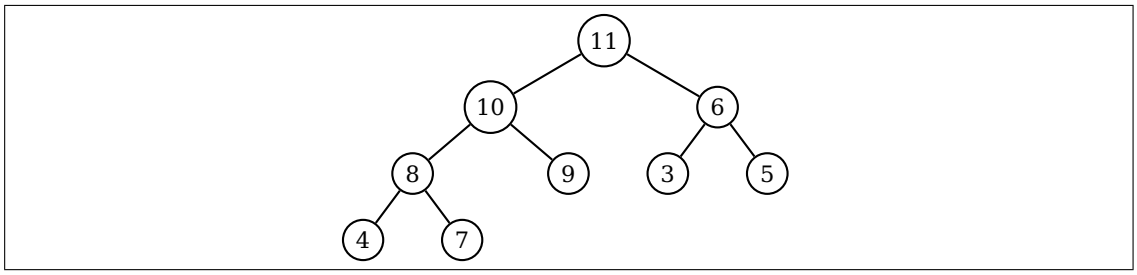

Figure 2: Tas : insertion de tous les éléments

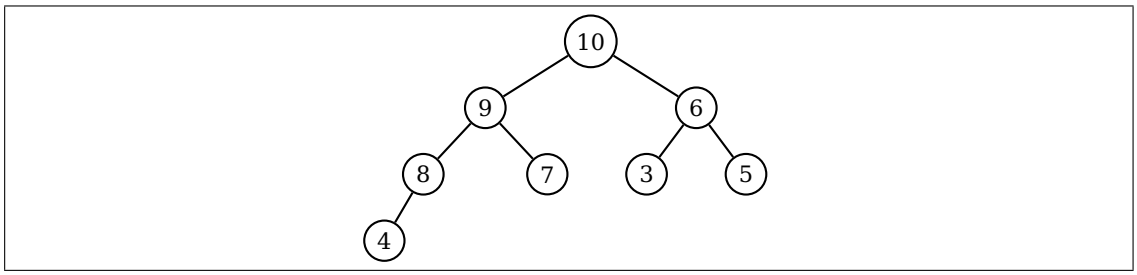

Figure 3: Tas : suppression de la racine (11)

Correction de l'exercice 3.

Pas de correction.

Correction de l'exercice 4.

C'est un exercice d'apprentissage du cours. Les solutions sont toutes données au chapitre tas du poly. Pour la première question : l'arbre quasi-parfait obtenu n'est pas un tas, car il ne vérifie pas la propriété de domination ($12 \not\leq 10$).

Correction de l'exercice 5.

1. On ne représente que le tas intermédiaire obtenu (fig. 4).
2. On ne représente que le tas intermédiaire obtenu (fig. 5).

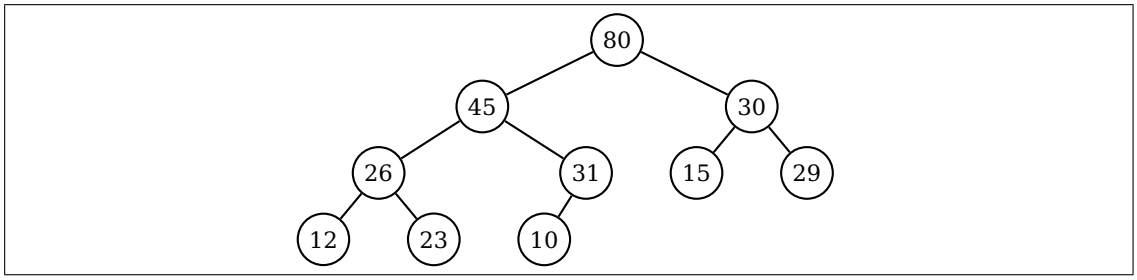

Figure 4: Le tas obtenu après tous les ajouts

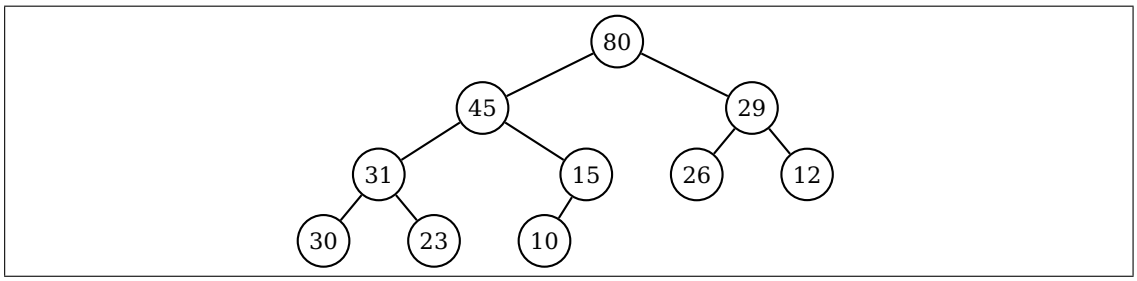

Figure 5: Le tas obtenu par le second algorithme

Correction de l'exercice 6.

L'idée est de former un tas de piles à partir de ce tableau en considérant que pour comparer deux piles on compare uniquement leurs sommets. Le tableau vu comme un arbre quasi-parfait est déjà dans le bon ordre pour la propriété de domination. On retire les éléments un à un en les dépilant de la pile en $T[0]$. Après chaque retrait :

- Si la pile en $T[0]$ est vide, on remplace $T[0]$ par la dernière pile $T[p-1]$, et on décrémente p.
- On rétablit la propriété de domination par maintien bas à partir de $T[0]$.

On effectue N retrait. Chacun de ces retraits prend au maximum un temps $c + \log p$, où c est une constante et où $\log p$ est la hauteur du tas (coût maximum en temps de la phase de maintien). Comme $p \leq N$, $\log p$ est majoré par $\log N$. Ainsi le temps mis par cet algorithme est majoré par $cN + N \log N$ où c est une constante, ce qui est bien en $O(N \log N)$.