Polytech'Lyon
Algo. Prog. 3A INFO
TD/TP : Tas binaires

Partie 1

Un arbre binaire est une structure de données qui peut se représenter sous la forme d'une hiérarchie dont chaque élément est appelé nœud, le nœud initial étant appelé racine. Dans un arbre binaire, chaque élément possède au plus deux éléments fils au niveau inférieur, habituellement appelés gauche et droit. Du point de vue de ces éléments fils, l'élément dont ils sont issus au niveau supérieur est appelé père. Au niveau le plus élevé il y a donc un nœud racine. Au niveau directement inférieur, il y a au plus deux nœuds fils. En continuant à descendre aux niveaux inférieurs, on peut en avoir quatre, puis huit, seize, etc. c'est-à-dire la suite des puissances de deux. Un nœud n'ayant aucun fils est appelé feuille. Le nombre de niveaux total, autrement dit la distance entre la feuille la plus éloignée et la racine, est appelé hauteur de l'arbre.
Afin d'éviter les structures chainées, proposez une méthode pour implémenter un arbre binaire en utilisant un tableau dynamique comme illustré dans la figure ci-dessous.

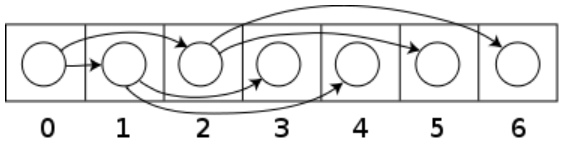

Fig 1. Un arbre binaire plongé dans un tableau

Partie 2

Un tas binaire est une structure de données en informatique utilisée notamment pour implémenter les files de priorité car elle permet d'accéder au maximum (rsp. minimum) d'un ensemble en temps constant. On peut la représenter par un arbre binaire qui vérifie deux contraintes :

- c'est un arbre binaire parfait : tous les niveaux excepté le dernier doivent être totalement remplis et si le dernier n'est pas totalement remplis alors il doit être rempli de gauche à droite
- c'est un tas : la clé de chaque nœud doit être supérieure ou égale (rsp. inférieure ou égale) aux étiquettes de chacun de ses fils.

L'ajout d'un élément *x* dans le tas se fait comme suit : On insère *x* à la prochaine position libre *i.e.* la position libre la plus à gauche possible sur le dernier niveau, puis tant que *x* n'est pas la racine de l'arbre et que *x* est strictement supérieur à son père on échange les positions entre *x* et son père. Comme illustré dans l'exemple suivant:

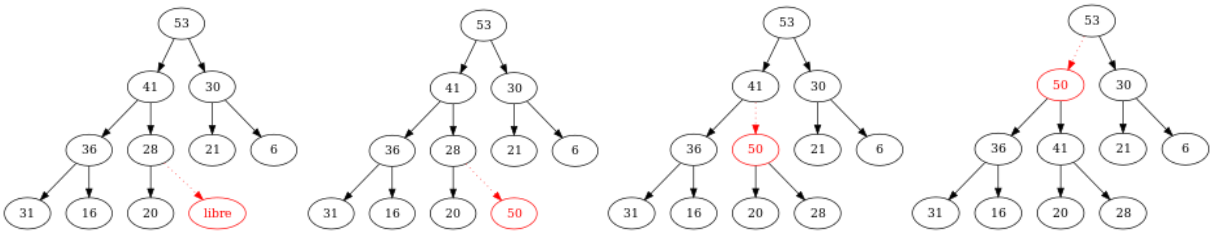

Fig 2. Ajout d'un élément dans un tas binaire.

Pour supprimer un élément *y* du tas, on met à sa place le nœud qui était en dernière position de l'arbre binaire (donc le nœud le plus à droite sur le dernier niveau) que l'on notera *x*. Puis, tant que *x* a des fils et que *x* est strictement inférieur à un de ses fils, on échange les positions entre *x* et le plus grand de ses fils.
Proposez des implémentations des opérations ajout et suppression en vous basant sur la structure définie dans la Partie 1 et prouvez que leurs complexités respectives sont de l'ordre O(log n).