TD8: Les tas

Exercice 1 – Rappels sur les tas

Q 1.1 Donner la définition de la structure de données abstraite appelée "tas".

Q 1.2 En utilisant uniquement l'opération `swap(pere, fils)`, construire le tas obtenu par insertion successives des entiers $[10, 2, 5, 4, 7, 15, 1, 3]$, la valeur de ces entiers servant directement de clé.

Q 1.3 Combien d'éléments peut-on avoir dans un tas de hauteur h (en nombre d'arcs) ?

Q 1.4 Étant donné n éléments à stocker dans un tas, écrire h, la hauteur du tas, en fonction de n.

Q 1.5 Quelle est la complexité-temps pire des cas d'insertion d'un élément dans un tas contenant n éléments ?

Exercice 2 – Tas ternaire

Afin de réduire la hauteur de l'arbre sous-jacent, on propose d'utiliser un tas ternaire à la place d'un tas binaire. Dans ce type de tas, chaque noeud possède au plus trois fils. La valeur de chaque noeud est inférieure à celle de ses fils.

Q 2.1 Représentez le tas ternaire construit à partir des éléments $[3, 6, 1, 13, 17, 18, 2]$.

Q 2.2 Étant donné n éléments à stocker dans un tas ternaire, écrire la hauteur h du tas en fonction de n. Est-ce que l'utilisation d'un tas ternaire améliore le coût d'insertion dans le pire des cas par rapport à un tas binaire ?

Q 2.3 En numérotant à partir de 1 les nœuds de l'arbre du haut vers le bas, niveau après niveau, et de la gauche vers la droite, quelles relations peut-on établir entre le numéro d'un nœud et ceux de ses trois fils ? Sachant que l'arbre est complet, décrire comment stocker un tas ternaire dans un tableau et écrire les fonctions nécessaires à sa gestion.

Q 2.4 En utilisant les relations définies précédemment, écrire une fonction qui supprime et retourne le plus petit élément d'un tas ternaire contenant des entiers.
Note Afin de maintenir une structure d'arbre ternaire tassé à gauche, il faut remplacer la racine par la feuille qui se trouve à l'extrémité droite.

Q 2.5 Quelle est la complexité-temps pire cas de la suppression d'un élément dans un tas ternaire contenant n éléments.

Exercice 3 – Tas contenant des éléments à deux clefs

On considère des couples (f, i) où f est un nombre réel et i est un entier compris entre 0 et K, où K est un nombre entier connu à l'avance. Ces couples doivent être stockés dans une structure de données telle que, pour une valeur $i \in \{1, ..., K\}$, la structure contienne au plus un élément (f, i). De plus, on veut que cette structure de données possède les primitives suivantes :

— un accès en $O(1)$ au couple (f, i) ayant la plus petite valeur f.

— insertion en $O(log(n))$ d'un couple (f, i).

— suppression en $O(log(n))$ de l'élément de plus petite valeur f.

— test en $O(1)$ de la présence ou non d'un élément (f, i) pour une valeur i donnée.

— suppression en $O(log(n))$ de l'élément (f, i) pour une valeur i donnée (s'il existe).

Q 3.1 Proposer une structure de données correspondant à cette description, ainsi que le code de ses primitives.

Q 3.2 Indiquer comment adapter ce code pour une structure de tas contenant des élements (f, c) où c est une valeur prise dans un ensemble quelconque fini $\mathcal{K}$. Indiquer la complexité des primitives de cette structure de données et discuter de la complexité moyenne.