Tas : principes et applications

1 Structure de tas

1. Rappeler les fonctions de manipulation d'un tas, avec les paramètres d'entrée et de sortie.
2. Est-ce que le tableau $T = [15, 9, 7, 4, 5, 6, 2, 1, 3, 0]$ représente un tas ? Si oui, donner sa représentation arborescente.
3. Simuler l'insertion de la clé 10 dans ce tas.
4. Simuler la suppression de la clé maximum.
5. Ecrire le pseudo-code de la procédure $AugmenterCle$.
6. Ecrire le pseudo-code de la procédure $InsererCle$.
7. Ecrire le pseudo-code du $TriParTas$.
8. Montrer que l'algorithme $ConstruireTasMax$ vu en cours est correct. On supposera que la procédure $EntasserMax$ est correcte.

2 Hauteur du tas

1. Donner le nombre de nœuds d'un arbre binaire complet de hauteur h.
2. Montrer qu'un arbre binaire complet avec n nœuds est de hauteur au plus $\log n$.
3. Montrer qu'un tas avec n clés est de hauteur au plus $\log n$.

3 Application des tas à l'algorithme de Dijkstra

Rappelons que pour une bonne efficacité, l'algorithme de Dijkstra est implanté en utilisant une structure de *tas*, mais ce sera un *tas min*. On notera n le nombre de sommets, m le nombre d'arêtes du graphe en entrée et F le tas utilisé.

1. Indiquer le nombre d'opérations de type $AjouterCle$, $ExtraireMin$, $DiminuerCle$ qui sont faites sur F pendant l'exécution de l'algorithme.
2. Pourquoi est-on sûr qu'à chaque modification d'une clé, cette clé diminue ?
3. Détailler la structure de données pour que l'algorithme fonctionne efficacement. En particulier, que doit-on garder dans le tas en plus des clés, et comment faire pour que les sommets du graphe accèdent en temps constant aux éléments du tas ?
4. En déduire la complexité de l'algorithme de Dijkstra implanté avec un tas.
5. Il existe une structure de donnée appelée *tas de Fibonacci*, qui fournit les mêmes services que les tas habituel, mais avec l'avantage que la fonction $DiminuerCle$ s'exécute en temps (amorti) $O(1)$. Quelle est la complexité de l'algorithme de Dijkstra implanté avec un tas de Fibonacci ?