Université Assane Seck de Ziguinchor
UFR sciences et Technologies
Département Informatique

Exercices TD (Feuille 5)

Consignes : Les exercices mis en évidence sont obligatoires pour valider le chapitre 5. Les autres exercices sont optionnels. Il vous est conseillé de les faire après les exercices obligatoires.

6.1.3 Montrer que pour un sous-arbre quelconque d'un tas max, la racine du sous-arbre contient la plus grande valeur parmi celles de ce sous-arbre.

6.1.4 Ou pourrait se trouver le plus petit élément d'un tas max, en supposant que tous les éléments sont distincts ?

6.1.5 Un tableau trié forme-t-il un tas min ?

6.1.6 La séquence « 23, 17, 14, 6, 13, 10, 1, 5, 7, 12 » forme-t-elle un tas max ?

6.1.7 Montrer que, si l'on emploie la représentation tableau pour un tas à n éléments, les feuilles sont les noeuds indexés par $n/2 + 1, n/2 + 2, \ldots, n$.

6.2.1 En prenant comme modèle la figure 6.2, illustrer l'action de ENTASSER-MAX(A, 3) sur le tableau A = « 27, 17, 3, 16, 13, 10, 1, 5, 7, 12, 4, 8, 9, 0 ».

6.2.2 En s'inspirant de la procédure ENTASSER-MAX, écrire du pseudo code pour la procédure ENTASSER-MIN(A, i) qui fait la même chose mais pour un tas min. Comparer le temps d'exécution de ENTASSER-MIN et celui de ENTASSER-MAX ?

6.2.3 Quel est l'effet d'un appel ENTASSER-MAX(A, i) quand l'élément $A[i]$ est plus grand que ses enfants ?

6.2.4 Quel est l'effet d'un appel ENTASSER-MAX(A, i) pour $i > taille[A]/2$?

6.2.5 Le code de ENTASSER-MAX est assez efficace en termes de facteurs constants, sauf peut-être pour l'appel récursif en ligne 10 qui risque d'entraîner certains compilateurs à générer

du code inefficace. Écrire une procédure ENTASSER-MAX efficace qui utilise une structure de contrôle itérative (boucle) au lieu d'un appel récursif.

6.2.6 Montrer que le temps d'exécution dans le pire des cas de ENTASSER-MAX sur un tas de taille n est $\boldsymbol{\Omega}$ (lg n). (*conseil :* Pour un tas à n noeuds, donner des valeurs de nœuds qui provoquent l'appel récursif de ENTASSER-MAX en chaque noeud d'un chemin reliant la racine à une feuille.)

6.3.1 En prenant modèle sur la figure 6.3, illustrer l'action de CONSTRUIRE-TAS-MAX sur le tableau A = « 5, 3, 17, 10, 84, 19, 6, 22, 9 ».

6.3.2 Pourquoi fait-on décroître l'indice de boucle i de la ligne 2 de CONSTRUIRE-TASMAX depuis *longueur*[A]/2 jusqu'à 1, au lieu de le faire croître de 1 à *longueur*[A]/2 ?

6.3.3 _ Montrer qu'il existe au plus _$n/2h+1$_ noeuds de hauteur h dans un tas quelconque à n éléments.

6.4.1 En s'aidant de la figure 6.4, illustrer l'action de TRI-PAR-TAS sur le tableau A = « 5, 13, 2, 25, 7, 17, 20, 8, 4 ».

6.4.2 Prouver la conformité de TRI-PAR-TAS à l'aide de l'invariant de boucle suivant : Au début de chaque itération de la boucle **pour** des lignes 2–5, le sous-tableau $A[1\,..\,i]$ est un tas max contenant les i plus petits éléments de $A[1\,..\,n]$, et le sous-tableau $A[i+1\,..\,n]$ contient les $n - i$ plus grands éléments de $A[1\,..\,n]$, triés.

6.4.3 Quel est le temps d'exécution du tri par tas sur un tableau A de longueur n déjà trié en ordre croissant ? Et en ordre décroissant ?

6.4.4 Montrer que le temps d'exécution du tri par tas dans le cas le plus défavorable est $\boldsymbol{\Omega}$ (n lg n).

6.4.5 _ Montrer que, quand tous les éléments sont distincts, le temps d'exécution optimal du tri par tas est **Ω** (*n* lg *n*).

6.5.1 Illustrer le fonctionnement de EXTRAIRE-MAX-TAS sur le tas *A* = « 15, 13, 9, 5, 12, 8, 7, 4, 0, 6, 2, 1 ».

6.5.2 Illustrer le fonctionnement de INSÉRER-TAS-MAX(*A*, 10) sur le tas *A* = « 15, 13, 9, 5, 12, 8, 7, 4, 0, 6, 2, 1 ». Utiliser le tas de la figure 6.5 comme modèle pour l'appel AUGMENTERCLÉ-TAS.

6.5.3 Écrire le pseudo code des procédures MINIMUM-TAS, EXTRAIRE-MIN-TAS, DIMINUER-CLÉ-TAS et INSÉRER-TAS-MIN qui implémentent une file de priorité min basée sur un tas min.

6.5.4 Pourquoi faut-il régler la clé du noeud inséré sur la valeur−∞, en ligne 2 de INSÉRERTAS-MAX, puisque l'étape immédiatement suivante sera de donner à la clé la valeur souhaitée ?

6.5.5 Prouver la conformité de AUGMENTER-CLÉ-TAS à l'aide de l'invariant de boucle suivant :

Au début de chaque itération de la boucle **tant que** des lignes 4–6, le tableau *A*[1 . . *taille*[*A*]] satisfait à la propriété de tas max, à une infraction potentielle près : *A*[*i*] risque d'être plus grand que *A*[PARENT(*i*)].

6.5.6 Montrer comment implémenter une file FIFO (first-in first-out) avec une file de priorité. Montrer comment implémenter une pile avec une file de priorité. (Files et piles sont définies à la section 10.1.)

6.5.7 L'opération SUPPRIMER-TAS(*A, i*) supprime dans le tas *A* l'élément placé dans le noeud *i*. Donner une implémentation de SUPPRIMER-TAS qui tourne en un temps $O(\lg n)$ pour un tas max à *n* éléments.

6.5.8 Donner un algorithme à temps $O(n \lg k)$ qui fusionne *k* listes triées pour produire une liste triée unique, *n* étant ici le nombre total d'éléments toutes listes confondues. (*Conseil :* Utiliser un tas min pour la fusion multiple.)