L3-INFO Algorithmique

UFR IM²AG
UNIVERSITÉ Grenoble Alpes

Quick 2 – 8 novembre 2019 – durée 1 h

Les documents et les téléphones (incluant smartphone, tablettes,... tout ce qui contient une interface réseau) interdits.
Seuls les dictionnaires pour les personnes de langue étrangère sont autorisés.
Le barème est indicatif.

Comme d'habitude, vous êtes vivement encouragés à illustrer vos recherches et vos réponses par des schémas.

Rappels On manipule les arbres binaires à l'aide des primitives suivantes :

- Pour un arbre non vide A, les appels de fonctions $G(A)$, $D(A)$ et $elem(A)$ renvoient respectivement le fils gauche, le fils droit et l'élément à la racine de A.
- On dispose d'une fonction $ArbreVide$ sans argument, permettant de créer un arbre vide.
- On dispose d'une fonction $Noeud$ prenant en argument deux arbres et un élément et permettant de créer un nouveau nœud (qui est donc la racine d'un nouvel arbre).

1 Inversion dans un tableau trié (10 points)

Soit un tableau T indexé de 1 à n trié en ordre croissant. On choisit deux élements (distincts) de ce tableau et on les échange.

L'objectif de cet exercice est de retrouver les deux éléments échangés pour pouvoir les remettre à leur place.

Question 1 :

Quel serait le coût d'un algorithme consistant simplement à trier à nouveau le tableau ? Justifier votre réponse.

Question 2 :

(a) Quel critère permet de déterminer l'indice du premier élément échangé ? Illustrer votre réponse par un schéma ou un exemple.
(b) Quel critère permet de reconnaître le second élément échangé ?
(c) Attention, un cas particulier peut se présenter. Expliquer lequel.
(d) Écrire un algorithme qui retrouve ces éléments et les remet en place, en ne parcourant qu'une seule fois le tableau.

Question 3 :

Plaçons-nous maintenant dans la situation où deux éléments ont été intervertis dans un arbre binaire de recherche : expliquez dans les grandes lignes comment il serait possible de remettre ces éléments à leur place.

Il n'est pas demandé ici d'écrire un algorithme complet.

2 À propos de l'algorithme de Huffman (10 points)

On rappelle ci-dessous le principe de l'algorithme de Huffman :

```
F = FàPVide()
pour chacun des K symboles s de l'alphabet faire
    z = Nœud ( ArbreVide (), s, ArbreVide () )
    Insérer dans F le nœud z avec pour priorité la fréquence de s
tant que il reste au moins 2 nœuds dans F faire
    x = Extraire (F)
    y = Extraire (F)
    z = Nœud ( x, NULL, y )  // Rappel : pour les nœuds internes
                                l'étiquette n'a aucune importance
    Insérer dans F le nœud z avec pour priorité la somme des priorités de x et de y
renvoyer Extraire (F)
```

On ne cherche pas à en démontrer la correction intégralement, on se contentera pour cet exercice des deux critères suivants.

Question 4 :

On veut démontrer la propriété suivante : « *le nœud racine de l'arbre renvoyé par l'algorithme de Huffman possède une priorité égale à 1* ».

(a) Énoncer un invariant de l'algorithme qui permettra de démontrer cette affirmation.

(b) Justifier que cet invariant est vérifié avant d'entrer dans la boucle **tant que**.

(c) Justifier que l'invariant est maintenu par chaque itération de la boucle **tant que**.

(d) En déduire la propriété souhaitée.

Question 5 :

Démontrer la terminaison de l'algorithme de Huffman à l'aide de la méthode du variant vue en cours.