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.

Réponse :

Les meilleurs algorithmes de tri s'exécutent en $\mathcal{O}(n \log n)$; on connaît par exemple le tri par tas dans ce cas.

Question 2 :

(a) Quel critère permet de déterminer l'indice du premier élément échangé ? Illustrer votre réponse par un schéma ou un exemple.

Réponse :

Un exemple de tableau avec interversion est le suivant :

0	1	2	3	4	5	6	7
A	B	G	D	E	F	C	H

où les élements d'indices 2 et 6 (C et G) ont été échangés. Il est facile de constater que le premier élément échangé est au plus petit indice i tel que $T[i] > T[i+1]$ (ici G supérieur à D).

Généralisons : ce n'est clairement pas le cas pour les éléments d'indice $< i$ (puisque le tableau est initialement trié) ; et la valeur qu'on a placée à l'indice i lors de l'échange provient d'un indice $j > i$, ce qui entraîne forcément que $T[j] \geqslant T[i+1]$.

(b) Quel critère permet de reconnaître le second élément échangé ?

Réponse :

De même, on voit que le second élément échangé est (en général) situé au seul indice $j > i+1$ tel que $T[j] < T[j-1]$, pour les mêmes raisons.

(c) Attention, un cas particulier peut se présenter. Expliquer lequel.

Réponse :

Dans le cas où on échange deux éléments consécutifs :

0	1	2	3	4	5	6	7
A	B	D	C	E	F	G	H

on ne « trouve pas » d'indice $j > i+1$ tel que $T[j] < T[j-1]$. Dans ce cas $j = i+1$, mais il faut avoir vérifié tout le segment droit du tableau pour le savoir.

(d) Écrire un algorithme qui retrouve ces éléments et les remet en place, en ne parcourant qu'une seule fois le tableau.

Réponse :

i := 0
tant que $T[i] < T[i+1]$ **faire**
  $i := i+1$
$j := i+2$
tant que $j < n$ *et* $T[j-1] < T[j]$ **faire**
  $j := j+1$
si $j = n$ **alors**
  $j := i+1$
Échanger $T[i]$ et $T[j]$.

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.

Réponse :

On sait que dans un arbre binaire de recherche, le parcours en profondeur infixe donne la liste des éléments de l'arbre en ordre croissant. Il faut donc appliquer le même algorithme que précédemment, en remplaçant le parcours du tableau par celui de l'arbre en ordre infixe.

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.

Réponse :

La somme des priorités des nœuds contenus dans la FàP F vaut 1.

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

Réponse :

Initialement on insère chaque symbole de l'alphabet avec pour priorité sa fréquence. Or la somme des fréquences de tous les symboles vaut nécessairement 1.

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

Réponse :

Lors d'une itération, on extrait de F deux nœuds x et y de priorités respectives p_x et p_y, et on insère un nouveau nœud de priorité $p_z = p_x + p_y$.

Si l'invariant est vérifié en début d'itération, la somme des priorités dans F vaut 1, et alors en fin d'itération, elle vaut donc :

$$1 - p_x - p_y + p_z \;=\; 1 - p_x - p_y + (p_x + p_y) \;=\; 1$$

L'invariant est donc vérifié pour l'itération suivante.

(d) En déduire la propriété souhaitée.

Réponse :

En sortie de boucle, la FàP F contient un unique nœud (cf condition du tant que), la priorité de ce dernier est donc égale à 1 d'après l'invariant. C'est ce nœud qui est pris pour racine de l'arbre de Huffman.

Question 5 :

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

Réponse :

Un variant convenable est le nombre de nœuds contenus dans F :

— c'est un nombre entier positif ;

— à chaque itération on extrait 2 nœuds et on en insère 1, donc le nombre de nœuds décroît de 1 exactement.