L3 Informatique **année 2014-2015**

Méthodes Algorithmiques

Examen du 27 Avril 2015

Responsable : Sophie Pinchinat

Documents autorisés

Exercice 1 : Énigme On considère l'algorithme suivant :

Algorithm 1 $XXX(A[1\dots n]$ d'entiers, k entier, $B[1\dots n]$ d'entiers

Require: $k = max_{1\leq i\leq n} A[i]$

1: **var** $C[0\dots k]$ d'entiers
2: **for** $i = 0$ **to** k **do**
3: $\quad C[i] \leftarrow 0$
4: **end for**
5: **for** $j = 1$ **to** longueur$[A]$ **do**
6: $\quad C[A[j]] \leftarrow C[A[j]] + 1$
7: **end for**
8: **for** $i = 1$ **to** k **do**
9: $\quad C[i] \leftarrow C[i] + C[i-1]$
10: **end for**
11: **for** $j =$ longueur$[A]$ **to** 1 **do**
12: $\quad B[C[A[j]] \leftarrow A[j]$
13: $\quad C[A[j]] \leftarrow C[A[j]] - 1$
14: **end for**

1) Exécuter l'Algorithme 1 pour les paramètres $A = [2, 5, 3, 0, 2, 3, 0, 3]$ et $k = 5$, selon les trois étapes entre les lignes $5-7$, $8-10$, et $11-14$, en ne faisant apparaître pour chacune des ces trois étapes que la valeur des tableaux C et B. Détailler toutefois les deux premières itérations de la boucle **for** de la ligne 11.

2) Quelle est la propriété vérifiée après l'exécution des lignes $5-7$?

3) Quelle est la propriété vérifiée après l'exécution des lignes $8-10$?

4) Que réalise l'Algorithme 1 ?

5) Quel serait l'effet de mettre **for** $j = 1$ **to** longueur$[A]$ **do** à la ligne 11 ?

Exercice 2 : Patron de fusion optimale Le tri par fusion utilise une procédure de fusion, que l'on nommera 2-Fusion, de deux sous-tableaux triés, dont on rappelle que le nombre de déplacements d'éléments vers le tableau fusionné est dans $O(n_1 + n_2)$, si le premier tableau est de taille n_1 et le second de taille n_2. On souhaite développer un algorithme de fusion efficace pour un nombre arbitraire de tableaux triés. Soient donc donnés $T_1, \dots, T_k$ de tailles respectives $n_1, \dots, n_k$. On prendra comme exemple de référence la séquence de tailles $20, 30, 10, 5, 30$.

1) Écrire un algorithme glouton A1 qui invoque 2-FUSION itérativement le long de la séquence d'entrée des tableaux, et calculer son temps d'exécution[1].

2) Sur notre exemple de référence, quel sera le temps d'exécution de A1 ?

3) On modifie l'algorithme A1 en choisissant de fusionner en priorité les deux tableaux de plus petites tailles parmi l'ensemble des tableaux restants à fusionner. Sur notre exemple de référence on obtiendra l'arbre de fusion en Figure 1.

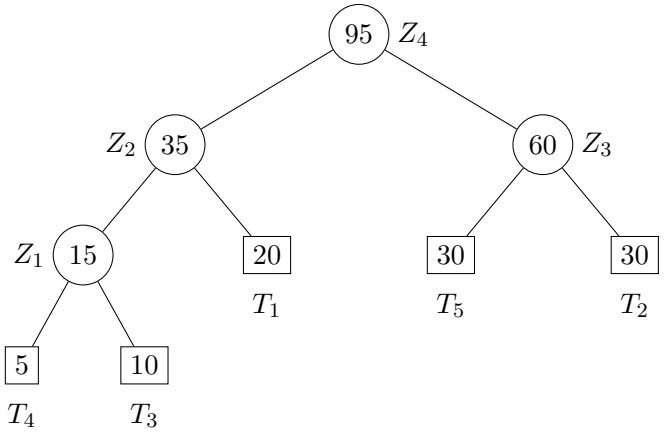

Figure 1: Un noeud interne contient le nombre total de déplacements pour fusionner les tableaux feuilles du sous-arbre. Les étiquettes Z_i sont les noms des tableaux intermédiaires créés pour la fusion, le noeud Z_4 étant le tableau final.

4) Écrire l'algorithme qui construit l'arbre de fusion pour une séquence arbitraire de tableaux (*indication : on pourra s'inspirer de l'algorithme de calcul de l'arbre préfixe pour le codage de Huffman*).

5) Quel lien peut-on établir entre l'algorithme A2 et l'algorithme de tri fusion ?

Exercice 3 : Programmation dynamique On considère $G = (V, \text{poids})$ un graphe orienté valué[2] pour lequel on fixe un sommet s. Pour tous $k \in \mathbb{N}$ et $v \in V$, on note $\delta(k, v)$ la valeur d'un plus court chemin de s à v en au plus k étapes.

1) Proposer une approche par programmation dynamique pour calculer la valeur d'un plus court chemin de s vers tout sommet v du graphe.

2) À quel algorithme cela fait-il penser ?

[1] on convient dans toute la suite que le temps d'exécution est donné par le nombre de déplacements d'éléments.

[2] avec poids : $V \times V \to \mathbb{N} \cup \{\infty\}$ et $\text{poids}(u, v) = +\infty$ s'il n'existe pas d'arc entre u et v.