Université Pierre et Marie Curie — **Master Informatique**
Lundi 5 Septembre 2005 — **MI015-ALGAV**

Examen de rattrapage

Durée : 3h. Tous documents autorisés

Exercice 1 Arbre couvrant et files binomiales (6 points)

Soit $G =< V, E >$ un graphe non orienté, connexe, et valué par une fonction de poids $w : E \rightarrow R$. Etant donné un sous-ensemble F de E, on appelle poids total de F la somme des poids des arêtes de F.

L'algorithme suivant permet de construire un arbre couvrant de poids minimum pour G, c'est-à-dire un sous-ensemble d'arêtes T, ne contenant pas de cycle, couvrant tous les sommets de V, et qui minimise le poids total $\sum_{(u,v)\in T} w(u, v)$.

`Algorithme AC(G)`
    $T \leftarrow \emptyset$
    `Pour chaque sommet` $v_i \in V(G)$
        $V_i \leftarrow \{v_i\}$
        $E_i \leftarrow \{(v_i, v) \in E(G)\}$
    `finPour`
    `Tant que le nombre d'ensembles` V_i `est supérieur à 1 Faire`
        `choisir l'un quelconque des` V_i
        `extraire de` E_i `l'arête` (u, v) `de poids minimum :` $u \in V_i$, $v \in V_j$
        `Si` $i \neq j$ `Alors`
            $T \leftarrow T \cup \{(u, v)\}$
            $V_i \leftarrow V_i \cup V_j$ `et on détruit` V_j
            $E_i \leftarrow E_i \cup E_j$
        `finSi`
    `finTantque`
`fin`

Question 1. Donner une implantation de cet algorithme à l'aide des opérations sur les files de priorité (construction d'une file, adjonction d'un élément, suppression du minimum, fusion de files, ...).

Question 2. Quelle est la complexité en temps de cet algorithme, dans le pire des cas,

- lorsque les E_i sont représentés par des tas (arbres croissants parfaits),
- lorsque les E_i sont représentés par des files binomiales,
- lorsque les E_i sont représentés par des files de Fibonacci.

Exercice 2 Recherche binaire dynamique (14 points)

La recherche binaire (dichotomique) dans un tableau trié consomme un temps logarithmique, mais le temps d'insertion d'un nouvel élément est linéaire par rapport à la taille du tableau. On peut améliorer le temps d'insertion en conservant séparément plusieurs tableaux triés.

Plus précisément, on suppose que l'on souhaite implanter les opérations de recherche et d'insertion sur un ensemble E ayant n éléments. Supposons que n s'écrive $b_{k-1} \dots b_1 b_0$ en base 2 ($i.e$ $n = b_{k-1}2^{k-1} + \dots + b_1 2^1 + b_0 2^0$). On a k tableaux triés $A_0, A_1, \dots, A_{k-1}$, de tailles respectives $2^0, 2^1, \dots, 2^{k-1}$. Chaque tableau A_i est soit plein, soit vide selon que $b_i = 1$ ou $b_i = 0$. Le nombre total d'éléments contenus dans les k tableaux est donc $b_0 2^0 + b_1 2^1 + \dots + b_{k-1}2^{k-1} = n$. Bien que chaque tableau soit trié, il n'existe pas de relation particulière entre les éléments des différents tableaux.

Question 4. Calculer la complexité de l'insertion dans le pire cas : d'une part en fonction de i, où i est tel que $A_0, A_1, \ldots, A_{i-1}$ sont pleins et A_i est vide ; et d'autrepart en fonction de n.

Question 5. Une séquence d'opérations est effectuée sur une structure de données. Soit n un entier naturel et $b_{k-1} \ldots b_1 b_0$ l'écriture de n en base 2. Soit $\alpha(n)$ le plus petit indice i tel que $b_i = 0$ (si tous les $b_{k-1}, \ldots, b_1, b_0$ sont égaux à 1 alors $\alpha(n) = k$). On suppose que la n-ième opération coûte $2^{\alpha(n)}$. Montrer que, pour $n = 2^k - 1$, le coût total d'une suite de n opérations est égal à $k2^{k-1}$. En déduire le coût amorti d'une opération, lorsqu'on effectue une suite de n opérations.

On considère une suite de n insertions dans la structure de données présentée dans l'introduction (tous les tableaux étant initialement vides). Montrer que le coût amorti d'une insertion est en $O(\log n)$.

Exercice 3 Problème de sac à dos (5 points)

On considère le problème du sac-à-dos suivant : un randonneur souhaite organiser un voyage et possède un ensemble de p objets. Chaque objet i possède un poids w_i et une utilité u_i. L'objectif du randonneur est de remplir son sac-à-dos de faon à maximiser sa fonction utilité tout en respectant la contrainte limitant le poids total du sac-à-dos à W.

Question 1. Formuler ce problème sous-forme de problème de plus long chemin dans un graphe orienté que vous définirez pour l'exemple ci-dessous avec $W = 6$:

j	1	2	3	4
u_j	40	15	20	10
w_j	4	2	3	1

Question 2. Comment peut-on transformer ce problème en un problème de recherche de plus court chemin ? Peut-on utiliser un algorithme de plus court chemin pour le résoudre ?