Université Paris 7

Maîtrise d'Informatique – Algorithmique

Examen janvier 2003

Duré de l'examen : 2 heures. Documents autorisés : notes de cours/TD. Le barème est indicatif.

Problème 1 : (4 points)

Soit $G = (V, E)$ un graphe orienté, pondéré par la fonction $w : E \to N$. On considère un sommet source $s \in V$, ainsi qu'un sous-ensemble de sommets *fiables* $F \subseteq V$.

Le but de l'exercice est de calculer pour chaque sommet $v \in V$ un chemin de poids minimal de s à v, qui en même temps maximise le nombre de sommets fiables.

Montrez comment on modifie l'algorithme de Dijkstra pour resoudre cette question.

Problème 2 : (5 points)

L'entrée pour ce problème est la matrice d'adjacence $A \in \{0, 1\}^{n \times n}$ d'un graphe non-orienté G, dont l'ensemble de sommets est $\{1, \ldots, n\}$. La procédure `CompConn` ci-dessous devra être complétée afin de calculer les composantes connexes de G. Autrement dit, on demande qu'à la fin de `CompConn` on ait $A[i, j] = 1$ ssi i et j sont connectés par un chemin.

```
Procedure CompConn (A[1..n, 1..n] matrice booleenne)
  debut
     t = n
     repeter t fois
     pour i de 1 a n faire
       pour j de 1 a n faire
         k = 1
         tant que (...) faire
           A[i,j] = A[i,k] et A[k,j]
           A[j,i] = A[i,j]
           k = k + 1
         fin tantque
       fin pour
     fin pour
    fin repeter
  fin
```

1. Compléter la condition de la boucle `tantque`.
2. Quelle est la différence entre `CompConn` et l'algorithme de Floyd-Warshall ? Trouvez un exemple qui montre qu'on ne peut pas choisir la valeur 1 pour t dans la première ligne du programme.
3. Montrer qu'on peut initialiser t par $\log_2(n)$ (au lieu de n), tout en préservant la validité de `CompConn`. Indiquez pour cela un invariant pour la boucle `repeter` et démontrez sa validité.
4. Indiquez le nombre minimal d'exécutions de la boucle `repeter` afin d'obtenir le bon résultat sur le graphe suivant : $V = \{1, \ldots, 8\}$, $E = \{ij \mid |i - j| = 1\}$. Justifiez votre réponse.

Problème 3 : (5 points)

Dans cet exercice on considère une séquence d'opérations d'insertions insert(i) (où $1 \leq i \leq n$) et de delete_min à partir de l'ensemble vide. Dans la séquence donnée, chaque élément de $\{1, \ldots, n\}$ est inséré exactement une fois.

Exemple : insert(4), insert(8), delete_min, insert(3), delete_min, insert(9), insert(2), insert(6), delete_min, delete_min, delete_min, insert(1), insert(7), delete_min, insert(5).

Le problème à resoudre est de trouver de manière efficace les valeurs retournées par les opérations delete_min (delete_min retourne la valeur -1, si l'ensemble est vide).

1. Indiquez la suite des valeurs retournées par les opérations delete_min sur l'exemple ci-dessus.

Une solution efficace pour ce problème est possible en utilisant la méthode union-find. On associe à la séquence donnée les sous-ensembles maximaux $S_1, S_2, \ldots, S_m$ de valeurs insérées entre 2 opérations delete_min consécutives (le premier delete_min vient après S_1 et le dernier delete_min vient après S_{m-1} ; un ensemble S_i peut être vide) :

Exemple : $m = 7$ (6 opérations delete_min), $S_1 = \{4, 8\}$, $S_2 = \{3\}$, $S_3 = \{9, 2, 6\}$, $S_4 = S_5 = \emptyset$, $S_6 = \{1, 7\}$, $S_7 = \{5\}$.

L'algorithme proposé est le suivant :

```
MIN : tableau [1 .. m-1]
// MIN[i] sera la valeur retournee par la i-eme operation delete_min

Procedure LesMin  (S1, ...., Sm)
  debut
      pour i de 1 a n faire
```

```
(1)         determiner j tel que i appartient a Sj
(2)         si (j != m)
(3)             alors MIN[j] = i
(4)             soit k > j le plus petit indice tel que Sk existe
(5)             Sk = Sk union Sj; supprimer Sj
            finsi
       fin pour
    fin
```

2. Comment doit-t-on initialiser le tableau `MIN` ?
3. Justifiez que la procédure `LesMin` calcule les valeurs correctes de `MIN`.
4. Decrivez les structures de données requises pour implementer cette procédure avec la méthode union-find aussi efficacement que possible.
5. Précisez pour chacune des lignes (1), (4) et (5) de cette procédure les opérations à effectuer sur la structure de donnée proposée.
6. Comment doit-on modifier la procédure `LesMins` si on remplace les opérations delete_min par delete_max ?

Problème 4 : (4 points)

Soit $G = (V, E)$ un graphe non-orienté et $s, t \in V$ deux sommets. Dans cet exercice on veut calculer le nombre maximal de chemins de s à t n'ayant pas de sommets communs (à part s, t).

On définit d'abord le graphe orienté $H = (V, F)$ en remplaçant chaque arête $uv \in E$ par deux arcs $(u, v), (v, u) \in F$. Ensuite on définit le graphe $H' = (V', F')$ en remplaçant chaque sommet $v \in V$ par 2 copies $v_1, v_2 \in V'$ avec $(v_1, v_2) \in F'$ et tel que chaque arc rentrant dans v sera un arc rentrant dans v_1, et chaque arc sortant de v sera un arc sortant de v_2 :

$$F' = \{(v_1, v_2) \in F' \mid v \in V\} \cup \{(x_2, v_1) \mid (x, v) \in F\} \cup \{(v_2, y_1) \mid (v, y) \in F\}$$

Tout arc de H' aura la capacité 1.

Montrez que la valeur maximale d'un flôt de s_2 à t_1 dans le graphe H' est égale au nombre maximal de chemins de s à t (dans G) n'ayant pas des sommets communs. Justifiez bien votre réponse.

Problème 5 : (4 points)

On considère une séquence d'opérations union-find dans laquelle les opérations unions précédent toutes les opérations find. On suppose que les opérations union-find sont réalisées comme decrit en cours (les ensembles sont des arbres ; find(a) retourne la racine de l'arbre contenant a ; union(a, b) fait l'union des deux arbres en mettant une des racines comme fils de l'autre racine). On suppose de plus que l'opération find se fait avec compression de chemins.

Montrez qu'une telle séquence de longueur m s'exécute en temps $O(m)$.