Partiel d'algorithmique
année 2006–2007

4 Décembre 2006

3 heures.
Exercices à rédiger sur deux copies distinctes.
Donner le fonctionnement de vos algorithmes avant d'écrire du pseudo-code.

1 Exercice 1:

Soit un graphe non-orienté, connexe, avec un ensemble de sommets $V = \{1, \ldots, n\}$ et $|E| = m$ arêtes de poids unitaire. On note A la matrice d'adjacence booléenne $n \times n$ telle que $A_{ij} = A_{ji} = 1$ si l'arête (i, j) est dans E et 0 sinon. Étant donné A, la matrice des distances D est une matrice d'entiers positifs telle que D_{ij} vaut la longueur du plus court chemin entre le sommet i et le sommet j. Les diagonales de A et D valent 0. Le diamètre d'un graphe est le maximum des plus court chemins entre toute paire de sommets.

Le but de cet exercice est de montrer que pour toute paire de sommets le calcul de la plus courte distance (APD) entre deux sommets, peut se faire en un peu plus du coût MM(n) d'une multiplication de 2 matrices $n \times n$.

Question 1. Soit $G'(V', E')$ le graphe obtenu en plaçant une arête entre chaque paire de sommets $i \neq j \in V$ qui sont à distance 1 ou 2 dans G. On notera A' la matrice d'adjacence de G' et D' la matrice des plus courtes distances dans G'.

a) Montrer que si $Z = A^2$, alors il existe un chemin de longueur 2 dans G entre chaque paire de sommets i et j si et seulement si $Z_{ij} > 0$. De plus, la valeur de Z_{ij} est le nombre de chemins distincts de longueur 2 entre i et j.

b) Supposons que le diamètre de G soit au plus 2. Montrer que G' est le graphe complet à n sommets (une arête entre chaque sommet) et exprimer dans ce cas D en fonction de A et A'.

Question 2. En général, G peut avoir un diamètre arbitrairement grand $\leq n$.

a) Montrer alors que pout toute paire $i, j \in V$,

- Si D_{ij} est paire, alors $D_{ij} = 2D'_{ij}$.
- Si D_{ij} est impaire, alors $D_{ij} = 2D'_{ij} - 1$.

On en déduit que la matrice D peut être calculée avec D' si on connaît la parité de tous les plus courts chemins.

b) Montrer que pour chaque paire de sommets distincts i et j dans G,

- Pour chaque voisin k de i, $D_{ij} - 1 \leq D_{kj} \leq D_{ij} + 1$.
- Il existe un voisin k de i tel que $D_{kj} = D_{ij} - 1$.

c) Montrer que pour chaque paire de sommets distincts i et j de G,

- Si D_{ij} est paire, alors $D'_{kj} \geq D'_{ij}$ pour chaque voisin k de i dans G.
- Si D_{ij} est impaire, alors $D'_{kj} \leq D'_{ij}$ pour chaque voisin k de i dans G. De plus, il existe un voisin k de i dans G tel que $D'_{kj} < D'_{ij}$.

Soit $\Gamma(i)$ l'ensemble des voisins de i dans G et $d(i)$ le degré de i. En déduire la caractérisation suivante: Pour toute paire de sommets distincts i et j de G:

- D_{ij} est paire si et seulement si $\sum_{k\in\Gamma(i)} D'_{kj} \geq D'_{ij}d(i)$.
- D_{ij} est impaire si et seulement si $\sum_{k\in\Gamma(i)} D'_{kj} < D'_{ij}d(i)$

Question 3.

a) Montrer comment calculer $\sum_{k\in\Gamma(i)} D'_{kj}$ par une multiplication matricielle dont un des arguments est D'. Remarquez que $Z_{ii} = d(i)$ pour tout i.

b) Proposer alors un algorithme récursif pour calculer D.

c) Analyser sa complexité en terme de multiplication matricielle utilisant la fonction $T(n, \delta)$ qui représente le temps de calcul de votre algorithme sur un graphe à n sommets et de diamètre δ donné en entrée.

2 Exercice 2:

Le but du problème est l'étude des sous-suites monotones de taille maximale d'un tableau.

On se donne un tableau A de n éléments entiers $A[1], \ldots, A[n]$. On appelle *sous-suite de longueur m* une suite d'indices $i_1, \ldots, i_m$ telle que $\forall k$, $1 \leq i_k \leq n$ et $i_k < i_{k+1}$. Une sous-suite est de plus *croissante* si $\forall k$, $A[i_k] \leq A[i_{k+1}]$, *décroissante* si $\forall k$, $A[i_k] \geq A[i_{k+1}]$, et *monotone* si elle est soit croissante, soit décroissante.

Question 1. *(Théorème de Erdös–Szekeres)* On note $\psi : [1 \dots n] \to [1 \dots n] \times [1 \dots n]$ la fonction qui à k associe deux entiers: la longueur maximale des sous-suites croissantes (respectivement décroissantes) de A dont le dernier élément est d'indice k.

Montrez que ψ est injective.

Déduisez-en que si $n \geq (N-1)^2 + 1$, alors il existe forcément une sous-suite monotone de taille N.

Question 2. Exprimez $\psi(k)$ en fonction de tous les $\psi(k')$ pour $k' < k$. Déduisez-en un algorithme polynomial qui détermine, par *programmation dynamique*, la longueur maximale des sous-suites monotones de A. Vous donnerez son pseudo-code, ainsi que sont coût asymptotique en temps et en mémoire, en fonction de n.

Question 3. Modifiez l'algorithme de la question précédente pour qu'il affiche, en plus de la longueur maximale des sous-suites monotones de A, la liste de toutes les sous-suites monotones de longueur maximale.

Dans la suite du problème, on ne s'intéresse désormais qu'aux sous-suites croissantes.

Question 4. Pour chaque i de 1 à n et chaque k, on considère les sous-suites croissantes de $A[1], \dots, A[i]$ de longueur k. On notera $\alpha_i(k)$ l'indice du dernier élément d'une telle sous-suite tel que $A[\alpha_i(k)]$ soit minimal.

1. Montrez que, pour tout i la suite des $A[\alpha_i(1)], A[\alpha_i(2)], \dots$ est croissante.
2. Montrez qu'il n'y a qu'un élément de α_i à modifier pour obtenir α_{i+1}. Donner un algorithme en temps $\mathcal{O}(\log n)$ pour trouver cet élément.
3. Proposez le pseudo-code d'un algorithme *glouton* basé sur α qui exhibe une sous-suite croissante de A de taille maximale. Quelle est sa complexité?

Question 5. *(Arbres de van Emde Boas)* Pour résoudre la dernière question, on définit une structure de données permettant de représenter un sous-ensemble S de $\{1, \dots, n\}$ et qui supporte efficacement les opérations suivantes: insertion et suppression d'un élément dans S, recherche de successeur (c'est à dire, du plus petit élément de S supérieur strictement à un entier donné) et de prédécesseur. On suppose maintenant que n est de la forme 2^{2^N}. On introduit maintenant la structure d'*arbres de van Emde Boas*. Un nœud de l'arbre, noté $B(S, n)$, représente le sous-ensemble S de $\{1, \dots, n\}$ de la manière récursive suivante:

- si $n \leq 2$, $B(S, n)$, contient simplement deux champs booléens qui indiquent, respectivement, si $1 \in S$ et si $2 \in S$,
- sinon, $B(S, n)$ est composé de:
  - un champ contenant $\min S$ (ou 0, si S est vide),

– un champ contenant max S (ou 0, si S est vide),

– $\sqrt{n}$ fils $B_i = B(S_i, \sqrt{n})$. Chacun représente le sous-ensemble $S_i \subseteq \{1, \ldots, \sqrt{n}\}$ suivant, pour $1 \leq i < \sqrt{n}$:

$$S_i = \{\ x \bmod \sqrt{n} \mid x \in S \text{ et } (i-1)\sqrt{n} < x \leq i\sqrt{n}\ \}$$

Ainsi, l'intervalle $1, \ldots, n$ est découpé en $\sqrt{n}$ intervalles de taille $\sqrt{n}$; B_i représente la partie de S incluse dans le $i-$ème intervalle. $\sqrt{n}$ étant toujours une puissance de 2, le calcul de $x \bmod \sqrt{n}$ est un simple masquage de bits.

– un fils qui représente l'ensemble $R \subseteq \{1 \ldots, \sqrt{n}\}$ des indices i tels que le $i-$ème intervalle n'est pas vide:

$$R = \{\ i \mid \exists x \in S \text{ et } (i-1)\sqrt{n} < x \leq i\sqrt{n}\ \}$$

(Le calcul de i tel que $(i-1)\sqrt{n} < x \leq i$ s'implante également grâce à un simple masquage de bits.)

1. Montrez que la recherche du successeur s'implante en temps $\mathcal{O}(N)$, c'est à dire, $\mathcal{O}(\log \log n)$, grâce à un algorithme récursif. Vous donnerez le pseudo-code de l'algorithme ainsi qu'une analyse mathématique de sa complexité. On ne demandera pas de donner l'algorithme de recherche du prédécesseur qui, par symétrie, s'implante de manière similaire. *(Indice: pour obtenir un coût en $\mathcal{O}(\log \log n)$, il faut que l'algorithme ne s'appelle par récurrence que sur un seul de ses fils.)*

2. La structure proposée ne permet l'insertion et la suppression d'un élément qu'avec un coût $\mathcal{O}(\log n)$ (considérer, par exemple, l'insertion dans un arbre vide). Pour descendre à un coût $\mathcal{O}(\log \log n)$, la définition de S_i est changée en S'_i, où

$$S'_i = S_i \setminus \{\min S_i\}$$

Montrez que cette définition ne change pas le comportement de l'algorithme de recherche de successeur. Donnez le pseudo-code des algorithmes d'insertion d'un élément de coût $\mathcal{O}(\log \log n)$. (On ne demande pas de donner l'algorithme de suppression qui a la même complexité.)

Question 6. Revenons maintenant aux sous-suites croissantes. On s'intéresse au cas particulier où A est une permutation de $1, \ldots, n$, où n est de la forme 2^{2^N}. Montrez comment utiliser un arbre de van Emde Boas pour exhiber en temps $\mathcal{O}(n \log \log n)$ une sous-suite croissante de A de taille maximale.