Examen d'algorithmique

Mardi 7 janvier 2014 12h–15h / Aucun document autorisé

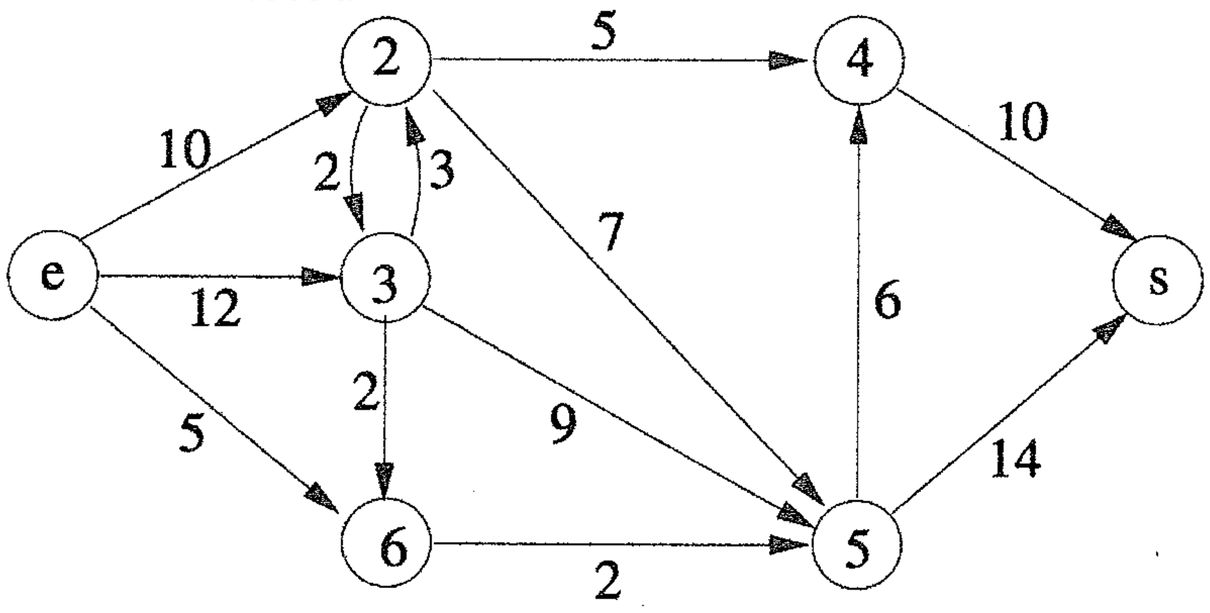
Mode d'emploi : Le barème est donné à titre indicatif.
La qualité de la rédaction des algorithmes et des preuves sera très fortement prise en compte pour la note.
On peut toujours supposer une question résolue et passer à la suite.

Exercice 1 : Diviser-pour-régner - 2 points

Ecrire un algorithme "Diviser-pour-régner" qui prend un tableau en paramètre et qui renvoie une paire constituée de la valeur minimale et de la valeur maximale du tableau. Donner sa complexité. Justifier l'algorithme.

Exercice 2 : Flots - 2 points

On considère le réseau de transport de la figure ci dessous. Appliquer l'algorithme de Ford-Fulkerson pour trouver un flot maximal. On donnera tous les flots intermédiaires et les graphes des augmentations obtenus par l'algorithme. En déduire une coupe de capacité minimale du réseau.



Exercice 3 : Calcul de notes - 3 points

On cherche à calculer les notes de semestre d'étudiants qui ont passé plusieurs unités d'enseignement (chacune d'elle reçoit un coefficient représentant un nombre de crédits). Pour valider un semestre, un étudiant doit valider 30 crédits. Il a pu suivre (et valider) un ensemble d'UEs qui dépassent les 30 crédits et pour calculer sa note finale, on cherche à trouver les UE validées qui lui permettent d'obtenir son semestre (30 crédits) avec la meilleure moyenne pondérée[1] possible. Pour cela, il est possible de diminuer le coefficient

1. par le poids en crédits

d'une UE si c'est nécessaire pour arriver à exactement 30 crédits (mais bien sûr, il n'est pas possible d'augmenter le crédit d'une note).

Par exemple, supposons que les notes sont les suivantes (avec le nombre de crédits entre parenthèses) : 12 (6), 17 (8), 11 (3), 15 (3), 14 (4), 8 (5), 13 (6), 18 (5). Il est alors préférable (pour l'étudiant) de prendre les notes 12 (6 dégradé en 4), 17 (8), 15 (3), 14 (4), 13 (6), 18 (5). Cela lui donne une moyenne de 15,1 (*ie* 453/30).

Proposer un algorithme glouton pour calculer la meilleure moyenne possible. Donner sa complexité. Justifier votre algorithme.

Exercice 4 : Sous-ensemble de somme exacte - 5 points

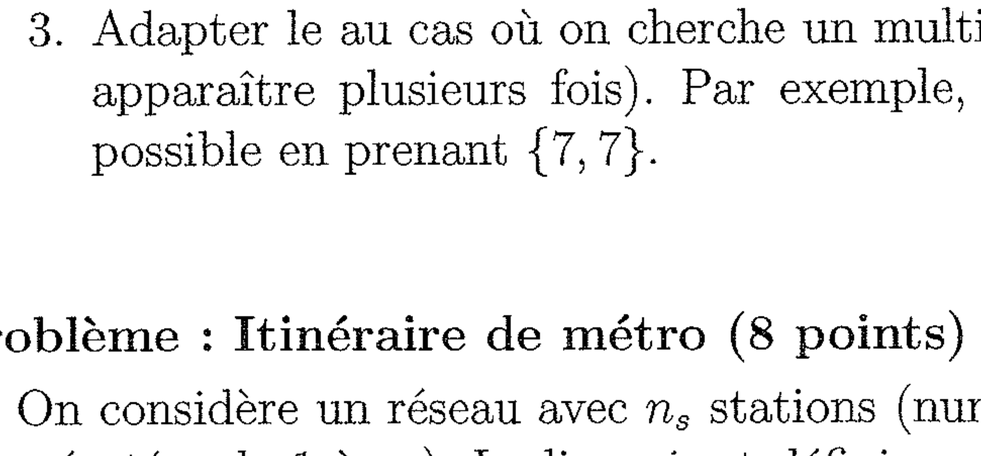
On cherche à résoudre le problème suivant : étant donnés un ensemble de n valeurs entières $\mathcal{E} = \{a_1, a_2, \ldots, a_n\}$ et une valeur cible V, on veut savoir si il est possible de trouver un sous-ensemble de $\mathcal{E}$ dont la somme des éléments vaut exactement V. Par exemple, pour $\mathcal{E} = \{2, 8, 3, 7\}$ et $V = 12$, c'est possible avec le sous-ensemble $\{2, 3, 7\}$, mais si on prend $V = 14$, ce n'est pas possible...

1. Proposer un algorithme de programmation dynamique pour résoudre ce problème. On cherchera à construire un tableau Booléen à deux dimensions $T[i, v]$ où $0 \leqslant i \leqslant n$ et $0 \leqslant v \leqslant V$ tel que $T[i, v]$ vaut VRAI si et seulement si il est possible de trouver un sous-ensemble de $\{a_1, \ldots, a_i\}$ de somme v. Justifier l'algorithme et donner sa complexité.
2. Appliquer l'algorithme sur l'exemple $\mathcal{E} = \{2, 8, 3, 7\}$ et $V = 15$.
3. Adapter le au cas où on cherche un multi-ensemble (où un même élément a_i peut apparaître plusieurs fois). Par exemple, pour $\mathcal{E} = \{2, 8, 3, 7\}$ et $V = 14$, c'est possible en prenant $\{7, 7\}$.

Problème : Itinéraire de métro (8 points)

On considère un réseau avec n_s stations (numérotées de 1 à n_s) et n_ℓ lignes de métro (numérotées de 1 à n_ℓ). La ligne i est définie par un tableau $L_i[-]$ d'entiers de taille t_i : $L_i[1]$ et $L_i[t_i]$ sont les numéros des deux stations extrémités de la ligne i. Sur cette ligne, les métros peuvent aller dans deux directions : de $L_i[1]$ à $L_i[t_i]$ ou de $L_i[t_i]$ à $L_i[1]$. Dans la première direction, la station $L_i[j+1]$ suit $L_i[j]$ (pour $j = 1, \ldots, t_i - 1$), et dans la seconde direction c'est $L_i[j]$ qui suit $L_i[j+1]$. Dans les deux cas, **on supposera toujours que le temps nécessaire pour aller de l'une à l'autre est $\delta = 90$ secondes.**

Dans la figure ci-dessous, on représente un réseau de métro de 8 stations ($n_s = 8$) et 3 lignes ($n_\ell = 3$).



$L_1 = [1, 2, 4, 8]$ et $t_1 = 4$
$L_2 = [1, 5, 3, 7, 4]$ et $t_2 = 5$
$L_3 = [6, 3, 4]$ et $t_3 = 3$

1. Pour chaque ligne i, on veut connaître les distances (en temps) séparant deux stations sur la ligne. Pour cela, on va construire une matrice Direct_i de taille $n_s \times n_s$ telle que le coefficient $\mathsf{Direct}_i[j,k]$ (avec $1 \leqslant j,k \leqslant n_s$) représente le temps pour aller de la station j à la station k par la ligne i (et il vaudra ∞ si les deux stations ne sont pas reliées par la ligne i, et il vaudra toujours 0 pour les coefficients de la diagonale $\mathsf{Direct}_i[k,k]$). On donne en annexe les matrices Direct_1, Direct_2 et Direct_3 de l'exemple.
   Donner un algorithme pour construire Direct_i à partir de L_i. Donner sa complexité.
2. A partir des n_ℓ matrices $\mathsf{Direct}_1, \ldots, \mathsf{Direct}_{n_\ell}$, donner un algorithme pour construire une matrice Direct de dimension $n_s \times n_s$ qui contient les temps minimaux pour aller d'une station à une autre par un trajet direct (sans correspondance) : le coefficient $\mathsf{Direct}[j,k]$ représente le temps minimal pour aller de j à k par un trajet direct (avec une des lignes du réseau), et ∞ si il n'y a pas de trajet direct entre elles (et 0 pour les éléments de la diagonale). Quelle est sa complexité ? (on donne en annexe la matrice Direct de l'exemple).
3. On veut à présent construire des matrices D_k de dimension $n_s \times n_s$ contenant les **temps minimaux** pour aller d'une station à une autre en au plus k changements de lignes (correspondances). Bien sûr on a : $\mathsf{D}_0 = \mathsf{Direct}$. Donner un algorithme pour calculer les matrices $\mathsf{D}_1, \ldots, \mathsf{D}_k$ pour un k fixé. On pourra notamment calculer D_{k+1} à partir de D_k et D_0 (en suivant les idées classiques de la programmation dynamique). **Ici on supposera que tout changement de ligne (toute correspondance) prend $\Delta = 240$ secondes.** Quelle est la complexité de cet algorithme ? (on donne en annexe la matrice D_1 de l'exemple).
4. On souhaite à présent trouver l'itinéraire correspondant aux temps minimaux inscrits dans D_k. Par exemple, pour $\mathsf{D}_1[6,1]$ (qui vaut 510), le trajet est : de la station 6, prendre la ligne 3, puis changer à la station 3 et prendre la ligne 2 et descendre à la station 1. Pour cela, on va construire en même temps que les matrices D_k, des matrices Via_k de dimension $n_s \times n_s$ telles que le coefficient $\mathsf{Via}_k[i,j]$ est le numéro de ligne du dernier tronçon sur le trajet allant de i à j en temps $\mathsf{D}_k[i,j]$ (bien sûr si $k = 0$ ou si $\mathsf{D}_0[i,j] = \mathsf{D}_k[i,j]$, il n'y a qu'un tronçon et $\mathsf{Via}_k[i,j]$ sera le numéro de ligne pour aller directement de i à j). On utilisera ∞ lorsque la liaison n'existe pas, et 0 pour la diagonale. Donner un algorithme pour calculer les matrice Via_k. Donner sa complexité. (on donne en annexe la matrice Via_0 de l'exemple).
5. A partir des matrices $\mathsf{D}_0, \mathsf{D}_1, \ldots, \mathsf{D}_k$ et $\mathsf{Via}_0, \mathsf{Via}_1, \ldots, \mathsf{Via}_k$, comment retrouver tout l'itinéraire (ici on se contentera des numéros de ligne à prendre) pour aller de i à j en temps $\mathsf{D}_k[i][j]$?

Annexe

Master theorem :

Soient $a \geqslant 1$, $b > 1$, $f(n)$ une fonction positive et $t(n) = \begin{cases} a \cdot t(\frac{n}{b}) + f(n) & \text{si } n > 1 \\ \Theta(1) & \text{si } n = 1 \end{cases}$;

— Si $f(n) = O(n^{\log_b a - \epsilon})$ avec $\epsilon > 0$, alors $t(n) = \Theta(n^{\log_b a})$
— Si $f(n) = \Theta(n^{\log_b a})$, alors $t(n) = \Theta(n^{\log_b a} \cdot \log n)$
— Si $f(n) = \Omega(n^{\log_b a + \epsilon})$ pour $\epsilon > 0$ et si $\exists c < 1$ tel que $a \cdot f(\frac{n}{b}) \leqslant c \cdot f(n)$ pour n assez grand, alors $t(n) = \Theta(f(n))$

Exemples pour le problème sur les réseaux de métro :

Questions 1 & 2. Les matrices Direct_1, Direct_2, Direct_1 et Direct de l'exemple sont :

$$\mathsf{Direct}_1 = \begin{pmatrix} 0 & 90 & \infty & 180 & \infty & \infty & \infty & 270 \\ 90 & 0 & \infty & 90 & \infty & \infty & \infty & 180 \\ \infty & \infty & 0 & \infty & \infty & \infty & \infty & \infty \\ 180 & 90 & \infty & 0 & \infty & \infty & \infty & 90 \\ \infty & \infty & \infty & \infty & 0 & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty & 0 & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty & \infty & 0 & \infty \\ 270 & 180 & \infty & 90 & \infty & \infty & \infty & 0 \end{pmatrix} \quad \mathsf{Direct}_2 = \begin{pmatrix} 0 & \infty & 180 & 360 & 90 & \infty & 270 & \infty \\ \infty & 0 & \infty & \infty & \infty & \infty & \infty & \infty \\ 180 & \infty & 0 & 180 & 90 & \infty & 90 & \infty \\ 360 & \infty & 180 & 0 & 270 & \infty & 90 & \infty \\ 90 & \infty & 90 & 270 & 0 & \infty & 180 & \infty \\ \infty & \infty & \infty & \infty & \infty & 0 & \infty & \infty \\ 270 & \infty & 90 & 90 & 180 & \infty & 0 & \infty \\ \infty & \infty & \infty & \infty & \infty & \infty & \infty & 0 \end{pmatrix}$$

$$\mathsf{Direct}_3 = \begin{pmatrix} 0 & \infty & \infty & \infty & \infty & \infty & \infty & \infty \\ \infty & 0 & \infty & \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & 0 & 90 & \infty & 90 & \infty & \infty \\ \infty & \infty & 90 & 0 & \infty & 180 & \infty & \infty \\ \infty & \infty & \infty & \infty & 0 & \infty & \infty & \infty \\ \infty & \infty & 90 & 180 & \infty & 0 & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty & \infty & 0 & \infty \\ \infty & \infty & \infty & \infty & \infty & \infty & \infty & 0 \end{pmatrix} \quad \mathsf{Direct}_4 = \begin{pmatrix} 0 & 90 & 180 & 180 & 90 & \infty & 270 & 270 \\ 90 & 0 & \infty & 90 & \infty & \infty & \infty & 180 \\ 180 & \infty & 0 & 90 & 90 & 90 & 90 & \infty \\ 180 & 90 & 90 & 0 & 270 & 180 & 90 & 90 \\ 90 & \infty & 90 & 270 & 0 & \infty & 180 & \infty \\ \infty & \infty & 90 & 180 & \infty & 0 & \infty & \infty \\ 270 & \infty & 90 & 90 & 180 & \infty & 0 & \infty \\ 270 & 180 & \infty & 90 & \infty & \infty & \infty & 0 \end{pmatrix}$$

Questions 3 & 4. La matrice D_1 et la matrice Via_0 de l'exemple sont :

$$\mathsf{D}_1 = \begin{pmatrix} 0 & 90 & 180 & 180 & 90 & 510 & 270 & 270 \\ 90 & 0 & 420 & 90 & 420 & 510 & 420 & 180 \\ 180 & 420 & 0 & 90 & 90 & 90 & 90 & 420 \\ 180 & 90 & 90 & 0 & 270 & 180 & 90 & 90 \\ 90 & 420 & 90 & 270 & 0 & 420 & 180 & 600 \\ 510 & 510 & 90 & 180 & 420 & 0 & 420 & 510 \\ 270 & 420 & 90 & 90 & 180 & 420 & 0 & 420 \\ 270 & 180 & 420 & 90 & 600 & 510 & 420 & 0 \end{pmatrix} \quad \mathsf{Via}_0 = \begin{pmatrix} 0 & 1 & 2 & 1 & 2 & \infty & 2 & 1 \\ 1 & 0 & \infty & 1 & \infty & \infty & \infty & 1 \\ 2 & \infty & 0 & 3 & 2 & 3 & 2 & \infty \\ 1 & 1 & 3 & 0 & 2 & 3 & 2 & 1 \\ 2 & \infty & 2 & 2 & 0 & \infty & 2 & \infty \\ \infty & \infty & 3 & 3 & \infty & 0 & \infty & \infty \\ 2 & \infty & 2 & 2 & 2 & \infty & 0 & \infty \\ 1 & 1 & \infty & 1 & \infty & \infty & \infty & 0 \end{pmatrix}$$