IFT438 Algorithmique (Été 2001) 28 juin 2001
Examen intra-trimestriel 13h30-15h30
Daniel Lemire, Ph.D. D7-3021
SOLUTIONAIRE

Nom :_____

Prénom :_____

Matricule :_____

L'examen compte 100 points et comprend 4 questions. Vous n'avez droit à aucune documentation. Vous devez répondre sur le questionnaire.

Aide-mémoire

Formule de Stirling

$$n! = n^n e^{-n} * \sqrt{2\pi n}(1 + \frac{1}{12n} + \frac{23}{288n^2} + ...)$$

Logarithmes

$$\log_b a^c = c \log_b a$$

Exponentielles

$$(a^b)^c = a^{bc}$$

Note : à moins de mention contraire, les logarithmes sont en base 2.

Question 1. Notation O, Θ, Ω **(30 points)**

1. A. Montrez que $\log n \in O(n)$. (10 points)

On utilise la règle de l'Hôpital.

$$\lim_{n \to \infty} \frac{\log n}{n} = \lim_{n \to \infty} \frac{1/n}{1} = 0$$

On peut ensuite utiliser la 2.10 des notes de Martin Beaudry.

1. B. Montrez que $3^n \in O(5^n)$ ainsi que $5^n \in \Omega(3^n)$ (20 points, 10 points pour chaque affirmation)

Parce que 3/5 < 1, nous avons

$$\lim_{n \to \infty} \frac{3^n}{5^n} = \lim_{n \to \infty} \left(\frac{3}{5}\right)^n = 0$$

et la prop. 2.10 nous donne $3^n \in O(5^n)$.

Puisque $3^n \in O(5^n)$ *, nous avons donc qu'il existe une constante* $c > 0$ *et un indice n0 tel que pour* $n > n_0$, $3^n > c5^n$ *ce qui signifie que* $5^n > \frac{1}{c}3^n$ *ce qui suffit pour conclure que* $5^n \in \Omega(3^n)$.

Question 2. Ordre des algorithmes (20 points)

On a vu en classe que l'algorithme de tri-fusion appartient à $O(n\log n)$. Pour en arriver à cette fonction ($n\log n$), nous avons supposé que tous les passages de paramètres se faisaient par pointeur. Sachant que le passage par valeur d'un tableau de taille N appartient à $O(N)$, refaites l'analyse de l'algorithme tri-fusion en passant à chaque fois les tableaux et sous-tableaux par valeur. Vous devez trouver $g(n)$ tel que l'algorithme est $O(g(n))$ et votre estimation doit être aussi efficace que possible. Comment se compare l'algorithme avec passage par valeur par rapport à l'algorithme avec passage par pointeur?

L'analyse classe de l'algorithme tri-fusion par avec une équation de la forme... (on suppose par souci de simplicité que n est une puissance de 2)

$$T(n)=2T(\frac{n}{2})+k_1 n+k_0 \quad (*)$$

où k est une constante. En effet, le coût d'une fusion est $\Theta(n)$. *On a vu en classe que la solution sera de la forme* $T(n)=k_1 n\log n-k_0$ *, on vérifie que c'est bien la solution...*

$$2(\frac{n}{2}k_1(\log n-1)-k_0)+k_1 n+k_0=k_1 n\log n-k_1 n-2k_0+k_1 n+k_0=k_1 n\log n-k_0 .$$

Dans le cas où l'on a un passage par valeur, en réalité, on ne fait qu'ajouter un terme en $\Theta(n)$ *à au membre de gauche de l'équation (*) ce qui ne change rien à la forme de la solution. Nous aurons toujours une solution en* $O(n\log n)$ *ce qui revient à dire que le passage par valeur ne pénalise pas la complexité de l'algorithme.*

Question 3. Récurrences (30 points)

Donnez la meilleure borne asymptotique de la forme $O(g(n))$ pour les récurrences : (Justifiez)

3.A $T(n)=4T(n/2)+n^5$ (15 points)

On pose $n=2^i$ *(strictement parlant, on ne résout donc que pour n une puissance de deux), ce qui donne* $T(2^i)=4T(2^{i-1})+2^{5i}$ *, on pose ensuite* $S(i)=T(2^i)$ *ce qui nous laisse*

$S(i)=4S(i-1)+32^i$ *(*).*

Le polynôme caractéristique homogène de cette dernière équation est $x-4$ *alors que le polynôme inhomogène est* $x-32$ *, nous avons donc comme solution* $S(i)=c_1 4^i+c_2 32^i$ *, on fait ensuite la substitution dans (*) pour obtenir* $c_2 32^i=4c_2 32^{i-1}+32^i$ *que l'on divise par* 32^{i-1} *ce qui donne* $28c_2=32$ *donc* $c2=\frac{8}{7}$ *. Nous avons par la suite* $S(i)=c_1 4^i+\frac{8}{7}32^i$ *ou* $T(n)=c_1 n^2+\frac{8}{7}n^5$ *. La borne recherchée est donc* $O(n^5)$ *. Il y a des solutions plus courtes.*

3.B $T(n)=2T(n/2)+\log_2 n$ (15 points)

L'approche est similaire. On pose $n=2^i$ *ce qui donne* $T(2^i)=2T(2^{i-1})+i$ *, on pose ensuite* $S(i)=T(2^i)$ *ce qui nous laisse*

$S(i)=2S(i-1)+i$ *(*).*

Le polynôme caractéristique homogène de cette dernière équation est $x-2$ *alors que le polynôme inhomogène est* $(x-1)^2$ *, nous avons donc comme solution* $S(i)=c_1 2^i+c_2+c_3 i$ *, on fait ensuite la substitution dans (*) pour obtenir* $c_2+c_3 i=2c_2+2(i-1)c_3+i$ *et finalement le couple d'équations* $c_2-2c_3=0$ *et* $0=c_3+1$ *d'où* $c_3=-1$ *et* $c_2=-2$ *. On a donc* $S(i)=c_1 2^i-i-2$ *et en conséquence,* $T(n)=c_1 n-\log_2 n-2$ *. Le terme dominant est* $c_1 n$ *, mais* c_1 *reste une inconnue. Il y a deux cas possibles, soit* $c_1>0$ *, soit* $c_1=0$ *. La meilleure borne est* $O(n)$ *, car on ne peut pas savoir que* $\log_2 n$ *est suffisant.*

Question 4. Algorithmiques voraces (20 points)

Vous conduisez sur l'autoroute de Montréal à Toronto et désirez faire le moins d'arrêts possible. Quand votre réservoir est plein, vous pouvez faire *n* kilomètres. Votre carte vous donne la position des stations-service qui sont respectivement à $x_1, x_2, ..., x_m$ kilomètres de Montréal. On suppose qu'il y a assez de stations-service par rapport à votre autonomie de manière à pouvoir faire le trajet. Donnez un algorithme vorace pour résoudre ce problème et démontrez (prouvez) qu'il est optimal.

Sans perte de généralité, on suppose que les x_i sont en ordre croissant. On suppose aussi que l'on part avec le plein d'essence d'une première station-service (x_0), même si cette supposition n'est pas nécessaire. On peut aussi supposer qu'il y a une station-service à l'arrivée. À chaque étape on minimise le nombre de stations ou bien, ce qui revient au même, on maximise la distance parcourue entre chaque arrêt. La première fois, on va donc choisir la station-service x_i la plus éloignée possible (étant donné notre autonomie). Ensuite, une fois à la station-service x_i on choisit la station-service x_j la plus éloignée possible. Et ainsi de suite, jusqu'à la destination. La solution peut être notée selon les stations-service visitée: $(x_{K_i})_{K_i \in J, i=1,...card(J)}$ en ordre croissant: $x_{K_1} < x_{K_2} < ... < x_{K_{card}(J)}$.

Reste maintenant à montrer que c'est un choix optimal. Supposons que nous avons un choix optimal que nous noterons $(x_{K'_i})_{K'_i \in J', i=1,...,card(J')}$ (on suppose implicitement qu'il existe au moins un choix optimal!). Il faut montrer que la cardinalité de J est la même que J'. On peut y arriver en remplaçant chacune des stations-service du choix optimal par un choix glouton sans changer la cardinalité. On observe que la suite $x_{K_1}, x_{K'_2}, ..., x_{K'_{card}(J')}$ contient $card(J')$ éléments (ce qui est évident). Cette nouvelle suite est aussi valable puisque x_{K_1} ayant été retenu par le choix glouton, elle est accessible par notre voiture lors du premier arrête, de plus, $x_{K_1} \geq x_{K'_1}$ par le principe glouton et donc la distance entre $x_{K'_2} - x_{K_1} \leq x_{K'_2} - x_{K'_1}$ et donc $x_{K'_2}$ est accessible à partir de la station-service x_{K_1}. On procède ensuite par induction. On suppose que les j premiers éléments ont été remplacés par des choix gloutons et que nous avons la suite $x_{K_1}, x_{K_2}, ..., x_{K_j}, x_{K'_j+1}, x_{K'_j+2}, ..., x_{K'_{card}(J')}$. Notons que $x_{K'_j+1} \leq x_{K_j+1}$ par le principe glouton et nous avons donc $x_{K'_j+2} - x_{K_j+1} \leq x_{K'_j+2} - x_{K'_j+1}$. La suite suite $x_{K_1}, x_{K_2}, ..., x_{K_j}, x_{K_j+1}, x_{K'_j+2}, ..., x_{K'_{card}(J')}$ est donc elle aussi valable et elle a toujours une cardinalité $card(J')$. Par induction, on peut donc conclure que le choix glouton est optimal. (Une preuve plus simple existe sans doute.)