ÉCOLE POLYTECHNIQUE

ÉCOLE SUPÉRIEURE DE PHYSIQUE ET CHIMIE INDUSTRIELLES

CONCOURS D'ADMISSION 2006

FILIÈRE MP - OPTION PHYSIQUE ET SCIENCES DE L'INGÉNIEUR **FILIÈRE PC**

COMPOSITION D'INFORMATIQUE

(Durée : 2 heures)

L'utilisation des calculatrices **n'est pas autorisée** pour cette épreuve.
Le langage de programmation choisi par le candidat doit être spécifié en tête de la copie.

⋆ ⋆ ⋆

Disque dur à deux têtes

On attachera une grande importance à la concision, à la clarté, et à la précision de la rédaction. On précisera en tête de copie le langage de programmation utilisé.

Le temps d'exécution $T(f)$ d'une fonction f est le nombre d'opérations élémentaires (addition, soustraction, multiplication, division, affectation, etc.) nécessaire au calcul de f. Lorsque ce temps d'exécution dépend d'un paramètre n, il sera noté $T_n(f)$. On dit que la fonction f s'exécute :

en temps $O(n^\alpha)$, s'il existe $K > 0$ tel que pour tout n, $T_n(f) \leq Kn^\alpha$.

Ce problème étudie des stratégies de déplacement des têtes d'un disque dur afin de minimiser le temps moyen d'attente entre deux requêtes au disque dur (de lecture ou d'écriture). Dans ce problème, le disque dur est représenté par une demi-droite $[0, +\infty)$ et possède deux têtes de lecture/écriture. Chacune des têtes peut aller indifféremment à n'importe quelle position sur le disque pour y lire ou écrire une donnée. Les deux têtes peuvent être au même endroit ou encore se croiser. On ne s'intéresse qu'aux temps de déplacement des têtes et non aux temps de lecture/écriture. Les deux têtes ont la même vitesse de déplacement. Le temps de déplacement d'une tête est supposé égal à la distance qu'elle parcourt.

Une requête r est un entier positif ou nul représentant l'emplacement du disque auquel l'une des deux têtes doit se rendre. Initialement les deux têtes sont chacune à la position 0.

Le disque dur est muni d'une mémoire (appelée cache) qui permet d'enregistrer n requêtes ($n > 0$) avant de les traiter. À chaque bloc de n requêtes présentes dans le cache, le contrôleur du disque dur doit alors satisfaire ce bloc de requêtes, dans leur *ordre* d'arrivée, en minimisant le déplacement *total* des deux têtes. L'ordre importe puisqu'une opération d'écriture peut précéder une autre opération de lecture ou d'écriture. Il faut donc déterminer pour chacune des n requêtes le numéro de la tête à déplacer de manière à minimiser la somme totale des temps de tous les déplacements.

Notes de programmation : On supposera que le langage utilisé permet de définir des fonctions qui retournent des tableaux. On pourra aussi supposer que le nombre de requêtes n

est une constante du programme. Enfin, une autre constante du programme `Infini` $= -1$ sera utilisée pour coder l'infini, noté ∞ dans la Partie II.B.

Partie I

A. Coût d'une séquence de déplacements

Un bloc de n requêtes est représenté par une suite de n entiers positifs ou nuls $\langle r_1, r_2, \ldots r_n \rangle$ rangés dans un tableau r de taille n. Une *séquence de déplacements* $\langle d_1, d_2, \ldots d_n \rangle$ est une suite de n entiers, 1 ou 2, rangés dans un tableau d indiquant à l'étape i qui de la première tête ($d_i = 1$) ou de la deuxième tête ($d_i = 2$) doit se déplacer à la position r_i ($1 \leq i \leq n$). Le *coût* d'une séquence de déplacements est la somme totale des distances parcourues par chacune des têtes.

Ainsi pour le bloc de requêtes $\langle 5, 2, 4 \rangle$, le coût de la séquence de déplacements $\langle 1, 1, 2 \rangle$ est $5 + 3 + 4 = 12$, alors que le coût de $\langle 1, 2, 1 \rangle$ vaut $5 + 2 + 1 = 8$.

Question 1 Écrire une fonction coutDe(r, d) qui calcule le coût d'une séquence de déplacements d pour le bloc de requêtes r.

Le coût optimal d'une suite de requêtes r est le plus petit coût des séquences de déplacements satisfaisant le bloc de requêtes r.

Question 2 Montrer qu'il existe toujours une séquence de déplacements de coût optimal qui commence par 1, c'est-à-dire commençant par déplacer la première tête.

Question 3 Combien de séquences de déplacements satisfont un bloc de requêtes r donné ?

B. Coût optimal pour deux requêtes

Dans cette partie, le cache est de taille 2 ($n = 2$). Il n'y a donc que deux requêtes r_1 et r_2. Par convention, la première tête sera toujours celle qui bouge sur la première requête.

Question 4 Donner une séquence de déplacements de coût minimal pour chacun des deux blocs de requêtes $\langle 10, 3 \rangle$ et $\langle 3, 10 \rangle$.

Question 5 Écrire une fonction coutOpt2(r_1, r_2) qui retourne un tableau d, de longueur 2, donnant une séquence de déplacements de coût optimal.

Partie II

A. Coût optimal pour trois requêtes

Dans cette partie, le cache est de taille 3 ($n = 3$). Il y a donc trois requêtes r_1, r_2 et r_3. Par convention, la première tête sera toujours celle qui bouge sur la première requête.

Question 6 On suppose que la fonction de la question 5 a été étendue au cas de trois requêtes en appliquant la même règle de décision à la troisième requête qu'à la deuxième requête. L'appliquer en justifiant sur l'exemple $\langle 20, 9, 1 \rangle$.

Question 7 Énumérer toutes les stratégies possibles sur l'exemple de la question précédente. En déduire que l'approche de la question 6 ne fournit pas la solution de coût minimal.

Question 8 Écrire une fonction $\mathsf{coutOpt3}(r_1, r_2, r_3)$ qui retourne un tableau d donnant une séquence de déplacements de coût optimal.

B. Coût optimal pour n requêtes

Dans cette partie, on calcule le coût minimal sans pour autant trouver une séquence de déplacements donnant ce coût. Par commodité, chacune des deux têtes peut effectuer indifféremment le premier déplacement.

On pose $r_0 = 0$ pour coder la position initiale des têtes. À un instant donné, la configuration des têtes du disque dur est représentée par une paire (i, j) codant le numéro des deux dernières requêtes respectivement satisfaites par chacune des deux têtes : la première tête a satisfait en dernier la $i^{ième}$ requête et la deuxième tête la $j^{ième}$ requête. Par convention, la configuration initiale est $(0, 0)$.

À chaque requête r_k, on associe la matrice $(n + 1) \times (n + 1)$ représentée par le tableau d'entiers à deux dimensions cout_k. L'élément $\mathsf{cout}_k[i][j]$ est égal au coût optimal pour atteindre la configuration (i, j), après avoir satisfait la $k^{ième}$ requête. On pose $\mathsf{cout}_k[i][j] = \infty$ si cette configuration n'est pas accessible.

Question 9 Expliquer comment calculer le coût optimal d'une suite de requêtes $\langle r_1, r_2, \ldots, r_n \rangle$ à l'aide du tableau correspondant cout_n.

Question 10 Montrer que les matrices $(\mathsf{cout}_k)_{0 \leq k \leq n}$ satisfont :
$\mathsf{cout}_0[0][0] = 0$ et $\mathsf{cout}_0[i][j] = \infty$ pour tout $i \neq 0$ ou $j \neq 0$;
$\mathsf{cout}_k[i][k]$ est le minimum de $|r_k - r_j| + \mathsf{cout}_{k-1}[i][j]$ pour $0 \leq j \leq n$;
$\mathsf{cout}_k[k][j] = \mathsf{cout}_k[j][k]$;
$\mathsf{cout}_k[i][j] = \infty$ si $i \neq k$ et $j \neq k$.

Question 11 Écrire une procédure $\mathsf{mettreAJour}(\mathsf{cout}, r, k)$ qui met à jour le tableau cout en fonction de la nouvelle requête r_k, de sorte que si cout contenait les valeurs du tableau cout_{k-1}, alors, après la mise à jour, cout contient les valeurs du tableau cout_k.

Question 12 En déduire une fonction $\mathsf{coutOpt}(r)$ permettant de trouver le coût minimal du bloc de n requêtes r. Donner le temps d'exécution de $\mathsf{coutOpt}(r)$ par rapport à n.

La matrice cout est très creuse. Après avoir satisfait la $k^{ième}$ requête, seule la $k^{ième}$ ligne et la $k^{ième}$ colonne peuvent contenir des valeurs différentes de ∞. De plus, comme la matrice cout est symétrique, seule la $k^{ième}$ ligne est à retenir.

Question 13 Écrire une nouvelle fonction $\mathsf{coutOpt}(r)$ qui calcule le coût minimal du bloc de n requêtes r en n'utilisant qu'un tableau cout à une dimension de taille $n+1$. Évaluer son nouveau temps d'exécution.

Indication : On remarquera que pour parvenir à la configuration (i, k), avec $i < k - 1$, nécessairement on doit venir de la configuration $(i, k - 1)$, en revanche pour la configuration $(k - 1, k)$ on peut provenir de n'importe quelle configuration $(k - 1, j)$.

* *
*