$1^{ère}$ année Magistère Informatique
Algorithmique 2007 - 2008

TD 6

Exercice 1. Nous rappelons le problème vu en cours qui consiste à faire l'appoint pour une somme donnée dans un système monétaire donné en minimisant le nombre de pièces :

Données : les valeurs des pièces $1 = v_1 < v_2 < \cdots < v_k$ et la somme n.

Solution : $n_1, n_2, \ldots, n_k$ tels que $n = n_1v_1 + n_2v_2 + \cdots + n_kv_k$.

Critère d'évalation : nombre de pièces.

L'algorithme glouton vu en cours est-il optimal pour les systèmes monétaires suivant ?

- $v_1 = 1$ et $v_i \geq 2v_{i-1}$ pour $1 \leq i \leq k$
- 1, 2, 5, 10, 20, 50, 100 (Euros)
- 1, 5, 10, 25, 50, 100 (US)
- 1, $2\frac{1}{2}$, 5, 10, 25, 50 (Portugais)
  Attention : n est toujours entier.
- 1, 3, 6, 12, 24, 30 (Anglais avant le système décimal)

Exercice 2. Le problème du choix d'activités s'énonce comme suit :

Soit un ensemble $S = \{a_1, a_2, \ldots, a_n\}$ de n activités qui veulent utiliser une ressource, par exemple une salle de conférences qui ne peut servir qu'à une seule activité à la fois. Chaque activité a_i a une heure de début d_i et une heure de fin f_i, avec $0 \leq d_i < f_i < \infty$. Si elle est sélectionnée, l'activité a_i a lieu pendant l'intervalle de temps $[d_i, f_i[$. Les activités a_i et a_j sont *compatibles* si les intervales $[d_i, f_i[$ et $[d_j, f_j[$ ne se chevauchent pas (*i.e.* $d_i \geq f_j$ ou $d_j \geq f_i$). Le *problème du choix d'acivités* consiste à choisir un sous-ensemble de taille maximale, d'activités mutuellement compatibles. Donner un algorithme qui résolve ce problème.

Exercice 3. Une *tâche de durée unitaire* est un travail, par exemple un programme à lancer sur un ordinateur, qui demande exactement une unité

de temps pour s'exécuter. Etant donné un ensemble fini E de tâches unitaires, un *ordonnancement* de E est une permutation de E spécifiant l'ordre dans lequel ces tâches doivent être effectuées. La première tâche de l'ordonnancement commence au temps 0 et se termine au temps 1, la deuxième commence au temps 1 et se termine au temps 2, *etc.*

Voici les entrées du *problème de l'ordonnancement de tâches de durées unitaire, avec dates d'échéance et pénalités, sur un processeur unique* :

- Un ensemble $E = \{1, 2, \ldots, n\}$ de n tâches unitaires,
- Un ensemble de n *dates d'échéance* $d_1, d_2, \ldots, d_n$ tel que chaque d_i vérifie $1 \leq d_i \leq n$ et chaque tâche est censée se terminer à la date d_i,
- Un ensemble de n *pénalités* positives $w_1, w_2, \ldots, w_n$, tel qu'une pénalité w_i survient si la tâche i n'est pas terminée à la date d_i.

On souhaite trouver un ordonnancement de E qui minimise le total des pénalités.

Exercice 4. Etant donné un ensemble $\{x_1, x_2, \ldots, x_n\}$ de n points sur une droite, décrire un algorithme qui détermine le plus petit ensemble d'intervalles fermés de longueur 1 qui contienne tous les points donnés. Prouver la correction de votre algorithme et donner sa complexité.

Exercice 5. Etant donnée une séquence $X = \langle x_1, \ldots, x_m \rangle$, une séquence $Z = \langle z_1, \ldots, z_k \rangle$, $k \leq m$, est une *sous-séquence* de X s'il existe une séquence $\langle i_1, \ldots, i_k \rangle$ strictement croissante d'indices de X tels que $\langle x_{i_1}, \ldots, x_{i_k} \rangle = \langle z_1, \ldots, z_k \rangle$.

Soient X et Y deux séquences, on dit que Z est une *sous-séquence commune* de X et de Y si Z est une sous-séquence de X et de Y.

Le *problème de la plus longue sous-séquence commune* est le suivant :

> Entrées : $X = \langle x_1, \ldots, x_m \rangle$ et $Y = \langle y_1, \ldots, y_n \rangle$.
>
> Sortie : Z : sous-séquence commune à X et à Y de longueur maximale.

Donner un algorithme qui résolve le problème de la plus longue sous-séquence commune.

Exercice 6. Problème du plus grand carré de 1.

> Entrée : une matrice M de taille $n \times m$ où les coefficients valent 0 ou 1.
>
> Sortie : la largeur maximum L d'un carré C de 1 dans M, ainsi que les coordonnées (i, j) du coin en haut à gauche d'un tel carré.

Donner un algorithme qui résolve le problème ci-dessus. Quelle est sa complexité ?

Exercice 7. Donner un algorithme en $\mathcal{O}(n\log\ n)$ pour trouver la plus longue sous-séquence monotone croissante d'une séquence de n nombres.

Exercice 8. On veut construire une tour la plus haute possible à partir de différentes briques. On dispose de n types de briques et d'un nombre illimité de briques de chaque type. Chaque brique de type i est un parallélépipède de taille (x_i, y_i, z_i) et peut être orientée dans tous les sens, deux dimensions formant la base et la troisième dimension formant la hauteur. Dans la construction de la tour, une brique ne peut être placée au dessus d'une autre que si les deux dimensions de la base de la brique du dessus sont strictement inférieures aux dimensions de la base de la brique du dessous.

Exercice 9. Problème de l'impression équilibrée d'un paragraphe sur une imprimante.

Le texte d'entrée est une séquence de n mots de longueurs l_1,l_2,. . .,l_n (mesurées en caractères). On souhaite imprimer ce paragraphe de manière équili-brée sur un certain nombre de lignes qui contiennent un maximum de M caractères chacune. Le critère d'équilibre est le suivant. Si une ligne donnée contient les mots i à j (avec $i \leq j$) et qu'on laisse exactement un espace entre deux mots, le nombre de caractères d'espacement supplémentaires à la fin de la ligne est $M - j + i - \sum_{k=i}^{k=j} l_k$, qui doit être positif ou nul pour que les mots tiennent sur la ligne. L'objectif est de minimiser la somme, sur toutes les lignes hormis la dernière, des cubes des nombres de caractères d'espacement présents à la fin de chaque ligne. Donner un algorithme de programmation dynamique permettant d'imprimer de manière équilibrée un paragraphe de n mots sur une imprimante.