Algorithmique TD4
L3 Informatique – ENS Cachan

8 octobre 2018

Exercice 1

On suppose qu'on a un système monétaire constitué de pièces de valeurs $1, p, p^2, \ldots, p^k$ pour $p \geq 2$ et $k \geq 0$.

1. Démontrer que l'algorithme glouton qui fait l'appoint en prenant en priorité les pièces les plus grosses est optimal, i.e., qu'il minimise le nombre de pièces utilisées.
2. Donner un exemple de système monétaire où l'algorithme glouton n'est pas optimal.
3. Pour un système monétaire quelconque, montrer comment résoudre le problème en temps $O(Mk)$ où M est le montant à payer et k est le nombre de pièces de monnaie.

Exercice 2

Décrire un algorithme efficace qui, étant donné un ensemble $\{x_1, \ldots, x_n\}$ de points sur une droite détermine le plus petit ensemble d'intervalles fermés de longueur 1 qui contiennent tous les points donnés.

Exercice 3

1. On considère le problème suivant, dit du sac à dos : On dispose de n objets ayant chacun une valeur v_i et un poids w_i, et d'un sac à dos pouvant contenir un poids total de W. Donner un algorithme permettant de trouver une remplissage optimal du cas à dos (i.e. qui maximise la valeur totale des objets dans le sac à dos), et analyser sa complexité en temps et en espace.
2. On considère maintenant la variante du problème du sac à dos, ou l'on est capable de prendre une fraction quelconque (entre 0 et 1) de chaque objet. Donner un algorithme pour pour cette variante du problème, et analyser sa complexité en temps et en espace.

Exercice 4

On considère le problème d'ordonnancement des tâches suivant. On dispose de n tâches $a_1, \ldots, a_n$ dites unitaires (i.e. chaque tâche prend exactement une unité de temps à effectuer). À chaque tâche sont associées une date d'échéance $1 \leq d_i \leq n$, et une pénalité w_i qui survient si et seulement si la tâche a_i est exécutée à une date postérieure à d_i. On cherche à trouver un ordonnancement des tâches E (c'est à dire une permutation de la liste des tâches) qui minimise le total des pénalités.

1. On dit qu'une tâche est en retard si elle se termine après sa date d'échéance, et en avance dans le cas contraire. Montrer qu'on peut toujours mettre un ordonnancement sous forme en avance d'abord, i.e. ou toutes les tâches en avance précèdent les tâches en retard.
2. Montrer que tout ordonnancement sous forme en avance d'abord, peut se mettre sous une forme canonique ou les tâches en avance sont triées par ordre croissant de date d'échéance.

On dit qu'un ensemble de tâches est indépendant s'il existe un ordonnancement de ses tâches tel qu'aucune ne soit en retard. On note $\mathcal{I}$ l'ensemble de tous les ensembles indépendants de tâches.

Étant donné un ensemble F de tâches, pour $1 \leq t \leq n$, on note $N_t(F)$ le nombre de tâches de F dont la date d'échéance est inférieure à t.

3. Montrer l'équivalence des propositions suivantes :
   (a) F est indépendant
   (b) $\forall 1 \leq t \leq n, N_t(F) \leq t$
   (c) Si les tâches de F sont ordonnancées par ordre monotone croissant de dates d'échéance, alors aucune des tâches n'est en retard.
4. Montrer que $(E, \mathcal{I})$ est un matroïde
5. En déduire un algorithme permettant de trouver l'ordonnancement optimal d'un ensemble de tâches.

Exercice 5

Soit Σ un alphabet fini et de cardinal supérieur ou égal à deux. On appelle codage binaire une application injective α de l'alphabet Σ dans $\{0,1\}^*$. En utilisant l'opération concaténation, α s'étend de manière naturelle en $\alpha : \Sigma^* \to \{0,1\}^*$.

Un codage est dit préfixe si aucune lettre n'est codée par un mot de code qui est préfixe du codage d'une autre lettre.

1. Montrer que pour un codage préfixe, α est injective sur Σ^*.
2. Montrer qu'on peut représenter un codage préfixe par un arbre binaire dont les feuilles sont les lettres de l'alphabet.

On dit qu'un codage est de longueur fixe quand toutes les lettres sont codées par un mot de code de même longueur. Mais on obtient des codes plus efficaces en associant des codes plus courts aux lettres qui apparaissent le plus fréquemment, quitte à devoir rallonger les codes des lettres qui apparaissent peu fréquemment.

On associe à chaque lettre a de Σ une fréquence d'apparition $f(a)$. Cette fréquence est généralement estimée à partir d'un ensemble de textes représentatif de la langue considérée.

On définit alors le coût d'un codage préfixe par la somme :

$$\sum_{a \in \Sigma} f(a) \cdot |\alpha(a)|$$

et on cherche un codage qui minimise ce coût.

3. Montrer qu'à un codage préfixe optimal correspond un arbre binaire où tout nœud interne a deux fils.
4. Montrer qu'il existe un codage préfixe optimal pour lequel les deux lettres dont le nombre d'occurences est le plus faible sont sœurs dans l'arbre.

Étant données x et y les deux lettres dont le nombre d'occurences est le plus faible dans w, on considère l'alphabet $\Sigma' = (\Sigma \setminus \{x, y\}) \cup z$ où z est une nouvelle lettre à laquelle on associe $f(z) = f(x) + f(y)$.

5. Soit T' l'arbre d'un codage optimal pour Σ', montrer que l'arbre T obtenu à partir de T' en remplaçant la feuille associée à z par un nœud interne ayant x et y comme feuilles représente un codage optimal pour Σ.
6. En déduire un algorithme recherchant un codage optimal et donner sa complexité.