

# Algorithmique TD5

## L3 Informatique – ENS Cachan

Guillaume Bury

16 octobre 2017

### Exercice 1

On suppose qu'on a un système monétaire constitué de pièces de valeurs  $1, p, p^2, \dots, p^k$  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.
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  $\{x_1, \dots, x_n\}$  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 sac 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, où l'on est capable de prendre une fraction quelconque (entre 0 et 1) de chaque objet. Donner un algorithme pour cette variante du problème, et analyser sa complexité en temps et en espace.

### Exercice 4

On considère le problème d'ordonnement des tâches suivant. On dispose de  $n$  tâches  $a_1, \dots, a_n$  dites unitaires (i.e chaque tâche prends exactement une unité de temps à effectuer). A chaque tâche est associé 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 exécuté à une date postérieur à  $d_i$ . On cherche à trouver un ordonancement 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 ordonancement sous forme en avance d'abord, i.e ou toutes les tâches en avance précèdent les tâches en retard.
2. Montrer que tout ordonancement sous forme en avance d'abord, peut se mettre sous une forme canonique où les tâches en avance sont triées par ordre croissant de date d'échéance
3. La suite en TD...

## Exercice 5

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

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,  $\alpha$  est injective sur  $\Sigma^*$ .
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  $\Sigma$  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'occurrences est le plus faible sont sœurs dans l'arbre.

Étant données  $x$  et  $y$  les deux lettres dont le nombre d'occurrences 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  $\Sigma'$ , 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  $\Sigma$ .
6. En déduire un algorithme recherchant un codage optimal et donner sa complexité.