# Algorithmique TD4
# L3 Informatique – ENS Cachan

10 août 2020

## Exercice 1

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 2

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 $\alpha : \Sigma^* \to \{0,1\}^*$.

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, $\alpha$ 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'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, 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.

## Exercice 3

On considère un alphabet $\Sigma$ tel que $\bullet \notin \Sigma$. On note $\Sigma_\bullet = \Sigma \cup \{\bullet\}$. On dispose d'une distance $d : \Sigma_\bullet \times \Sigma_\bullet \to \mathbb{N}$. Cette distance s'étend aux mots $x, y$ de même longueur par la formule $d(x, y) = \sum_{i \leq |x|} d(x[i], y[i])$.
Soit $x$ un mot de $\Sigma^*$ de longueur $m$, $k \geq m$ et $\alpha$ une fonction strictement croissante de $[1, m]$ dans $[1, k]$. Le mot $x_\alpha$ de longueur $k$ est défini ainsi : pour tout $i \leq m$, $x_\alpha[\alpha(i)] = x[i]$ et pour $i \leq k$ tel que $\alpha^{-1}(i) = \emptyset$, $x_\alpha[i] = \bullet$.
Soit $y$ un mot de $\Sigma^*$ de longueur $n$. On définit la *distance d'édition* entre $x$ et $y$ par :

$$\begin{aligned}\Delta(x, y) = \min(d(x_\alpha, y_\beta) \mid & \max(m, n) \leq k \leq m + n, \\ & \alpha : [1, m] \to [1, k], \alpha \text{ strictement croissante} \\ & \beta : [1, n] \to [1, k], \beta \text{ strictement croissante })\end{aligned}$$

1. Démontrer que pour tout $1 \leq i \leq m$ et $1 \leq j \leq n$,

$$\begin{aligned}\Delta(x[1, i], y[1, j]) = \min(&\Delta(x[1, i-1], y[1, j]) + d(x[i], \bullet), \\ &\Delta(x[1, i], y[1, j-1]) + d(\bullet, y[j]), \\ &\Delta(x[1, i-1], y[1, j-1]) + d(x[i], y[j]))\end{aligned}$$

2. Proposer un algorithme (basé sur la programmation dynamique) qui calcule $\Delta(x, y)$. Etablir sa complexité en fonction de $m$ et $n$.

On définit $e(i, j)$ pour $0 \leq i \leq m$ et $0 \leq j \leq n$ par :

$$e(i, j) = \min(\Delta(x[1, i], y[k, j]) \mid 1 \leq k \leq j + 1)$$

i.e. le minimum de la distance d'édition entre $x[1, i]$ et un suffixe de $y[1, j]$.

3. Démontrer que pour tout $1 \leq i \leq m$ et $1 \leq j \leq n$,

$$\begin{aligned}e(i, j) = \min(&e(i-1, j) + d(x[i], \bullet), \\ &e(i, j-1) + d(\bullet, y[j]), \\ &e(i-1, j-1) + d(x[i], y[j]))\end{aligned}$$

4. Proposer un algorithme (basé sur la programmation dynamique) qui renvoie un suffixe de $y$ qui minimise la distance d'édition avec $x$. Etablir sa complexité en fonction de $m$ et $n$.
5. Proposer un algorithme basé sur l'algorithme précédent qui renvoie un sous-mot $y[k, j]$ (avec $0 \leq k \leq j + 1 \leq n + 1$) qui minimise la distance d'édition avec $x$. Etablir sa complexité en fonction de $m$ et $n$.

## Exercice 4

Étant donné un ensemble $\{x_1, \ldots, x_n\} \subset \mathbb{R}$ de points sur une droite, on cherche à déterminer le plus petit ensemble d'intervalles fermés de longueur 1 qui contiennent tous les points donnés.

1. Montrez qu'il existe une solution optimale dans laquelle la borne inférieure de chaque intervalle est dans $\{x_1, \ldots, x_n\}$.
2. Proposez un algorithme permettant de résoudre ce problème et calculez sa complexité.