# TD 3 – Algorithmes gloutons & programmation dynamique

**Exercice 1.** *À la recherche des tables de multiplication*

Soient deux entiers $x$ et $y$ codés sur $n$ bits (on supposera que $n$ est une puissance de 2). On souhaite multiplier ces deux entiers entre eux, en travaillant au niveau des bits. La multiplication de deux entiers de $n$ bits se fait de manière triviale en $\mathcal{O}(n^2)$, la multiplication d'un entier par une puissance de 2, et l'addition de 2 entiers se font en temps linéaire $\mathcal{O}(n)$.

1. La méthode diviser pour régner n'est pas toujours meilleure qu'un algorithme naïf. Pour illustrer cela, donnez un algorithme diviser pour régner ayant une complexité en $\mathcal{O}(n^2)$ pour multiplier deux entiers $x$ et $y$ de $n$ bits chacun.
2. Proposez un algorithme diviser pour régner ayant une complexité inférieure à $\mathcal{O}(n^2)$. Donnez la formule de récurrence pour votre algorithme et sa complexité finale (en $\mathcal{O}$). On supposera pour simplifier que l'addition/multiplication de deux nombres de taille $n$ est un nombre de taille $n$.

**Exercice 2.** *Codage de Huffman*

Soit $\Sigma$ un alphabet fini de cardinal au moins deux. Un *codage binaire* est une application injective de $\Sigma$ dans l'ensemble des suites finies de 0 et de 1 (les images des lettres de $\Sigma$ sont appelées *mots de code*). Il s'étend de manière naturelle par concaténation en une application définie sur l'ensemble $\Sigma^\star$ des mots sur $\Sigma$. Un codage est dit *de longueur fixe* si toutes les lettres dans $\Sigma$ sont codées par des mots binaires de même longueur. Un codage est dit *préfixe* si aucun mot de code n'est préfixe d'un autre mot de code.

1. Le décodage d'un codage de longueur fixe est unique. Montrer qu'il en est de même pour un codage préfixe.
2. Expliquer comment représenter un codage préfixe par un arbre binaire dont les feuilles sont les lettres de l'alphabet. Donner un exemple.

On considère un texte dans lequel chaque lettre $c$ apparaît avec une fréquence $f(c)$ non nulle. À chaque codage préfixe de ce texte, représenté par un arbre $T$, est associé un coût défini par

$$B(T) = \sum_{c \in \Sigma} f(c) l_T(c)$$

où $l_T(c)$ est la taille du mot binaire codant $c$. Si $f(c)$ est exactement le nombre d'occurrences de $c$ dans le texte, alors $B(T)$ est le nombre de bits du codage du texte.
Un codage préfixe représenté par un arbre $T$ est *optimal* si, pour ce texte, il minimise la fonction $B$.

3. Montrer qu'à un codage préfixe optimal correspond un arbre binaire où tout nœud interne a deux fils. Montrer qu'un tel arbre a $|\Sigma|$ feuilles et $|\Sigma| - 1$ nœuds internes.
4. **(a)** *Propriété de choix glouton*. Montrer qu'il existe un codage préfixe optimal pour lequel les deux lettres de plus faibles fréquences sont soeurs dans l'arbre (autrement dit leurs codes sont de même longueur et ne diffèrent que par le dernier bit).

   Étant donnés $x$ et $y$ les deux lettres *de plus faibles fréquences* dans $\Sigma$, on considère l'alphabet $\Sigma' = \Sigma - \{x, y\} + \{z\}$, où $z$ est une nouvelle lettre à laquelle on donne la fréquence $f(z) = f(x) + f(y)$. Soit $T'$ l'arbre d'un codage optimal pour $\Sigma'$.

   **(b)** *Propriété de sous-structure optimale*. 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 $\Sigma$.
5. En déduire un algorithme pour trouver un codage optimal et donner sa complexité. À titre d'exemple, trouver un codage optimal pour $\Sigma = \{a, b, c, d, e, g\}$ et $f(a) = 45$, $f(b) = 13$, $f(c) = 12$, $f(d) = 16$, $f(e) = 9$ et $f(g) = 5$.

**Exercice 3.** *Impression équilibrée*

Le problème est l'impression équilibrée d'un paragraphe sur une imprimante. Le texte d'entrée est une séquence de $n$ mots de longueurs $\ell_1, \ell_2, \ldots, \ell_n$ (mesurées en caractères). On souhaite imprimer ce paragraphe de manière équilibrée 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 une espace [1] entre deux mots, le nombre de caractères d'espacement supplémentaires à la fin de la ligne est $M - j + i - \sum_{k=i}^{j} \ell_k$, 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.

1. Est-ce que l'algorithme glouton consistant à remplir les lignes une à une en mettant à chaque fois le maximum de mots possibles sur la ligne en cours, fournit l'optimum ?
2. Donner un algorithme de programmation dynamique résolvant le problème. Analyser sa complexité en temps et en espace.
3. Supposons que pour la fonction de coût à minimiser, on ait simplement choisi la somme des nombres de caractères d'espacement présents à la fin de chaque ligne. Est-ce que l'on peut faire mieux en complexité que pour la question 2 ?
4. (*Plus informel*) Qu'est-ce qui à votre avis peut justifier le choix de prendre les cubes plutôt que simplement les nombres de caractères d'espacement en fin de ligne ?

**Exercice 4.** *Gloutonneries matroïdesques*

**Définition.** Soit $S$ un ensemble fini et $\mathcal{I}$ une famille de parties de $S$. Alors $(S, \mathcal{I})$ est un *matroïde* si
- — hérédité : pour tout $X \in \mathcal{I}$, pour tout $Y \subset X$, $Y \in \mathcal{I}$ ;
- — échange : $\forall X, Y \in \mathcal{I}$ tels que $|X| < |Y|$, $\exists x \in Y \setminus X$ tel que $X \cup \{x\} \in \mathcal{I}$.

Les éléments de $\mathcal{I}$ sont appelés les *indépendants* du matroïde.

1. Montrer que si $(S, \mathcal{I})$ est un matroïde et $T$ une partie de $S$, alors si $X$ et $Y$ sont deux indépendants de $S$ inclus dans $T$ et qu'ils sont maximaux pour l'inclusion dans $T$, alors $|X| = |Y|$.

Soit $S$ un ensemble fini et $\mathcal{I}$ un ensemble de parties de $S$. Une *fonction de coût* pour $S$ est une fonction $c : S \to \mathbb{R}_+$. Elle est naturellement étendue à $\mathcal{I}$ en posant $c(X) = \sum_{x \in X} c(x)$. On considère l'algorithme suivant :

**Algorithme :** `Glouton`$(S, \mathcal{I}, c)$

```
1 Ordonner les éléments de S = {s_1, ..., s_n} par coût décroissant
2 X ← ∅
3 pour i de 1 à n faire
4     si X ∪ {s_i} ∈ I alors
5         X ← X ∪ {s_i}
```

2. Montrer que si `Glouton` trouve un ensemble $X \in \mathcal{I}$ de coût maximal quelque soit la fonction de coût $c$, alors $(S, \mathcal{I})$ est un matroïde.
3. Soit $(S, \mathcal{I})$ un matroïde. On considère une fonction de coût $c$ ainsi qu'un ensemble de coût maximal $X_{\text{opt}} \in \mathcal{I}$. On suppose que `Glouton` renvoie $X$ avec $c(X) < c(X_{\text{opt}})$.
   **(a)** Montrer qu'on peut supposer que $|X| = |X_{\text{opt}}|$.
   **(b)** On note $X = \{x_1, \ldots, x_p\}$ et $X_{\text{opt}} = \{y_1, \ldots, y_p\}$, rangés par coût décroissant. Montrer que $c(x_1) \geq c(y_1)$.
   **(c)** Soit $i$ le plus petit indice tel que $c(x_i) < c(y_i)$, et $Y = \{s \in S : c(s) \geq c(y_i)\}$. Montrer que $\{x_1, \ldots, x_{i-1}\}$ est un indépendant maximal pour l'inclusion dans $Y$.

1. En typographie *espace* est un mot féminin.

**(d)** Conclure.

**Exercice 5.** *Bibliothèque*

La bibliothèque planifie son déménagement. Elle comprend une collection de $n$ livres $b_1, b_2, \ldots, b_n$. Le livre $b_i$ est de largeur $w_i$ et de hauteur $h_i$. Les livres doivent être rangés dans l'ordre donné (par valeur de $i$ croissante) sur des étagères identiques de largeur $L$.

1. On suppose que tous les livres ont la même hauteur $h = h_i, 1 \leq i \leq n$. Montrer que l'algorithme glouton qui range les livres côte à côte tant que c'est possible minimise le nombre d'étagères utilisées.
2. Maintenant les livres ont des hauteurs différentes, mais la hauteur entre les étagères peut se régler. Le critère à minimiser est alors l'encombrement, défini comme la somme des hauteurs du plus grand livre de chaque étagère utilisée.
   **(a)** Donner un exemple où l'algorithme glouton précédent n'est pas optimal.
   **(b)** Proposer un algorithme optimal pour résoudre le problème, et donner son coût.
3. On revient au cas où tous les livres ont la même hauteur $h = h_i, 1 \leq i \leq n$. On veut désormais ranger les $n$ livres sur $k$ étagères de même longueur $L$ à minimiser, où $k$ est un paramètre du problème. Il s'agit donc de partitionner les $n$ livres en $k$ tranches, de telle sorte que la largeur de la plus large des $k$ tranches soit la plus petite possible.
   **(a)** Proposer un algorithme pour résoudre le problème, et donner son coût en fonction de $n$ et $k$.
   **(b)** On suppose maintenant que la la taille d'un livre est en $2^{o(kn)}$. Trouver un algorithme plus rapide que le précédent pour répondre à la même question.