TD 07 – Carcajou (Gulo gulo)

Exercice 1. *Sortez couverts !*

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

Exercice 2. *Algorithme de Kruskal (1956)*

Soit $G = (V, E)$ un graphe non orienté. On note $\mathcal{F}$ l'ensemble des forêts de G, c'est-à-dire $\mathcal{F} = \{F \subseteq E : \text{le graphe } (V, F) \text{ est sans cycle}\}$.

1. Montrer que $(E, \mathcal{F})$ est un matroïde.

On suppose qu'on dispose d'une fonction de poids $w : E \to \mathbb{R}_+$ sur les arêtes du graphe.

2. Déduire de la question précédente un algorithme glouton pour calculer un arbre couvrant de poids minimal dans un graphe, c'est-à-dire un ensemble $T \subseteq E$ d'arêtes de poids minimal tel que le graphe (V, T) est un arbre.

Exercice 3. *« Nous ne travaillons qu'à remplir la mémoire » (Montaigne)*

On souhaite enregistrer sur une mémoire de taille L un groupe de fichiers $P = (P_1, \dots, P_n)$. Chaque fichier P_i nécessite une place a_i. Supposons que $\sum a_i > L$: on ne peut pas enregistrer tous les fichiers. Il s'agit donc de choisir le sous ensemble Q des fichiers à enregistrer.
On pourrait souhaiter le sous-ensemble qui contient le plus grand nombre de fichiers. Un algorithme glouton pour ce problème pourrait par exemple ranger les fichiers par ordre croissant des a_i.
Supposons que les P_i soient ordonnés par taille ($a_1 \leq \dots \leq a_n$).

1. Écrivez un algorithme (en pseudo-code) pour la stratégie présentée ci-dessus. Cet algorithme doit renvoyer un tableau booléen S tel que $S[i] = 1$ si P_i est dans Q et $S[i] = 0$ sinon. Quelle est sa complexité en nombre de comparaisons et en nombre d'opérations arithmétiques ?
2. Montrer que cette stratégie donne toujours un sous-ensemble Q maximal tel que $\sum_{P_i \in Q} a_i \leq L$.
3. Soit Q le sous-ensemble obtenu. À quel point le quotient d'utilisation $(\sum_{P_i \in Q} a_i)/L$ peut-il être petit ?

Supposons maintenant que l'on souhaite enregistrer le sous-ensemble Q de P qui maximise ce quotient d'utilisation, c'est-à-dire celui qui remplit le plus de disque. Une approche *gloutonne* consisterait à considérer les fichiers dans l'ordre décroissant des a_i et, s'il reste assez d'espace pour P_i, on l'ajoute à Q.

4. On suppose toujours les P_i ordonnés par taille croissante. Écrivez un algorithme pour cette nouvelle stratégie.
5. Montrer que cette nouvelle stratégie ne donne pas nécessairement un sous-ensemble qui maximise le quotient d'utilisation. À quel point ce quotient peut-il être petit ? Prouvez-le.

Exercice 4. *Codage de Huffman (1952)*

Soit Σ un alphabet fini de cardinal au moins deux. Un *codage binaire* est une application injective de Σ dans l'ensemble des suites finies de 0 et de 1 (les images des lettres de Σ 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 Σ. Un codage est dit *de longueur fixe* si toutes les lettres dans Σ 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 Σ, 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 Σ'.

   (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 Σ.

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 $f(g) = 5$.