TD 04 – Algorithmes gloutons

(Matroide) **Exercice 1.** *Matroides*

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`$(S, \mathcal{I}, c)$

```
Ordonner les éléments de S = {s_1, ..., s_n} par coût décroissant
X ← ∅
pour i de 1 à n faire
    si X ∪ {s_i} ∈ I alors
        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 $X = \{x_1, \dots, x_p\}$ et $X_{\text{opt}} = \{y_1, \dots, y_p\}$, 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 $\{x_1, \dots, x_{i-1}\}$ est un indépendant maximal pour l'inclusion dans Y.
   - **(d)** Conclure.

(Kruskal) **Exercice 2.** *Algorithme de Kruskal*

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.

(Couverture) **Exercice 3.** *Couverture par intervalles*

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.

(Memoire) **Exercice 4.** *Utilisation de la memoire*

On souhaite enregistrer sur une mémoire de taille L un groupe de fichiers $P = (P_1, \ldots, P_n)$. 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 ($a_1 \leq \ldots \leq a_n$).

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.