DM n°1

Exercice 1. *Plus grand et plus petit de n entiers*

Dans cet exercice, on s'intéresse au calcul (simultané) du maximum et du minimum de n entiers. On mesure la **complexité dans le pire des cas et en nombre de comparaisons** des algorithmes.

1. Donner un algorithme naïf et sa complexité.

Une idée pour améliorer l'algorithme est de regrouper *par paires* les éléments à comparer, de manière à diminuer ensuite le nombre de comparaisons à effectuer.

2. Décrire un algorithme fonctionnant selon ce principe et analyser sa complexité.

Nous allons étudier l'optimalité d'un tel algorithme en fournissant une borne inférieure sur le nombre de comparaisons à effectuer. Nous utiliserons la méthode de l'*adversaire*.
Soit A un algorithme qui trouve le maximum et le minimum. Pour une donnée fixée, au cours du déroulement de l'algorithme, on appelle *novice* (N) un élément qui n'a jamais subi de comparaisons, *gagnant* (G) un élément qui a été comparé au moins une fois et a toujours été supérieur aux éléments auxquels il a été comparé, *perdant* (P) un élément qui a été comparé au moins une fois et a toujours été inférieur aux éléments auxquels il a été comparé, et *moyens* (M) les autres. Le nombre de ces éléments est représenté par un quadruplet d'entiers (i, j, k, l) qui vérifie bien sûr $i + j + k + l = n$.

3. Donner la valeur de ce quadruplet au début et à la fin de l'algorithme. Exhiber une stratégie pour l'adversaire, de sorte à maximiser la durée de l'exécution de l'algorithme. En déduire une borne inférieure sur le nombre de tests à effectuer.

Exercice 2. *Matrices de tœplitz*

Une *matrice de Tœplitz* est une matrice $n \times n$ $(a_{i,j})$ telle que $a_{i,j} = a_{i-1,j-1}$ pour $2 \leq i, j \leq n$.

1. La somme de deux matrices de Tœplitz est-elle une matrice de Tœplitz ? Et le produit ?
2. Trouver un moyen d'additionner deux matrices de Tœplitz en $\mathcal{O}(n)$.
3. Comment calculer le produit d'une matrice de Tœplitz $n \times n$ par un vecteur de longueur n ? Quelle est la complexité de l'algorithme ?

Exercice 3. *Un petit jeu de cartes*

On considère le jeu suivant à deux personnes :
n cartes sont disposées sur la table, face visible. La carte i a la valeur v_i. On suppose les valeurs toutes distinctes. Les joueurs ramassent tour à tour une carte à l'extrémité gauche ou droite de l'alignement de cartes, jusqu'à ce que toutes les cartes aient été ramassées. Le but pour un joueur est de ramasser des cartes telles que la somme des valeurs soit la plus grande possible.

1. Montrer que la stratégie gloutonne, qui choisit toujours parmi les deux cartes possibles celle de plus grande valeur, ne maximise pas nécessairement la valeur totale des cartes ramassées par le premier joueur.
2. Donner un algorithme en $O(n^2)$ pour calculer la stratégie optimale du premier joueur, c'est-à-dire celle qui lui garantit une valeur totale maximale, même si le deuxième joueur joue de façon optimale.

Exercice 4. *Codage de Huffman*

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 $f(g) = 5$.