TD 14 – Révisions

(FFT) **Exercice 1.** *Multiplication rapide de polynômes*

Pour simplifier cet exercice, on travaille avec des polynômes dans $\mathbb{C}[X]$ et on évaluera la complexité des algorithmes par le nombres d'opérations nécessaires dans $\mathbb{C}$ (addition, soustraction, multiplication et division). La complexité des algorithmes sera évaluée en fonction des degrés des polynômes en entrée.

1. Donner un algorithme naïf pour multiplier deux polynômes. Quelle est sa complexité ?
2. En vous inspirant de l'algorithme de multiplication d'une matrice de Toeplitz par un vecteur vu au partiel, décrivez un algorithme s'effectuant en $O(n^{log_2(3)})$.
   (*Indice :* commencez par essayer de multiplier deux polynômes de dégré 1).
3. Soit P un polynôme de degré n et $a_0, \dots, a_n \in \mathbb{C}$ deux à deux distincts. Exprimer P en fonction de $P(a_0), \dots, P(a_n)$. En déduire un algorithme pour calculer les coefficients de P à partir de la donnée des $P(a_0), \dots, P(a_n)$. Quelle en est la complexité ?
4. Réciproquement, si P est donné par la liste de ses coefficients, donner un algorithme pour calculer $(P(a_0), ..., P(a_n))$. Quelle en est la complexité ?
5. Supposons qu'on représente désormais des polynômes P et Q de degrés au plus $n/2$ par les vecteurs $(P(a_0), \dots, P(a_n))$ et $(Q(a_0), \dots, Q(a_n)) \in \mathbb{C}^{n+1}$. Quelle est la complexité de multiplier deux polynômes avec cette représentation ?

Nous allons maintenant voir un algorithme pour multiplier des polynômes en $O(nlog(n))$ opérations dans $\mathbb{C}$ en utilisant la transformée de Fourier rapide (FFT). L'idée principale est d'utiliser la remarque précédente pour ramener la multiplication de polynômes à une multiplication point à point en passant par une interpolation. On reviendra à la représentation initiale par une évaluation multipoint. On choisira les points d'évaluations très particuliers que sont les racines n-ième de l'unité. On pose $\omega_n^k = e^{2i\pi k/n}$.

6. Pour simplifier, on suppose que $P = \sum_{k=0}^{n-1} p_k X^k$ en prenant n une puissance de deux. Donner un algorithme diviser-pour-régner qui calcule les coefficients de $\tilde{P}(X) = \sum_{k=0}^{n-1} P(\omega_n^k) X^k$ en $O(nlog(n))$ opérations.
7. Montrer que $p_k = \frac{\tilde{P}(\omega_n^{-k})}{n}$. En déduire comment calculer les coefficients de P à partir de ceux de $\tilde{P}$ et conclure.

(PetitJeu) **Exercice 2.** *Petit Jeu*

On considère le jeu suivant à deux personnes. Une suite de n cartes sont disposée sur la table, face visible. La carte i a la valeur v_i On suppose ces valeurs toutes distinctes. Les joueur ramassent tour à tour une carte à l'extrémité gauche ou droite de l'alignement, jusqu'à ce que toutes les cartes aient été ramassées. Le but pour un joueur est de ramasser des cartes de valeur totale (somme des valeurs) maximum.

1. Montrer que la stratégie gloutonne qui choisit toujours, parmi les deux cartes possibles, celle qui a la valeur la plus élevée, ne maximise pas nécessairement pas 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 valeur totale maximale même si le deuxième joueur joue de façon optimale.

(Recoloriage) **Exercice 3.** *Recoloriage de Mots*

Étant donné un ensemble $\mathcal{C}$ de couleurs et un entier n, on dit qu'une tableau $T : \{1, \dots, n\} \to \mathcal{C}$ est convexe si les positions qui ont une même couleur sont consécutives, *i.e.* si $T(i) = T(j) = c$ avec $i \leq j$ alors pour tout $i \leq k \leq j$ on a $T(k) = c$.
Étant donné T non convexe, on dit qu'une couleur c est problématique s'il existe des positions $i < k < j$ telles que $T(i) = T(j) \neq T(k)$.
Soit T un tableau de taille n, et N le nombre de couleurs problématiques pour T.

1. Donner un algorithme en $O(2^N n^2)$ qui donne en sortie le nombre minimum de valeurs de T à modifier pour rendre T convexe.
2. Le modifier afin qu'il donne en sortie un tableau S convexe minimisant le nombre de i tels que $T(i) \neq S(i)$.