

Programmation impérative et structures de données mutables

Exercice 1 :

On se donne n points du plan notées $(M_i)_{i \in \llbracket 1, n \rrbracket}$, de coordonnées $(x_i, y_i)_{i \in \llbracket 1, n \rrbracket}$, et l'on cherche le couple de points dont la distance est minimale. On représentera les points par un tableau dont les éléments sont de type `float*float`.

1. Écrire une fonction `dist` qui prend en paramètre deux tableaux x et y et deux entiers i et j et qui renvoie la distance entre les points M_i et M_j .
2. Écrire une fonction `dist_min` qui résout le problème en testant toutes les combinaisons possibles. Quelle est sa complexité?

On va utiliser une approche de type "diviser pour régner".

On suppose la famille $(x_i)_{i \in \llbracket 1, n \rrbracket}$ croissante. On sépare le nuage en deux nuages, situés de part et d'autre de la droite d'équation $x = x_0$. On note d_g (resp. d_d) les distances minimales entre les points du sous-nuage gauche (resp. droite) et $d = \min(d_g, d_d)$. On s'intéresse alors aux points du nuage dont l'abscisse est entre $x_0 - d$ et $x_0 + d$ en remarquant que pour trouver les points de cette bande les plus proches, il suffit de parcourir les points dans l'ordre des ordonnées croissantes et de tester les distances avec uniquement les 7 prochains points.

3. Écrire une fonction `indices_bande_oc` prend en paramètre deux tableaux x et y et deux flottants $x_{\min}$ et $x_{\max}$ et qui renvoie un tableau contenant les indices des points dont l'abscisse est comprise entre $x_{\min}$ et $x_{\max}$, trié par ordonnées croissantes.
4. Écrire une fonction `dist_min_DR` qui prend en paramètre deux tableaux x et y et qui renvoie la distance minimale entre deux points du nuage. Quelle est sa complexité?

Exercice 2 : Autour de la multiplication matricielle (X 2022)

1. Écrire une fonction `identite` : `int-> matrix` prenant en entrée un entier n non nul et renvoyant la matrice identité de $\mathcal{M}_n(\mathbb{R})$.
2. Écrire une fonction `scalaire` : `float array -> float array -> float` prenant en entrée deux vecteurs u et v et renvoyant le produit scalaire $u \cdot v$.
3. Écrire une fonction `mul` : `matrix -> matrix -> matrix` qui calcule le produit $A \times B$ des matrices A et B de flottants passées en argument. Quelle est sa complexité en nombre de multiplications?

On suppose maintenant que l'on a plusieurs matrices $A_1, \dots, A_r$ à multiplier.

On note $k_0, \dots, k_n$ les entiers tels $A_i \in \mathcal{M}_{k_{i-1}, k_i}(\mathbb{R})$, et $p_{i,j}$ le nombre minimal de multiplications à réaliser pour calculer le produit $A_i \times \dots \times A_j$.

4. Écrire une fonction `min_mul` : `int array -> int` qui renvoie la valeur $p_{1,n}$ à partir du tableau des $(k_i)_{0 \leq i \leq n}$ passé en argument. On fera une version naïve et une version optimisée grâce au principe de programmation dynamique.
5. Écrire une fonction `mul_opt` : `float array array array -> float array`, qui réalise le produit matriciel du tableau de matrices passée en argument en effectuant $p_{1,n}$ multiplications matricielles.