Les programmes doivent être écrits en langage Python et les requêtes de base de données en langage SQL. Les candidat·e·s sont libres de définir et de programmer toute fonction auxiliaire qu'ils ou elles jugent nécessaires pour répondre aux question posées. On veillera dans ce cas à définir précisément le rôle de chaque fonction introduite, ses paramètres et son résultat. On pourra également utiliser librement les fonctions de la bibliothèque standard Python, en particulier celles du module `math`*.*

Lorsque le sujet demande l'écriture d'une fonction Python, la réponse doit commencer par l'entête de la fonction (instruction `def`*). D'autre part, si le sujet précise que la fonction prend un paramètre d'un certain type ou qui répond à une certaine condition, la fonction n'a pas à vérifier la conformité de l'argument reçu.*

La lisibilité des codes produits, tant en Python qu'en SQL, est un élément important d'appréciation.

Les calculs de complexité doivent être détaillés. On pourra admettre qu'ajouter n fois des éléments à la fin d'une liste initialement vide (avec la fonction `liste.append(element)`*) a une complexité $O(n)$.*

Le sujet comporte 5 pages.

Prologue

Dans un univers inspiré du roman *1984* de George Orwell, les fonctionnaires de la police de la pensée, au service de l'ordinateur Big Brother, surveillent les citoyens et les citoyennes. Big Brother souhaite pouvoir intervenir rapidement si une menace sérieuse se profile (par exemple en cas de regroupement, de colportation de rumeur ou d'impression de livre).

Votre rôle dans ce devoir sera de programmer des fonctions supplémentaires sur Big Brother pour le rendre plus efficace.

Le devoir est constitué de 4 parties très largement indépendantes. Il est conseillé de faire la partie III avant la partie II si vous ne passez pas le concours de l'école Polytechnique.

I Surveillance d'individus

1.1 Analyse grossière

Dans cette partie, on suppose que Big Brother a attribué à chaque individu une note de dangerosité (plus la note est élevée, plus l'individu est dangereux). Ces notes sont stockées dans la liste `notes`, de telle sorte que `notes[i] = k` signifie que l'individu numéro `i` a une note `k`. On dispose également d'une fonction `envoyer_en_prison(i)` qui envoie l'individu numéro `i` en prison [1]. On supposera que toutes les notes sont distinctes.

► **Question 1** Écrire une fonction `eliminer_le_chef(notes)` qui prend en argument une liste de notes et envoie en prison l'individu ayant la note la plus élevée. Cette fonction ne renvoie aucune valeur et ne modifie pas le tableau.

► **Question 2** Écrire une fonction `priorites(notes)` qui renvoie un tableau contenant les mêmes valeurs que `notes`, mais triées par ordre croissant. On n'exige pas un tri efficace ici. Préciser la complexité de l'algorithme utilisé dans le pire cas. Le tableau `notes` ne doit pas être modifié. On n'autorise pas l'utilisation des fonctions `sort` ou `sorted`.

► **Question 3** Écrire une fonction `rompre_sedition(notes)` qui envoie en prison les 10% des individus avec les notes les plus élevées (si le nombre d'individus n'est pas un multiple de 10, on arrondira au nombre entier supérieur).

1. Cette fonction, dont le code est trop complexe pour figurer dans le sujet, consiste essentiellement à émettre un avis de recherche.

1.2 Analyse plus poussée

On s'intéresse dans cette sous-partie à la manière dont Big Brother établit les notes de dangerosité des individus. On dispose pour cela d'une base de donnée SQL qui liste des délits commis par les individus. Cette base est constituée de deux tables :

— La table `citoyens`, qui possède un enregistrement par personne, et dont les attributs sont :
  — `matricule`, le numéro de matricule de chaque individu, clé primaire de la table,
  — `nom`, le nom de l'individu,
  — `profession`, la profession de l'individu,
  — `prison`, valeur entière indiquant le nombre de jours passés en prison par l'individu jusqu'à aujourd'hui.

— La table `incidents`, qui possède un enregistrement par délit commis, et dont les attributs sont :
  — `id`, un identifiant unique, clé primaire de la table,
  — `matricule`, le numéro de matricule du citoyen en faute,
  — `type`, qui peut valoir `'crime'` ou `'terrorisme'`,
  — `date`, valeur entière qui indique l'année du délit.

► **Question 4** Indiquer par quelles requêtes sur cette base (rédigées en SQL) on peut obtenir les valeurs suivantes :

a) Le matricule de l'individu qui a passé le plus de jours en prison.

b) Le nombre de délits de chaque type commis en 1984.

c) Le nombre de délits (toutes catégories confondues) commis par des journalistes.

d) Les noms des trois personnes ayant commis le plus de délits de type terrorisme en 1984.

II Surveillance de groupes

2.1 Méthode naïve

L'objectif de cette partie est de prévenir toute manifestation en envoyant préventivement en prison deux individus qui commenceraient à se rassembler.

Un individu est représenté ici par un triplet de trois valeurs `(i,x,y)` où :

— `i` représente le matricule de l'individu,
— `x` représente l'abscisse de l'individu,
— `y` représente l'ordonnée de l'individu.

On se donne une liste globale `ind` de n individus *triés par ordre de matricule croissants*. Ainsi, dans cette liste, le matricule du 12e individu s'obtient avec `ind[11][0]`. Le tableau de la figure 1 présente un exemple de telle liste.

FIGURE 1 – Exemple de tableau `ind`

	`i=`	0	1	2	...	n
Matricule	`t[i][0]=`	1	7	12	...	45
Abscisse	`t[i][1]=`	1.2	24.1	-1.4	...	23
Ordonnée	`t[i][2]=`	-3	45	2.5	...	-12

► **Question 5** Écrire une fonction `distance(x1,y1,x2,y2)` qui renvoie la distance euclidienne entre le point $M1(x_1, y_1)$ et le point $M_2(x_2, y_2)$.

► **Question 6** Écrire une fonction `position(i)` qui renvoie le couple de coordonnées de l'individu possédant le matricule `i`, dont on suppose qu'il figure dans le tableau `ind`. On garantira une complexité en $O(\ln n)$, et on justifiera cette complexité.

Ainsi, avec le tableau précédent, on souhaite que `position(7)` renvoie `(24.1,45)`.

► **Question 7** Écrire une fonction naïve `plusproches()` qui renvoie la paire de matricules correspondant aux deux individus les plus proches de la liste `ind`. Quelle est la complexité de cette fonction ?

2.2 Méthode sophistiquée

On suppose dans cette partie qu'on dispose de deux nouveaux tableaux globaux `ind_x` et `ind_y` contenant les mêmes individus (ou triplets) que `ind` mais rangés respectivement par valeur de `x` ou de `y` croissantes. On admettra qu'il est possible d'obtenir ces tableaux une fois pour toutes avec une complexité $O(n \ln n)$.

► **Question 8** Écrire une fonction `partie_gauche(a)` qui renvoie la liste des individus dont l'abscisse est inférieure ou égale à a, *triés par abscisses croissante* avec une complexité $O(n)$.

On supposera dans la suite qu'on dispose également d'une fonction `partie_droite(a)` qui renvoie la liste des individus d'abscisse strictement supérieure à a, triés également par abscisses croissantes.

Pour la mise en place de l'algorithme de la paire de points la plus proche, on procède par la méthode du *diviser pour régner*, illustré sur la figure 2 :

i - On sépare les individus en deux parties (qu'on supposera de tailles égales) suivant la médiane des abscisses, qu'on notera x_0.

ii - On calcule récursivement le couple le plus proche dans la moitié gauche et dans la moitié droite, on note d_0 la plus petite de ces deux distances.

iii - On cherche s'il existe une paire de points (M_1, M_2) tels que M_1 est dans la moitié gauche, M_2 dans la moitié droite, et $d(M_1, M_2) < d_0$.

iv - Si on en trouve une (ou plusieurs), on renvoie la plus petite. Sinon, on renvoie d_0.

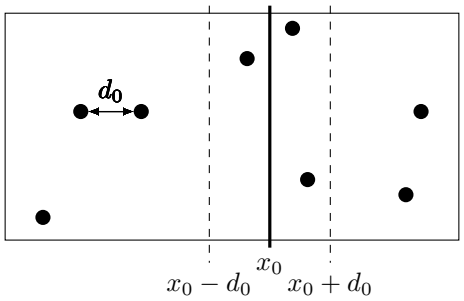

FIGURE 2 – Illustration du diviser pour régner

► **Question 9** Justifier qu'on peut se contenter de chercher les points M_1, M_2 de l'étape iii dans les points dont l'abscisse appartient à $I_0 = [x_0 - d_0, x_0 + d_0]$.

► **Question 10** Écrire une fonction `bande_centrale(x0,d0)` qui renvoie la liste des individus dont l'abscisse est dans I_0, *triés par ordonnée croissante*. On impose une complexité $O(n)$ à justifier.

► **Question 11** Montrer que deux points M_1 et M_2 situés à distance inférieure à d_0 l'un de l'autre sont au plus à 7 cases d'intervalle dans le tableau obtenu à la question précédente.

On pourra montrer qu'un rectangle à préciser de dimensions $2d_0 \times d_0$ contient au plus 8 points.

► **Question 12** En déduire une fonction `fusion(x0,d0)` qui prend en argument les valeurs x_0 et d_0 introduites précédemment, et renvoie le couple de matricules des points les plus proches, situés à au plus 7 cases de distance l'un de l'autre dans le tableau `bande_centrale(x0,d0)`. On impose une complexité en $O(n)$.

► **Question 13** Écrire une fonction `individus_trop_proches(t)` qui renvoie un triplet contenant les matricules des deux individus les plus proches dans la liste d'individus t, et la distance entre eux. On supposera que la liste t est triée par abscisses croissantes.

► **Question 14** Si on note k la taille du tableau `t`, justifier que la complexité $C(k)$ de la fonction `individus_trop_proches` vérifie :

$$C(k) \leqslant 2C(k/2) + O(n)$$

► **Question 15** En déduire la complexité $C(n)$. On pourra se limiter au cas où n est une puissance de 2.

III Surveillance de prison

Big Brother constate que les citoyens en prison sont assez uniformément malheureux. Pour expliquer ce curieux phénomène, on propose dans cette partie une modélisation de la manière dont évolue le bonheur de citoyens dans un environnement encadré.

On place n^2 prisonniers et prisonnières sur un carré de taille $(n-1)\times(n-1)$. Le bonheur est noté entre 0 et 10 comme une fonction $b : [0, n-1]\times[0, n-1]\times\mathbb{R} \to [0, 10]$. Ainsi, $b(x, y, t) = h$ signifie que l'individu situé aux coordonnées (x, y) a une note de bonheur de h à l'instant t.

On modélise le transfert de bonheur par une équation du même type que la diffusion thermique, à savoir :

$$\frac{\partial b}{\partial t}(x, y, t) = \alpha\,\Delta b(x, y, t) - \beta$$

où :

— α est une constante, appelée *coefficient de diffusivité du bonheur*,

— β est une constante, appelée *disparition locale de bonheur*,

— Δ désigne l'opérateur Laplacien, c'est à dire qu'on a $\Delta b(x, y, t) = \dfrac{\partial^2 b}{\partial x^2}(x, y, t) + \dfrac{\partial^2 b}{\partial y^2}(x, y, t)$.

En réalité, nos prisonniers ne sont pas un milieu continu. On va donc considérer une matrice `B` de dimension $n \times n$ telle que la valeur `B[i][j]` correspond au bonheur du prisonnier en j-ième position dans la i-ième rangée (avec $0 \leqslant i, j \leqslant n-1$).

On note l'approximation classique, pour une petite valeur de h :

$$\frac{\partial b}{\partial x}(x) = \frac{b(x + h/2) - b(x - h/2)}{h}$$

► **Question 16** Soit i et j compris entre 1 et $n-2$. Justifier qu'on utilise :

$$\Delta b(i, j) = \texttt{B[i+1][j]} + \texttt{B[i-1][j]} + \texttt{B[i][j-1]} + \texttt{B[i][j+1]} - 4\texttt{B[i][j]}$$

Écrire une fonction `delta_b(B,i,j)` qui calcule cette valeur.

En se basant sur la méthode d'Euler, on va maintenant considérer qu'on passe de la matrice `B1` (à l'instant t) à la matrice `B2` (à l'instant $t + dt$) avec l'approximation :

$$\frac{\texttt{B2[i][j]} - \texttt{B1[i][j]}}{dt} = \alpha\,\texttt{delta\_b(B1,i,j)} - \beta$$

Notons enfin que les prisonniers sur le bord ont un niveau de bonheur toujours égal à 10, car ce sont des agents infitrés placés là par Big Brother. Toutes les autres personnes ont un niveau de bonheur nul initialement.

► **Question 17** Programmer une fonction `evolution(alpha,beta,k,dt)` qui renvoie la matrice `B` définie précédemment, au bout d'un temps `k` × `dt` écoulé, en utilisant les valeurs `alpha` et `beta` comme définies ci-dessus.

IV Surveillance de quartier

Pour Big Brother, les quartiers de la ville sont assimilés à une grille rectangulaire de dimension `n` × `m`. Certains quartiers sont tranquilles alors que d'autres sont en révolte. On attribue la valeur -1 aux quartiers tranquilles, et on attribue aux quartiers en révolte une valeur entière correspondant au jour où ils passent en révolte.

De plus, la révolte se répand : un quartier se révolte au jour $k+1$ si au moins deux quartiers adjacents [2] sont en révolte au jour k. Un quartier en révolte le reste ensuite indéfiniment.

Un certain nombre de quartiers sont en révolte initialement et ont donc une valeur de 0. L'objectif est donc de prédire quels quartiers seront finalement en révolte et à partir de quel moment. On montre sur la figure 3 comment évolue la contamination dans un cas particulier de quartier.

$$\begin{bmatrix} 0 & -1 & -1 & -1 \\ -1 & -1 & 0 & -1 \\ 0 & -1 & -1 & -1 \end{bmatrix} \to \begin{bmatrix} 0 & -1 & -1 & -1 \\ 1 & -1 & 0 & -1 \\ 0 & -1 & -1 & -1 \end{bmatrix} \to \cdots \to \begin{bmatrix} 0 & 3 & 4 & -1 \\ 1 & 2 & 0 & -1 \\ 0 & 3 & 4 & -1 \end{bmatrix}$$

FIGURE 3 – Un exemple de quartier représenté au jour 0, 1, et 4.

► **Question 18** Écrire une fonction `contamination(m)` qui transforme la matrice `m` correspondant à un état initial de quartiers (avec uniquement des -1 et des 0) en une matrice finale.

► **Question 19** Montrer que si k cases sont contaminées initialement, on ne peut pas avoir plus de k^2 cases contaminées à la fin.

2. Les diagonales ne comptent pas