Exercice 1 – base de données

On considère la base de données de gestion des séries dans une plateforme de diffusion dont le schéma relationnel est le suivant :

```
serie(id_serie INT, titre_serie TEXT, annee_crea_serie INT, annee_fin_serie INT, pays_serie TEXT)

saison(id_sais INT, num_sais INT, annee_lance_sais INT, id_serie INT)

episode(id_epis INT, num_epis INT, titre_epis TEXT, duree_epis INT, id_sais INT)

plateforme(id_pf INT, nom_pf TEXT, pays_siege_social_pf TEXT, nom_creat_pf TEXT)

diffusion(id_sais INT, id_pf INT, pays_diff TEXT, annee_diff INT)
```

II représente ainsi cinq tables qui constituent une base de données de gestion des séries dans les plateformes de diffusion :
- La **série**, décrite par un identifiant de type entier, un titre de type texte, une année de création, une année de fin, et un pays d'origine.
- La **saison** d'une série représentée par un identifiant, un numéro de saison, une année de lancement.
- L'**épisode** d'une saison qui est représenté par un identifiant, un numéro d'épisode, un titre, une durée (en minutes) et la saison à laquelle il appartient.
- La **plateforme**, caractérisée par un identifiant, un nom, le pays d'origine du siège social, et le nom du créateur de la plateforme
- La **diffusion**, qui permet de caractériser le fait qu'une saison (d'une série) est diffusée sur une plateforme donnée, dans un pays donné, une année donnée.

Les différents pays sont représentés par leur code : USA pour les Etats-Unis d'Amérique, FR pour la France, D pour l'Allemagne, GB pour la Grande-Bretagne...

Les clés primaires des deux premières tables sont soulignées.

Par exemple, dans la table `serie`, les attributs sont `id_serie` qui est un entier, `titre_serie` qui est du texte, `annee_crea_serie` qui est un entier, `annee_fin_serie` qui est un entier et `pays_serie` qui est du texte.

► **Q1.** Donner une clé primaire de la table `episode`.
Même question avec la table `diffusion`

Pour les questions suivantes, écrire une requête permettant de répondre à la question.

► **Q2.** Afficher les informations sur les séries françaises créées en 2022.

► **Q3.** Afficher la série française créée il y a le plus longtemps (ou l'une d'entre elles si plusieurs datent de la même année).

► **Q4.** Afficher les titres des épisodes de la saison 1 de la série d'identifiant 123 en les classant par ordre croissant de numéros.

► **Q5.** Afficher, pour la série GAME OF THRONES, les numéros de saison classés par orde croissant ainsi que la durée totale en minutes pour chaque saison.

► **Q6.** Afficher les titres des séries dont au moins deux saisons ont été diffusées (même partiellement) en France en 2023 par la plateforme NFX, ainsi que le nombre exact de saisons diffusées.

► **Q7.** Afficher les titres des séries dont au moins un épisode a été diffusé en 2023 aux USA mais aucun en France la même année.

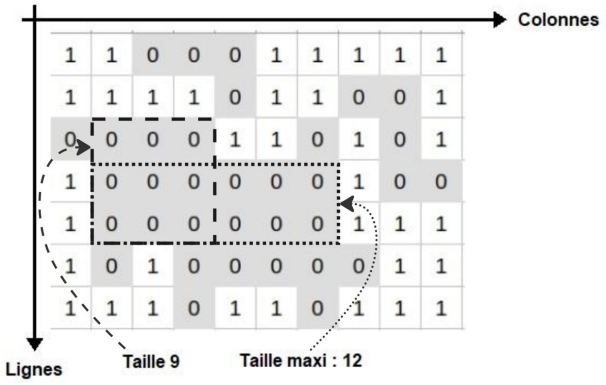
Exercice 2

Cet exercice utilise les listes Python mais seules les opérations qui suivent sont autorisées. Si `L` est une liste :

- `len(L)` renvoie la longueur de la liste L, c'est-à-dire son nombre d'éléments, complexité en O(1) ;
- `L[i]` désigne l'élément d'indice `i` de `L` (où `i` est compris entre `0` et `len(L)-1`), complexité en O(1) ;
- `L.append(e)` modifie la liste `L` en lui ajoutant l'élément `e` en dernière position, complexité en O(1) ;
- `L.pop()` renvoie le dernier élément de la liste `L` (supposée non vide) et supprime l'occurrence de cet élément en dernière position de la liste, complexité en O(1) ;
- les itérations sur une liste peuvent être effectuées directement (une liste Python est un objet itérable) ou à l'aide de la fonction range ;
- les listes peuvent être créées avec des boucles, ou en compréhension (par exemple `[0 for i in range(5)]`) ou avec une syntaxe de la forme `[0]*5`.

D'autre part, les fonctions `max` et `min` ne peuvent être utilisées que pour considérer le maximum ou le minimum de deux éléments (et non pour une liste).

Le but de ce problème est de détecter le plus grand rectangle dans une image. Plus précisément, on se donne un tableau à deux dimensions dont les valeurs sont 0 ou 1 et on cherche un rectangle constitué uniquement de 0 d'aire maximale (l'aire étant le nombre total de cases de ce rectangle) :



L'image sera représentée par une liste de taille p (nombre de lignes) dans laquelle chaque élément est une liste de taille q (nombre de colonnes) constituée de 0 et de 1. L'exemple de départ est représenté par la liste :

```
G = [[1, 1, 0, 0, 0, 1, 1, 1, 1, 1],
     [1, 1, 1, 1, 0, 1, 1, 0, 0, 1],
     [0, 0, 0, 0, 1, 1, 0, 1, 0, 1],
     [1, 0, 0, 0, 0, 0, 0, 1, 0, 0],
     [1, 0, 0, 0, 0, 0, 0, 1, 1, 1],
     [1, 0, 1, 0, 0, 0, 0, 0, 1, 1],
     [1, 1, 1, 0, 1, 1, 0, 1, 1, 1]]
```

On appellera «grille» une liste `G` du type précédent, on appellera «ligne» d'une telle grille l'un des éléments de cette liste `G` (les lignes sont numérotées à partir de 0). Par exemple, la ligne 2 de `G` est :

```
>>> G[2]
[0, 0, 0, 0, 1, 1, 0, 1, 0, 1]
```

On définit de même les colonnes de la grille. On désignera par G_{ij} l'entier `G[i][j]` d'une grille `G`.

► **Q1.** Écrire une fonction `dimensions(G)` qui prend en argument une liste de listes d'entiers, qui vérifie que chaque ligne comporte le même nombre d'entiers et qui renvoie un couple `(p,q)` correspondant aux dimensions (lignes puis colonnes) de `G` et qui renvoie `(0,0)` lorsque les lignes ne sont pas toutes de même taille. Par exemple :

```
>>> dimensions([[0,0,0], [0,0,0], [0,0,0], [0,0,0]])
(4, 3)
>>> dimensions([[0,0,0], [0,0,0], [0,0,0], [0,0]])
(0, 0)
```

► **Q2.** Écrire une fonction `EstRectangle(G, l1, l2, c1, c2)` qui prend en argument une grille, ainsi que 4 entiers et qui renvoie `True` lorsque tous les entiers G_{ij} pour `l1` $\leqslant i <$ `l2` et `c1` $\leqslant j <$ `c2` sont nuls et `False` sinon (on suppose que les arguments sont cohérents avec la grille et on ne demande pas de le vérifier dans cette fonction).

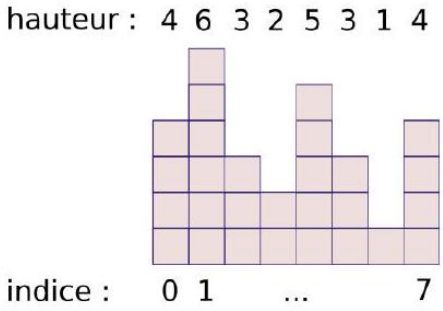
► **Q3.** Proposer un jeu de tests (deux ou trois tests utilisant `assert` par exemple) pour illustrer la fonction précédente en justifiant brièvement ce choix.

► **Q4.** Quelle est la complexité (temporelle) de la fonction `EstRectangle` ?

► **Q5.** *Version brute*
Écrire une fonction `PlusGrandRectangle(G)` qui renvoie la taille du plus grand rectangle (de 0) dans la grille `G` en testant tous les rectangles possibles à l'aide de boucles imbriquées.

► **Q6.** Si on suppose que `G` est une grille carrée de taille $n \times n$, donner une ordre de grandeur de la complexité temporelle de cette fonction `PlusGrandRectangle` (en $O(n^{\alpha})$).

On va maintenant s'intéresser à un cas particulier du problème : on se donne un histogramme de longueur n, c'est-à-dire une liste d'entiers naturels `H` dans laquelle chaque entier représente la hauteur d'une colonne de cet histogramme :



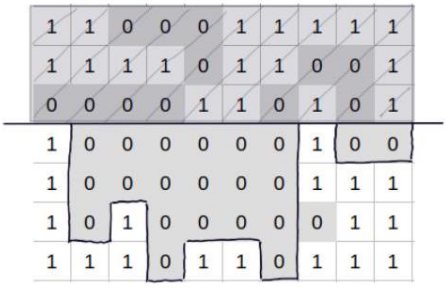
Cet histogramme sera représenté par une liste `H=[4,6,3,2,5,3,1,4]`. Dans ce qui suit, la longueur de la liste `H` sera notée n.

► **Q7.** On se donne une liste d'entiers `H` de taille n représentant un histogramme et on note $\text{hmax}_{i,j}$ la hauteur minimale (la valeur minimale de `H`) entre les positions i et j (comprises). Expliquer brièvement pourquoi, pour chaque valeur de i, on a les relations suivantes :

$$\text{hmax}_{i,i} = \texttt{H}[i] \quad \text{et} \quad \text{hmax}_{i,j} = \begin{cases} 0 & \text{si } j < i \\ \min\left(\text{hmax}_{i,j-1}, \texttt{H}[j]\right) & \text{si } j > i \end{cases}.$$

► **Q8.** Écrire une fonction `CalculHauteurs(H)` qui prend en argument une liste `H` (de taille n) et qui renvoie une liste de listes `L` (avec n listes de longueur n) telle que `L[i][j]` contienne $\text{hmax}_{i,j}$.

Nous reprenons désormais le problème d'origine. On se donne une grille G et on cherche le plus grand rectangle de 0 dans cette grille. On remarque que si ce plus grand rectangle commence en ligne i, alors ce plus grand rectangle est le plus grand rectangle d'un histogramme obtenu en retirant les lignes 0 à $i-1$ de la grille G et dont les hauteurs sont les zéros sous cette base (on regarde vers le bas par rapport à ce qu'on a fait avant) :



On est donc ramené à chercher le plus grand rectangle dans l'histogramme `H=[0,3,2,4,3,3,4,0,1,1]`.

La première étape consiste à transformer la grille afin de pouvoir obtenir rapidement les histogrammes «sous une ligne».

On veut créer une nouvelle grille, de la même taille que celle de départ mais telle que, en case (i, j), on ait le nombre de 0 consécutifs au dessous de la case (i, j) (cette case étant comprise). Pour une case en ligne i, colonne j, on a donc deux possibilités :

– la case contient 1 : la valeur sera 0 ,

– la case contient 0 : le nombre de 0 consécutifs en dessous vaut donc 1 de plus que celui en ligne $i+1$, même colonne.

Sur l'exemple de départ, on doit obtenir :

1	1	0	0	0	1	1	1	1	1
1	1	1	1	0	1	1	0	0	1
0	0	0	0	1	1	0	1	0	1
1	0	0	0	0	0	0	1	0	0
1	0	0	0	0	0	0	1	1	1
1	0	1	0	0	0	0	0	1	1
1	1	1	0	1	1	0	1	1	1

0	0	1	1	2	0	0	0	0	0
0	0	0	0	1	0	0	1	3	0
1	4	3	5	0	0	5	0	2	0
0	3	2	4	3	3	4	0	1	1
0	2	1	3	2	2	3	0	0	0
0	1	0	2	1	1	2	1	0	0
0	0	0	1	0	0	1	0	0	0

► **Q9.** Compléter la fonction `ColonneZero(G)` qui effectue ce travail et renvoie la grille avec les tailles des colonnes de zéro. Si G est de taille $p \times q$, la complexité de la fonction devra être en $O(p.q)$ (et donc en $O\left(n^2\right)$ si G est de taille $n \times n$).

► **Q10.** On suppose que l'on dispose d'une fonction `PgrHistogramme(H)` qui renvoie la taille du plus grand rectangle d'un histogramme avec une complexité linéaire par rapport au nombre de colonnes de l'histogramme.
En utilisant des fonctions vues précédemment, écrire une fonction `PgrUltime(G)` qui renvoie la taille du plus grand rectangle avec une complexité en $O\left(n^2\right)$ si G est de taille $n \times n$).

► **Q11.** Afin de comprendre l'intérêt d'un tel algorithme, on s'intéresse à des images que l'on représente numériquement sous la forme d'une liste de listes dont chaque case est un triplet d'entiers `(a, b, c)` dans $[\![0, 255]\!]$ correspondant au codage RGB du pixel (avec autant de colonnes par ligne).
Compléter la fonction `convertir(image)` qui prend en argument une image sous ce format et renvoie une grille de 0 et de 1 : 0 si la moyenne des trois entiers est inférieure ou égale à 120 et 1 sinon.