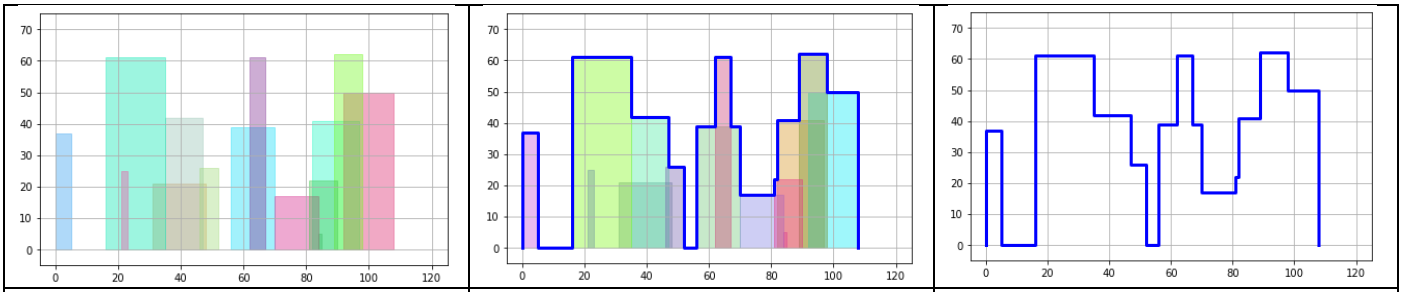
Principe de l'algorithme skyline

On s'intéresse dans ce sujet à l'algorithme «skyline» (qui signifie «horizon» et désigne ici la «New york Skyline» : la ligne d'horizon de New-York modelée par les gratte-ciels) : étant donné une série de rectangles tous alignés par le bas, déterminer la ligne d'horizon correspondante. Voici un exemple avec New-York :

Et voici un exemple correspondant à l'algorithme que nous souhaitons mettre en œuvre :



Données de départ : ensemble de rectangles	Traitement	Données en sortie : skyline correspondante

Représentation d'un ensemble de rectangles et représentation de la skyline

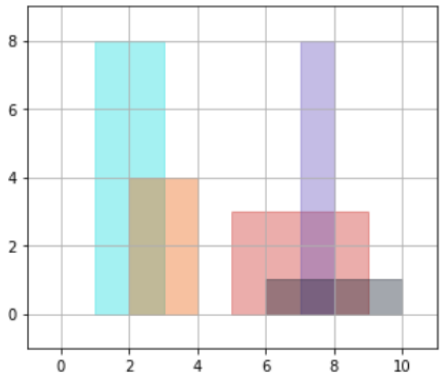
On travaille dans un repère comme ci-dessus.

Chaque rectangle est représenté par un triplet (g, h, d) où :
- g représente l'abscisse du côté gauche du rectangle ;
- h représente la hauteur du rectangle ;
- d représente l'abscisse du côté droit du rectangle.

L'ensemble de rectangles est alors représenté par un tableau (list Python)

La skyline est représentée par une liste(*) de couples de la forme (x, h) où, lorsque (x1, h1) et (x2, h2) se succèdent dans la liste, h1 désigne la hauteur de la skyline entre l'abscisse x1 et l'abscisse x2.

(*) : «liste» au sens de «vraie liste» (et pas tableau) comportant une tête de liste et une queue de liste, par exemple implémentée à l'aide d'une classe Cellule.



<u>Exemple :</u> on considère l'ensemble de cinq rectangles ci-contre. Il serait représenté par le tableau suivant :

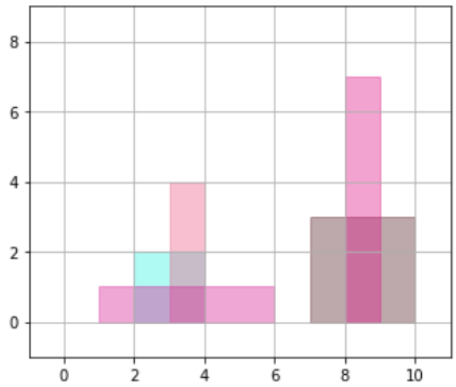
```
[(1, 8, 3), (2, 4, 4), (5, 3, 9), (7, 8, 8), (6, 1, 10)]
```

Quant à la skyline correspondante elle serait donnée par la liste suivante (tête de liste à gauche) :

```
| (0, 0) | (1, 8) | (3, 4) | (4, 0) | (5, 3) | (7, 8) |
(8, 3) | (9, 1) | (10, 0) |
```

Question 1 :

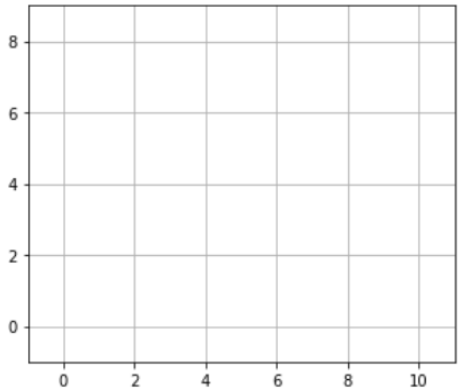
Donner les représentations des cinq rectangles ainsi que de la skyline ci-contre.



Question 2 :

Dessiner ci-contre les cinq rectangles correspondant au tableau de triplets ci-dessous :

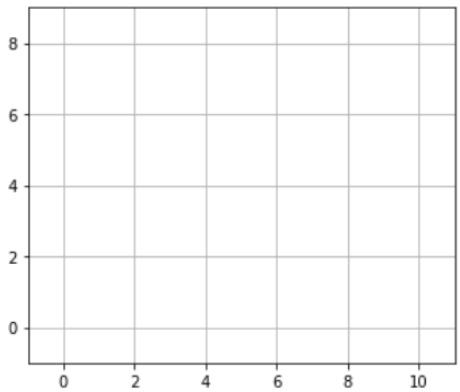
`[(1, 5, 4), (2, 4, 3), (5, 8, 7), (6, 6, 8), (7, 3, 10)]`



Question 3 :

Dessiner ci-contre la skyline correspondant à la liste ci-dessous :

| (0, 0) | (2, 3) | (3, 7) | (4, 2) | (5, 8) | (6, 0) | (8, 3) | (9, 5) | (10, 0) |



Principe de l'algorithme donnant la skyline à partir du tableau de rectangles

Nous allons ici utiliser un algorithme récursif «diviser pour régner» :

- Le cas de base est le cas où le tableau de rectangles ne contient aucun ou un seul rectangle : dans ce cas obtenir la skyline est assez accessible

- Le cas général consiste à :
  - diviser le tableau de rectangles en deux
  - obtenir les deux skylines de chacune des deux moitiés du tableau
  - fusionner les deux skylines : obtenir la skyline qui correspondrait à la superposition des deux

La trame de cet algorithme est donc la même que celle des autres algorithmes «diviser pour régner» déjà étudiés. Il possède néanmoins ses propres spécificités : d'une part la fonction `skyline_base` permettant d'obtenir la skyline dans les cas de base et d'autre part fonction `fusion` permettant d'effectuer la fusion de deux skylines. Et on comprend assez intuitivement que c'est la fusion qui risque de demander un peu plus de travail.

Fonction `skyline_base` : étude préliminaire

Question 4 :

Quelle est la skyline correspondant à un tableau ne contenant aucun rectangle ?

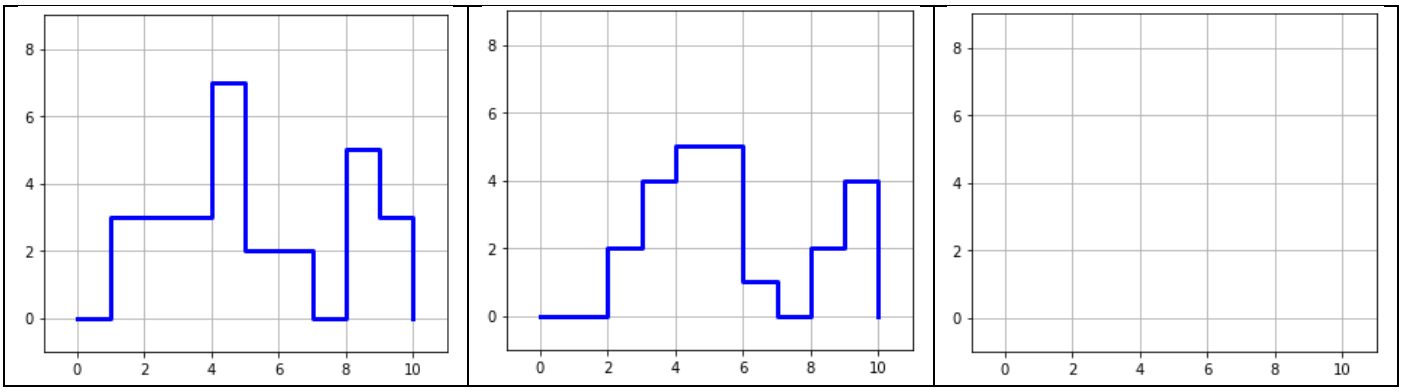
Question 5 :

Quelle est la skyline correspondant à un tableau contenant un unique rectangle `(g, h, d)` ?

Fonction fusion : étude préliminaire

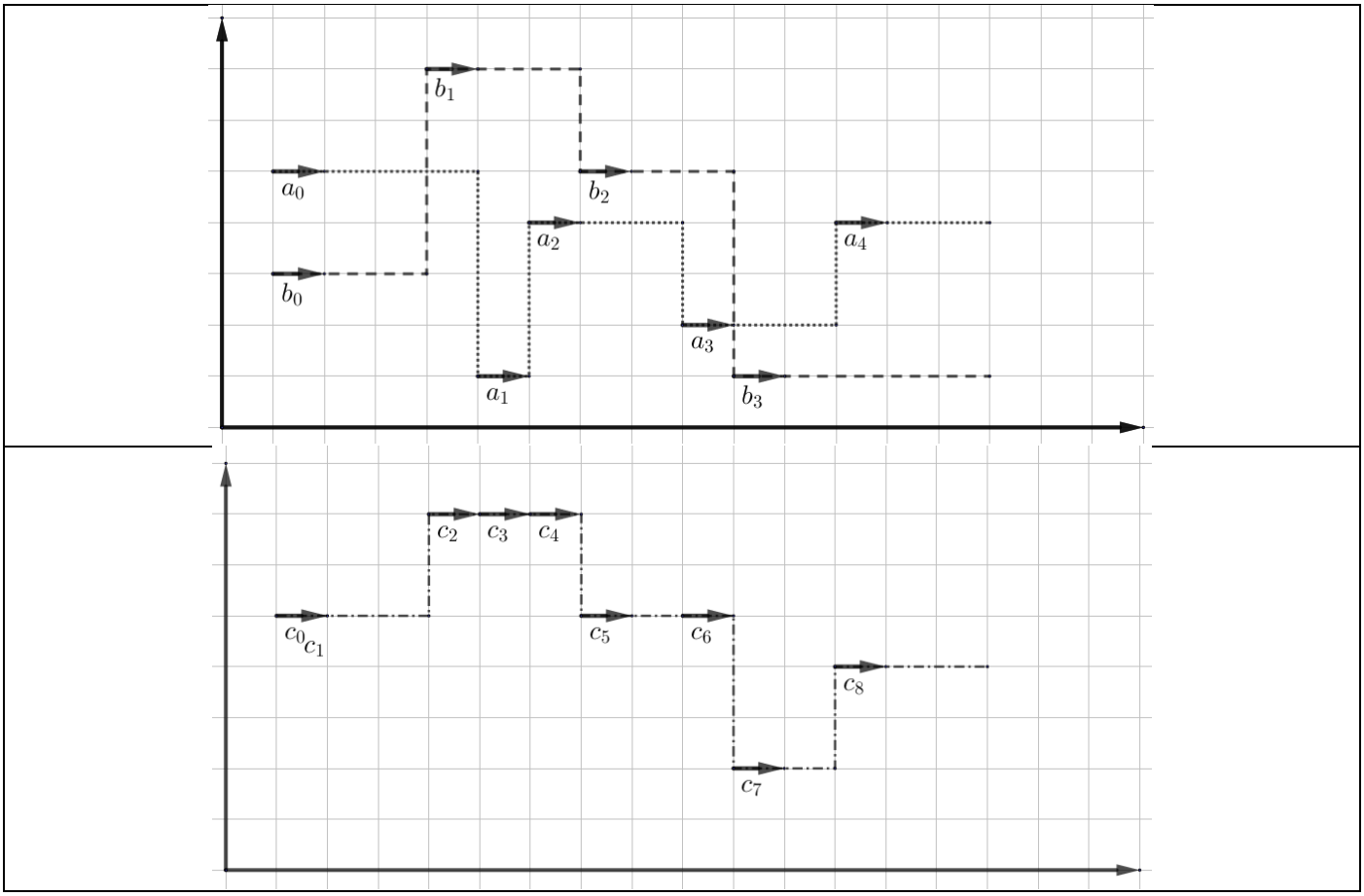
Question 6 :

On considère les deux skylines ci-dessous : dessiner ci-dessous à droite la skyline correspondant à la fusion des deux skylines (c'est-à-dire la skyline que l'on obtiendrait si l'on superposait les rectangles des deux skylines de gauche).



Question 7 :

Un exemple d'exécution de l'algorithme de fusion est donné ci-dessous : chaque flèche indique la donnée (abscisse, hauteur) d'un élément des listes skylines. En haut les deux skylines à fusionner, en bas la skyline fusion.



Le principe de l'algorithme récursif de fusion est donné ci-dessous, où :
- `sk1` et `sk2` sont les deux skylines de départ et `skr` est la skyline obtenue en faisant la fusion de `sk1` et `sk2`,
- `sk.tete` désigne le couple `(abscisse, hauteur)` situé en tête de liste et `sk.queue` désigne la queue de liste,
- `h1` et `h2` désignent les dernières hauteurs lues dans sk1 et sk2 : en fonction de leurs valeurs, on gardera la même hauteur ou au contraire on utilisera une nouvelle hauteur.

```
fusion(sk1, sk2, h1, h2) :
    Si sk1 (ou sk2) est vide renvoyer sk2 (ou sk1)
    Sinon :
        Définir abscisse1 et abscisse2 comme les abscisses de la tête des listes sk1 et sk2
        Si abscisse1 est plus petite que abscisse2 :                    #on travaille sur sk1
            - actualiser la valeur de h1 avec la hauteur de la tête de sk1
            - actualiser la liste choisie sk1 en la remplaçant par la queue de sk1
            - renvoyer la liste ayant :
                - pour tête le tuple (abscisse1, max(h1, h2))
                - pour queue fusion(sk1, sk2, h1, h2)
        Si abscisse2 est plus petite que abscisse1 :                    #on travaille sur sk2
            - actualiser la valeur de h2 avec la hauteur de la tête de sk2
            - actualiser la liste choisie sk2 en la remplaçant par la queue de sk2
            - renvoyer la liste ayant :
                - pour tête le tuple (abscisse2, max(h1, h2))
                - pour queue fusion(sk1, sk2, h1, h2)
```

a) Entourer ou surligner les deux lignes de l'algorithme qui montrent qu'il est récursif.
b) Appliquer cet algorithme pour fusionner les deux skylines ci-dessous :

```
sk1 = | (0, 0) | (3, 3) | (5, 7) | (6, 0) | (9, 8) | (10, 0) |
sk2 = | (0, 0) | (1, 1) | (4, 2) | (8, 0) |
```

Implémenter l'algorithme skyline

Vous disposez d'un notebook avec des fonctions de dessin et de génération de villes aléatoires qui vous permettront de tester votre code.

Il vous faudra normalement :

- Une fonction récursive `skyline(tableau)` qui correspond au corps principal de l'algorithme du type «diviser pour régner» décrit plus haut.

- Une fonction `skyline_base(tableau)` chargée de gérer la skyline des cas de base, comme indiqué plus haut, où le tableau ne contient que zéro ou un seul rectangle.

- Une fonction récursive `fusion(sk1, sk2, h1, h2)` effectuant la fusion nécessaire comme indiqué ci-dessus.