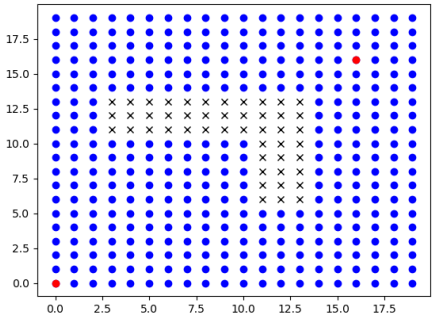
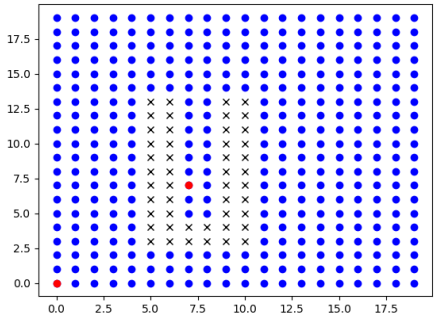
TD informatique du chapitre 22 : Algorithme de Dijkstra

Exercice 1 : Contournement d'un obstacle

On considère une grille carré dont tous les points à coordonnées entières sont les sommets du graphe (points bleus) à l'exception de points exclus (croix noires).



Grille 1



Grille 2

Chaque point de la grille est relié avec ses voisins directs avec un poids qui vaut la distance entre les deux sommets. L'objectif de cet exercice est de trouver le plus court chemin reliant les 2 points rouges et de le tracer.

Pour cet exercice, on utilisera les fonctions graphiques suivantes :

- `import matplotlib.pyplot as plt` : importation abrégée du module graphique,
- `plt.plot(i,j,'options')` : tracé du point de coordonnées (i, j) avec les options :
  - `x` : forme de croix,
  - `o` : forme ronde,
  - `^`: forme triangulaire,
  - `b` : couleur bleue,
  - `r` : couleur rouge,
  - `g` : couleur verte,
  - `k` : couleur noire.

  Par exemple, pour un rond bleu : `plt.plot(i,j,'ob')`
- `plt.plot([x1,x2],[y1,y2],'options')` : tracé de la ligne reliant les points $(x1, y1)$ et $(x2, y2)$ avec les options précédentes,
- `plt.show()` : affichage du graphique,
- `plt.pause(t)` : pause de durée t avant l'affichage du graphique suivant.

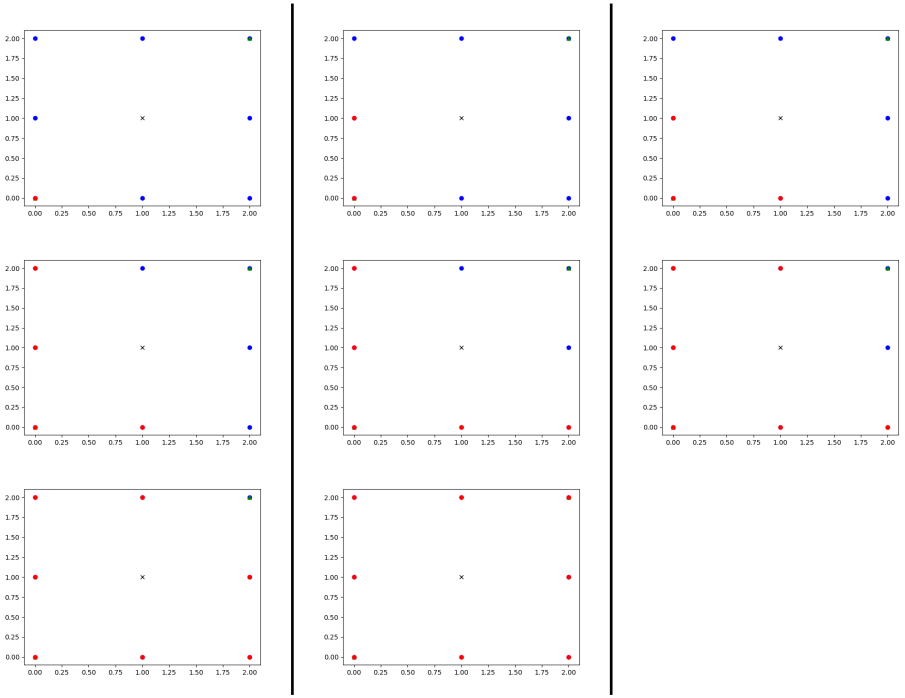
On considère les fonctions suivantes (voir le fichier TD22_Dijkstra.py) :

- `grille_init` qui prend comme arguments : un entier `n`, une liste `l` et un entier `b` et qui :
  - affiche le graphique où les points du carré $[\![0, n-1]\!]$ sont représentés par des ronds bleus à l'exceptions les points $(0,0)$ (départ) et (b, b) (arrivée) qui sont représentés par des triangles verts et des points de l qui sont représentés par des croix noires,
  - renvoie la liste S des points de $[\![0, n-1]\!]$ qui ne sont pas dans l, qui représente les sommets du graphe.
- `grille_graphe` qui prend comme arguments : un entier `n`, une liste `l` et un entier `b` et qui renvoie la représentation sous forme de dictionnaire du graphe dont les sommets sont donnés par `grille_init(n,l,b)` et dont chaque sommet est relié à ses voisins directs avec un poids égal à la distance entre les deux points.

1. Ecrire une fonction `grille_Dijkstra` qui prend comme arguments : un entier `n`, une liste `l`, un entier `b` et un flottant `t` et qui :
   - applique l'algorithme de Dijkstra au graphe `grille_graphe(n,l,b)` en partant du sommet $(0,0)$ et en arrêtant l'algorithme lorsque le sommet (b, b) est atteint,
   - affiche successivement, avec une pause de durée `t`, les graphiques partant de `grille_init(n,l,b)` dans lesquels les points visités sont marqués par des ronds rouges,
   - renvoie le dictionnaire dont les clés sont des sommets et dont les valeurs sont le sommet permettant d'atteindre la clé.

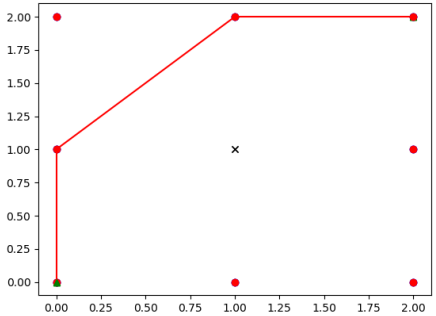
   Par exemple :

```
>>> grille_Dijkstra(3,[(1,1)],2,0.1)
{(1, 0) : (0, 0), (0, 1) : (0, 0), (1, 2) : (0, 1), (0, 2) : (0, 1),
 (2, 0) : (1, 0), (2, 1) : (1, 0), (2, 2) : (1, 2)}
```



2. Ecrire une fonction grille_trace_Dijkstra qui prend comme arguments : un entier n, une liste l, un entier b et un flottant t et qui trace les graphiques successifs donnés par grille_Dijkstra(n,l,b,t) puis qui trace le chemin obtenu reliant $(0,0)$ à (b,b).
Par exemple :

```
>>> grille_trace_Dijkstra(3,[(1,1)],2,0.1)
```



3. Utiliser la fonction grille_trace_Dijkstra pour obtenir les plus courts chemins pour les grilles 1 et 2.