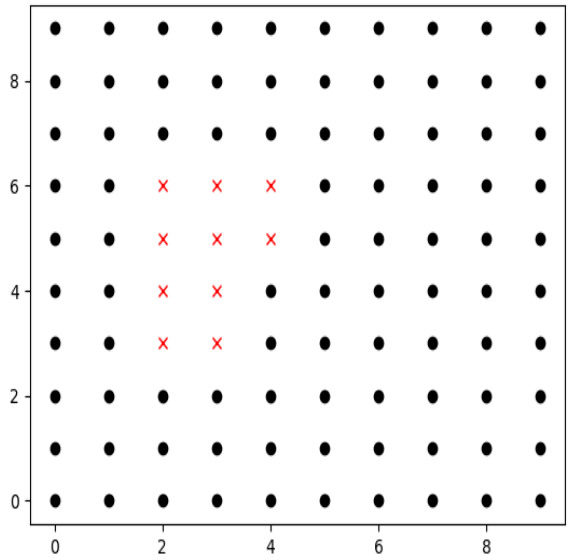
On considère une grille carré dont tous les points à coordonnées entières sont les sommets graphe (en noir) à l'exception de points exclus (croix rouges).



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

Pour appliquer l'algorithme de Dijkstra , vous devez écrire le code des fonctions suivantes en suivant les spécifications.

Les sommets représentés par les croix rouges sont des obstacles et ne font pas partie du graphe.
Dans le projet , la liste `obstacles` est une liste contenant tous les tuples (i,j) qui sont des obstacles.

```
def sommets_graphe(n,m,obstacles):
    '''
    renvoie la liste de tous les sommets (i,j)
    du graphe qui ne sont pas dans obstacles
    '''



def distance(i1,j1,i2,j2) :
    '''
    renvoie la distanse de Manhattan entre
    les points (i1,j1) et (i2,j2)
    '''

def sommetsVoisins(n,m,k,i,j):
    '''
    renvoie la liste des voisins du sommet
    (i,j).C'est donc une liste de tuples
    '''
```

```
def graphe(n,m,obstacles):
    '''
    renvoie le dictionnaire dont les clés sont les sommets
    du graphe et la valeur associée la liste des tuples:
    ((k,l),d) où (k,l) est un sommet voisin et d la distance
    de la clé à (k,l)
    '''


def tracer_graphe(n,m,obstacles):
    '''
une fonction qui trace avec matplotlib la grille
représentant les sommets du graphe
    '''



def Extraire_Min(F,dicoDistPred):
    '''
    renvoie le sommet de distance minimale depuis l'origine.
     en entrée F est une liste (représentyant une file)
     dicoDistPred est un dictionnaire
     '''


def Dijksra(G,origine):
    '''
    G est le dictionnaire du graphe
    origine : sommet de départ de l'algorithme

    sortie : l'algorithme renvoie un dictionnaire
    DistancesPredecesseurs du type:
    {(0,0):[0,None] , (1,0):[1,(0,0)]...}
    qui contient les distances depuis origine
    '''
```