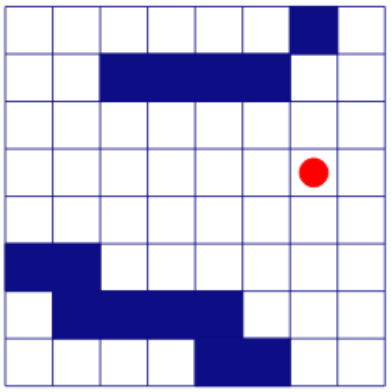
ExerciceType Bac
NSI *Terminale*

Épreuve écrite
Année .....

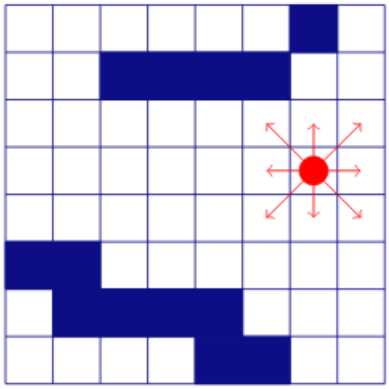
Exercice 5

Files - Tableaux - Graphes - BFS

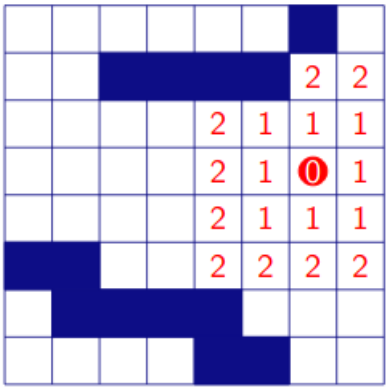
On se place sur une grille rectangulaire de taille 8×8 dont certaines cases sont inaccessibles (obstacles) et d'un jeton placé sur la grille.



Le jeton se déplace comme le Roi au échec (sur les 8 cases adjacentes).



- La case de départ est accessible en 0 coup.
- Ses voisines sont accessibles en 1 coup.
- Les voisines de ses voisines sont accessibles en 2 coups.
  - En tenant compte des obstacles.
  - Seulement si elles n'ont pas été atteintes avant.



N	N	N	N	N	N	-1	N
N	N	-1	-1	-1	-1	N	N
N	N	N	N	N	N	N	N
N	N	N	N	N	N	0	N
N	N	N	N	N	N	N	N
-1	-1	N	N	N	N	N	N
N	-1	-1	-1	-1	N	N	N
N	N	N	N	-1	-1	N	N

On représente cette grille par un tableau T dont les lignes et les colonnes sont numérotées de 0 à 7. La case en haut à gauche est repérée par (0, 0) et la case en bas à droite par (7, 7).

- La case de départ (3, 6) est marquée à 0.
- Les obstacles sont marqués à –1.
- les autres cases sont marquées d'un N (non atteintes).
- On accède à la valeur d'une case avec l'instruction `T[i][j]`

? Question 1:

1. Quel est le nombre minimal de coups pour atteindre la case (0,0) depuis la case départ ?
   6 coups
2. Donner les coordonnées des cases non atteignables qui ne sont pas des obstacles.
   (0,6),(0.7),(1,7),(2,7) et (3,7)
3. Donner les coordonnées des voisines atteignables de la case (1,6)
   (0,5),(0,7),(1,7),(2,5),(2,6),(2,7)

? Question 2:

1. Écrire en Python une fonction `marquee` qui prend en paramètre un tuple qui contient les coordonnées d'une case et qui renvoie **`False`** si la case contient un "N" et **`True`** sinon.
2. Écrire en Python une fonction `marquer` qui prend en paramètres un tuple (contenant les coordonnées d'une case) et une valeur (`val`) et qui remplace la valeur de cette case par : `val + 1`.

En imaginant cette grille comme un graphe où les sommets sont les coordonnées des cases marquée d'un "N".
Deux sommets sont alors adjacents (reliés par une arête) si les cases sont des voisines.

? Question 3:

On rappelle que l'ordre d'un graphe est le nombre de sommets du graphe et que le degré d'un sommet est le nombre d'arêtes issues de ce sommet.

1. Quel est l'ordre du graphe représentant cette grille ?46
2. Quel est le degré du sommet (1,6) ?6

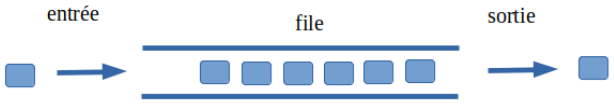
Dans la suite de l'exercice, **on suppose que l'on a accès à une fonction** `voisins` qui prend en paramètre un tuple (contenant les coordonnées d'une case) et qui renvoie une liste de tuples contenant les coordonnées de ses voisines atteignables (qui ne sont pas des obstacles).

On souhaite écrire une fonction `distances` prenant en paramètre un tuple(contenant les coordonnées de la case de départ) et qui effectue un parcours en largeur (BFS) de la grille en marquant les cases de la distance (en nombre de coups) les séparant de la case de départ.

On dispose d'une classe `File`:

- `f = File()` —> crée la file vide `f`.
- `f.est_vide()` —> renvoie **`True`** si la file est vide.
- `f.enfiler(val)` —> enfile `val` dans `f`.
- `f.defiler()` —> défile `f` et renvoie la valeur. (**`None`** si la file est vide).

Le principe de la file



Le principe du parcours est décrit ci-dessous:

- On marque la case de départ avec 0
- On l'enfile
- Tant que la file n'est pas vide
  - On défile dans `case_en_cours`
  - On récupère la valeur et les voisines de `case_en_cours`
  - Pour chacune des voisines
    - Si elles ne sont pas déjà marquées
      - On les marquent et on les enfilent

? Question 4:

1. Au second tour de la boucle, donner un exemple de case qui pourrait être traitée.(1,6)
2. Écrire en Python la fonction `distances` en utilisant les fonctions `marquee`, `marquer` et `voisins`.

```
def marquee(t):
    i,j=t[0],t[1]
    if T[i][j] == None:
        return False
    return True
def marquer(t,val):
    i,j=t[0],t[1]
    T[i][j] = val + 1

def distances(entree):
    file=File()
    marquer(entree,-1)
    file.enfiler(entree)
    while file.est_vide() == False:
        x=file.defiler()
        #valeur=T[x[0]][x[1]]
        valeur=valeurs(x)
        voisines=voisins(x)
        for vois in voisines:
            if marquee(vois)==False:
                marquer(vois,valeur)
                file.enfiler(vois)
```