

5.6

TD

5.6.1 Labyrinthe

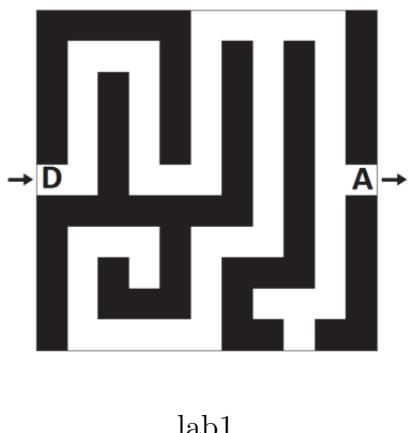
Représentation d'un labyrinthe

On modélise un labyrinthe par un tableau à deux dimensions à n lignes et p colonnes avec n et p des entiers strictement positifs.

Les lignes sont numérotées de 0 à n -1 et les colonnes de 0 à p-1.

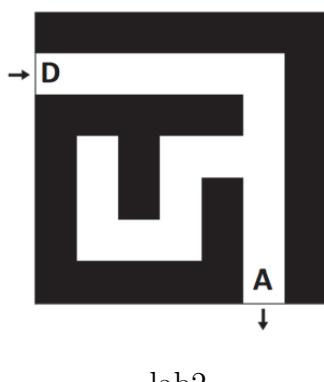
La case en haut à gauche est repérée par (0,0) et la case en bas à droite par (n-1,p-1). Dans ce tableau :

- 0 représente une case vide, hors case de départ et arrivée,
- 1 représente un mur,
- 2 représente le départ du labyrinthe,
- 3 représente l'arrivée du labyrinthe. Ainsi, en Python, le labyrinthe ci-dessous est représentée par le tableau de tableaux lab1.



```
lab1 = [
    [1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1],
    [1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1],
    [1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1],
    [1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1],
    [1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1],
    [1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1],
    [2, 0, 1, 0, 0, 0, 1, 0, 1, 0, 3],
    [1, 1, 1, 1, 1, 1, 0, 1, 0, 1],
    [1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1],
    [1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1],
    [1, 0, 1, 1, 1, 0, 1, 0, 0, 0, 1],
    [1, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1]
]
```

1. Le labyrinthe ci-dessous est censé être représenté par le tableau de tableaux lab2. Cependant, dans ce tableau, un mur se trouve à la place du départ du labyrinthe. Donner une instruction permettant de placer le départ au bon endroit dans lab2.



```
lab2 = [
    [1, 1, 1, 1, 1, 1, 1],
    [1, 0, 0, 0, 0, 0, 1],
    [1, 1, 1, 1, 1, 0, 1],
    [1, 0, 1, 0, 0, 0, 1],
    [1, 0, 1, 0, 1, 0, 1],
    [1, 0, 0, 0, 1, 0, 1],
    [1, 1, 1, 1, 1, 3, 1]
]
```

2. Écrire une fonction `est_valide(i, j, n, m)` qui renvoie True si le couple (i,j) correspond à des coordonnées valides pour un labyrinthe de taille (n,p) , et False sinon.
On donne ci-dessous des exemples d'appels.

```
>>> est_valide(5, 2, 10, 10)
True
>>> est_valide(-3, 4, 10, 10)
False
```

3. On suppose que le départ d'un labyrinthe est toujours indiqué, mais on ne fait aucune supposition sur son emplacement. Compléter la fonction `depart(lab)` ci-dessous de sorte qu'elle renvoie, sous la forme d'un tuple, les coordonnées du départ d'un labyrinthe (représenté par le paramètre `lab`). Par exemple, l'appel `depart(lab1)` doit renvoyer le tuple $(5, 0)$.

```
def depart(lab) :
    n = len(lab)
    m = len(lab[0])
    ...
    ...
```

4. Écrire une fonction `nb_cases_vides(lab)` qui renvoie le nombre de cases vides d'un labyrinthe (comprenant donc l'arrivée et le départ).
Par exemple, l'appel `nb_cases_vides(lab2)` doit renvoyer la valeur 19.

Recherche d'une solution dans un labyrinthe

On suppose dans cette partie que les labyrinthes possèdent un unique chemin allant du départ à l'arrivée sans repasser par la même case. Dans la suite, c'est ce chemin que l'on appellera solution du labyrinthe. Pour déterminer la solution d'un labyrinthe, on parcourt les cases vides de proche en proche. Lors d'un tel parcours, afin d'éviter de tourner en rond, on choisit de marquer les cases visitées. Pour cela, on remplace la valeur d'une case visitée dans le tableau représentant le labyrinthe par la valeur 4.

1. On dit que deux cases d'un labyrinthe sont voisines si elles ont un côté commun. On considère une fonction `voisines(i,j,lab)` qui prend en arguments deux entiers i et j représentant les coordonnées d'une case et un tableau `lab` qui représente un labyrinthe. Cette fonction renvoie la liste des coordonnées des cases voisines de la case de coordonnées (i,j) qui sont valides, non visitées et qui ne sont pas des murs. L'ordre des éléments de cette liste n'importe pas.
Ainsi, l'appel `voisines(1, 1, [[1, 1, 1], [4, 0, 0], [1, 0, 1]])` renvoie la liste $[(2, 1), (1, 2)]$. Que renvoie l'appel `voisines(1, 2, [[1, 1, 4], [0, 0, 0], [1, 1, 0]])` ?
2. On souhaite stocker la solution dans une pile chemin. Cette liste contiendra les coordonnées des cases de la solution, dans l'ordre. Pour cela, on procède de la façon suivante :

- Initialement :
 - Déterminer les coordonnées du départ : c'est la première case à visiter ;
 - empiler les coordonnées de la case départ à la pile chemin.
- Tant que l'arrivée n'a pas été atteinte :
 - on marque la case visitée avec la valeur 4 ;

Exercice 5.23

On considère la suite :

```
l= [1, 3, 0, 5, 7, 9, 2, 20, 13, 15, 4, 18, 31, 38, 33, 35, 21, 8, 34, 37, 23, 17, 19, 6, 11, 7]
```

On considère que l'on sort du labyrinthe si le nombre 51 a été entièrement visité ! Ecrire une fonction python qui détermine si l'on sort du labyrinthe !
