Réalisation d'un labyrinthe

Il s'agit de créer un chemin-labyrinthe parfait dans un carré de taille $N \times N$ (par exemple avec $N = 8$) : c.a.d. un chemin qui permet de relier tous les points du carré (entre deux points du carré il existe un et un seul chemin, sous-chemin du chemin-labyrinthe, qui relie ces deux points). On pourra considérer de manière conventionnelle que l'entrée et la sortie de ce labyrinthe sont situés à deux coins diagonalement opposés du carré.

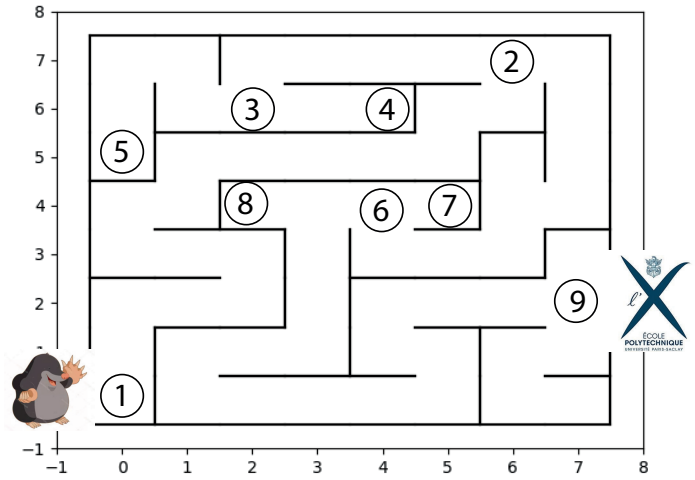

FIGURE 1 – Chemin-labyrinthe obtenu pour $N = 8$

I. Fonctions de base

N est une variable globale.

Q 1 - Définir un tableau `est_visitee` de taille $N \times N$ contenant pour chaque case la variable `False` Cette matrice permettra d'indiquer pendant la construction du chemin si une case donnée a été atteinte ou non. Elle sera modifiée chaque fois qu'une case sera visitée pour la première fois.

Q 2 - Quelle est le type de variable de `est_visitee[i][j]` ?

Q 3 - Écrire une fonction `case_possible` de paramètre `case=(i,j)` qui renvoie `False` si on est en dehors du carré et qui renvoie `True` sinon.

On vérifiera :

```
>>>case_possible( (1,3 )
True
>>>case_possible( (1,9 )
False
```

Q 4 - Écrire une fonction `visite` de paramètre `case=(i,j)` qui mute sur place `est_visitee [i][j]` en `True` si `case` est dans le carré.

Q 5 - Écrire une fonction `cases_voisines` de paramètre `case=(i,j)` qui donne dans une liste `L` de longueur au maximum 4, les cases adjacentes à `case` et non encore visitées dans le carré.

On vérifiera :

```
>>>case_possible( (0,0) )
[1,0],[0,1]
```

Q 6 - Créer une fonction `case_alea` qui choisit au hasard une case dans une liste non vide `L`. On vérifiera que cette fonction est valide que si la liste est non vide. (On importera le module `random` et on pourra utiliser la fonction `randint`)

II. Construction du chemin-labyrinthe

1 - Principe

Il reste à définir la construction du labyrinthe en partant de (0,0) , en choisissant un chemin possible à chaque étape et en étudiant toutes les possibilités. La fonction demandée `Labyrinthe()` retourne la liste des points du chemin commençant à (0,0) et visitant les cases du plateau.

- Une pile définie par une liste `p` contient au départ le point (0,0) et contiendra toutes les cases dont la succession définit le chemin ;
- une liste `chemin` est initialisée et contiendra la liste des case visitées par la procédure.

L'algorithme est le suivant :

- tant que la pile `p` n'est pas vide, on examine la case `sommet` stockée au sommet (on dépile).
- Le tableau `est_visitee` est mis à jour
- on stocke les coordonnées de la case `sommet` dans la liste `chemin`.
- On examine tous les déplacements possibles à partir de `sommet` et, si il y en a au moins un :

— on replace `sommet` sur `p`
— on choisit une case au hasard parmi les déplacements possibles que l'on place sur la pile `p`.

2 - Réalisation

Q 7 - À l'aide de la description de l'algorithme, compléter la procédure `labyrinthe` renvoyant le chemin-labyrinthe.