Projet Labyrinthe - support

Nom	Prénom	Courriel	Groupe	Github
AIT BELKACEM	Moncef Karim	moncef.ait-belkacem@universite-paris-saclay.fr	LDDIM2	MK8BK
LABOURET	Lucas	lucas.labouret@universite-paris-saclay.fr	LDDIM2	Lucas-Labouret

https://github.com/MK8BK/projet_labyrinthe

Génération de pseudo-labyrinthes

- Représentation des murs verticaux et horizontaux par des nombres binaires
- Énumération des dits nombres
- Création d'un pseudo-labyrinthe pour chaque couple possible

légende: pseudo-labyrinthe de dimensions 3×4 généré aléatoirement

Il est représenté par deux nombres binaires:

- Murs verticaux: { 1 0 0 1 0 0 0 0 }
- Murs horizontaux: { 0 0 1 0 0 1 1 0 }

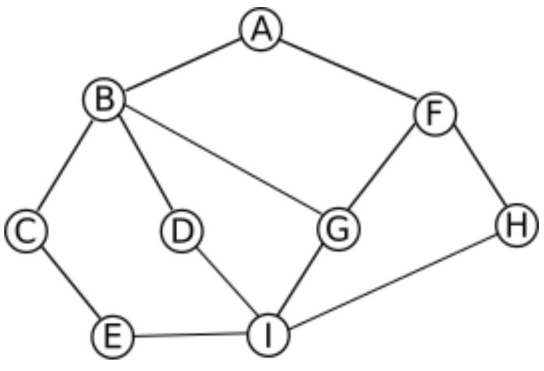
(Connaissant bien sûr le nombre de colonnes de la grille)

Introduction aux graphes

Nœuds, arêtes

Définitions:

- Un graphe est un ensemble de nœuds (sommets, points) reliés ou non par des arêtes.
- Un chemin est une succession de nœuds reliés par des arêtes.
- On dit qu'un graphe est connexe s'il existe toujours au moins un chemin reliant deux sommets.

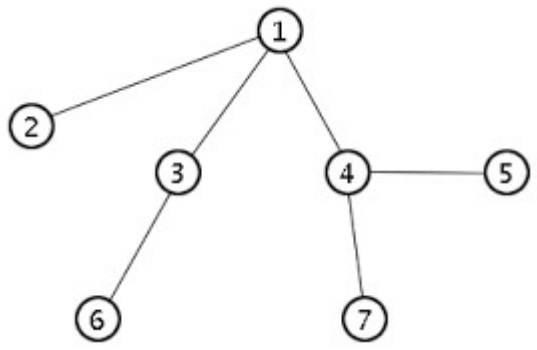


légende: chaque cercle est un nœud, chaque ligne est une arête

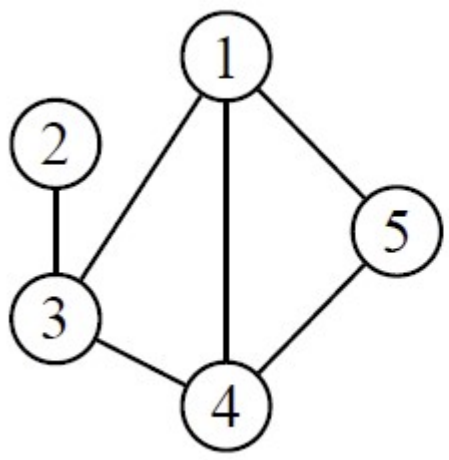
Arbres (couvrants)

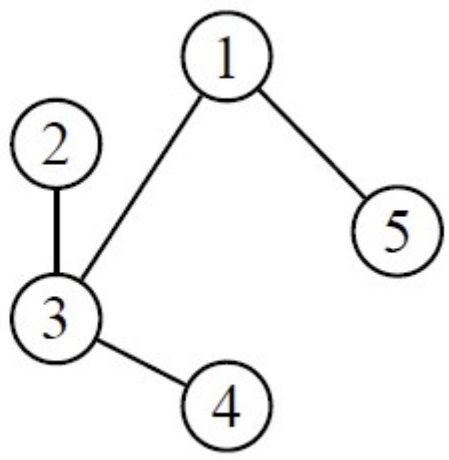
Définition:

- un arbre est un graphe acyclique et connexe, c'est à dire qu'il existe exactement un chemin entre deux sommets donnés
- un arbre A couvre un graphe G (connexe) ssi A contient tous les nœuds de G et toutes les arêtes de A appartiennent a G



légende: exemple d'arbre





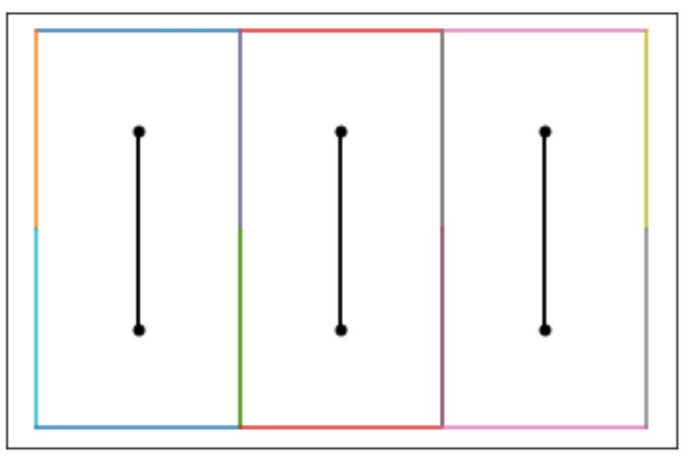
légende: un graphe G (à gauche), un arbre couvrant G (à droite)

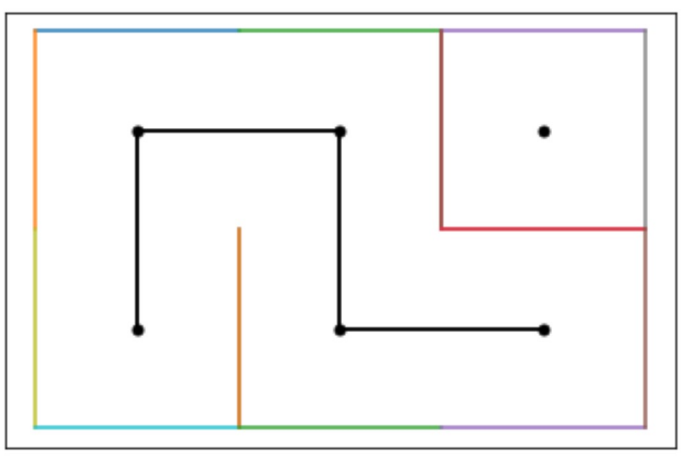
Visualisation et lien avec les labyrinthes

Afin de visualiser des pseudo-labyrinthes et des labyrinthes, on utilise la librairie matplotib.

On affiche à la fois les murs du labyrinthe mais aussi le graphe (connexe ou non) associé.

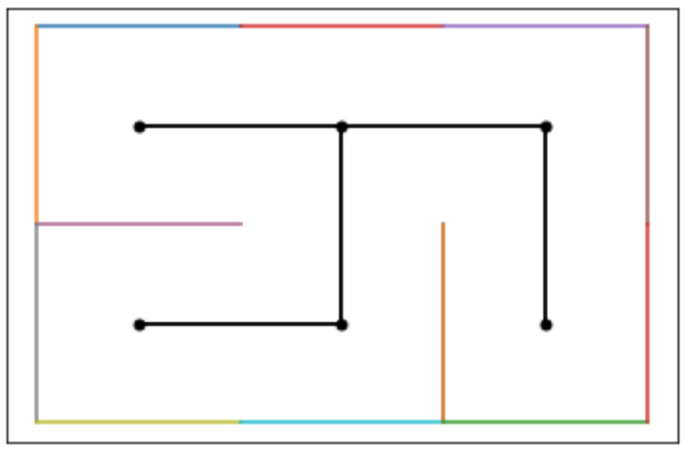
Chaque case de la grille est un nœud, chaque vide entre deux cases est une arête.





légende: exemples d'affichage de pseudo-labyrinthes

Chaque pseudo-labyrinthe est associé à un unique graphe.



légende: exemple d'affichage d'un labyrinthe

On remarque que le graphe associé à un vrai labyrinthe est un arbre couvrant de la grille.

Génération de labyrinthes (brute force)

On présente ici notre première tentative de génération de tous les labyrinthes de dimensions $n \times m$.

Principe de fonctionnement

On génère tous les pseudo-labyrinthes de dimensions $n \times m$,

on vérifie pour chaque pseudo-labyrinthe si c'est un labyrinthe.

Parcours de graphe

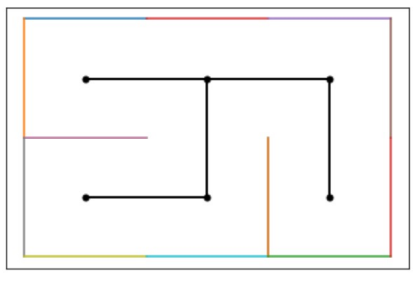
Pour vérifier si un pseudo-labyrinthe est un labyrinthe, on implémente une fonction `get_cell_occurences()`.

On part de la cellule (0,0), et pour chaque cellule de la grille, on donne le nombre de chemins la liant a (0,0).

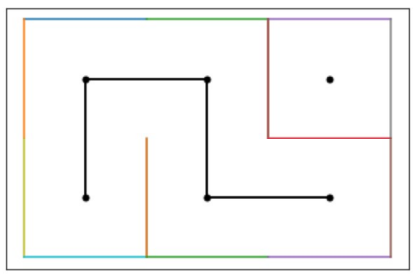
On classe ces valeurs dans une matrice de dimensions $n \times m$.

La matrice associée à un vrai labyrinthe ne contient que des 1.

$$W = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$



$$W = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$



Problèmes

Le nombre de pseudo-labyrinthes de dimensions $n \times m$ est : $2^{n(m-1)+m(n-1)}$

Le tableau ci-dessous récapitule ces valeurs de 1 à 4.

	1	**2**	**3**	**4**
1	1	2	4	8
2	2	16	128	1024
3	4	128	4096	2^{17}
4	8	1024	2^{17}	2^{24}

On remarque que le nombre de pseudo-labyrinthes évolue exponentiellement lorsque les dimensions de la grille évoluent linéairement.

Un algorithme type brute force aura donc une complexité exponentielle.

Génération de labyrinthes : transformation d'arbres couvrants

Principe de fonctionnement

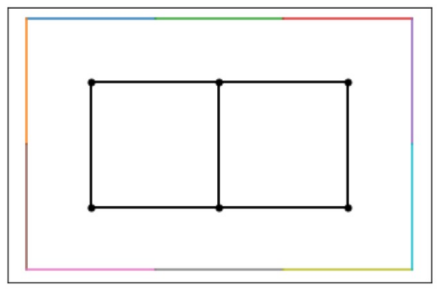
On génère tous les labyrinthes de taille 2×3 en partant d'un labyrinthe triviale, et en le transformant en d'autres labyrinthes jusqu'à épuisement des transformations possibles.

La racine d'un arbre est un sommet arbitraire (ce graphe étant non orienté).

On prend pour racine la case `(0,0)`.

On définit un cul-de-sac comme une extrémité de l'arbre différente de sa racine.

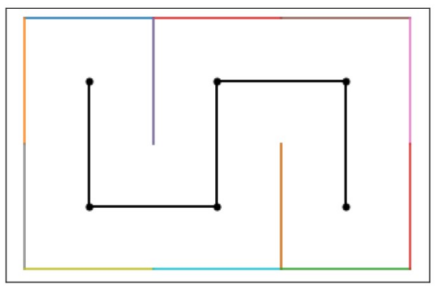
Prenons l'exemple des labyrinthes de dimension 2×3



Etape 1

On considère le labyrinthe canonique et son graphe associé,

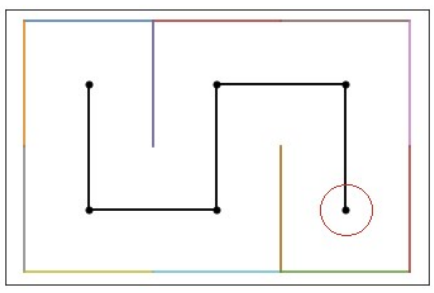
on l'ajoute à la liste des labyrinthes `non_traites` .



Etape 2

On cherche les culs-de-sac du labyrinthe (ici il y en a 1)

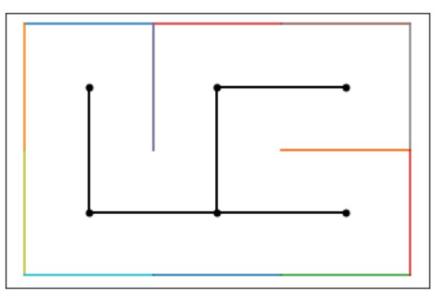
On retire le labyrinthe de la liste `non_traites` .



Etape 3

On déplace le mur autour du cul-de-sac.

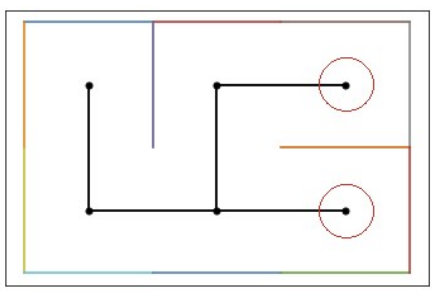
On ajoute le nouveau labyrinthe a la liste `non_traites` .



Etape 4

On cherche les culs-de-sac du labyrinthe (ici il y en a 2)

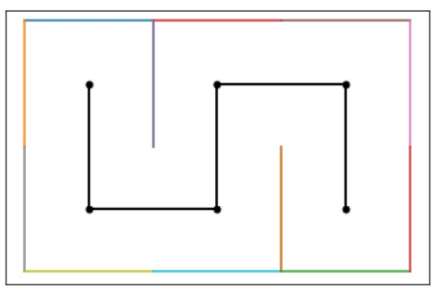
On retire le labyrinthe de la liste `non_traites` .



Etape 5

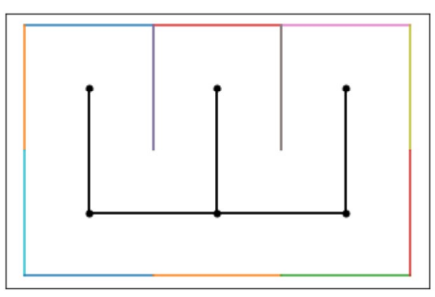
On gère le cul-de-sac du bas:

- On déplace le mur autour de ce cul-de-sac
- Ce labyrinthe a déjà été créé, on ne le rajoute pas à la liste `non_traites`



On gère le cul-de-sac du haut:

- On déplace le mur autour de ce cul-de-sac
- Ce labyrinthe est nouveau, on l'ajoute à la liste `non_traites`



Etape 6

On applique les étapes 4 à 6 sur le dernier labyrinthe de la liste `non_traites` , tant que la liste n'est pas vide.

Problèmes

- Redondance des labyrinthes générées
- Démonstration manquante