Rapport fonctionnel phase 1

L'objectif final de ce projet est de générer une carte dans laquelle des murs sont placés aléatoirement, et à partir d'un point de départ et d'un point d'arrivée, d'évaluer et d'afficher graphiquement un plus court chemin. La première partie se limite au calcul du plus court chemin pour une grille en 2 dimensions et à un affichage simple de l'itinéraire déterminé dans la console Python.

1- Description du programme et spécificités

Notre programme reprend le squelette proposé à l'exception de la partie "constante" qui a été supprimée. Dans l'optique d'inclure directement le bonus "choix des paramètres de densité et de dimensions", ces paramètres sont des variables retournés par une fonction interagissant avec l'interface. L'algorithme de calcul du trajet le plus court est celui mis en place en TD.
Nous travaillons intégralement avec des tableaux de type np.array afin de faciliter l'affichage console.
Nous avons choisi d'ajouter une fonction vérifiant la solvabilité du problème, en effet la génération des coordonnés des murs étant aléatoire, il est possible que le point de départ ou d'arrivé soit entouré de murs (cf glossaire des fonctions).

2- Instruction d'utilisation

Afin d'exécuter, programme, il faut lancer le sous-programme projet_console. Une interface apparaît vous demandant de donner la densité de mur souhaitée, les dimensions de la grille ainsi que les coordonnées du point de départ et d'arrivée.
Le programme gère les entrées biaisées, et à chaque étape une légende permet à l'utilisateur de décoder l'affichage proposé.
L'ensemble des sous-programmes doit être placé dans le même dossier.

3- Synthèse des principaux problèmes rencontrés

Dans cette première partie, nous avons rencontré plusieurs difficultés :

- Au début du projet, pour gagner en efficacité, nous nous sommes réparti les fonctions à écrire. En réunissant les lignes de codes, nous avons été confrontés à des problèmes d'indices sur les positions dans nos matrices qui c'est avéré, au final été très chronophage. Il aurait été bon de fixer d'emblée une convention de travail sur les indices.
- À la fin du projet, le nombre important de fonctions et l'interaction entre des fonctions écrites dans différents sous-programmes nous ont posé problème. Afin d'éviter ce problème, nous avons décidé de ne pas modifier les matrices générées par une fonction en utilisant une autre fonction, et d'écrire un maximum de fonctions à paramètre.

4- Structure du programme et décomposition en sous-parties :

Le squelette proposé comme point de départ est le suivant :

- 1- Fonctions logiques : Regroupe toutes les fonctions calculatoires du projet
- 2- Fonctions console : Ensemble des fonctions interagissant avec la console

- 3- Projet console : Programme principal pour la partie 1
- 4-Test
- 5-(Constantes : regroupe toutes les variables constantes et globales)

Dans l'optique d'inclure directement certains items de la phase 3 (bonus) dans notre code, nous avons choisi de ne pas conserver le sous-programme "constantes", les dimensions de notre grille et la densité n'étant pas des constantes.
Nous ne développerons pas le sous-programme "test" utilisé pour la mise au point de nos fonctions uniquement.

5 - Glossaire exhaustif des fonctions

A- Fonctions logiques

- comptage_1

Cette fonction prend comme argument une matrice (c'est-à-dire une liste de liste). Elle permet de compter le nombre de fois que le nombre 0 est présent dans cette matrice. Cette fonction est appelée dans la fonction generer_grille et permet de placer le nombre suffisant de murs sur la grille.

- generer_grille

Cette fonction prend comme argument la densité rentrée par l'utilisateur ainsi que les dimensions X et Y voulues par l'utilisateur. Cette fonction calcule le nombre de murs à placer dans la grille (c'est à dire le nombre de 1), génère une grille ayant les bonnes dimensions et place les murs de manière aléatoire sur la grille. Elle s'arrête quand le nombre de murs calculé est égal au nombre de murs présent dans la grille.

- generer_PD_PA

Cette fonction prend comme argument une matrice ainsi que les coordonnées dans la grille des points de départ et d'arrivée. Elle place ensuite sur la grille le point de départ (avec le numéro 2) et le point d'arrivée (avec le nombre 3).

- Voisin

Cette fonction prend comme argument une liste "l" contenant les coordonnées en X et Y d'un point ainsi qu'une liste "listes" contenant notamment les dimensions de la matrice. Elle teste si les 4 voisins d'un point sont dans la grille (cela peut ne pas être le cas si le point dont on recherche les voisins est situé contre une ou deux extrémités de la grille). Si c'est le cas, elle ajoute les coordonnées des voisins du point à une liste et retourne cette liste.

- matrice_des_distances

Préalable : numero_case_libre, numero_case_arrivee, numero_case_mur sont choisis de tel sorte à être supérieur à la distance maximale réalisable dans la grille. Ce calcul est réalisé dans le sous-programme projet_console
Cette fonction prend comme argument la grille contenant les 1 et les 0 symbolisant les espaces vides et les murs, un nombre "numero_case_libre" qui va remplacer les 1 de la grille, "numero_case_mur" qui va remplacer les 0 de la grille, "numero_case_arrivee" qui va remplacer le nombre 3 symbolisant l'arrivée, ainsi qu'une liste contenant la densité de murs, les dimensions de la grille, les coordonnées du point de départ et d'arrivée.
Cette fonction va placer sur les murs, les cases libres ainsi que la case arrivée respectivement les valeurs "numero_case_mur", "numero_case_libre" et "numero_case_arrivee". Ensuite, en partant du

départ jusqu'à l'arrivée, le programme va pour chaque point de la grille qui n'est pas un mur la distance qui le sépare du point de départ.

- test_si_resolvable

Ce programme prend comme argument une matrice, les coordonnées du point de départ et d'arrivée ainsi que le numéro associé à un mur. Il permet de savoir si il est possible de trouver un chemin du point de départ au point d'arrivée.
Ce programme va comparer le nombre de voisins au nombre de murs qui entourent les points de départ et d'arrivée. Si l'un des deux points est entouré de murs, alors le programme n'est pas résolvable.

B- Fonctions console

- test_entier :

Cette fonction prend en argument une valeur de type autre que "INT" et renvoie un niveau logique 0 ou 1 en fonction de la possibilité de convertir cet argument en un entier ou non. Cette fonction est utilisée dans les fonctions qui suivent pour prévenir les entrées biaisées à la console.

- test_longueur :

Ici, nous vérifions que la longueur d'une liste est bien de 2, comme pour la fonction précédente, elle trouve son utilité dans la vérification des entrées console.

- aquisition_donnees_position_depart (fonction récursive) :

Interroge l'utilisateur sur les coordonnées souhaitées pour le point de départ et le point d'arrivée. Une liste est créée à partir de ces informations. Si le résultat des deux fonctions précédents appliqué à cette liste sont des niveaux logiques 1, une conversion en valeurs entières est opérée, sinon la fonction est rappelée.

- aquisition_donnees_position_stop

Fonctionne sur le même principe que la fonction précédente, mais pour les coordonnées du point d'arrivée.

- aquisition_donnees_position

Nous concaténons ici les résultats retournés par les deux fonctions précédentes et vérifions que le point de départ et le point d'arrivé de soient pas confondus. Cette fonction est itérative et se rappelle tant que les points de départ et d'arrivée sont confondus. Elle ne prend pas d'arguments, étant placée dans le même sous programme que aquisition_donnees_position_stop et aquisition_donnees_position_stop.

- aquisition_donnees

Cette fonction est en charge de récupérer auprès de l'utilisateur les données de construction de la grille, à savoir :

- Le paramètre de densité (compris entre 10% et 50%)
- Les dimensions en X et y de la grille (comprisent entre 5 et 10)

Pour chacun de ces éléments a été mise en place une structure de vérification du type de donnée saisie par l'utilisateur sous forme de boucle "while", en utilisant notamment la fonction test_entier.
Ces données sont finalement retournées dans une liste de dimension 3.

C- Projet console

Ce sous-programme est le cœur du programme : il utilise l'ensemble des fonctions décrites précédemment afin de renvoyer la solution au problème.
Une seule fonction est définie dans ce sous-programme.

- trajet_le_plus_court

Cette fonction utilise la matrice renvoyée par la fonction matrice_des_distances.
En exploitant cette matrice, elle renvoie le trajet et la distance minimale du point de départ et d'arrivée.