MOGPL

LA BALADE DU ROBOT

24 novembre 2015

Benjamin Sportich Emmanuel de Bézenac
Université Pierre et Marie Curie
Année Universitaire 2015-2016

Table des matières

1 INTRODUCTION

Nous avons décider de développer le programme en Python pour faciliter la gestion de la mémoire et les listes. Nous avions aussi connaissances des librairies : , facilitant la création, la gestion des graphes ainsi que l'affichage.

2 FORMULATION DU PROBLÈME

Intéressons nous d'abord a une seule instance du problème de la balade du robot, c'est-à-dire, étant donné une matrice A de taille MxN où $\forall i \in 0,...,M-1$, $\forall j \in 0,...,N-1$, $A_{i,j} = 0$, soit $A_{i,j} = 1$, avec 1 étant défini comme un obstacle, et 0 comme une case libre. Soit O cet ensemble d'obstacles. Nous avons également un point de départ S (et une direction D) et un point d'arrivée E. La question est de trouver un chemin de durée la plus courte, étant donné une matrice A de taille $M.N$ instanciée avec $|O|$ obstacles, d'un point et d'une direction de départ, ainsi qu'un point d'arrivée, sachant que nous pouvons uniquement nous déplacer le long des cases qui sont libres.

Pour répondre à cette question, nous avons choisi de formuler le probème sous la forme d'un graphe orienté. Soit $G = (V, E)$ ce graphe orienté, et V et E ses noeuds et ses arcs (respectivement).

On peut modéliser le problème avec un graphe orienté G=(V,E). L'ensemble des nœuds représente l'ensemble des positions possibles du robots sur les rails du grand magasin, c'est à dire sa localisation sur la grille et son orientation. Sauf dans le cas des bords, un quadruplet de nœud n'existe que si le croisement correspondant sur le grillage ne touche aucun obstacle.

Il existe un arc entre le nœud A et le nœud B si et seulement si le robot peut passer de la position correspondant au nœud A à la position correspondant au nœud B en une et une seule commande.

3 EVALUTION - COMPLEXITÉ

Rappelons $G = (V,E)$, notre graphe directionnel associé à un instance du problème. Soit $|O|$ le nombre d'obstacles dans le dépôt. Nous avons donc par construction que $|V| = 4((M+1)(N+1)-|O|)$, donc $|V| \leq 4(M+1)(N+1)$. De plus, $|E| \leq 5|V|$, puisqu'il y a au plus 5 arcs crées à partir d'un noeud : les arcs étiquetés $D, G, a1, a2$, et $a3$. Donc $|E| \leq 20(M+1)(N+1)$.

Complexité de la création du graphe :
Un noeud est crée en $O(1)$, et un arc en $O(1)$ donc le graphe est crée en $O(|E|+|V|)$, ou encore en $O(M.N)$.

Complexité du parcours en profondeur :
Nous pouvons observer que le parcours en profondeur ne remet jamais un noeud dans la file. Les opérations de mise en file et de sortie de file prennent un temps $O(1)$, ou un temps de $O(|V|)$. Puisqu'on visite visite au plus tout les noeuds dans la liste d'adjacence, le temps total attribué à celui-ci est $O(|E|)$. La complexité du parcours en largeur est donc $O(|E|+|V|)$. Donc pour une instance donnée de taille $M.N$, nous avons donc que le parcours en profondeur du graphe associé est $O(M.N)$.

Pour retrouver le plus court chemin à partir de la fin, il faudra parcourir tout les noeuds jusqu'à atteindre le noeud de départ. Cette étape de backtracking est effectuée en $O(|V|)$, donc en $O(M.N)$.

Complexité de notre méthode :
- Etape 0 : Entrée des paramètres/Interface
coût :
- Etape 1 : Lecture d'un bloc
coût :
- Etape 2 : Création du graphe correspondant
coût :
- Etape 3 : Calcul du plus court chemin (Parcours en largeur)
coût :
- Etape 4 : Traçage du plus court chemin calculé précédemment (BFS)
coût :

- Etape 5 : Affichage du résultat

coût :

Ces 6 étapes sont répétées à chaque bloc.

La complexité d'un tel algorithme est O( )

4 ESSAIS NUMÉRIQUES

4.1 En fonction de la taille de la grille

4.2 En fonction du nombre d'obstacles

———————————————-

5 INTERFACE UTILISATEUR

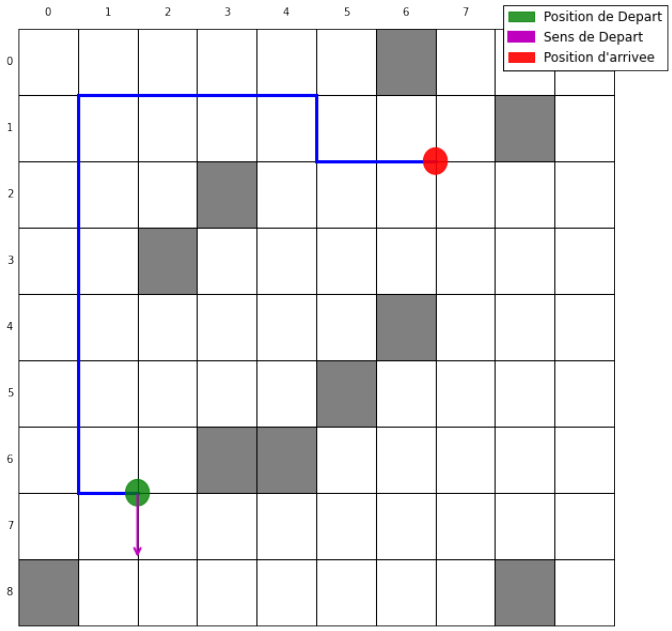

FIGURE 1: Interface de la solution du problème énoncé dans le projet.

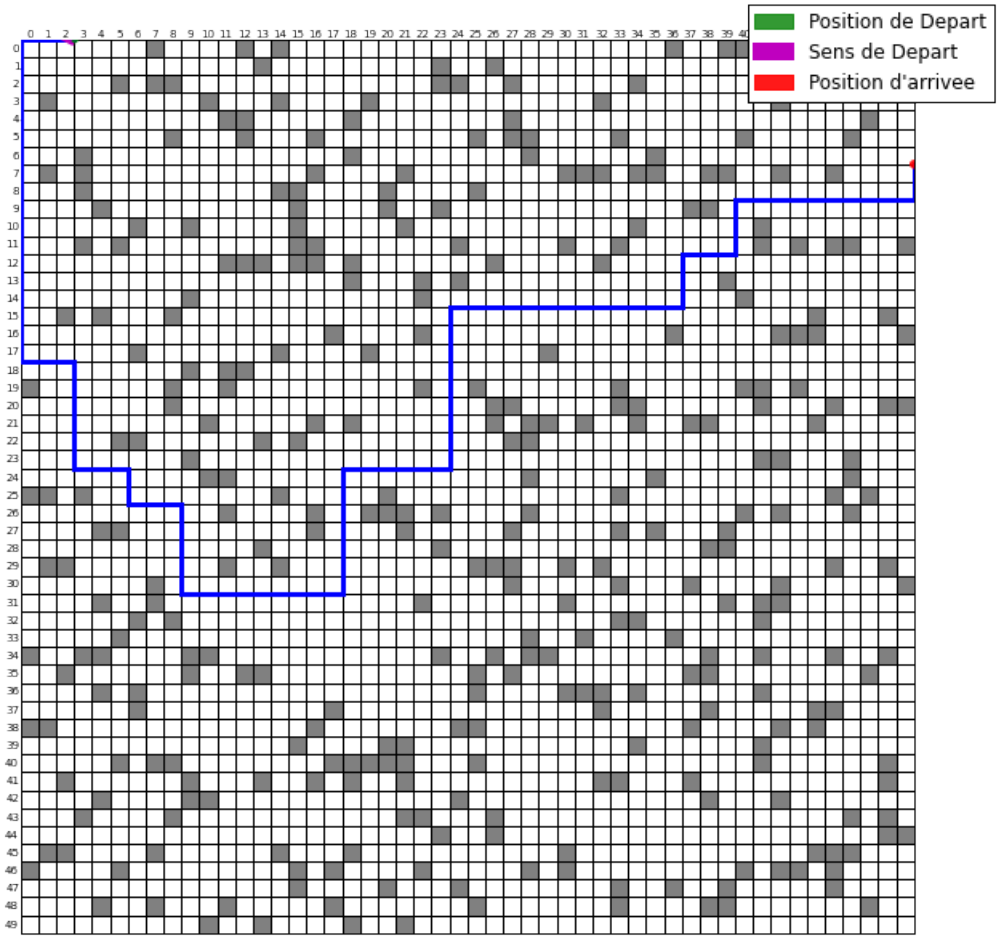

FIGURE 2: Exemple d'instance de taille 50, avec obstacles.

6 CONCLUSION