TD no1 : Génération de trajectoires

L'objectif de ce TD est d'implémenter et de tester une variante de l'algorithme A* pour la recherche d'un chemin dans une carte.

L'algorithme étudié est le « A* pondéré ». Le pseudo code est identique à celui du A*, vu en cours, mais la fonction d'évaluation du coût d'un chemin passant par un nœud s est remplacée par : $f(s) = g(s) + \varepsilon * h(s, n_{goal})$, avec le coefficient $\varepsilon \geq 0$.

I. Etude de l'algorithme

I.1. Si l'on choisit $\varepsilon = 1$ ou $\varepsilon = 0$, quels algorithmes retrouve-t-on ?

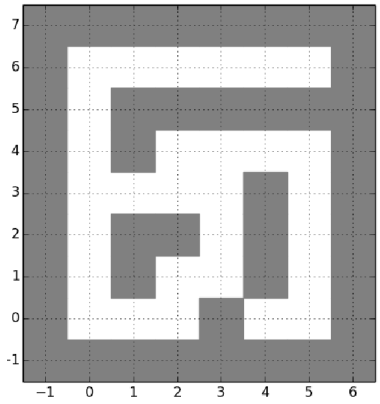
I.2. Quel serait l'intérêt de choisir $\varepsilon > 1$? Quelle propriété risque-t-on de perdre ?

II. Implémentation et test de l'algorithme A*

Le fichier Python *Astar.py* définit plusieurs classes d'objets qui seront nécessaires à l'implémentation :

> *Map* : définit un objet de type carte par la donnée de ses dimensions en X et Y, d'une grille d'occupation (0 : case libre, 1 : case occupée), des coordonnées X et Y de chaque point de la grille d'occupation, et du graphe correspondant.
>
> *Graph* : définit un objet de type graphe par la donnée du nombre et de la liste de ses nœuds, de la matrice d'adjacence, et du degré d'adjacence (4 ou 8).
>
> *Node* : définit un objet de type nœud par la donnée de son numéro (identifiant), de ses coordonnées x et y et des données nécessaires à l'utilisation d'un algorithme de recherche de chemin : coût g et coût f, numéro du nœud parent.

II.1. Prendre connaissance du code fourni et l'utiliser pour représenter la carte suivante (la bordure extérieure n'est pas à représenter : uniquement les cases de coordonnées 0 à 5 en abscisses et 0 à 6 en ordonnées) :



II.2. Compléter la fonction *manhattanDist* permettant de calculer la distance de Manhattan entre deux nœuds. Modifier les fonctions *heuristic* et *distance* de manière à ce que l'algorithme de recherche de chemin utilise la distance de Manhattan. Tester la version de l'algorithme correspondant au A*, en

utilisant cette distance et un degré d'adjacence de 4, pour trouver un chemin entre le point de départ de coordonnées (0,0) et le point d'arrivée de coordonnées (5,0).

II.3. A partir de la liste *closedList* renvoyée comme résultat de l'algorithme A* à la question précédente, compléter la méthode *buildPath* de la classe *Map* permettant de reconstruire le chemin généré sous forme de liste contenant les numéros des nœuds, depuis le nœud de départ jusqu'au nœud d'arrivée. Affichez ensuite ce chemin en utilisant la fonction *plotPathOnMap* ainsi que l'arbre couvrant à l'aide de la fonction *plotExploredTree*.

II.4. Compléter la méthode *generateAdjacencyMatrix* de la classe *Map* pour étendre sa définition à un degré d'adjacence de 8.

II.6. Compléter la fonction *euclidianDist* permettant de calculer la distance Euclidienne entre deux nœuds. Modifier les fonctions *heuristic* et *distance* de manière à ce que l'algorithme de recherche de chemin utilise la distance euclidienne. Tester la version de l'algorithme correspondant au A*, en utilisant cette distance et un degré d'adjacence de 8, pour trouver un chemin entre le point de départ de coordonnées (0,0) et le point d'arrivée de coordonnées (5,0). Affichez le chemin trouvé sur la carte. Qu'observe-t-on ? Est-ce le comportement souhaité ? Si non, que faudrait-il modifier en conséquence ?

III. Etude des performances de l'algorithme « A* pondéré »

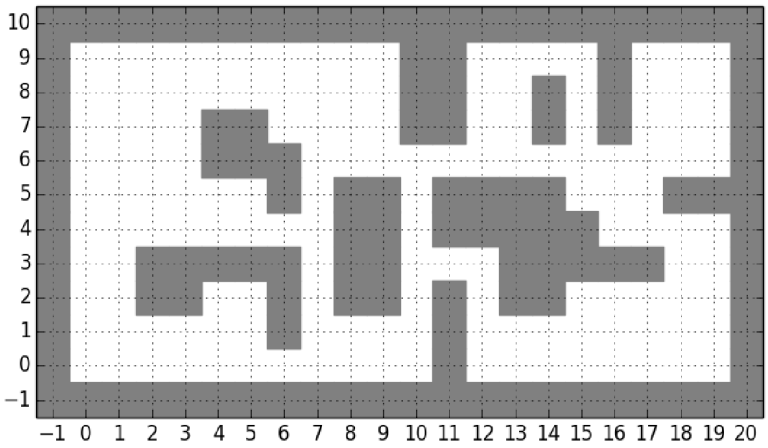
III.1. Compléter la fonction *buildPath* pour calculer et afficher la longueur d'un chemin trouvé.

A partir du fichier *Map_20x10.py* (cf. carte page suivante), relever pour différentes valeurs de ε (= 0, 1, 2, 3) la longueur du chemin trouvé pour aller du nœud de coordonnées (0,0) à celui de coordonnées (19,9). Utiliser la distance euclidienne et un degré d'adjacence de 8.

On remplira le tableau ci-dessous en y reportant également le nombre d'itérations réalisées par l'algorithme pour trouver le chemin et le nombre de nœuds explorés.

Conclure quant à l'intérêt du coefficient ε.

Valeur de ε	0	1	2	3
Longueur du chemin				
Nombre d'itérations				
Nombre de nœuds explorés				


10
9
8
7
6
5
4
3
2
1
0
−1
−1 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20