Ch 21bis. Graphe - le plus court chemin - Algorithme A Star.

1 Problématique

Lorsque l'on cherche le plus court chemin entre deux sommets, départ et arrivée, le plus rapidement possible et en évitant les obstacles éventuels, l'algorithme A* (prononcer A star) est fait pour ça ! Cet algorithme est principalement utilisé dans les jeux vidéos :

FIGURE 1 – Exemple de chemin

C'est un algorithme de recherche de chemin dans un graphe. C'est l'un des plus efficaces en la matière. Il ne donne pas toujours la solution optimale mais il donne très rapidement une bonne solution.
Au premier abord, on pourrait se dire que pour trouver un chemin d'un point à un autre il faut commencer par se diriger vers la destination. Et bien... c'est justement cette idée qu'utilise l'algorithme A* ! L'idée est très simple : à chaque itération, on va tenter de se rapprocher de la destination. Pour cela, on va donc privilégier les possibilités directement les plus proches de la destination, en mettant de côté toutes les autres. Toutes les possibilités ne permettant pas de se rapprocher de la destination sont mises de côté, mais pas supprimées. Elles sont simplement mises dans une liste de possibilités à explorer si jamais la solution en cours s'avère mauvaise. En effet, on ne peut pas savoir à l'avance si un chemin va aboutir ou s'il sera le plus court. Il suffit que ce chemin amène à une impasse pour que cette solution devienne inexploitable.
L'algorithme va donc d'abord se diriger vers les chemins les plus directs. Et si ces chemins n'aboutissent pas ou bien s'avèrent mauvais par la suite, il examinera les solutions mises de côté. C'est ce retour en arrière pour examiner les solutions mises de côté qui nous garantit que l'algorithme nous trouvera toujours une solution (si tant est qu'elle existe, bien sûr).

2 Comment fonctionne cet algorithme

Il est basé sur l'algorithme de `Dijkstra` auquel est ajouté une `heuristique`.

En algorithmique, une heuristique est une méthode de calcul qui fournit rapidement une solution réalisable, pas nécessairement optimale ou exacte, pour un problème d'optimisation difficile (wikipédia).

Cette heuristique est définie dans la fonction :

$$f(s) = g(s) + h(s)$$

Avec :

- $g(s)$ est le coût du chemin optimal partant du sommet initial jusqu'au sommet s ;
- $h(s)$ le coût estimé du reste du chemin partant de s jusqu'à un état satisfaisant de l'arrivée. $h(s)$ est une heuristique.

2.a Principe

Dans des applications de recherche de chemin sur une image ou un quadrillage par exemple, il est possible de connaître pour chaque sommet la distance « à vol d'oiseau » ou « à taxi-distance » de chaque sommet à l'arrivée. On peut utiliser une de ces heuristiques et, au lieu de choisir le sommet ayant la plus petite distance depuis le départ parmi les sommets visités, on peut choisir la relation :

$$f(s) = d(depart, s) + d'(s, fin)$$

On choisit alors la valeur `min` des f(s) des sommets visités.
Remarque : la valeur calculée peut être mémorisée sous forme d'entier (voir exemple).

Vous pouvez calculer ces distances de la manière que vous voulez, distance euclidienne, distance de `Manhattan` ou autre, elles peuvent convenir.
La distance de `Manhattan`, appelée aussi taxi-distance, est la distance entre deux points parcourue par un taxi lorsqu'il se déplace dans une ville où les rues sont agencées selon un réseau ou quadrillage (Fig. 2). Un taxi-chemin est le trajet fait par un taxi lorsqu'il se déplace d'un sommet du réseau à un autre en utilisant les déplacements horizontaux et verticaux du réseau.

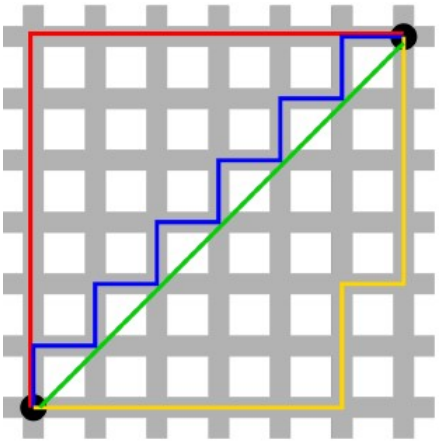
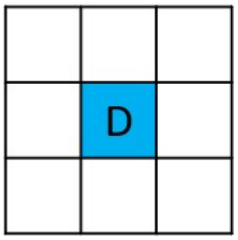


FIGURE 2 – distance de Manhattan contre distance euclidienne

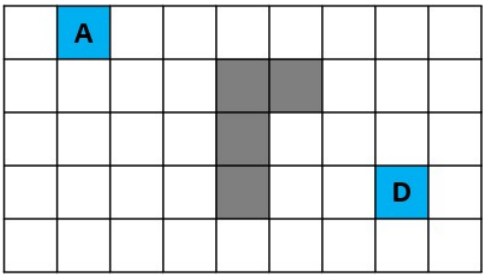
2.b Exemple de construction sur un quadrillage

Toutes les cases du quadrillage sont des sommets et les sommets voisins sont les cases ayant un côté ou un angle commun.

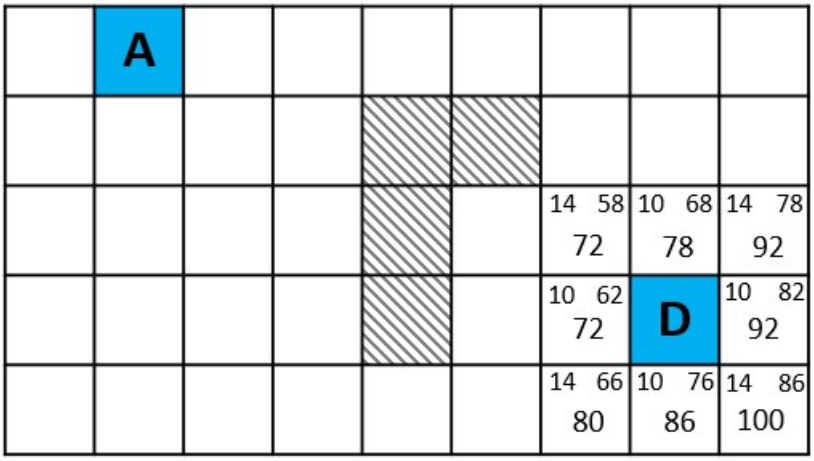
Chaque case voisine par le côté est à une distance 10 soit 1×10 et les cases en diagonale à une distance 14 soit $\sqrt{2} \times 10$ arrondi à la valeur entière.



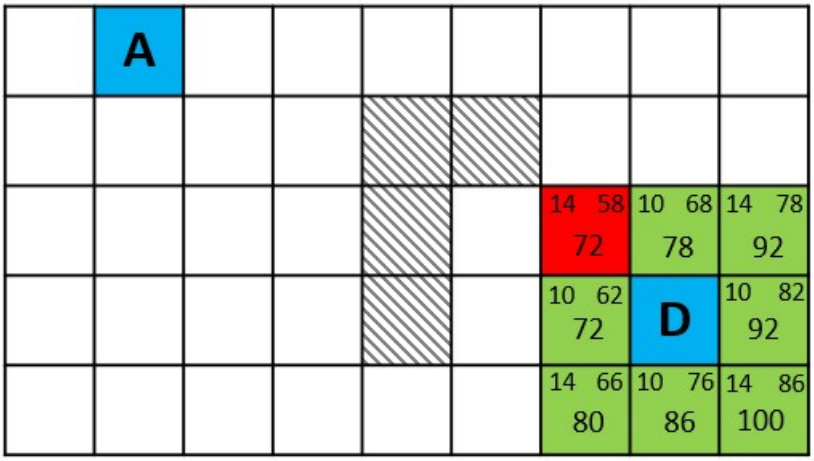
On cherche à déterminer le plus court chemin entre D et A.



Étape 1 : On évalue la distance à l'origine, la distance de la fin et le coût global



Étape 2 : On choisit le sommet dont le coût global est le plus faible. En cas d'égalité, on choisit le sommet le plus proche de l'arrivée.



Étapes 3, 4, 5 et 6 :

	A							
						14 58 72	10 68 78	14 78 92
						10 62 72	D	10 82 92
						14 66 80	10 76 86	14 86 100

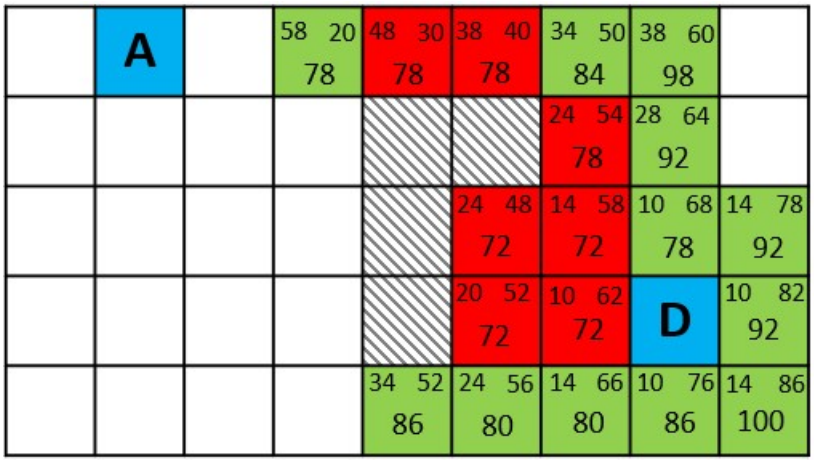
Étape 7 :

	A				38 40 78	34 50 84	38 60 98	
						24 54 78	28 64 92	
					24 48 72	14 58 72	10 68 78	14 78 92
					20 52 72	10 62 72	D	10 82 92
				34 52 86	24 56 80	14 66 80	10 76 86	14 86 100

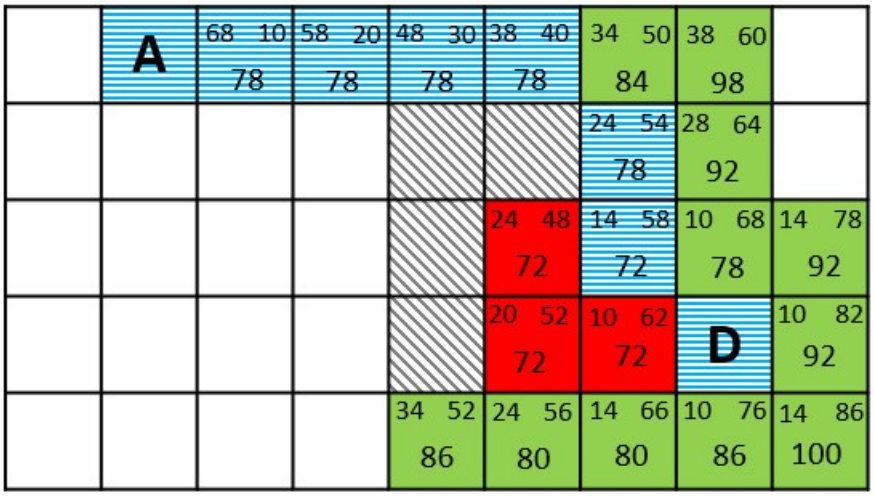
Étape 8 :

	A			48 30 78	38 40 78	34 50 84	38 60 98	
						24 54 78	28 64 92	
					24 48 72	14 58 72	10 68 78	14 78 92
					20 52 72	10 62 72	D	10 82 92
				34 52 86	24 56 80	14 66 80	10 76 86	14 86 100

Étape 9 :



Finalement le chemin le plus court :



3 Implémentation en python

3.a Structure de données

Un sommet du graphe peut être représenté sous forme d'une liste dans un dictionnaire avec :

- la valeur de `G` (de type `int`), c'est la distance pour aller du point de départ au sommet considéré ;
- la valeur de `H` (de type `int`), c'est la distance pour aller du sommet considéré au point d'arrivée ;
- la valeur de `F` (de type `int`), c'est la somme des deux précédents mémorisée pour ne pas la recalculer ;
- le parent, représenté par ses coordonnées.

Initialement, un sommet non visité prend comme valeurs : {(l,c) :[inf, None, inf, None]}
Si le point O(0,0) est le point de départ et A(5,9) est le point d'arrivée, alors le sommet B(0,1) voisin de O prend comme valeurs :
{(0,1) :[10, 106, 116, (0,0)]}
On peut aussi choisir l'écriture suivante,
{(0,1) :{'G' :10,'H' :106,'F' :116,'pred' :(0,0)}}

3.b Fonctions à définir

3.b.i L'heuristique

```
def heuristique(pt1:tuple,A:tuple)->int:
"'Calcul de la valeur de l'heuristique avec la méthode du cours, 14 pour un
déplacement diagonal et 10 pour un déplacement sur le côté
entrées : pt1, tuple ou list des coordonnées et A, tuple ou list des coordonnées
sortie : H : integer, valeur de l'heuristique "'
    a=abs(pt1[0]-A[0])
    b=abs(pt1[1]-A[1])
    if a<=b:
        return (a*14+(b-a)*10)
    else:
        return (b*14+(a-b)*10)
```

3.b.ii Le graphe pondéré

```
G=np.ones((5,9))
G[1,4]=-10
G[1,5]=-10
G[2,4]=-10
G[3,4]=-10
M={}
ligne,colonne=G.shape
for i in range(ligne):
    for j in range(colonne):
        if not G[i,j]==-10.0:
            M[i,j]=[np.inf,None,np.inf,None]
```

3.b.iii L'algorithme A*

Plan du cours

Références :

https ://khayyam.developpez.com/articles/algo/astar/

https ://www.youtube.com/watch ?v=-L-WgKMFuhE