esiea
ECOLE D'INGENIEURS
DU MONDE NUMERIQUE

THEORIE DES GRAPHES

12) L'ALGORITHME A*

L'algorithme A* est un algorithme qui a été proposé en 1968 par une équipe de chercheurs en IA (Intelligence Artificielle). Il permet, à partir d'un graphe, de déterminer **un** plus court chemin d'un sommet s dit sommet initial ou source à un sommet b dit sommet terminal ou encore but.

L'algorithme A* est une extension de l'algorithme de Dijkstra.

De par sa rapidité, l'algorithme A* est très utilisé dans les jeux de stratégie tel le jeu de Taquin ou encore lorsqu'on souhaite déplacer un robot d'un point à un autre tout en évitant les obstacles.

Contrairement à l'algorithme de Dijkstra qui explore en largeur tous les sommets du graphe, l'algorithme A* utilise une exploration en profondeur et n'explore que les sommets qui se rapprochent le plus du but recherché, avec possibilité de retour arrière lorsqu'un meilleur sommet est trouvé.

Pour diriger l'exploration vers le but recherché et, de ce fait, diminuer le nombre de nœuds à explorer et donc, le temps de calcul, l'algorithme A* se base sur une stratégie heuristique, c.-à-d., une stratégie de recherche qui utilise les connaissances du domaine.

La stratégie heuristique utilisée par l'algorithme A* utilise une fonction d'évaluation f(n) qui donne une estimation du meilleur plus court chemin du sommet initial au but et ce, tout en passant par le sommet n.

$$\mathbf{f(n) = g(n) + h(n)}$$

où,

n est un sommet du graphe ;

g(n) est la longueur du plus court chemin du sommet initial s au sommet n ;

h(n) est une estimation de la longueur du plus cout chemin du sommet n au but où, **h(n)** est une fonction heuristique qui **doit être admissible**, c.-à-d., une heuristique qui ne surestime pas (dépasse pas) la longueur réelle du sommet n au but.

Comme exemple d'une heuristique admissible, on peut citer la distance à vol d'oiseau du sommet n au but, distance qui ne dépasse jamais la distance réelle laquelle est la distance par la route.

Remarques

L'algorithme A* exige que la fonction heuristique h soit admissible et ce, afin de garantir que le chemin trouvé, s'il existe, est le meilleur.

Si l'on se permet d'utiliser une heuristique non admissible au sein de l'algorithme A*, alors ce dernier trouvera toujours un chemin s'il existe de la source au but. Toutefois, le chemin trouvé peut ne pas être le meilleur.

Si h1(n) et h2(n) sont deux heuristiques admissibles et donc, deux heuristiques qui conduisent à un chemin optimal, et si $0 \leq h1(n) \leq h2(n) \leq$ longueur réelle, alors le nombre de nœuds exploré par h2 est inférieur ou égal au nombre de nœuds exploré par h1.

Ainsi, face à un problème donné auquel peut correspondre plusieurs heuristiques admissibles, il convient donc de choisir l'heuristique qui se rapproche le plus de la longueur réelle et ce, pour des raisons de rapidité de l'algorithme.

Algorithme A*

Soient :

G = [X, U] un graphe valué ;

b : Le sommet but recherché ;

s : Le sommet source à partir duquel on cherche le plus court chemin ;

O : La liste dite liste ouverte qui est à construire (donc un résultat) et qui contient les sommets à examiner ;

F : La liste dite liste fermé qui est à construire (donc un résultat) et qui contient les sommets déjà examinés ;

Le principe de l'algorithme de A* est le suivant.

- ✓ **Initialisation :**

  On pose :

  O = {(s, f(s), vide)}

  F = ∅

  Si s = b, on s'arrête. Le but est atteint. Sinon, on ajoute le triplet {(s, f(s), vide) à F, et on démarre avec le sommet n=s.

- ✓ **1ère étape:**

  On considère tous les successeurs succ de n (ou tous les voisins de n si le graphe n'est pas orienté).

  Pour chaque succ de n, on détermine f(succ).

- ✓ **2ème étape:**

  Pour chaque succ de n,

  - Si succ n'appartient pas à O et succ n'appartient pas à F, on ajoute le triplet (succ, f(succ), n) à O où, n est le père de succ.
  - Si succ est déjà dans O ou dans F, et si la nouvelle valeur de f(succ) est plus petite que l'ancienne, alors on supprime l'ancien triplet de O ou de F et on ajoute le nouveau triplet à O.

- ✓ **3ème étape:**

  On considère le successeur succ de la liste O tel que f(succ) est minimum.

  Si O est vide, on s'arrête. Le problème n'admet pas de plus cout chemin de s à b.

- ✓ **4ème étape:**

  On ajoute succ à F et on le supprime de O.

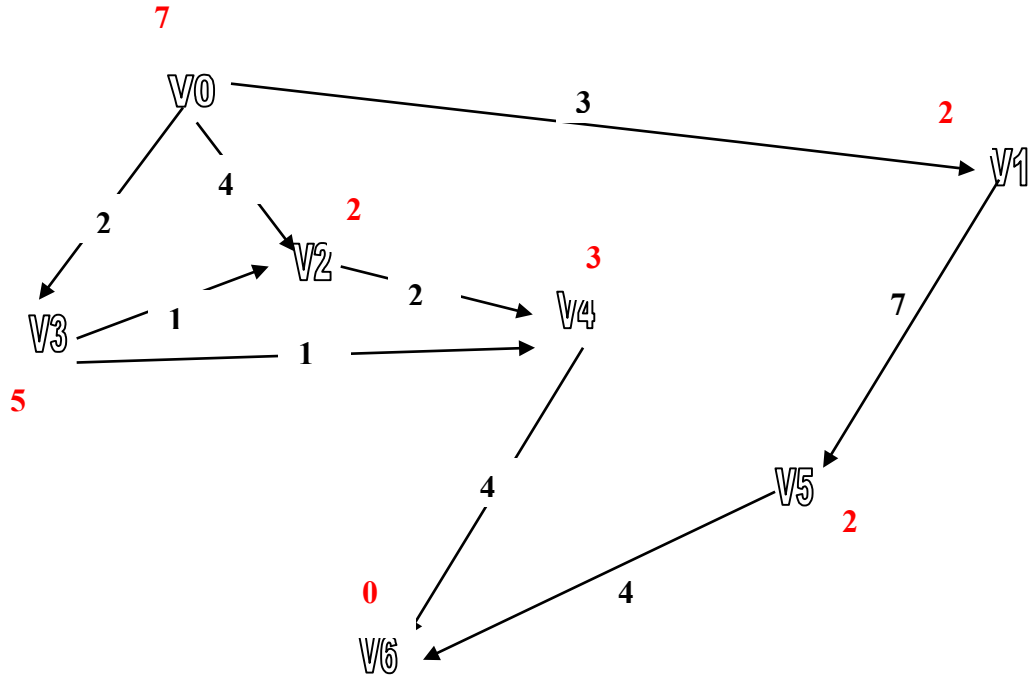
- ✓ **5ème étape:**

  On va à la 1ère étape avec comme sommet n le sommet succ

Exemple

Considérons le graphe suivant qui représente les routes entre certaines villes, et supposons que l'on veuille un plus court chemin de la ville V0 à la ville V6.

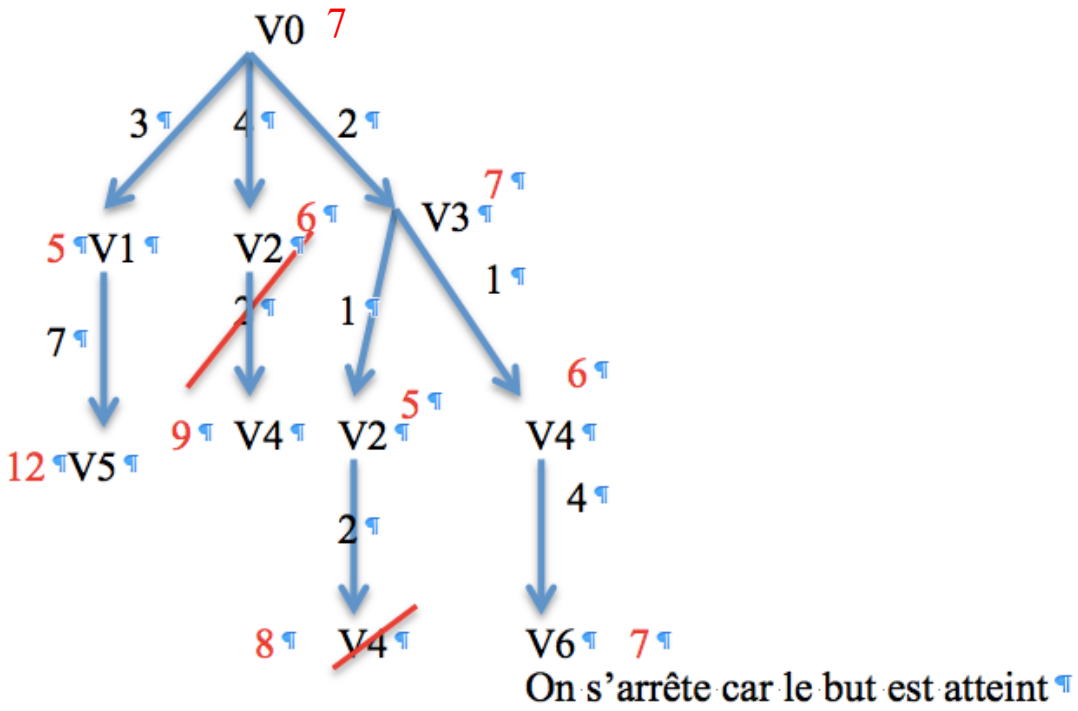
Par ailleurs, considérons comme heuristique admissible, h(Vi), la distance à vol d'oiseau de la ville Vi à la ville V6.



La liste ouverte O	La liste Fermée F
{(V0, 7, vide)} {(V0, 7, vide)}	∅ {(V0, 7, vide)}
{(V1, 5, V0), (V2, 6, V0), (V3, 7, V0)} {(V2, 6, V0), (V3, 7, V0)}	{(V0, 7, vide)} {(V0, 7, vide), (V1, 5, V0)}
{(V5, 12, V1), (V2, 6, V0), (V3, 7, V0)} {(V5, 12, V1), (V3, 7, V0)}	{(V0, 7, vide), (V1, 5, V0)} {(V0, 7, vide), (V1, 5, V0), (V2, 6, V0)}
{(V4, 9, V2), (V5, 12, V1), (V3, 7, V0)} {(V4, 9, V2), (V5, 12, V1)}	{(V0, 7, vide), (V1, 5, V0), (V2, 6, V0)} {(V0, 7, vide), (V1, 5, V0), (V2, 6, V0), (V3, 7, V0)}
{(V2, 5, V3), (V4, 6, V3), (V5, 12, V1)} {(V4, 6, V3), (V5, 12, V1)}	{(V0, 7, vide), (V1, 5, V0), ~~(V2, 6, V0)~~ (V3, 7, V0)} {(V0, 7, vide), (V1, 5, V0), (V3, 7, V0), (V2, 5, V3)}

{(V4, 6, V3), (V5, 12, V1)} {(V5, 12, V1)}	{(V0, 6, vide), (V1, 5, V0), (V3, 7, V0), (V2, 5, V3)} {(V0, 6, vide), (V1, 5, V0), (V3, 7, V0), (V2, 5, V3), (V4, 6, V3)}
{(V6, 7, V4), (V5, 12, V1)} {(V5, 12, V1)}	{(V0, 6, vide), (V1, 5, V0), (V3, 7, V0), (V2, 5, V3), (V4, 6, V3)} {(V0, 6, vide), (V1, 5, V0), (V3, 7, V0), (V2, 5, V3), (V4, 6, V3), (V6, 7, V4)} Comme on a atteint le but V6, on s'arrête

Le chemin optimal est donc V0→V3→V4→V6.



Exercice

Ecrivez l'algorithme A*.