STEFANOS
Alexandre
21905831

Compte rendu

TP7-8 Partie B

La partie B du TP7-8 est sur l'algorithme de Dijkstra et donc la recherche du plus court chemin. Étudions dans un premier temps le Pseudo-code vu dans le cours du professeur Birmelé :

Data: Un graphe connexe valué G de fonction de poids ω et un sommet u
Result: Un arbre couvrant T et la distance $D(v) = d_G(u, v)$ pour tout v

$T = \{u\}$
for $v \in V(G)$ **do**
  $dist\_prov(v) := \infty$; $pere(v) := \emptyset$
end
$dist\_finale(u) := 0$; $dernier\_ajout := u$

while $V(T) \neq V(G)$ **do**
  for v *voisin de* $dernier\_ajout$ **do**
    if $dist\_finale(dernier\_ajout) + \omega(dernier\_ajout, v) < dist\_prov(v)$
    then
      $dist\_prov(v) :=$
       $dist\_finale(dernier\_ajout) + \omega(dernier\_ajout, v)$
      $pere(v) := dernier\_ajout$
    end
  end
  Selectionner $v \notin V(T)$ tel que $dist\_prov$ est minimum
  Ajouter $(pere(v), v)$ a T
  $dist\_finale(v) := dist\_prov(v)$
  $dernier\_ajout := v$
end

Algorithm 6: Algorithme de Dijkstra

En entrée, on donne un graphe connexe valué G de fonction de poids ω et un sommet u de G. Le but est de construire un sous-graphe de T dans lequel les différents sommets sont classés par ordre croissant de leur distance minimale au sommet de départ. La distance correspond à la somme des poids des arcs empruntés.
On considère un graphe valué G et un sommet u de G afin de construire un arbre couvrant T_u tel que, pour tout sommet v, la distance de u à v est la même dans G et dans T_u.
En sortie, on retourne un arbre couvrant T et la distance $D(v) = d_G(u, v)$ pour tout v.
La complexité est en $O(n^2)$. La complexité de cet algorithme peut être réduit en utilisant des tas de Fibonacci par exemple.

L'algorithme que nous avons codé est une mise en forme de l'algorithme de Dijkstra en Java. Cependant, il y a une majeure différence : le choix du nœud à explorer. On le choisit en fonction de la somme entre la distance temporaire et une heuristique à définir et non en

ayant la plus petite distance temporaire comme dans l'algorithme de Dijkstra. C'est l'algorithme A* (A Star).
L'heuristique utilisé est la distance euclidienne (distance estimée jusqu'au dernier nœud).
J'ai utilisé la mise en forme (println, classes, etc) des exemples sur Wikipedia Dijkstra, de même pour l'exemple dans le README.txt .
Mon projet est séparé en plusieurs classes : Arrete, AstarAlgo, DijkstraAlgo, Graphe, Main, Sommet et Tableau.

Mini Projet

J'ai choisi le projet Labyrinthe.
Mon projet est séparé en plusieurs classes : Arrete, Graphe, Labyrinthe, Main et Sommet.