Projet Info 2nd Semestre :

Trouver le plus court chemin dans un graphe grâce à l'algorithme A*

Lilian Billod

Mathis Sauret

Introduction

Ce projet d'informatique de fin de second semestre a pour but de nous faire exploiter toutes les connaissances acquises durant les cours, notamment sur les différentes structures étudiées comme les listes chaînées, les piles et les tas entre autres. L'objectif est de pouvoir déterminer le plus court chemin entre deux sommets d'un graphe grâce à l'algorithme A* (astar). Il revient donc à nous de choisir quelles structures implémenter et comment organiser notre code pour relever le défi.

Sommaire

1.Implantation

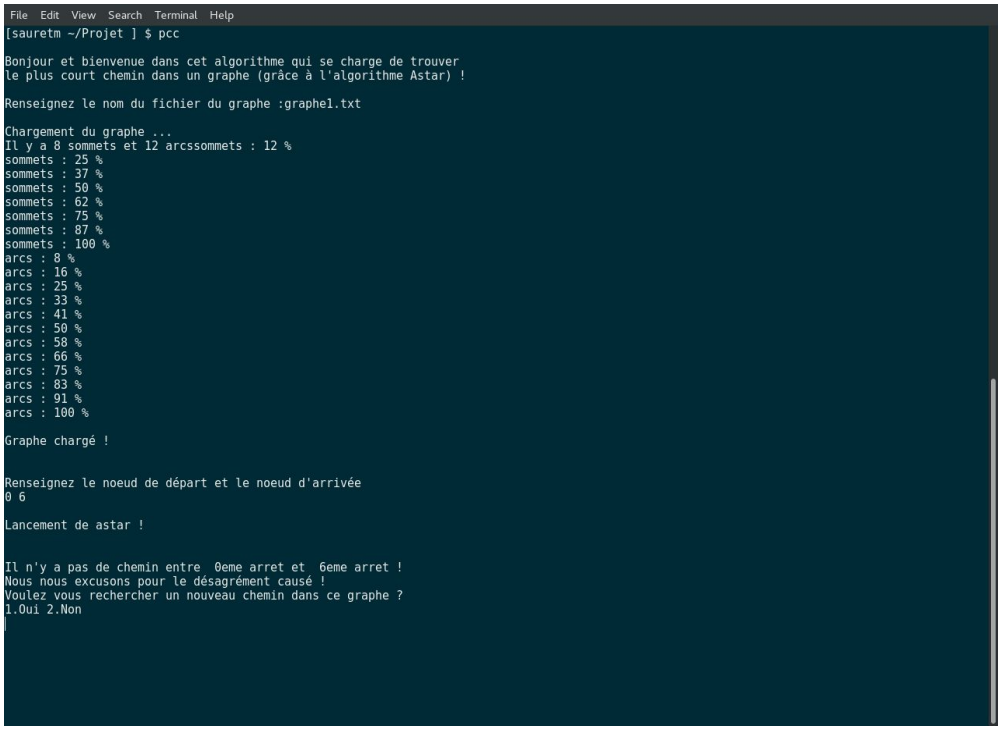
A. Etat du logiciel : ce qui fonctionne,ce qui ne fonctionne pas

Le logiciel est nommé pcc et permet de trouver le plus court chemin entre deux sommets d'un graphe. En premier lieu, le programme demande à l'utilisateur d'entrer un nom de fichier contenant un graphe. Le programme lit ensuite le graphe. Il détecte n'importe quelle erreur de lecture ou bien l'absence de fichier. Une fois la lecture terminée, il demande à l'utilisateur le numéro du noeud de départ et du noeud d'arrivée qu'il cherche à relier. Le programme fait un test pour s'assurer que les valeurs soient bien des véritables sommets puis il exécute l'algorithme A* sur le graphe avec ces données. Il écrit ensuite le coût du plus court chemin trouvé et écrit ensuite la suite des sommets pour parcourir ce chemin.

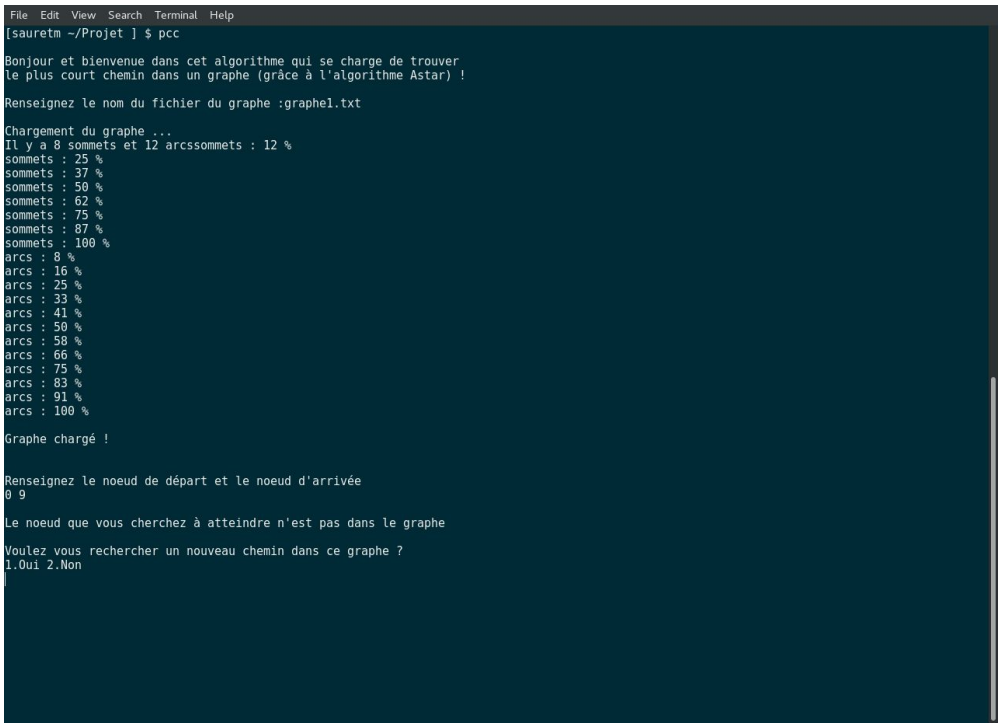
B. Tests effectués

Nous avons testé l'algorithme sur la plupart des graphes fournis, celui-ci fonctionne bien et rapidement sur des graphes de tailles moyennes (jusqu'à environ 20000 sommets). Au dessus l'algorithme fonctionne toujours mais nécessite un temps de calcul un peu plus important (de 30 secondes à 5 minutes).
Tout au long du développement nous avons aussi utilisé plusieurs autres programmes de debug pour tester séparément chacun des aspects du code. Ces fichiers de tests nous ont permis de valider au fur et à mesure du développement chacun des morceaux du code.

C. Exemple d'exécution



Test avec un sommet hors du graphe



Test avec un sommet inatteignable

```
File  Edit  View  Search  Terminal  Help

[sauretm ~/Projet ] $ pcc

Bonjour et bienvenue dans cet algorithme qui se charge de trouver
le plus court chemin dans un graphe (grâce à l'algorithme Astar) !

Renseignez le nom du fichier du graphe :graphe1.txt

Chargement du graphe ...
Il y a 8 sommets et 12 arcs
sommets : 12 %
sommets : 25 %
sommets : 37 %
sommets : 50 %
sommets : 62 %
sommets : 75 %
sommets : 87 %
sommets : 100 %
arcs : 8 %
arcs : 16 %
arcs : 25 %
arcs : 33 %
arcs : 41 %
arcs : 50 %
arcs : 58 %
arcs : 66 %
arcs : 75 %
arcs : 83 %
arcs : 91 %
arcs : 100 %

Graphe chargé !


Renseignez le noeud de départ et le noeud d'arrivée
0 5

Lancement de astar !

Le plus court chemin a un coût de 22.000000

 0eme arret
  |
  v
 1er arret
  |
  v
 4eme arret
  |
  v
 5eme arret

Voulez vous rechercher un nouveau chemin dans ce graphe ?
1.Oui 2.Non
```

Test fonctionnel

```
File  Edit  View  Search  Terminal  Help
  Sommets49519
  |
  v
  Sommets49965
  |
  v
  Sommets49529
  |
  v
  Sommets49528
  |
  v
  Sommets49908
  |
  v
  Sommets49907
  |
  v
  Sommets49906
  |
  v
  Sommets49905
  |
  v
  Sommets49966
  |
  v
  Sommets49548
  |
  v
  Sommets49547
  |
  v
  Sommets50003
  |
  v
  Sommets50072
  |
  v
  Sommets49583
  |
  v
  Sommets49582
  |
  v
  Sommets49585
  |
  v
  Sommets50001

Le plus court chemin a un coût de 2664951.000000

Voulez vous rechercher un nouveau chemin dans ce graphe ?
1.Oui 2.Non
```

Test avec un gros fichier (grapheColorado.txt)

D. Optimisations et extensions

Le code utilise un tri tas pour l'algorithme astar ce qui permet de gagner du temps d'exécution. De plus, tous les sommets et les arcs sont stockées en un seul endroit en mémoire, ce qui permet d'économiser de la place.
Le code est bien commenté ce qui permet un travail en groupe.
Les structures utilisées dans le code sont tout à fait exploitables pour un autre algorithme ou une autre utilisation.

Lors de la lecture, l'algorithme affiche un pourcentage de complétion pour indiquer où en est la lecture des sommets et des arcs.

L'ensemble du code source est disponible sur github. L'utilisation de github nous a grandement facilité la tâche pour travailler ensemble sur le même projet.

L'algorithme est découpé en diverses fonctions chargées de diverses tâches et séparées en deux fichiers principaux. Le premier structure.c regroupe toutes les fonctions de bas sur les structures implémentées. Le second fonctions.c regroupe les fonctions les plus complexes chargées comme l'algorithme astar ou bien la fonction de lecture d'un graphe dans un fichier. Les structures utilisées sont quant à elles définies dans le header structure.h.

Nous avons utilisés plusieurs structures toutes fonctionnelles :

-GRAPHE est composée de deux informations, n le nombre de sommets et tab qui est un tableau de T_SOMMET

-T_SOMMET représente un sommet du graphe, il contient toutes les informations dessus : son nom, ses coordonnées, son numéro et la listes des arcs vers ses voisins.

-T_ARC représente un arc entre deux sommets du graphe, il contient le numéro du sommet d'arrivée et le coût pour l'atteindre.

-L_ARC est une liste chainée de T_ARC

-L_SOMMET est une liste chainée de T_SOMMET

2.Suivi

A. Problèmes rencontrés

La lecture de fichier a nécessité beaucoup de test afin d'aboutir à un résultat correct. Pour commencer, certains des fichiers fournis afin de tester les fonctions n'avaient pas la même norme que celle indiquée dans le sujet. Parfois, il manquait les coordonnées du point, parfois ils les arcs n'étaient pas présents, nous avons dû modifier ces fichiers tests dans un premier temps pour faire des tests.
Nous avons aussi eu un soucis avec la fonction strcpy, en effet nous cherchions à copier un char* dans un autre char* avec la fonction strcpy. Nous pensions que la fonction allouerait seule la place mémoire pour la destination. Or ce n'est pas le cas et nous avions une erreur de segmentation. Le debugger gdb nous a permis de retrouver la source de l'erreur. Dorénavant nous utilisons la fonction strdup, qui elle effectue l'allocation mémoire avant la copie.

Pour la lecture du fichier, l'algorithme affiche un pourcentage de complétion pour indiquer où en est la lecture des sommets et des arcs

Pour l'implémentation des structures, nous n'avons eu aucun gros problèmes, la plupart des fonctions avaient déjà été codées lors des TPs du semestre 2. Il nous a suffit de les réadapter ici. Cette étape a cependant mis un peu de temps car nous voulions être sûrs que tout fonctionne. Nous avons donc fait beaucoup de tests.

Lors de l'implémentation de Astar, nous avons fait une première version utilisant pour LF et LO une structure de liste de sommets appelée L_SOMMET. Sur les petits fichiers, cela fonctionnait très bien. Cependant, pour les plus gros fichiers, les problèmes apparaissaient. Une telle structure impose de faire à chaque fois une copie du sommet entier (nom, coordonnées, voisins) à chaque mouvement et cela est extrêmement lourd en mémoire. Si bien, qu'une erreur de segmentation survenait presque à coup sûr pour les gros fichiers.
Nous avons rectifié cela en changeant totalement les structures LF et LO. Nous utilisons désormais des tableaux d'entiers.

B. Planning effectif

Nous nous sommes réparti le travail au préalable, Mathis s'est occupé de la lecture des fichiers et Lilian s'est occupé de coder les structures nécessaires ainsi que l'algorithme astar. Cela nous a pris une dizaine d'heure environ pour arriver à un code fonctionnel.

Nous avons choisi de rassembler tout notre travail sur github afin que celui-ci soit accessible partout par chacun des membres.

C. Ce qu'on a appris et ce qui nous manque

Nous avons pu mettre en pratique la plupart des acquis du second semestre. Nous nous sommes rendu compte du temps et du travail nécessaire afin d'arriver à un résultat convenable. Il manque encore beaucoup de chose à notre algorithme. Tout d'abord, nous aurions aimé avoir plus de temps pour optimiser le fonctionnement de l'algo astar. Le temps de réponse de l'algorithme pour les gros fichier est clairement trop importants.
De plus, nous aurions bien aimé avoir les moyens de faire une interface graphique pour notre algorithme afin que le graphe soit affiché ainsi que le plus court chemin.

3.Conclusion

Pour conclure, ce projet nous a enrichi. Il nous a aider à renforcer nos compétences informatiques en relevant un défi de taille. Il nous a permis de nous confronter à la réalité du travail en équipe.