

TP3 Algo Avancée

TP Graphes

Da Costa Tom | Mottino Loris | Verrier Martin

# Introduction

Lien du dépôt Github : https://github.com/iogepryga/ALG\_TP3.git

Lors de ce TP nous allons manipuler une nouvelle structure de donnée : le graphe. Nous allons améliorer notre compréhension et notre maitrise de celle-ci à travers des fonctions de manipulation des graphes que nous allons devoir coder ainsi que des algorithmes déjà connu comme le parcours en largeur, en profondeur, ou encore l‘algorithme de Dijkstra par exemple.

# Le code

Lors de ce TP nous avons codé de nombreuses fonctions et allons donc prendre quelques minutes pour en expliquer certaines ici.

**void afficher\_graphe\_largeur (pgraphe\_t g, int r)**

Pour cette fonction nous utilisons une pile. Nous allons utiliser cette file afin de sauvegarder les sommets que nous voulons parcourir plus tard dans l’algorithme.

Plus précisément, nous allons parcourir un à un les sommets par étage (on appel ici étage l’ensemble des sommets à une même distance du sommet de départ) tout en stockant dans une pile les arcs de chaque sommet qui mènent aux sommets de l’étage inférieur, pour ensuite pouvoir parcourir un à un les sommets de l’étage du dessous et ainsi de suite.

**int hamiltonien(pgraphe\_t g, chemin\_t c)**

Pour vérifier si le chemin donné en paramètre est bien un chemin hamiltonien, on parcourt le chemin en initialisant à 1 tous les champs tmp de chaque sommet parcouru lors du parcours du chemin. Ensuite il nous suffit alors de reparcourir tout le graphe à l’aide de la liste chainé contenant l’ensemble des sommets du graphes. Si un sommet n’a pas sa valeur tmp à 1, alors le chemin donné ne parcourt pas ce sommet, et donc il n’est pas hamiltonien.

# Conclusion

Lors de ce TP nous avons appris à manipuler une nouvelle structure de donnée : le graphe. Nous avons codé toutes sortes de fonctions afin d’exécuter des algorithme connu (parcours en largeur, profondeur, Dijkstra …) mais également des fonctions de manipulation des graphes.