Rapport projet AG44

Corentin Menier & Loïc MARTIN

Automne 2018

Table des matières

Constitution du dossier

Notre projet a pour but de permettre la représentation de Graph, par la création de Vertex et d'Edge, pour y appliquer différentes fonctions de tri et de recherche.

Il est composé de 16 fichiers différents, dans 3 dossiers séparés.

Dans le dossier principal, on peut trouver :

- 5 fichiers .cpp : Graph.cpp, Vertex.cpp, Edge.cpp, Heap.cpp et Client.cpp
- 4 fichiers .h : Graph.h, Vertex.h, Edge.h et Heap.h
- Un Makefile permettant de compiler et d'utiliser ces fichiers

Dans le dossier intitulé « Files », il y a 4 exemples de graphs. Chaque exemple vient avec un fichier .txt lisible par notre programme grâce à la fonction fileToGrap(), et un fichier .png qui représente le graph. Il y a un exemple de graph de chaque type : un graph non orienté représenté par une matrice, un graph orienté représenté par une matrice, un graph non orienté représenté par une liste et un graph orienté représenté par une liste.

- 4 fichiers .png : exampleDirectedMatrix.png, exampleDirectedList.png, exampleUndirectedList.png et exampleUndirectedMatrix.png
- 4 fichiers .txt : exampleDirectedMatrix.txt, exampleDirectedList.txt, exampleUndirectedList.txt et exampleUndirectedMatrix.txt

Dans le dossier intitulé « Readings » se trouvent :

- Le README.md
- L'énoncé du projet
- Ce rapport nommé « Rapport projet AG44 »

Les classes

Notre projet fonctionne sur la création de 3 classes : Graph, Vertex et Edge.

Graph

La classe Graph contient les attributs suivants :

- **nbVertex** (int) : le nombre de Vertex dans le graph
- **time** (int) : un entier utiliser pour le DFS
- **isDirected** (boolean) : indique si le graph est orienté ou pas
- **isMatrix** (boolean) : indique si le graph est représenté par une matrice ou par une liste d'adjacence
- **vertexList** (vector<Vertex*>) : la liste de tous les vertex du graph
- **edgeList** (vector<Edge*>) : la liste de tous les edge du graph
- **adjencyList** (vector<vector<vector<int> > >) : la liste d'adjacence. Pour l'exemple adjencyList[i][j][k], j représente les différents edge reliés au Vertex i, adjencyList[i][j][0] le vertex destination et adjencyList[i][j][1] le poids de l'edge.
- **adjencyMatrix** (vector<vector<int> >) : la matrice d'adjacence. Pour l'exemple adjencyMatrix[i][j], adjencyMatrix[i][j] est le poids de l'edge reliant i en vertex source et j en vertex destination.

Un graph ne peut pas avoir son adjencyList et son adjencyMatrix de renseigné en même temps. L'un des deux est forcément null. Pour savoir lequel, utiliser le booléen isMatrix. Pour changer le type de représentation d'un graph, utiliser matrixToList() ou listToMatrix().

Vertex

La classe Vertex contient les attributs suivants :

- **id** (int) : l'identifiant du Vertex
- **color** (int) : deux entiers utilisés dans le BFS
- **dist** (int) : deux entiers utilisés dans le BFS
- **d** (int) : utilisé dans le DFS
- **f** (int) : utilisé dans le DFS
- **distance** (int) : utilisé dans relax
- **pred** (Vertex*) : utilisé dans le BFS
- **predecessor** (Vertex*) : utilisé dans plusieurs algorithmes

Edge

La classe Edge a les attributs suivants :

- **id** (int) : l'identifiant de l'edge
- **weight** (int) : le poids de l'edge
- **src** (Vertex*) : le vertex source de l'edge
- **dst** (Vertex*) : le vertex destination de l'edge