# Specification

We shall define two classes named *Graph* (representing a *directed graph*) and *WeightedGraph* (representing a *weighted, directed graph*).

We need some auxiliary classes:

***AdjacentVertex*** – represents an adjacent vertex associated with a given vertex; also contains the cost of the edge between the two, and an overloaded equality operator

***CustomHash*** – a custom hashing type, used for the unordered\_map which contains the cost of each Edge

***Edge*** – contains the two vertices which determine an edge, as well as an overloaded equality operator (again, required by the unordered\_map)

***VectorIterator*** – iterator over a list of multiple AdjacentVertex

The class ***Graph*** will provide the following methods:

Graph();

void clearGraph();

int getTotalNrVertices();

int getNrEdges();

int getNrVertices();

bool isActiveVertex(int vertex);

bool isEdge(int srcVertex, int destVertex);

int getInDegree(int vertex);

int getOutDegree(int vertex);

VectorIterator inEdgesIterator(int vertex);

VectorIterator outEdgesIterator(int vertex);

void addEdge(int srcVertex, int destVertex);

void removeEdge(int srcVertex, int destVertex);

void addVertex();

void removeVertex(int vertex);

The class ***WeigthedGraph*** will provide the following methods:

WeightedGraph();

void clearGraph();

int getEdgeCost(int srcVertex, int destVertex);

void modifyEdgeCost(int srcVertex, int destVertex, int newCost);

void addEdge(int srcVertex, int destVertex, int cost);

void removeEdge(int srcVertex, int destVertex);

void printGraph();

void removeVertex(int vertex);

WeightedGraph(const WeightedGraph& originalGraph);

# Implementation

Class ***Graph*** will have the following data members:

int nrActiveVertices;

int nrTotalVertices;

int nrEdges;

vector <vector <AdjacentVertex>> inEdges;

vector <vector <AdjacentVertex>> outEdges;

Class *WeightedGraph* will have the following data members:

unordered\_map <Edge, int, CustomHashFunction> costEdges;