# **Specification**

We will define two classes: **Graph** and ***WeightedGraph***, representing a Directed Graph and a Weighted Directed Graph, respectively. We will use two lists of neighbors for each vertex, to facilitate the parsing of both inbound and outbound edges for a given vertex. In addition, the WeightedGraph will contain a dictionary which will map each edge to a cost (real value).

Each vertex is uniquely identified by an integer denoting its index, and each edge is denoted by the two vertices which it unites.

We need some auxiliary classes:

***AdjacentVertex*** – represents an adjacent vertex associated with a given vertex; contains the other vertex and 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**; contains a custom hashing function

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

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

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

# **Graph()**

Creates a new, empty graph

# **void initEmptyGraph(int nrVertices)**

Initializes an already built graph with a given number of vertices and no edges

# **void clearGraph()**

Clears the entire content of the graph

# **int getTotalNrVertices()**

Returns the total number of vertices of the graph (including those which have been deleted)

# **int getNrEdges()**

Returns the number of edges of the graph

# **int getNrVertices()**

Returns the number of ‘active’ (non-deleted) vertices of the graph

# **bool isActiveVertex(int vertex)**

Checks if a vertex has been deleted or not

# **bool isEdge(int srcVertex, int destVertex)**

Checks if an edge, determined by its 2 endpoints, exists or not

# **int getInDegree(int vertex)**

Returns the in degree of a vertex

# **int getOutDegree(int vertex)**

Returns the out degree of a vertex

# **VectorIterator inEdgesIterator(int vertex)**

Returns an iterator over the inbound edges of a given vertex

# **VectorIterator outEdgesIterator(int vertex)**

Returns an iterator over the outbound edges of a given vertex

# **void addEdge(int srcVertex, int destVertex)**

Adds an edge from srcVertex to destVertex to the graph

# **void removeEdge(int srcVertex, int destVertex)**

Removes the edge from srcVertex to destVertex from the graph

# **void addVertex()**

Adds a new vertex to the graph

# **void removeVertex(int vertex)**

Removes a given vertex from the graph

The class ***WeigthedGraph*** is inherited from ***Graph*** and will provide the following methods:

# **WeightedGraph()**

Creates a new, empty, weighted graph

# **void clearGraph()**

Clears the entire content of the graph (overwrites the method from Graph)

# **int getEdgeCost(int srcVertex, int destVertex)**

Determines the cost of an edge determined by its 2 endpoints

# **void modifyEdgeCost(int srcVertex, int destVertex, int newCost)**

Modifies the cost of an edge determined by its 2 endpoints

# **void addEdge(int srcVertex, int destVertex, int cost)**

Adds a new edge, determined by its 2 endpoints and a given cost, to the graph

# **void removeEdge(int srcVertex, int destVertex)**

Removes an edge, determined by its 2 endpoints, from the graph (overwrites the method from Graph)

# **void printGraph()**

Prints the entire graph to the screen

# **void removeVertex(int vertex)**

Removes a given vertex from the graph (overwrites the method from Graph)

# **WeightedGraph(const WeightedGraph& originalGraph)**

Copy constructor for the WeightedGraph type

# **WeightedGraph& operator = (const WeightedGraph& originalGraph)**

Overloads the assignment operator

# **Implementation**

The class ***Graph*** will have the following data members:

# **int nrActiveVertices**

The number of non-deleted vertices in the graph

# **int nrTotalVertices**

The total number of vertices in the graph (including the deleted ones)

# **int nrEdges**

The number of edges in the graph

# **vector <vector <AdjacentVertex>> inEdges**

The list of lists of inbound adjacent vertices for each vertex

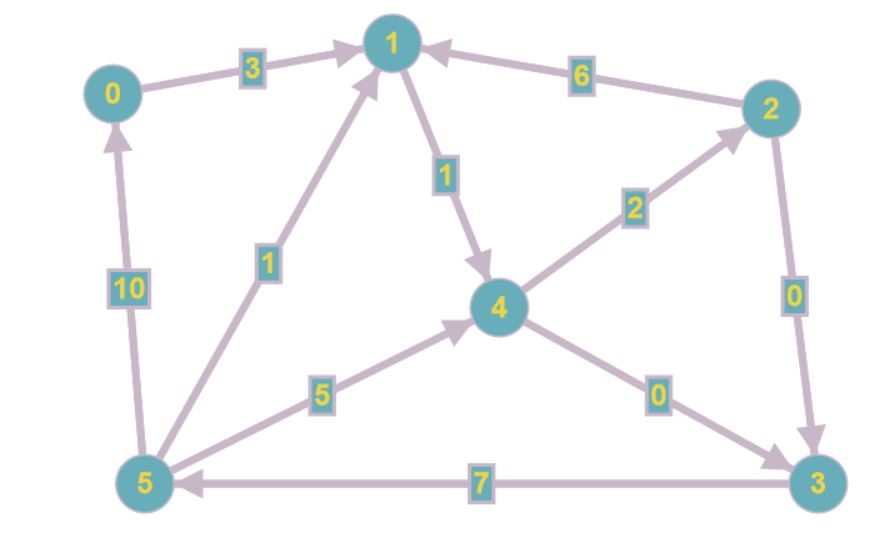
# **vector <vector <AdjacentVertex>> outEdges**

The list of lists of outbound adjacent vertices for each vertex

Class ***WeightedGraph*** provides the following data members (in addition to those in ***Graph***):

# **unordered\_map <Edge, int, CustomHashFunction> costEdges**

Structure which maps each edge to a cost (real number)



costEdges = {

(0, 1): 3

(1, 4): 1

(2, 1): 6

(2, 3): 0

(3, 5): 7

(4, 2): 2

outEdges = {

0: [1]

1: [4]

2: [1, 3]

3: [5]

4: [2, 3]

5: [0, 1, 4]

}

inEdges = {

(4, 3): 0

(5, 0): 10

(5, 1): 1

(5, 4): 5

}

0: [5]

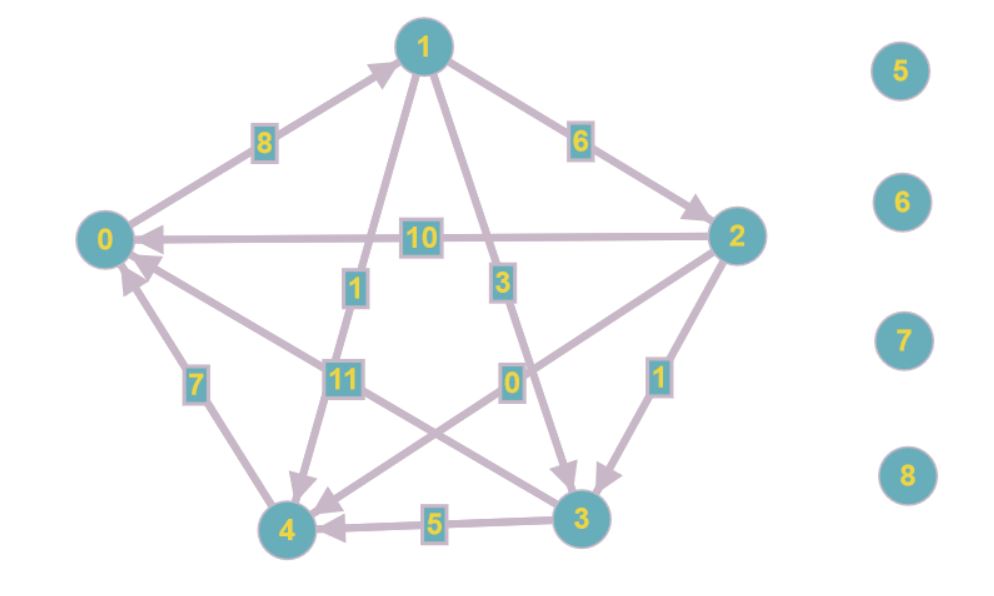
1: [0, 2, 5]

2: [4]

3: [2, 4]

4: [1, 5]

5: [3]

}

costEdges = {

(0, 1): 8

(1, 2): 6

(1, 3): 3

(1, 4): 1

(2, 10): 10

(2, 3): 1

(2, 4): 0

(3, 0): 11

(3, 4): 5

(4, 0): 7

}

outEdges = {

0: [1]

1: [2, 3, 4]

2: [0, 3, 4]

3: [0, 4]

4: [0]

5: []

6: []

7: []

8: []

}

inEdges = {

0: [2, 3, 4]

1: [0]

2: [1]

3: [1, 2]

4: [1, 2, 3]

5: []

6: []

7: []

8: []

}