**Specification**

Graph class Create the graph and all does all operation on graph (add/remove edge/vertex, cost of edge etc)

Each edge is member of two double-linked lists, a list of all edges that are outbound from the same vertex, and a list of all edges inbound to the same vertex. Each vertex has a pointer to one of the edges in each list.

Every edge has 4 parameters ( start vertex, end vertex, cost and id). Everytime an edge is added the edge\_id increases, stat vertex, end vertex and cost are read from file.

Methods defined here:

1.\_\_init\_\_(self, fileName)

2.readGraph(*self*, fileName)

Function that read from file and create a graph

-First read the first line: the number of vertices and number of edges

-creates empty lists for in and out bounds vertices

-then add edges; read from file and create an id for every edge

-in case of error, it prints a message

3.addEdge(self, vertexStart, vertexEnd, cost, id)

Function that add an edge to graph

-check if the edge we want to add exists;

-if not then add the edge with its cost and a new id

4.addVertex(self, vertex)

add a vertex

-create empty list for in and out bound for vertex

-then append the list of vertices and add the vertex

5.edgeIDExists(self, edgeID)

return True if edge exist; else return False

- check in edgeId`s list and return true or false

6.getCostOf(self, edgeID)

Function that return costs of an edge

- check in both directions if there s and edge and return its cost

7.getCostOfEdge(self, vStart, vEnd)

return the cost of an edge, if doesn’t exist returns None

-check in both directions if there an edge and return its cost if true

8. getEndpointsOf(self, edgeID)

Function that return edgeId

* Check if exist then return, else raise an exception

9.inDegreeOf(self, vertex)

return in degree of a vertex

- return length of in of a vertex

10.inboundEdgesOf(self, vertex)

return in bounds of a vertex

- return list of in bounds of a vertex

11.isEdge(self, vertexStart, vertexEnd)

-return True or False if the end Vertex is in list of start Vertex (means it is edge)

12.isolatedNodes(self)

creates a list with isloated nodes

- creates an empty list andcheck every vertices if have in or out bounds; if not then appends the empty list and add the vertex

13.modifyCostOf(self, edgeID, newCost)

modify the cost of an edge

-check if the edge exist then it modify cost if true

14.outDegreeOf(self, vertex)

return out degree of a vertex

-return length of out bounds of a vertex

15.outboundEdgesOf(self, vertex)

return out bounds of a vertex

-return the list of out bounds of a vertex

16.removeEdge(self, vertexStart, vertexEnd)

Function that remove an edge to graph

-check if edge exist between vertexStart and vertexEnd

-if true, removes the bounds, cost, and id of that edge

17. removeVertex(self, vertex)

remove a vertex

-remove vertex from vertices lists

-remove in and out bounds

18. vertexExists(self, vertex)

check if vertex exists

-return true or false if vertex is in vertices list

----------------------------------------------------------------------

**Data descriptors defined here**:

costs - dictionary for costs (a cost is set to keys: vertexStart and vertexEnd )

edgeIDs – dictionary for edges IDs

edges – number of edges

graphIn - dictionary (vertexStart and vertexEnd are keys; vertexStart – list of vertices where vertexStart has edges with every vertex from the list )

graphOut – dictionary

vertices – number of vertices

verticesList – list of vertices