Representation

```
def __init__(self):
   self._dictIn = {} – dictionary of dictionaries
   self._dictOut = {} - – dictionary of dictionaries
   self._dictCosts = {} – dictionary with pairs as keys
   self._vertices = 0
   self._edges = 0
```

Specification

Class DoubleDictGraph provides the following methods:

def __init__(self)

Constructs a graph without vertices or arcs.

def vertices(self)

Returns the number of vertices.

def edges(self)

Returns the number of edges.

def is_edge(self,x, y)

Checks whether or not there is an arc between x and y.

def is_vertice(self,n)

Checks whether or not n is a vertex.

def add_vertex(self)

Adds a new vertex to the graph.

def remove_vertex(self, vertex)

Removes the vertex n from the graph.

Precondition: n is a vertex .

def add_edge(self, x, y, cost)

Adds an edge to the graph.

Precondition: x and y are existent vertices and the edge x-y doesn’t exist.

def remove_edge(self, x, y)

Removes an edge from the graph.

Precondition: x-y is an edge.

def get_vertices(self)

Returns a list of all vertices.

def get_in_degree(self, vertex)

Returns the in degree of a given vertex.

Precondition: vertex is in the graph.

def get_out_degree(self, vertex)

Returns the out degree of a given vertex.

Precondition: vertex is in the graph.

def parse_outbound(self, vertex)

Returns the list of outbound neighbors of a given vertex.

Precondition: vertex is in the graph.

def parse_inbound(self, vertex)

Returns the list of inbound neighbors of a given vertex.

Precondition: vertex is in the graph.

def get_cost(self, x, y)

Returns the cost of a given edge.

Precondition: x and y are vertices in the graph.

def modify_cost(self,x, y,newValue)

Changes the cost of a given edge.

Precondition: the edge is in the graph.

def copy(self)

Returns a deep copy of the graph.

def get_costs(self):

Returns the dictionary of costs.

External functions:

def loadGraphs(graph, filename)

Loads a graph from a text file in the memory.

def storeGraph(graph, filename)

Stores a graph from memory to a text file

def generateRandomGraph(vertices, edges)

Returns a random generated graph with a given number of vertices and edges