Graph lab 1 06.02.2023

Input file:

5 7

0 1 -1

1 1 9

1 2 6

2 1 7

2 3 0

3 0 10

3 1 5

Operations: -add a vertex

!!all operations have preconditions

!!1. is\_vertex(v)

!!2. add\_vertex(v) – check if vertex

!!3. is\_edge(u,v)

!!4. add\_edge(u,v) – check

!!5.remove\_edge(u,v)

Ex: remove (1,1) – remove from the list of the succesors, predecessors, costs!!

Remove (2,3)

!!6.remove\_vertex(v)

Ex: remove(1)

We take 1 and go to din as k, we parse the predecessors of 1

!!7.write\_graph()

2 -1

3 0 10

4 -1

6 -1

2,4,6 – isolated vertex

3, 0 – vertex, create an edge with value 10

Another function for reading a file!!

Din(predecesors)

0-[3]

1-[0]-[1]-[2]-[3]

2-[1]

3-[2]

4-[]

6-[]

Dout(successers)

0-[1]

1-[1]-[2]

2-[1]-[3]

3-[0]-[1]

4-[]

6-[]

Dcosts(ex:tuple corresponding to edges)

<0,1> = -1

<1,1> = 9

<1,2> = 6

<2,1> = 7

<2,3> = 0

<3,0> = 10

<3,1> = 5

Create e class graph with all 3 dictionaries(Din, Dout, Dcosts)

**Required operations:**

* get the number of vertices; = nr of keys
* parse (iterate) the set of vertices; = return an iterrable object(list)
* given two vertices, find out whether there is an edge from the first one to the second one, and retrieve the *Edge\_id* if there is an edge (the latter is not required if an edge is represented simply as a pair of vertex identifiers); = not necessary to use an edge\_id(we can face a start and end point of an edge), is\_edge function
* get the in degree and the out degree of a specified vertex;=in -nr of predecesors,!check existence of vertex! Out-nr of succcesors
* parse (iterate) the set of outbound edges of a specified vertex (that is, provide an iterator). For each outbound edge, the iterator shall provide the *Edge\_id* of the current edge (or the target vertex, if no *Edge\_id* is used).
* parse the set of inbound edges of a specified vertex (as above);
* get the endpoints of an edge specified by an *Edge\_id* (if applicable);
* retrieve or modify the information (the integer) attached to a specified edge.=!edge must exist!, we go with that tuple, return the value or modify the value
* The graph shall be modifiable: it shall be possible to add and remove an edge, and to add and remove a vertex. Think about what should happen with the properties of existing edges and with the identification of remaining vertices. You may use an abstract Vertex\_id instead of an int in order to identify vertices; in this case, provide a way of iterating the vertices of the graph.
* The graph shall be copyable, that is, it should be possible to make an exact copy of a graph, so that the original can be then modified independently of its copy. Think about the desirable behaviour of an Edge\_property attached to the original graph, when a copy is made.
* Read the graph from a text file (as an external function); see the format below.
* Write the graph from a text file (as an external function); see the format below.
* Create a random graph with specified number of vertices and of edges (as an external function).=create a new object of type graph

Implement a menu!!!

Reading a file – 2 different functions

Documentatie-functii argumente,preconditie,descriere dictionare