Csoka Ervin – 938 (Prelungiri)

Representation

Vertex = { v | v represents a vertex in a directed graph; it is uniquely identified by an ID and holds references to the inbound and outbound Edges }

Vertex:
- vID : Integer
- in : Set<Edge>
- out : Set<Edge>
- degreeIn : Integer
- degreeOut : Integer

Edge = {e | e represents an edge in a directed graph; it is uniquely identified by the 2 vertices that it connects and has an associated weight }

Edge:
- v1 : Vertex
- v2 : Vertex
- weight : Integer

Graph = { g | g represents a directed graph; it holds a set of Vertex and a set of Edge }

Graph:
- vertices : Set<Vertex>
- edges : Set<Edge>
- noVertices : Integer
- noEdges : Integer

Interface

Graph()
- *Construct an empty graph*

Graph(filename : String)
- *Read the graph from the CSV list at @filename*

Graph(Graph other)
- *Copy constructor*

getVertices() : Set<Vertex>
- *Return the set of vertices of the graph*

getEdges() : Set<Edge>
- *Return the set of edges of the graph*

getNoVertices() : Integer
- *Return the number of vertices of the graph*

getNoEdges() : Integer
- *Return the number of edges of the graph*

Csoka Ervin – 938 (Prelungiri)

getVertexById(vid : int) : Optional<Vertex>
Search for a vertex in the graph, given it's id.
Return Optional<Vertex> - empty if a Vertex with the given @vid does not exist; holds the Vertex otherwise
getEdge(v1id : int, v2id : int) : Optional<Edge>
Search for an Edge in the graph given the ids for it's 2 vertices
Return Optional<Edge> - empty if an Edge with the given vertices as does not exist; holds the Edge otherwise

vertices() : Stream<Vertex>
Used to parse the set of vertices
edges() : Stream<Edge>
Used to parse the set of edges
existsVertex() : boolean
Check the existence of a vertex specified by its ID.
Return true if the vertex exists, false otherwise.
existsEdge(v1 : Vertex, v2 : Vertex) : Optional<Edge>
Check the existence of an edge specified by the vertices it connects
Return Optional<Edge> - empty if an Edge with the given vertices as does not exist; holds the Edge otherwise

addEdge(edge : Edge)
Add edge @e to the set of edges.
removeEdge(edge : Edge)
Remove edge @e from the set of edges
addVertex(vertex : Vertex)
Add vertex @vertex to the set of vertices
removeVertex(vertex : Vertex)
Remove vertex @vertex from the set of vertices

generateGraph(noVertices : Integer, noEdges : Integer) : Graph
Generates a random directed graph with @noVertices vertices and @noEdges edges
saveGraph(graph : Graph, filename : String)
Write @graph to @filename in the following manner:
Number_of_vertices Number_of_edges
Vertex_1(outbound) Vertex_2(inbound) Weight
Throws IOException if the file could not be created
readGraph(filename : String) : Graph
Read a directed graph from @filename
Throws FileNotFoundException if the specified file could not be found
Throws IllegalStateException if the read values for vertex cound and edge count don't match the actual number of edges and vertices