Alex-Tudor Petrean
#916

Graph algorithms - practical work no. 1

Documentation

- Language of implementation: C++
- Representation in memory: double list of neighbours for each vertex with costs
  - std::vector<int> for inbound neighbours
  - std::vector<int> for outbound neighbours
  - std::vector<int> for costs
- Method specification for graph:
  - Constructor
    - Will take no parameter
    - Will initialize all vectors as empty
  - Destructor
    - Will clear all memory
  - unsigned int getVertices()
    - Will return the total number of vertices as an unsigned integer
  - bool isEdge(int v1, int v2)
    - Will return true if there is an edge between v1 and v2
    - Because of the representation and the directed nature of the graph, calling the same method with the arguments swapped might return a different value
  - int getInDegree(int v)
    - Will compute and return the in degree of the vertex v sent as a parameter
  - int getOutDegree(int v)
    - Will compute and return the out degree of the vertex v sent as a parameter
  - int getCost(int v1, int v2)
    - will get the cost of the edge between v1 and v2 and return it as an integer
    - Because of the representation and the directed nature of the graph, calling the same method with the arguments swapped might return a different value
  - void setCost(int v1, int v2, int newCost)
    - will set the new cost on the edge between v1 and v2
  - void addVertex(int v)
    - will add a vertex to the graph

    - if the vertex already exists, an exception will be thrown
- void deleteVertex(int v)
    - will delete a vertex from the graph
    - if the vertex does not exist, an exception will be thrown
- void addEdge(int v1, int v2, int cost)
    - will add an edge with cost between the vertices v1 and v2
    - if the edge already exists, an exception will be thrown
- void deleteEdge(int v1, int v2)
    - will delete the edge between the vertices v1 and v2
    - if the edge does not exist, an exception will be thrown
- Graph* copyGraph()
    - will copy the graph object and return a pointer to the copy
- void readGraph(FILE* fileDesc)
    - will populate the graph from the file with the given file descriptor
- void writeGraph(FILE* fileDesc)
    - will write the graph to the file given by the descriptor
- void generateRandom(int v, int e)
    - will generate a random graph object with v vertices and e edges

The method specification might modify along the development of the class, and the iterators will be specified later.