**Object Oriented Programming**

**Ex2**

Part A

**By:**

Hagai Hen 313414872

Nir Geron 315874925

1) **Literature**:

1. <https://www.geeksforgeeks.org/travelling-salesman-problem-greedy-approach/>
2. <https://www.geeksforgeeks.org/dijkstras-shortest-path-algorithm-greedy-algo-7/>
3. <https://www.youtube.com/watch?v=XB4MIexjvY0>
4. <https://sites.google.com/site/gson/gson-user-guide>
5. <https://gist.github.com/raydac/df97493f58b0521fb20a>

**2) Algorithm Explaining**

This time we've been asked to build an algorithm for graphs and make an interface that shows the best solution for the "Travelling Salesman Problem", which gave us a list of place that we must visit and returns the quickest way to pass the whole places.

**Algorithm Classes**

MyGraphAlgo

This class implements the interface "DirectedWeightedGraphAlgo", For solving the "Travelling Salesman Problem" we need to find the quickest and most efficient way for planning the route, we were assisted Dijkstra's algorithm in greedily.  
Dijkstra's algorithm is an algorithm for finding the shortest paths between nodes in a graph.  
To find which point will be the best place for start we checked each couple of points to find the fastest start, and after that we checked the whole other points with the second point, and so on.  
In def "isConnected" we check if the graph is connected so we check if the graph is connected with "Depth first search algorithm" that help us to make sure we can reach each node and then we transpose the graph with the def "getTranspose" and check again with DFS if the graph is connected

In def "Center" we check the shortest path (by using Dijkstra Algo) for each node, and than we choose the longest path. from all the longest values, we choose the minimum to be the center.

In def "ShortestPath" we use Dijkstra Algo for find the shortest path for the src node.  
  
  
MyGraph

This class implements the interface DirectedWeightedGraph, the main function it's Mygraph that implements JSON with GSON and copy the data into two hashmaps (Nodes and Edges). And we have another function calls "connect" that make an edge between two nodes and give them a weight.

LandmarkThis class implements the interface GeoLocation, it's provided a point from three definition type, and has a function that calculate the distance between two 3D points.  
  
Edge

This class implements the interface EdgeData, it's provided an edge that has IDs of source and destination of edge and another variable that save the weight of the edge.

In addition, each edge has a tag that has a three-color option, white, gray and black and use it when we do a Depth first search.  
  
Node

This class implements the interface NodeData. it has some of variable, a landmark, id, tag, info

Generate Package

This package is take the detail of MyGraph type and convert to GenerateGraph, and by this class we generate the json file.

**3)UML diagram**

Diagram

Description automatically generated

How To Run:

Using Ex2 class, you have to use the function "runGUI" and enter a json file. In the interface you will see some button:  
Save – for saving the json file of the specific graph

Load – for loading another json file  
Center – will paint the center point, in the second click paint will remove.  
Shortest Path – will paint the shortest path between the to point that you entered, , in the second click paint will remove.

TSP – will returns the order of the nodes that you entered

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Case | Shortest Path | Center | TSP | isConnected | Load |
| 1,000 Nodes | 0.3 sec | 48.2 sec | 1.2 sec | 0.07 sec | 0.08 sec |
| 10,000 Nodes | 2.7 | - | 12.8 sec | 0.5 sec | 0.5 sec |
| 100,000 Nodes |  |  |  | 33.4 sec | 8.5 sec |

You can also run the GUI using this command "java -jar Ex2.jar <json\_file\_src>" in CMD / terminal. [Download Here](https://github.com/HagaiHen/OOP_Ex2/blob/master/src/Ex2/Ex2.jar)