Algorithm Section Documentation.

Author: Belal Hmedan

Saturday, December 7, 2019 7:14 AM

Introduction:

This Section handles data provided from Data Structure Provided by the Model which depends on OSM file, to get the shortest path between two points, each point represents Location on the Map.

==========

Main Classes :

1- Class MyGraphBuilder : This Class mainly takes the nodes included inside Ways only from the Model as an input, then builds the Vertices and the Edges Between Vertices to construct a Graph which we can take it as an input to the Next stage which is Dijkstra Algorithm.
Note: I'm using Boost Graph Library to implement this Graph Building here.

Member Functions :

1.1 MyGraphBuilder(Model) This Constructor takes the Model Data Structure as an input, then Stores it in the Private variable from the same Type : OurModel.

1.2 generateGraph() Void type Function, takes the WayMap
(all nodes included in ways and there is Path between them) from The Model
(So it's obviously Doesn't work with isolated nodes) and Generates The Graph.

1.3 distance(Vertex Nod1_ID, Vertex Nod2_ID) double Type Function, it takes the ID of 2 nodes as An input, and returns the Euclidean Distance between that Nodes.

1.4 Accessors:

A. getGraph() Function Type graph_t (which in turn is adjacency list) it Returns the Graph Stored in the Private Member of The Class: MyGraph.

B. getGraphMap() Function Type MapGraph (which is just a mapping between real ID idType, And Vertex Number) it returns the Graph Stored in the Private Member of The Class: MyGraphMap.

1.5 Mutators:

A. setGraph(graph_t) Function Type void it assigns the input parameter Graph into the private Member of The Class: MyGraph.

B. setGraphMap(GraphMap) Function Type void it assigns the input parameter GraphMap into The Private Member of the Class: MyGraphMap.

1.5 printGraph() Function Type void it prints numbers of Vertices and Edges, and has more Capabilities for printing, but I commented that part for time saving.

2- Class MyAlgorithm : This Class Mainly takes the Graph which was built in The past Class, the GraphMap (which links between The Original Model that Represents the OSM File nodes ID, and Vertices numbers in our Graph),
And the source node which we want start from to get the shortest path to give The Shortest path between two nodes.
Note: I'm using Dijkstra Algorithm from Boost Library to implement this Shortest Path finding here.

Member Functions :

2.1 MyAlgorithm(graph_t *const&* ,GraphMap ,idType) This Constructor takes 3 Parameters as input: 1. The Graph itself, 2. GraphMap which links the Model to The Graph, and 3. The Source Node, then returns all shortest paths to all other nodes in the graph.

2.2 getShortPath(GraphMap ,idType) Function Type Path(which in turn is a Vector of nodes idType) it takes two parameters, the GraphMap, and the Destination Node ID, and returns ShortPath Vector of Nodes which represents The Shortest Path between two Nodes .

Note: in case there was no path between provided nodes, this Vector will contain one default node "1540689869" which refers to Aldi Store, so that no crash happens to this part due to returning empty Vector.

2.3 Accessors:

A. getGraph()Function Type graph_t (which in turn is adjacency list) it returns the Graph Stored in the Private Member of The Class: MyGraph2.

B. getFlag() Function Type bool, it refers to that there is no path between the Nodes in case it Was 0, while it's default Value is 1.

2.4 Mutators:

A. setShortPath(idType) Function Type void takes the Node (idType) and puts it in the Path Vector ShortPath, as a part of the whole path.

B. setGraph(graph_t) Function Type void it assigns the input parameter Graph into the private Member of The Class: MyGraph2.

C. ResetShortPath() Function Type void , this function clears the Path vector, and makes it empty again.

2.5 PrintRawData() *const* Function Type void prints all data about the distance and Edges between two nodes which is a part of the whole shortest path(group of Edges) between source Node and all Other Nodes.

2.6 PrintPath() Function Type void prints the Nodes idType in the Shortest Path one by one.

3- Class ShortPath : This class added later to simplify things, it takes 3 Parameters, Source Node, Destination Node, and Model, and it gives the Path (Shortest Path) which is nothing except a victor Of Nodes idType.

Member Functions :

3.1 ShortPath(idType ,idType , Model) This Constructor takes 3 parameters as mentioned before, Source Node, Destenation Node, and The Model, and it builds the graph, applies the Algorithm, then stores the result in the Private Member mypath, which is a Path(Vector of Nodes idType).

3.2 Accessors:

A. getModel() Function Type Model, it returns OurModel, a Private Member of the Class where We stored our Model (Data Structure).

B. getSource() Function Type idType returns the ID of the Source Node.

C. getDestination() Function Type idType returns the ID of the Distenation Node.

D. getYourPath() Function Type Path returns mypath, a Private Member of the Class where We store our Shortest Path Vector.

3.3 Mutators:

A. setMyPath(Path) Function Type void assigns the parameter input to mypath, a Private Member of the Class where We store our Shortest Path Vector.

B. setSource(idType) Function Type void assigns the parameter input to Source_ID;a Private Member of the Class where We store our Source Node ID.

C. setDestination(idType) Function Type void assigns the parameter input to Destination_ID; a Private Member of the Class where We store our Destination Node ID.

D. setModel(Model) Function Type void assigns the parameter input to OurModel; a Private Member of the Class where We store our Model (Data Structure).

3.4 printMyPath() Function Type void prints the Nodes idType in the Shortest Path one by one.

End