ECE 250 - Project 5
Shortest Path using Dijktsra's Algorithm
Design Document
Ayushi Tiwari, UW UserID: attiwari
Apr 24th, 2020

1. Overview

Class 1: Node

Description:
Helped in creating a linked list for the vector.

Member Variables:
1. String City
2. Node type pointer next
3. Double edgeWeight

Class 2: linkedList
It inplements the linked list data structure to implement the undirected graph.

Member Variables:
1. Node type pointer head
2. Node type pointer tail

Class 3: undirectedGraph

Description:
It is the implementation of undirected graph which has implements all the command functions for the project and also builds a vector of type linked list to store city information.

Member Variables:
1. Vector of type linked list cityList
2. Node type pointer info
3. Integer vertex
4. Integer edge

Member Functions (operations):
1. insert: for the command i to insert the name of the city as a node in the linked list
2. assignEdge: Updates or add the distance of the road if the conditions of the input are valid
3. search: Searches the city name and return the index if it exists
4. degree: returns the number of vertices a node is connected to.
5. graphNodes: returns the total number of nodes i.e. the variable vertex.
6. graphEdges: returns the number of edges i.e. the variable edge.
7. distance: returns the edge weight between two vertices from the adjacency matrix.
8. clear: it clears the whole linked list

2. Constructors/Destructors

Class Node (Constructor and Destructor):

This class has a constructor to create an object for the vector.

All other classes do not need a constructor or a destructor as they have different functions and the memory is deallocated in the functions.

3. Test Cases

Test 1: Add a lot of cities and connect edges between them check for exceptions, search for existing and non-existing cities

Test 2: Check the other functions like graph_nodes and graph_edges , clear, find the shortest distance and the distance between the cities.

Test File Example 1:

```
i Toronto
i Markham
i Oshawa
i Belleville
i Kingston
i Cornwall
i Ottawa
i Montreal
setd
Toronto;Markham;31.3
setd Toronto;Oshawa;59.9
setd
Ottawa;Montreal;199.0
setd
Cornwall;Montreal;114.0
setd
Markham;Ottawa;374.0
graph_nodes
graph_edges
s Belleville
s Cornwall
degree Oshawa
degree Cornwall
degree Toronto
d Oshawa;Ottawa
d Belleville;Kingston
d Toronto;Montreal
shortest_d
Toronto;Montreal
print_path
Toronto;Montreal
clear
graph_nodes
```

Test File Example 2:

```
i City0
i City1
i City2
i City3
i City4
setd City0;City1;10.0
setd City0;City2;5.0
setd City1;City2;3.0
setd City2;City3;9.0
setd City2;City4;2.0
setd City4;City3;6.0
setd City1;City3;1.0
graph_nodes
graph_edges
s City3
s City5
s City4
degree City2
degree City4
degree City0
d City0;City2
d City2;City3
d City2;City4
shortest_d City0;City4
clear
graph_nodes
```

4. Performance

For class undirectedGraph:

The insert function’s time complexity is O(V) as we are inserting a node in the linked list and its linear time.

The assignEdge function runs in linear time O(V) as it has for loops and conditional statements.

The search function also runs in linear time O(V) because it uses a for loop to search for the city.

The degree function consists of a for loop, so it has a time complexity of O(V).

The graphNodes runs in constant time as it returns a single variable vertex.

The graphEdges runs in constant time as it returns a single variable edge

The Distance function just returns the value of that edge from the matrix so it is also implemented in constant time O(1).

The clear fiunction runs a for loop in O(V) times.