-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------User Manual

Graph Gratification

This program creates a graph abstract data type to find different aspects of a directed graph of cities connected to each other. We were to add functions to manipulate the data in certain ways such as taking in a file with data and displaying the graph or adding or deleting an edge to the graph.

1. Executing the Program

Start visual studio with all the files into one project. Go to digraph.t to see all the functions you’re able to use to manipulate the data. Hit start without debugging and program executes. Follow prompts to use programs.

2. Input

2.1 Input Requirements

When entering the file name make sure the path can be read into. Must choose a number option from the menu to begin using one of the function options. When entering the city names, they must match up exactly to how they are written in the file. When entering two cities please leave a space between the cities. Enter the exit option(15) to end program.

2.2 Input Restrictions

I added virtually zero error checking on this program so it can easily break with any wacky input. Just stick to using numbers corresponding to what you want from menu and use the exact same characters from the data being input.

3. Output

There is a few outputs that can come up, first being the output of the graph displayed is an early one as option 2 DISPLAY current graph. Option 3 IS VERTEX in graph tells you whether an input data is a vertex in the graph, it will tell you yes, it is or no, it is not. Option 4 IS UNI\_DIRECTIONAL EDGE in the graph checks if an input edge with two data points exists. Option 5 IS BI-DIRECTIONAL EDGE in graph checks if an input edge with two data points exists. Option 6 ADD VERTEX to graph will say added to graph. Option 7 DLETE VERTEX in graph just deletes a node in the graph. Option 8 ADD UNI-DIRECTIONAL EDGE to the graph just adds the edge and weight of two data points. Option 9 ADD BI-DIRECTIONAL EDGE to the graph just adds the edge and weight of two data points. Options 10 DELETE UNI-DIRECTIONAL EDGE in the graph gets rid of an edge in the graph. Option 11 DELETE BI-DIRECTIONAL EDGE in the graph gets rid of an edge in the graph. Option 12 BREADTH FIRST TRAVERSAL of the graph shows the breadth first search on the given data. Option 13 DEPTH FIRST TRAVERSAL of the graph show the depth first search from the given data. Option 14 VIEW menu again displays the entire menu. Option 15 EXIT the program does just that terminates the program.

Programmer Manual

Graph Gratification

1. Problem Description

The program takes in a file with data for a graph to be read in. Once it receives a readable file it will give you 15 Options from the meu on what to do with the graph, either print it out, or add a vertex, or delete an edge, or any of the other options.

2. Data Types and Classes

There is a class of digraph and two structs, vertex and edge representation.

2.1 Structs

The vertex struct has a name and an edgelist

The edge representation has name and weight

2.2 Digraph

This class has:

Data members: num\_of\_verticies, vector<vertex graph array>

Member functions: Digraph, ~Digraph, set\_size, getDigraph, get\_size, isVertex, isUniEdge, isBiDirEdge, addVertex, deleteVertex, addUniEdge, addBiDirEdge, deleteUniEdge, deleteBiDirEdge, printDigraph(),breadth, depth, getOneVertex, getTwoVerticies

Variable:

Public Member Functions:

Digraph(); // consturctor

~Digraph(); // destructor

void set\_size(ifstream& inFile); //

void getDigraph(ifstream& inFile);

int get\_size();

int isVertex(V& v);

int isUniEdge(V& v1, V& v2);

int isBiDirEdge(V& v1, V& v2);

int addVertex(V& v);

bool deleteVertex(V& v);

int addUniEdge(V& v1, V& v2, W& w);

int addBiDirEdge(V& v1, V& v2, W& w);

bool deleteUniEdge(V& v1, V& v2, W& w);

bool deleteBiDirEdge(V& v1, V& v2, W& w);

void printDigraph();

void breadth(V& start\_vertex);

void depth(V& start\_vertex);

void getOneVertex(V& v1);

void getTwoVerticies(V& v1, V& v2);