CS 2302

Fall 2019

Lab Report #6

John Rodriguez

Due: November 15, 2019

Professor: Olac Fuentes

TA: Anindita Nath

Introduction

For lab 6 we were given the tasks of implementing and insert edge function, a delete edge function, and a display function for an adjacency matrix and an edge list. Then we had to create three functions that convert the graph type to an adjacency list, adjacency matrix, and an edge list for all three of the graph files we created for the lab. We then could use the as\_AL function on the adjacency matrix graph and edge list graph so we could use the draw function given to us. The second part of the lab is to implement six diﬀerent solutions to a problem using the three graph representations and the two search algorithms being breadth first search, and depth first search.

Proposed Solution Design and Implementation

I approached this lab by first focusing on implementing an insert edge, delete edge, and display functions for the graph\_AM program and the graph\_EL program. Using the insert edge, delete edge, and display functions given to us in the graph\_AL program as a template I was able to create these functions for the graph\_AM program with minor changes so it would work with an adjacency matrix and the display function for the adjacency matrix was very easy to implement. For the edge list program, I did the same but the implementation for the edge list versions was much simpler to implement. I used the test\_graphs program given to use to make sure I was implementing the functions correctly when creating them for the AM and EL programs.

The next part of the programs I worked on was implementing functions that can convert a graph into one of the three graph type implementations we learned. When converting to the same type of graph in the programs I just returned the graph and did nothing to it. When converting a graph to a different type of graph I would create a graph of the type wanted and make it same size of the graph I wished to change. Then I transferred the information from the starting graph to the different type graph and return it. I did this for the three graph programs that are part of our lab. For the draw function in the graph\_AM and graph\_EL programs I would first convert the graph to be a adjacency list then used the draw function given in the graph\_AL program to draw the graphs in the other two graph programs. I added test to the test\_graph program so I could see if the functions were performing correctly.

For the second part of the lab where we had to solve the fox, chicken, grain, farmer problem I first focused on creating the breath first search and depth first search functions for all three programs. I then created a function called path steps for all three graph programs which showed the steps when the breath first search or depth first search functions are used on a graph. I then edited the test\_graphs program so the tests could be performed on the type of graph the user wants to test.

The final part of the lab I worked on was creating a menu where I imported the graph\_AL, graph\_AM, graph\_EL, and test\_graphs programs in it. I then made it where the user could choose the type of graph they wanted, if they wanted to run the tests on the graph or solve the problem given in part two of the lab. If the user chooses to run the test on there selected graphs the test\_graphs program runs with the users selected type of graph and displays the results and the running time it took to perform the tests. If the user chooses to solve the fox, chicken, grain, farmer problem they will then be prompted to choose if they want to perform a breadth first search or a depth first search. Then bases on their selection the program displays the path taken, the running time for the search, and draws the graph for the problem.

Experimental Results

**Graph Tests using AL:**

**A screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generatedA close up of text on a white background

Description automatically generatedA close up of text on a white background

Description automatically generated**

**Graph Tests using AM:**

**A close up of a logo

Description automatically generatedA screenshot of a cell phone

Description automatically generatedA close up of text on a white background

Description automatically generatedA close up of a logo

Description automatically generatedA drawing of a person

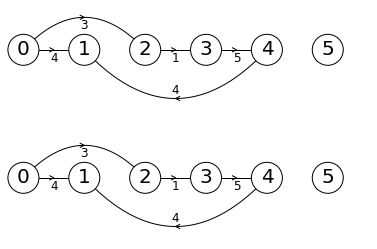
Description automatically generated**

**Graph Tests using EL:**

**A screenshot of a cell phone

Description automatically generatedA close up of text on a white background

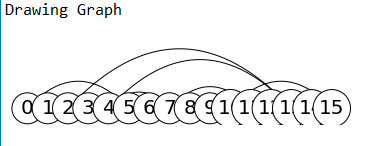
Description automatically generatedA close up of a logo

Description automatically generated**

**Part 2 Problem Adjacency List BFS:**

**A screenshot of a cell phone

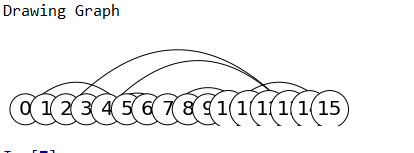
Description automatically generated**

****

**Part 2 Problem Adjacency List DFS:**

**A screenshot of a cell phone

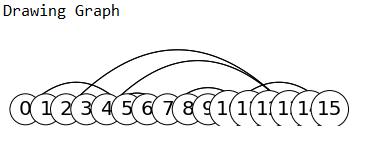
Description automatically generated**

****

**Part 2 Problem Adjacency Matrix BFS:**

**A close up of a map

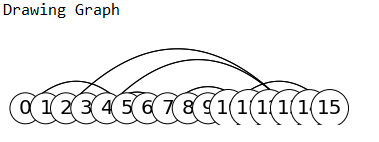
Description automatically generated**

****

**Part 2 Problem Adjacency Matrix DFS:**

**A screenshot of a cell phone

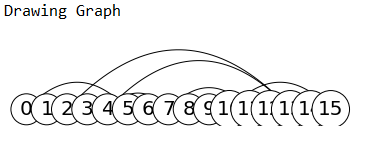
Description automatically generated**

****

**Part 2 Problem Edge List BFS:**

**A screenshot of a social media post

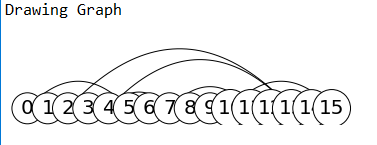
Description automatically generated**

****

**Part 2 Problem Edge List DFS:**

**A screenshot of a cell phone

Description automatically generated**

****

**Graph Tests Running Times:**

|  |  |
| --- | --- |
| Graph Type | Running Time |
| Adjacency List | 0.2963836193084717 seconds |
| Adjacency Matrix | 0.35929012298583984 seconds |
| Edge List | 0.2806055545806885 seconds |

**Breadth First Search Running Times:**

|  |  |
| --- | --- |
| Graph Type | Running Time |
| Adjacency List | 0.0 seconds |
| Adjacency Matrix | 0.0 seconds |
| Edge List | 0.0 seconds |

**Depth First Search Running Times:**

|  |  |
| --- | --- |
| Graph Type | Running Time |
| Adjacency List | 0.0 seconds |
| Adjacency Matrix | 0.0 seconds |
| Edge List | 0.0 seconds |

Conclusion

This lab helped me become more comfortable when using adjacency list type graphs, adjacency matrix type graphs, and edge list type graphs. Also, I now have a better understanding about how a depth first search works when going threw a graph. I also have a better understanding of how a breadth first search works and how it moves threw a graph. The second part of the lab showed me how graphs could be used to solve problems like the fox, chicken, grain, and farmer problem in part two of the lab. The running times for all three types of graphs during the tests were very close and the running times when performing a depth first search and breadth first search on all types of graph types was zero for all.

Appendix

**graph\_AL.py**

A screenshot of a cell phone

Description automatically generated

A screenshot of a cell phone

Description automatically generated

A screenshot of a social media post

Description automatically generated

A screenshot of a cell phone

Description automatically generated

**graph\_AM.py**

**A screenshot of a social media post

Description automatically generated**

**A screenshot of a cell phone

Description automatically generated**

**A screenshot of a social media post

Description automatically generated**

**A screenshot of a cell phone

Description automatically generated**

**graph\_EL.py**

**A screenshot of a social media post

Description automatically generated**

**A screenshot of a social media post

Description automatically generated**

**A screenshot of a cell phone

Description automatically generated**

**A screenshot of a cell phone

Description automatically generated**

**test\_graphs.py**

**A screenshot of a social media post

Description automatically generated**

**A screenshot of a cell phone

Description automatically generated**

**A screenshot of a cell phone

Description automatically generated**

**Lab 6.py**

**A screenshot of a social media post

Description automatically generated**

**A screenshot of a cell phone

Description automatically generated**

I certify that this project is entirely my own work. I wrote, debugged, and tested the code being presented, performed the experiments, and wrote the report. I also certify that I did not share my code or report or provided inappropriate assistance to any student in the class