Lab Assignment 6

ISMAEL VILLALOBOS

PROFESSOR FUENTES

TA: DITA NATH

11-19-19

Introduction

For this lab we were tasked with creating program test_graphs.py demonstrates functions to build, modify, and display graphs using an adjacency list representation, implemented in the class graph_AL.py. Our task consisted of implementing the same functions using an adjacency matrix representation and an edge list representation.

Proposed Solutions

Part1- For this part we completed the implementations of insert_edge , delete_edge , and display for graph_EL.py and graph_AM.py. We already had a leg up on this with the partner led assignment from class so this allowed for us to have something to work off on for this lab. Using the provided test_graphs.py from the class webpage.

In addition to this we had to create as _AL(), as _EL(), and as_AM() respectively. These functions allowed us to go from the graph we constructed say we made a graph represented as an adjacency list and return the same graph represented as either an Edge List or and Adjacency Matrix.

Next, for each graph implementation we created the search functions Depth First Search and Breath First Search, given the pseudocode that was covered in class it was relatively easy to follow and just required minor changes for implementation between Edge Lists, Adjacency Lists and Adjacency Matrixes

Additionally, For this reports purposes and to ensure that my graphs were correct. In my code there is a method that draws the graphs. In essence they should all be the same.

Part 2

We were given the following riddle:

You have a fox, a chicken and a sack of grain. You must cross a river with only one of them at a time. If you leave the fox with the chicken, he will eat it; if you leave the chicken with the grain, he will eat it. How can you get all three across safely?

For this riddle we were given rules and scenarios in which if it was valid, they should be connected. The valid edges were then shown as graphs

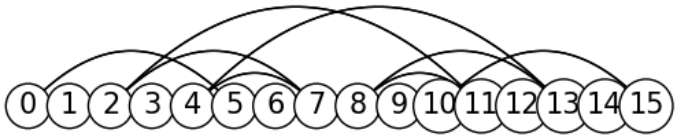
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
1001	9
1010	10
1011	11
1100	12
1101	13
1110	14
1111	15

Experimental Results

DFS and BFS

The result of the code is the best path given both algorithms from (0,0,0,0) in which the fox,chicken, grain and myself are all on the starting side of the river to (1,1,1,1) which everyone is on the opposite side. In this case vertex 0 to vertex 15

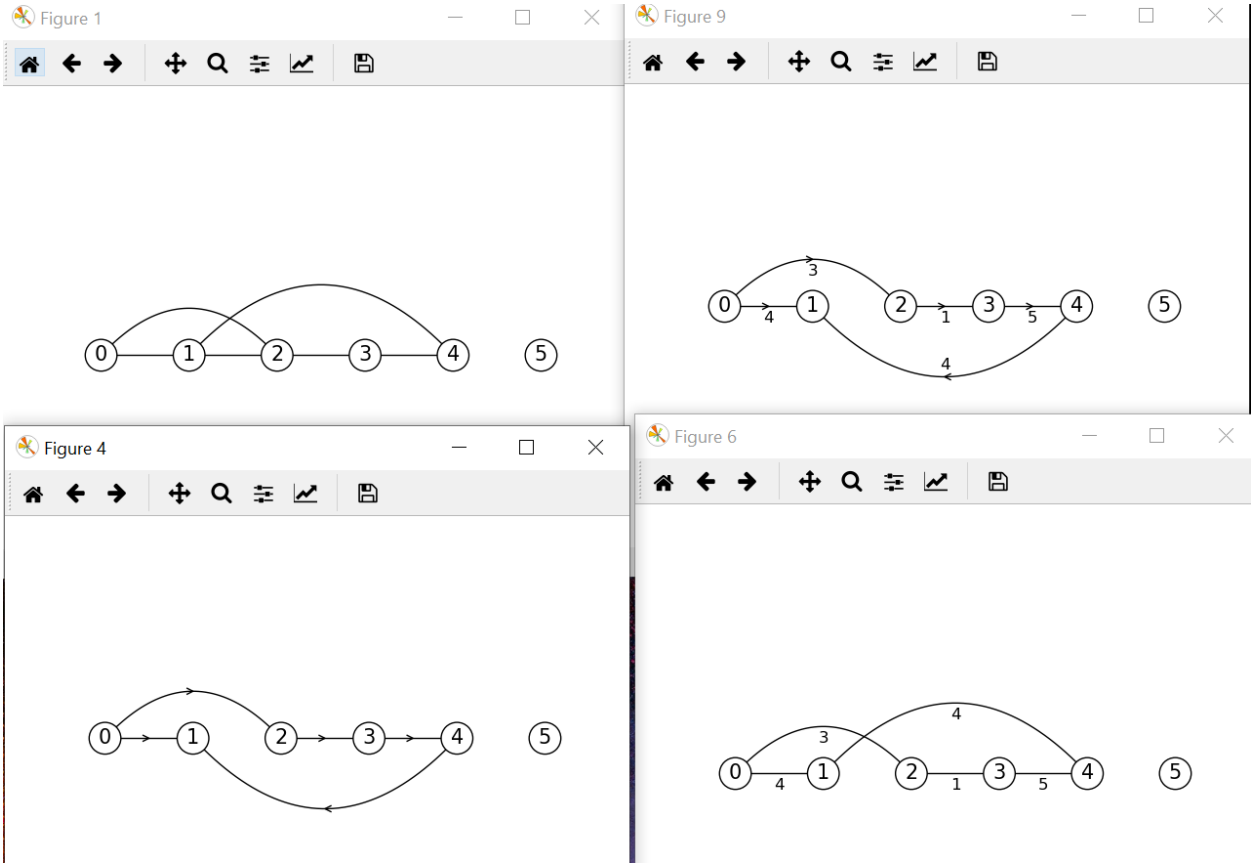
```
In [63]: runfile('C:/Users/Ismael/Desktop/CS2302Fall19/Lab6V2.py', wdir='C:/Users/Ismael/Desktop/
CS2302Fall19')
Reloaded modules: graph_AL, graph_AM, graph_EL
Adjancency List DFS:
[-1, -1, 11, -1, 5, 0, -1, 4, 13, -1, 11, 8, -1, 4, -1, 10]
0 5 4 13 8 11 10 15
Adjacency List BFS:
[-1, -1, 7, -1, 5, 0, -1, 4, 13, -1, 11, 2, -1, 4, -1, 10]
0 5 4 7 2 11 10 15

Adjancency Matrix DFS:
[-1, -1, 11, -1, 5, 0, -1, 4, 13, -1, 11, 8, -1, 4, -1, 10]
0 5 4 13 8 11 10 15
Adjacency Matrix BFS:
[-1, -1, 7, -1, 5, 0, -1, 4, 13, -1, 11, 2, -1, 4, -1, 10]
0 5 4 7 2 11 10 15

Edge List DFS:
[-1, -1, 11, -1, 5, 0, -1, 4, 13, -1, 11, 8, -1, 4, -1, 10]
0 5 4 13 8 11 10 15
Edge List BFS:
[-1, -1, 7, -1, 5, 0, -1, 4, 13, -1, 11, 2, -1, 4, -1, 10]
0 5 4 7 2 11 10 15
```



This graph was produced for all graph representations including EL, AL and AM

Test Graph Representation

In order to assure that all functions and implementations were correct graphs. If they are correct, all graphs should be the same.



Big O Notation

The Time complexity of BFS is O(V + E), where V stands for vertices and E stands for edges.

The Time complexity of DFS is also O(V + E), where V stands for vertices and E stands for edges.

Conclusion

Graph implementations can be shown in a variety of ways including Adjacency List, Adjacency Matrix, or Edge List. Unfortunately for our implantation I received 0.0 for runtimes so I was unable to calculate that. I also learned that DFS uses stack and BFS uses a queue.

Academic Honesty

"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."

Name: Ismael Villalobos

Appendix:

'''

Ismael Villalobos

11-17-19

Lab Assignment 6

Professor Fuentes

TA-Dita Nath

Purpose:

'''

```
import graph_AL as graphAL
import graph_AM as graphAM
import graph_EL as graphEL


# build three graph representation:
# - adjacency list
# - adjacency matrix
# - edge list
def buildGraphRepresentations():
    # vars
```

```
    v = 16
    weighted = False
    directed = False

    # build adjacency list
    AL = graphAL.Graph(v, weighted, directed)
    AL.insert_edge(0, 5)
    AL.insert_edge(2, 11)
    AL.insert_edge(2, 7)
    AL.insert_edge(4, 5)
    AL.insert_edge(4, 7)
    AL.insert_edge(4, 13)
    AL.insert_edge(8, 11)
    AL.insert_edge(8, 13)
    AL.insert_edge(10, 11)
    AL.insert_edge(10, 15)

    # build adjacency matrix, build edge list
    AM = AL.as_AM()
    EL = AL.as_EL()

    # return all three representations
    return AL, AM, EL

def graphs(AL, AM, EL):
    AL.draw()
    AM.draw()
    EL.draw()

if __name__ == "__main__":
```

```
AL, AM, EL = buildGraphRepresentations()

# adjacency list
# depth first
print("Adjancency List DFS:")
print(AL.DFS(0, 15))
AL.printDFS(0, 15)
# breadth first
print("Adjacency List BFS:")
print(AL.BFS(0, 15))
AL.printBFS(0, 15)

print()

# adjancency matrix
# depth first
print("Adjancency Matrix DFS:")
print(AM.DFS(0, 15))
AM.printDFS(0, 15)
# breadth first
print("Adjacency Matrix BFS:")
print(AM.BFS(0, 15))
AM.printBFS(0, 15)

print()

# edge list
# depth first
print("Edge List DFS:")
```

```
print(EL.DFS(0, 15))
EL.printDFS(0, 15)
# breadth first
print("Edge List BFS:")
print(EL.BFS(0, 15))
EL.printBFS(0, 15)

print()
```