**CS 2302 Data Structures**

**Fall 2019**

**Lab Report #6**

Due: November 19th, 2019

Professor: Olac Fuentes

TA: Anindita Nath

**Introduction**

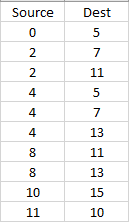
For this lab we were tasked with three methods (insert\_edge, delete\_edge, and display) for an adjacency list, adjacency matrix and edge list then also create three more methods in each file that would allow us to create different graph representations of each (as\_AL, as\_AM, as\_EL) and used them to covert each graph representation to an adjacency list and draw it within the spyder IDE. After all these methods where tasked with using these different graph representations to find the path of a word problem using breadth first search and depth first search.

**Proposed Solution Design and Implementation**

**Part 1:**

First, I created each method for an adjacency list because I felt it was the easiest. Then used similar techniques to create the same method for an adjacency matrix and edge list. After I went back each graph file and added the three methods (as\_AL, as\_AM, as\_EL) an when completed would allow any of the three graph representations to be converted to the other two representations. To test that the conversions worked I used them to draw each graph using the .draw() method.

**Part 2:**

 Next, in all three graph files I created two more methods (DFS “Depth first search” & BFS ”Breadth first search”) to find the solution path for the word problem, “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?” My initial solution was (0,5,4,7,2,11,10,15) as this is what I came up with working out the problem on paper. To find this I first inputted all valid edges into an adjacency list.

They were valid because the binary representations of the numbers did not break any rule of the word problem when they were moving to their destinations. After words to save time I used the as\_AM and as EL methods to convert the adjacency list to an adjacency matrix and an edge list.

**Experimental Results:**

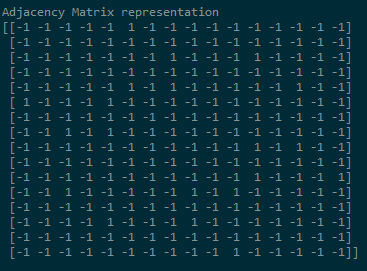
Adjacency list:

The Path List after Breadth First Search:



The Path List after Depth First Search:



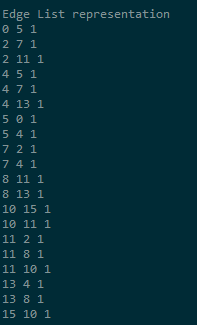
Adjacency Matrix:

The Path List after Breadth First Search:



The Path List after Depth First Search:



Edge List:

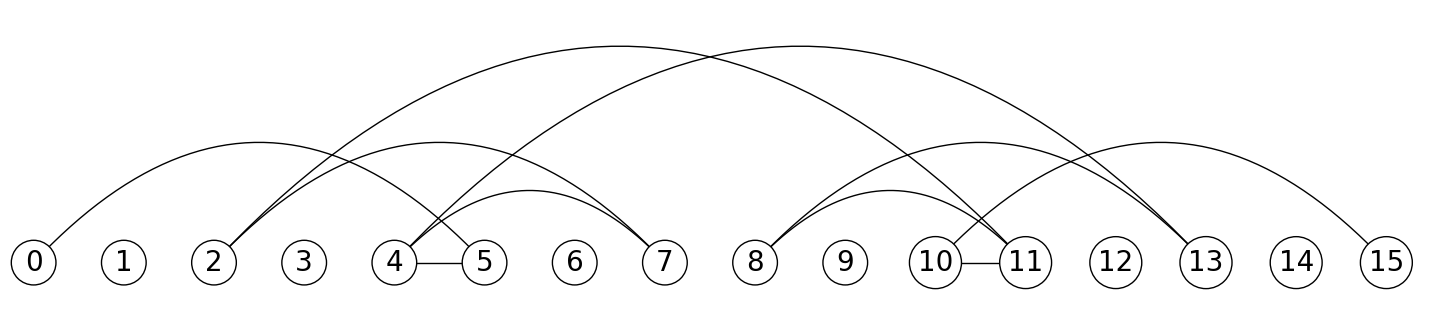
The Path List after Breadth First Search:

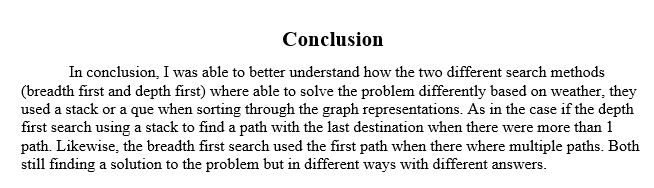


The Path List after Depth First Search:



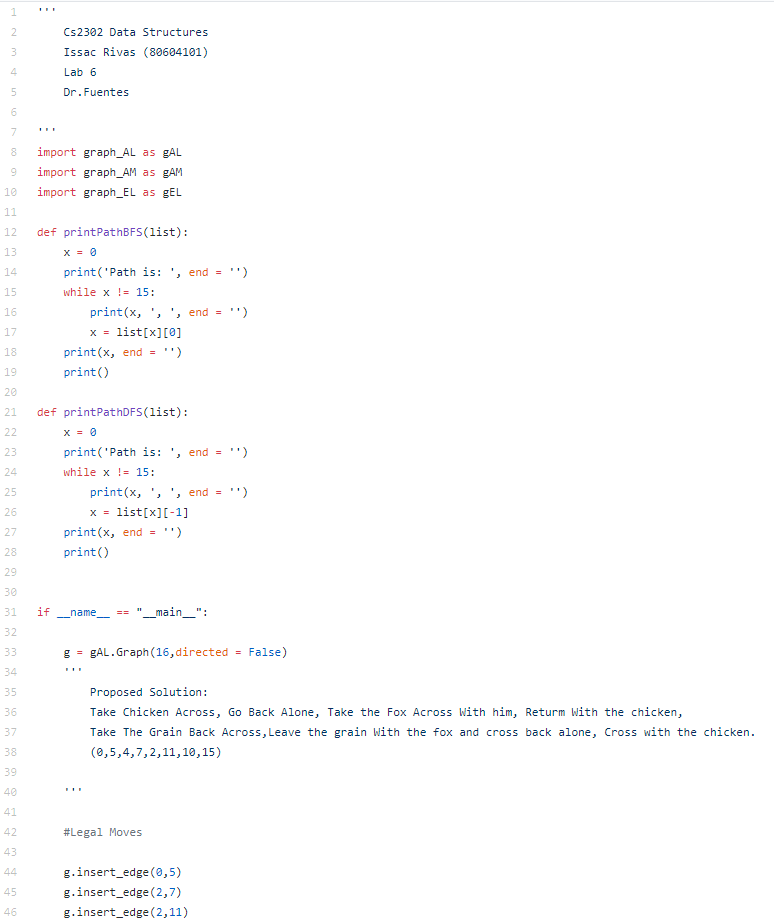
Drawing:

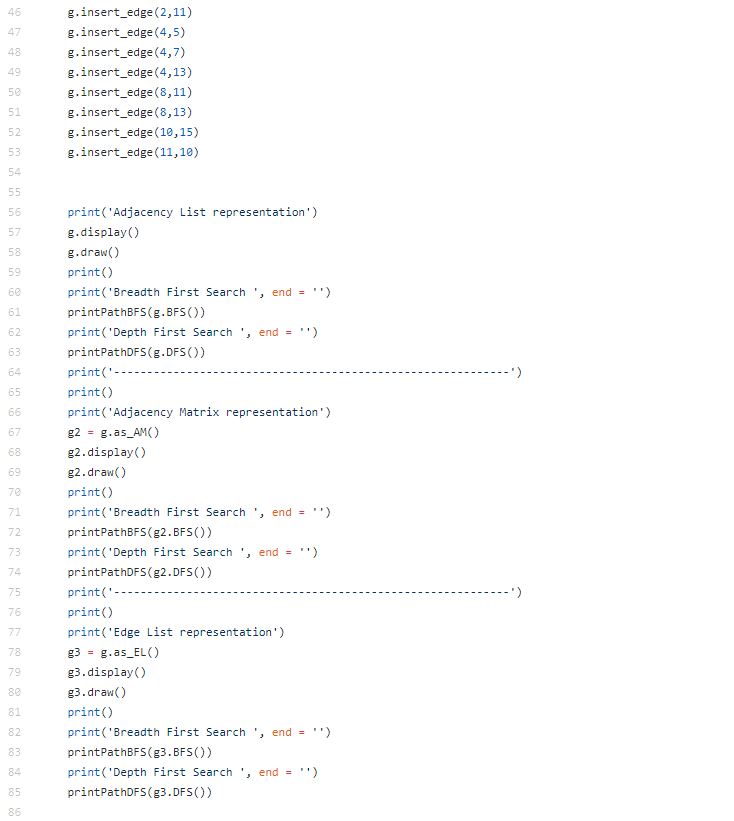




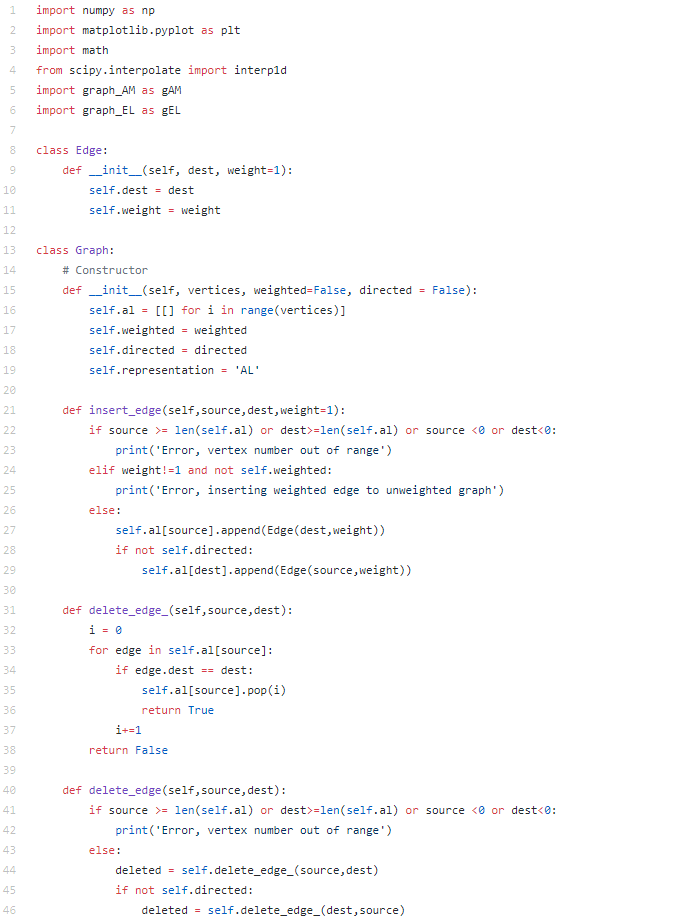
**Appendix**

**Main.py:**



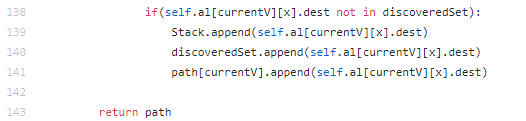


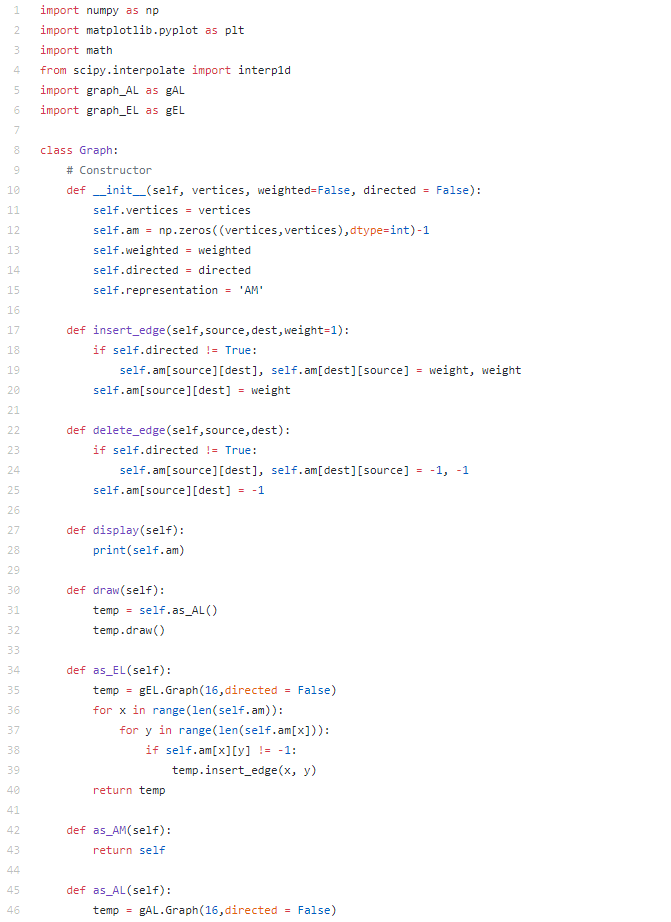
**graph\_AL.py:**

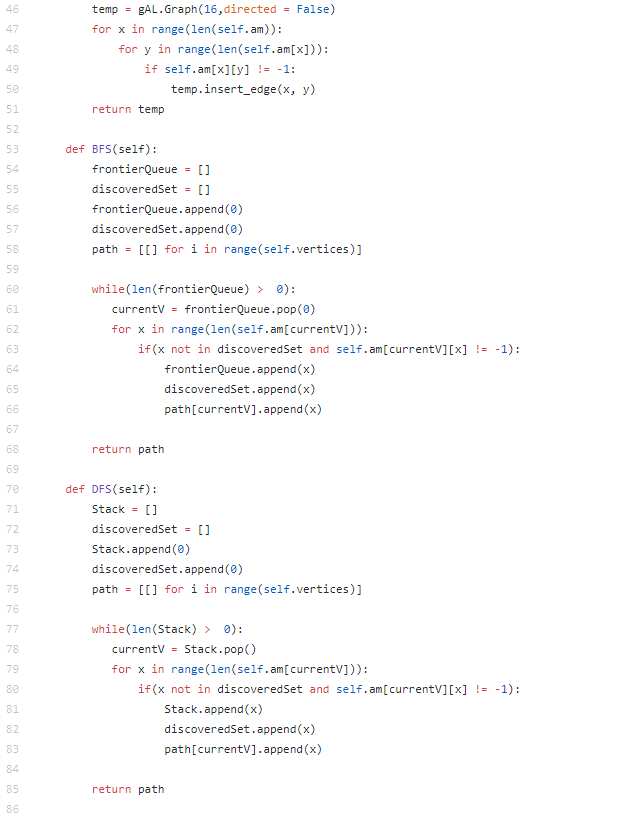




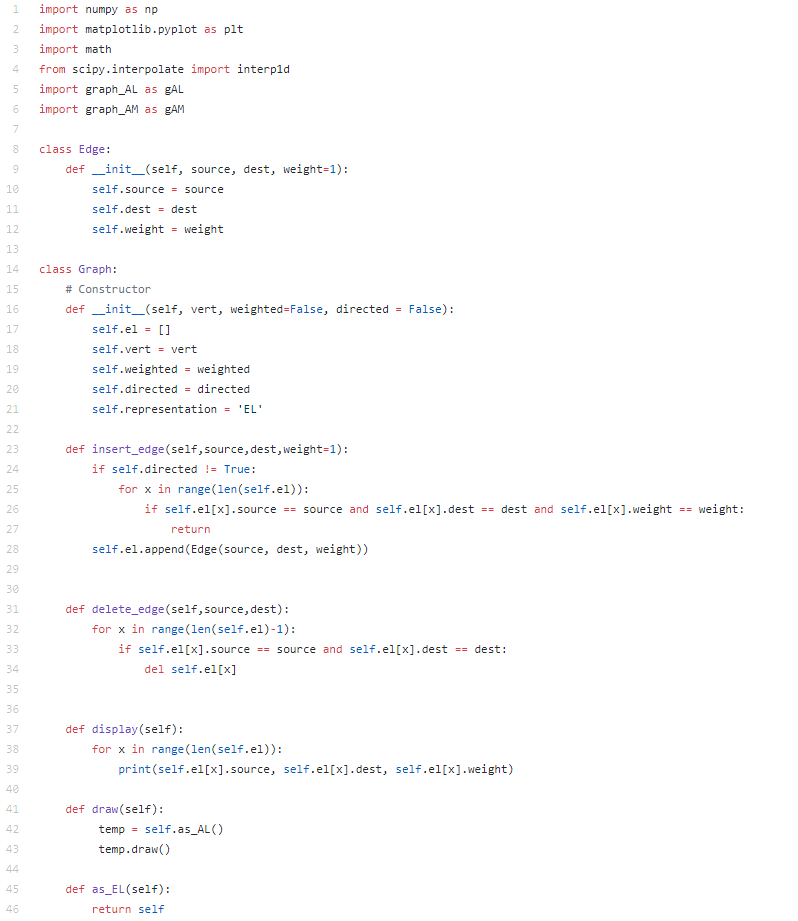


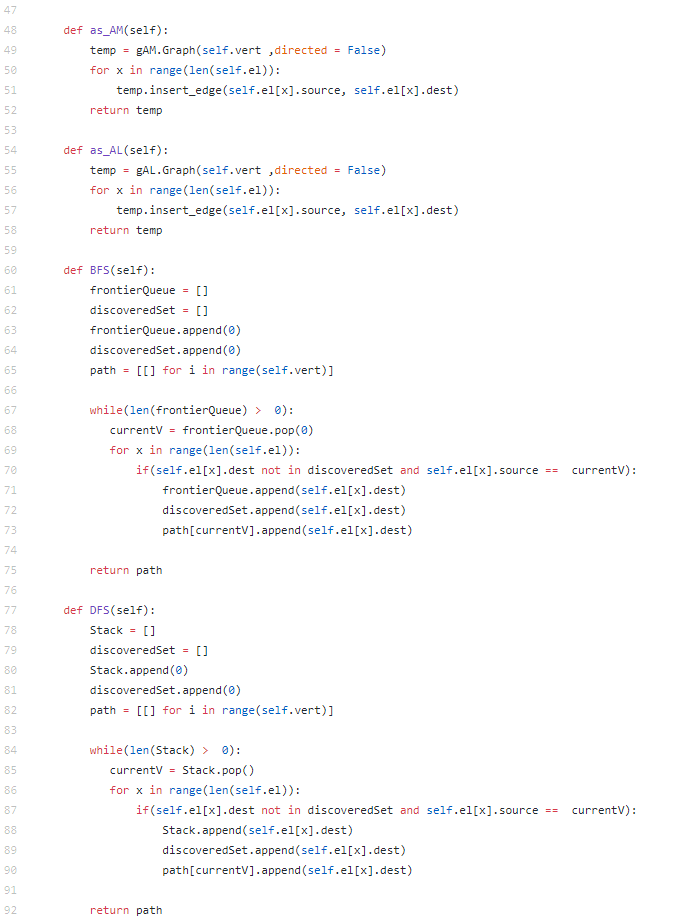


**Graph\_AM.py:**



**Graph\_EL.py:**





I Issac Rivas, certify that this project is entirely my own work. I wrote, debugged, and tested the code being presented, preformed 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.