**CS 2302 Data Structures**

**Fall 2019**

**Lab Report #6**

Due: 11/15/2019

Professor: Olac Fuentes

TA: Anindita Nath

**Introduction**

For this lab we were asked to implement an insert\_edge, delete\_edge, and a display function for Adjacency Matrix and Edge List. We were also asked to add the function as\_AL that returns an adjacency list representation of the graph, a function as\_AM that returns an adjacency matrix representation of the graph, and a function as\_EL that returns an edge list representation of the graph. For part 2, we were asked to implement a solution to the fox, chicken, and a sack of grain problem. For the problem of the fox, chicken, and a sack of grain, we have to get the fox, the chicken, and the sack of grain to get across the river safely without letting the fox and the chicken to be alone. The sack cannot stay alone with the chicken.

**Proposed Solution Design and Implementation**

Part 1:

Operation 1:

For this operation, I just inserted an edge at [source][dest] if the graph was directed. I made an if statement in which it checked if the graph was directed, and if it was, I inserted an edge conncecting [source][dest] and [dest][source]. For deleting an edge I approached it the same way. The only difference that I made was finding the edge first and then deleting the [source][dest] (if it was directed). If the graph was undirected, I deleted both [source][dest] and [dest][source]

Operation 2:

For this operation, I just created a list in which I appended the edges that were given.

Operation 3:

For this operation, I was not able to create it.

Part 2:

For this operation, I checked which steps were valid and the ones that were valid, I saved them into a list. I then inserted the edges into a graph.

**Experimental Results**

Part 1:

Operation 1:

For this operation, I had a very hard time understanding how exactly to iterate through the things. I had to try many different things until something actually worked. I then discovered how they worked and just looked for [source] and [dest].

Operation 2:

For this operation, I also had a hard time figuring out how to iterate through things. I first tried iterating through a non-existing ‘list’ and then realized that what I was trying to do was to create a list in which I appended the source and destiny.

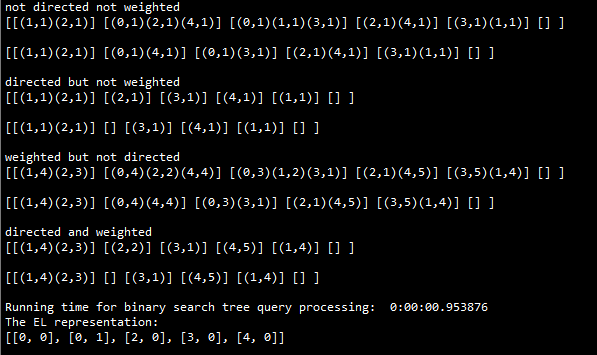
Operation 3:

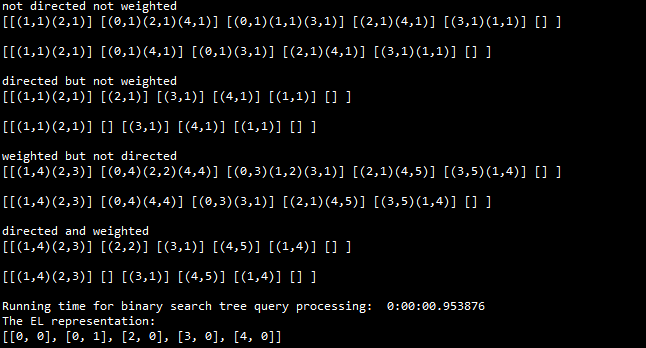
For this operation, I was not able to create it.

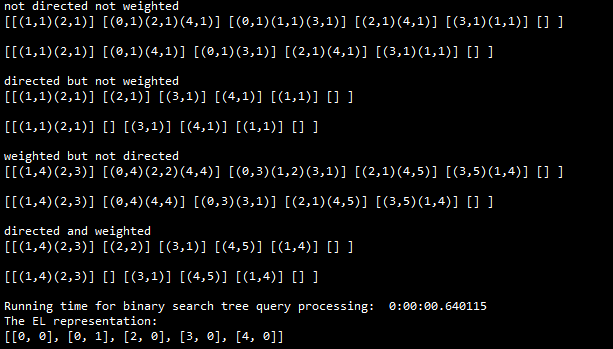
Part 2:

For this operation, it was hard coming up with a graph that represented the path that the fox, chicken, and the sack of grain must take so they do not eat each other. I first approached it with an adjacency matrix and then moved forward from there

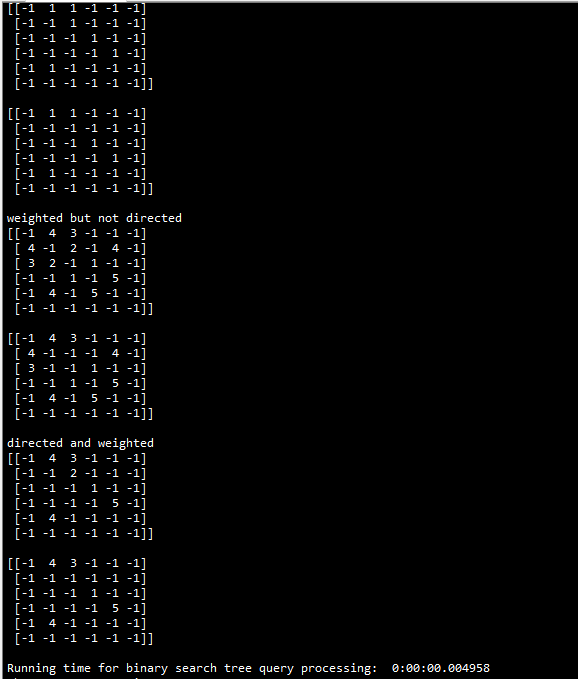
AL: (I ran it three times to have different running time results)

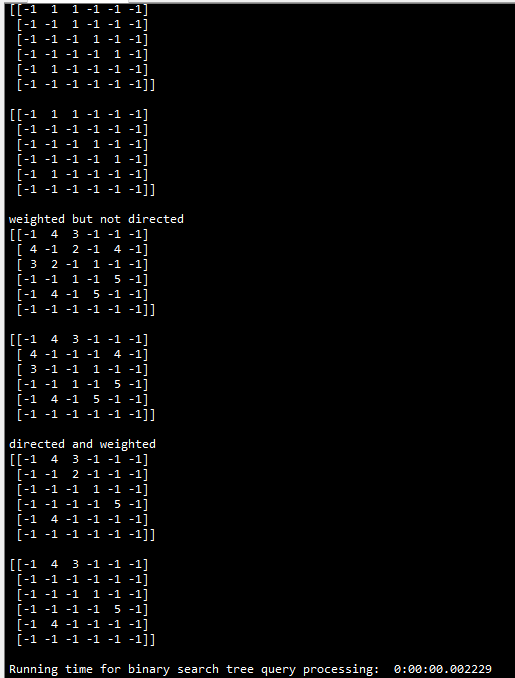


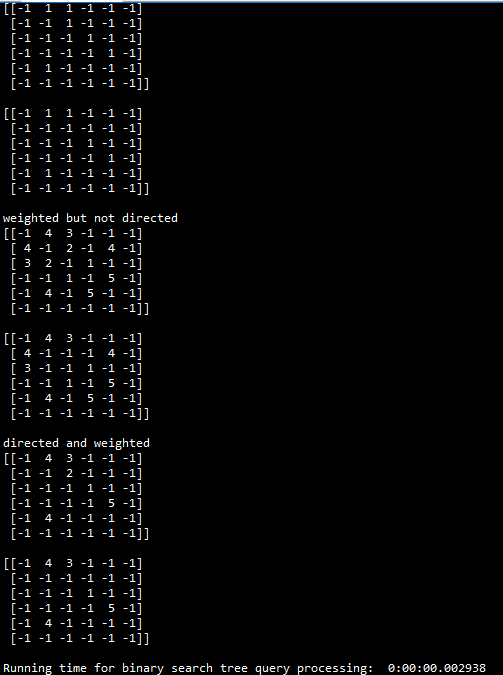




AM: (I ran it three times to have different running time results)

****

****

****

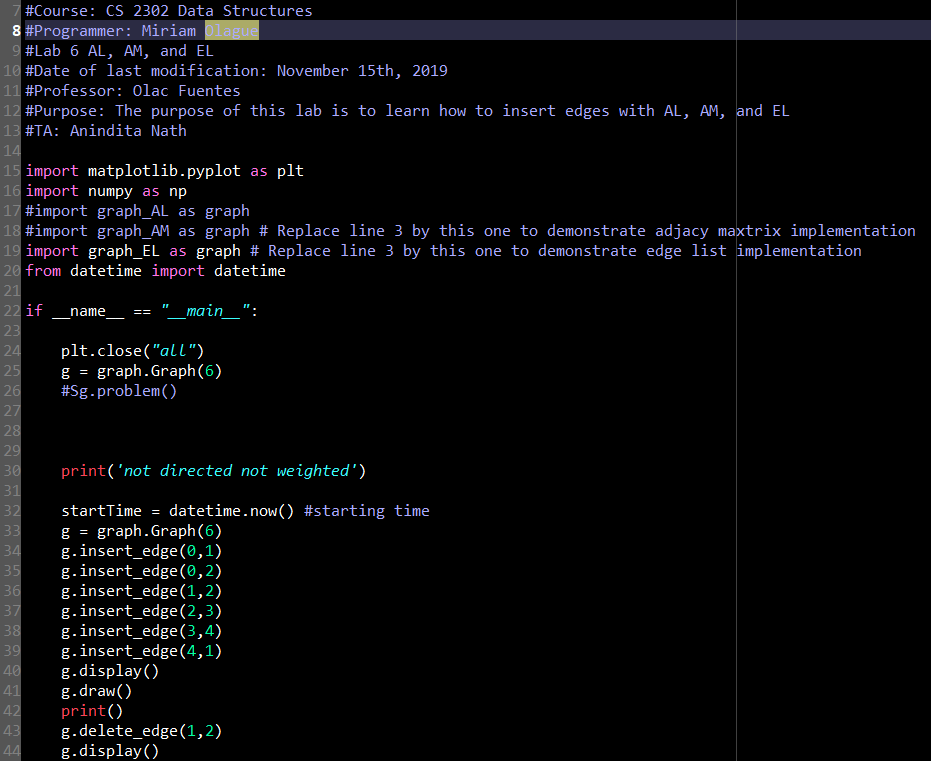
****

As the results show, AM is faster at inserting/deleting the edges when compared to AL. I was unable to finish EL.

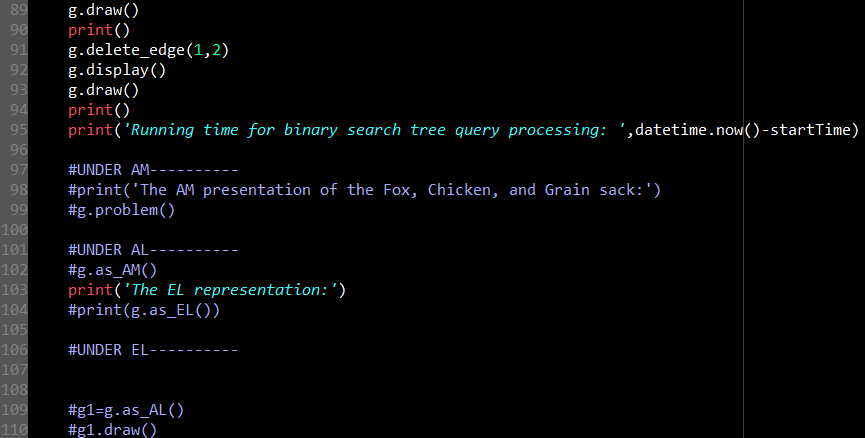
**Conclusion**

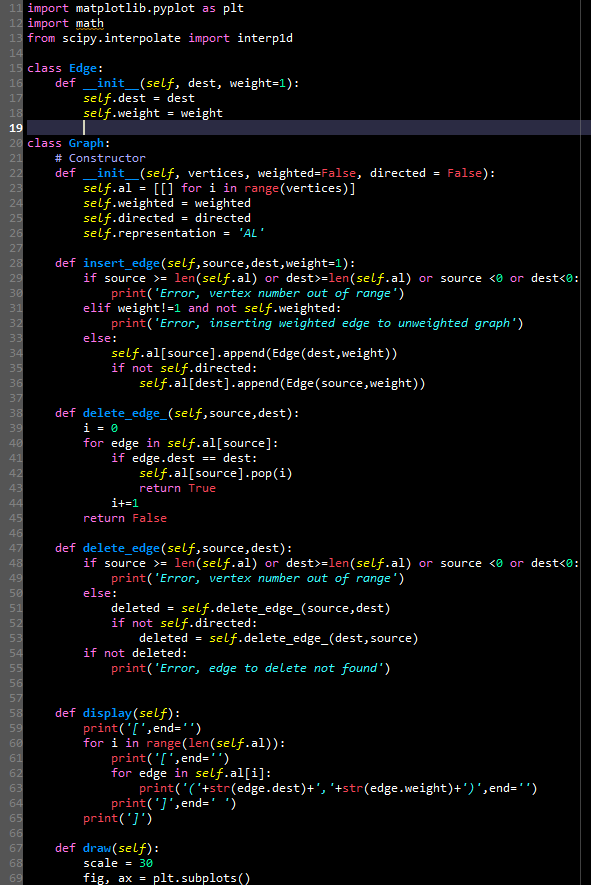
For this lab I learned how to access elements from an AL and an AM. I was able to insert edges and understand the concept of both AL and AM. I also learned that AM is faster than AL when inserting/deleting edges. For EL, I was not able to change the weight. I was only able to insert edges only if they were directed or undirected or unweighted.

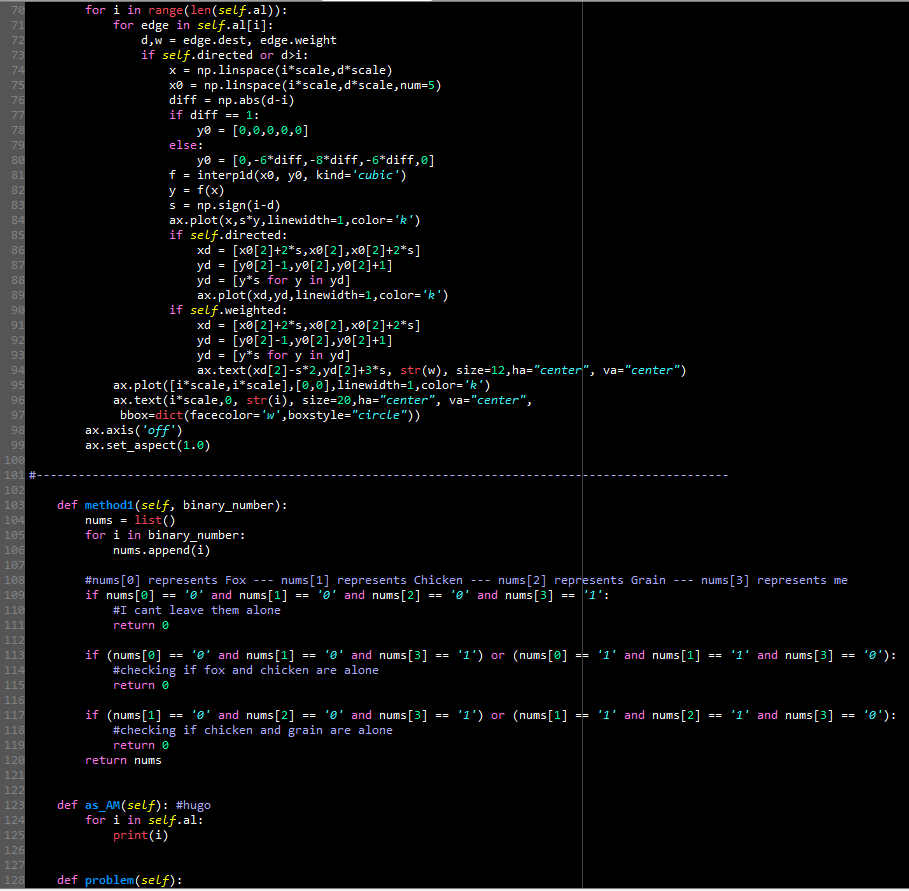
**Appendix**

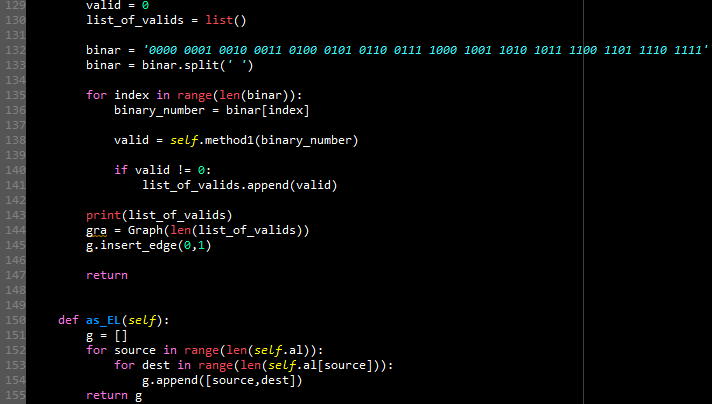
****

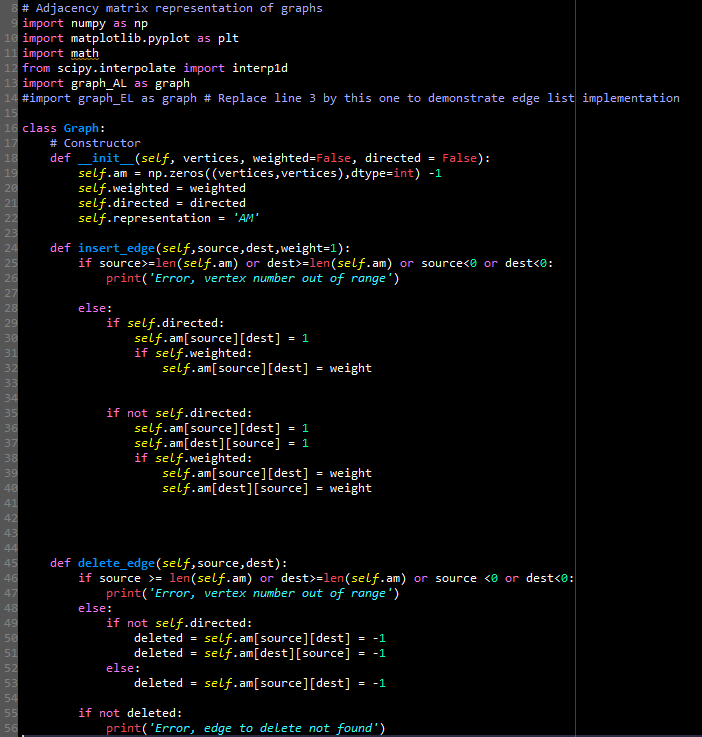
****

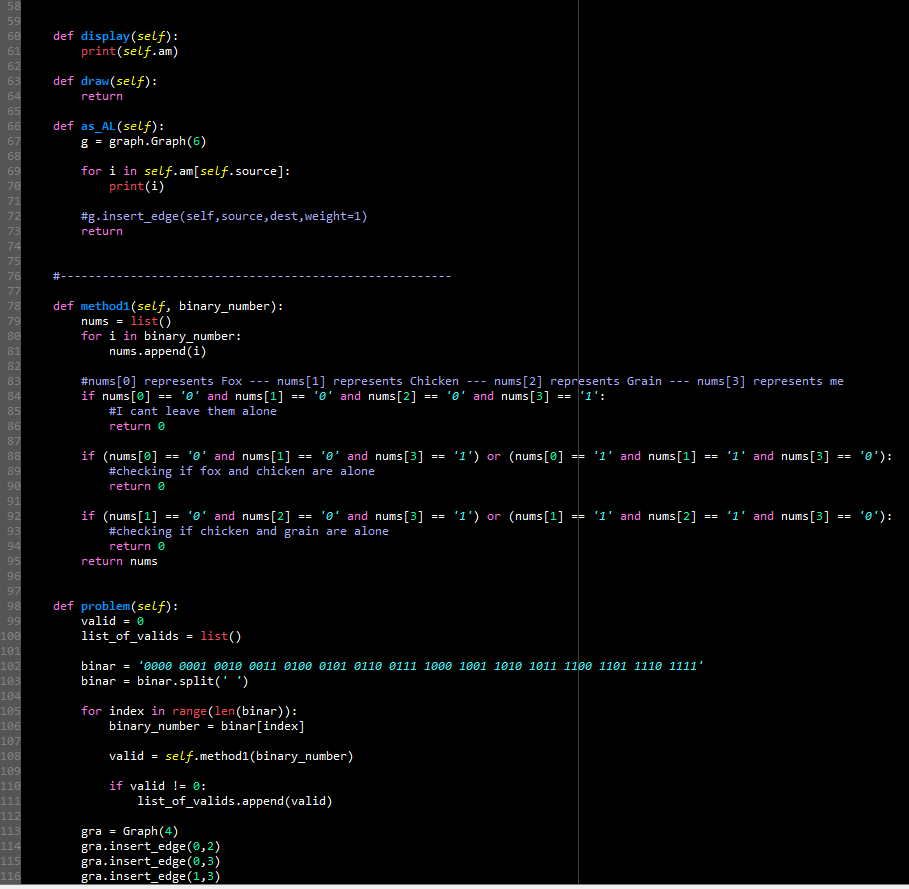
****

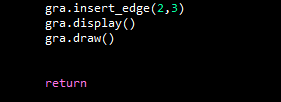
****

****

****

****

****

****

I certify that this project is entirely my own. 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.