**CS 2302 Data Structures**

**Fall 2019**

**Lab Report #7**

Date: December 4th, 2019

Professor: Olac Fuentes

TA: Anindita Nath

Programmer: Miriam Olague

**Introduction**

For this lab, we were asked to implement a randomized algorithm with a maximum of n trials. Once we created a randomized graph, we had to figure out if the graph had a Hamiltonian cycle. We were asked to solve this with randomized Hamiltonian and backtracking. We were also asked to modify the distance function that was provided in class to allow certain replacements.

**Proposed Solution Design and Implementation**

**Part #1:**

For this part, I created a function that creates random graphs with a certain amount of vertices and a certain amount of edges. I then created another method that checked if it had a Hamiltonian cycle. I called the random graph function from the Hamiltonian cycle function and checked if it was valid. If the graph was valid, it showed the path and counted it as True. If the graph was not valid, it showed the path it took and added a -1 to the path to show why it was not valid and counted it as False.

**Part #2:**

For this part, I created a function that checked if the given graph had a Hamiltonian cycle by using backtracking instead. I created an adjacency matrix with the edges I inserted and checked if it was valid or not.

**Part #3:**

For this part, I took the edit distance code of Dr. Fuentes and modified it so it checks if the letters are both vowels, or consonants and created a method that checks if the letters are the same.

**Experimental Results:**

**Part #1:**

For this part, I had no idea how to actually work with randomized things in general. I didn’t even know how to get a random number. I then looked up how to create a random number generator and then implemented it on a graph. I looked up the definition of a Hamiltonian Cycle and worked some examples by hand. I wanted to create a for loop that created a random graph three times and when the graphs were created, I wanted to check if there was a Hamiltonian Cycle. I checked if the graphs were valid in general and if they were, I wanted to show the path and if it was True or False.

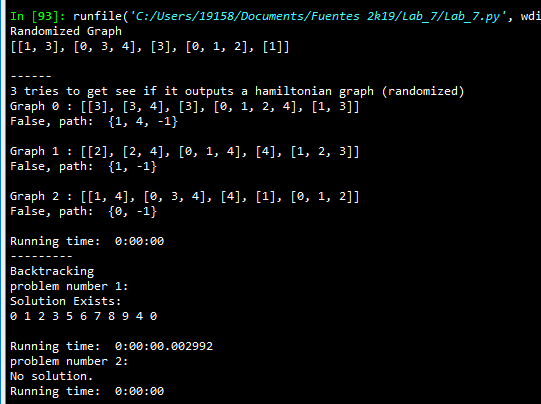
**Part #2:**

For this part, I had a vague idea of what backtracking was. The hard part was implementing it. I did some examples that Dr. Fuentes gave us and I figured out what I wanted to do coding it. I looked up examples of how to create a method for backtracking and I then figured it out.

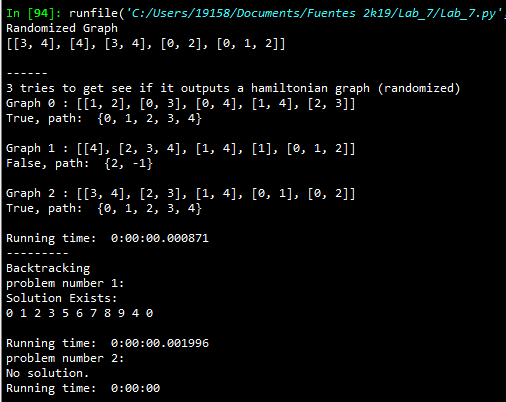
**Part #3:**

For this part, I did not understand very well what I was being asked to do. I had to read it a couple of times to understand what it was asking. I then got Dr. Fuentes’s code from his website and took a look at it. I then traced some examples by hand and looked up examples of how edit distance worked. After I got a good idea of what I needed to do, I started working on the code part. I came up with instances to see if I had the right answer when compared to my code’s output.

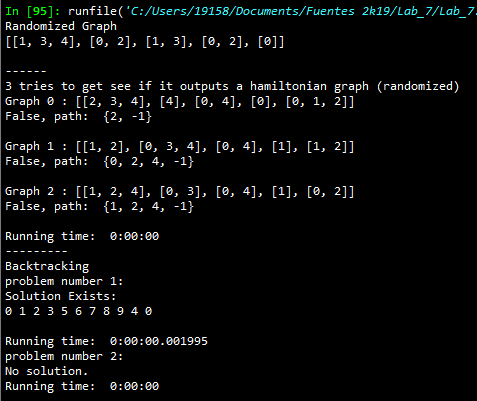
Trial 1: same backtracking problem and creating three random graphs of 4 vertices and edges

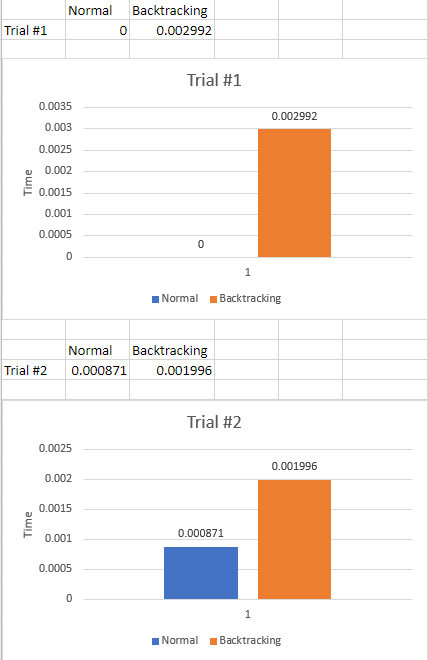


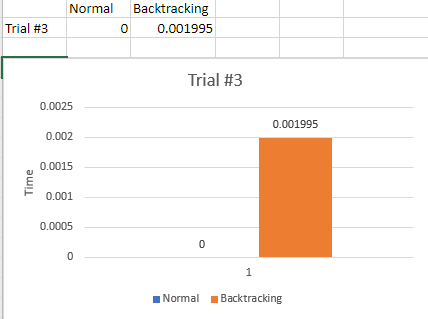
Trial 2: same backtracking problem and creating three random graphs of 4 vertices and edges



Trial 3: same backtracking problem and creating three random graphs of 4 vertices and edges





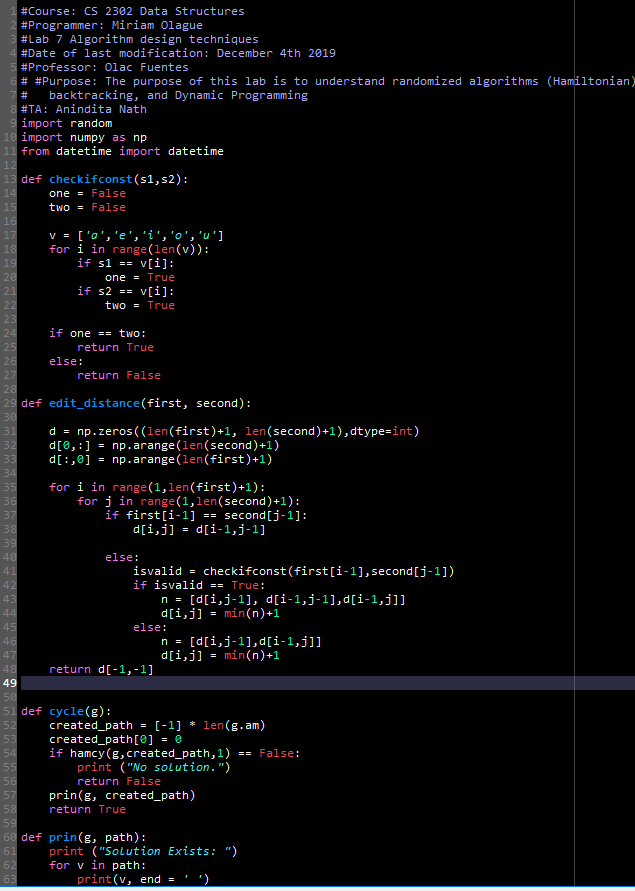


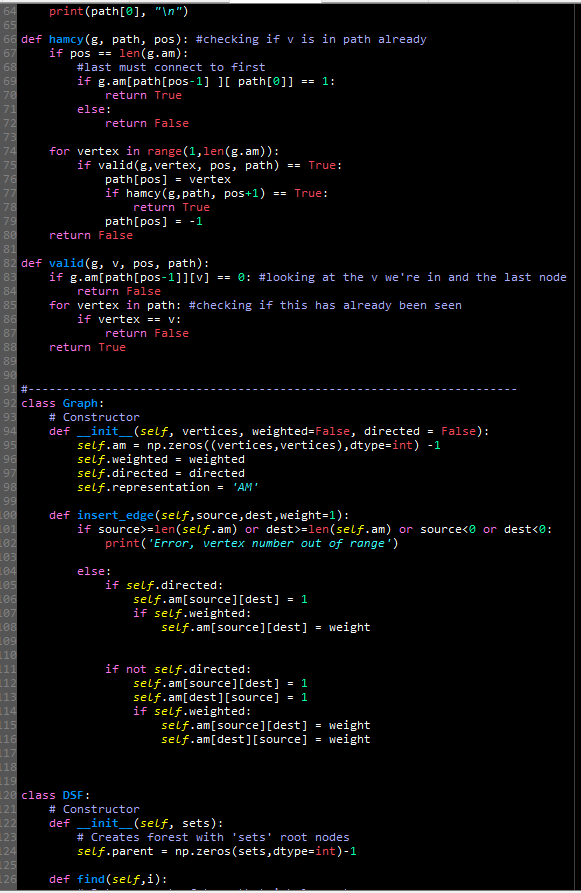
As the results show, creating three randomized graphs takes less time when finding a Hamiltonian cycle when compared to finding a Hamiltonian cycle of a graph using backtracking.

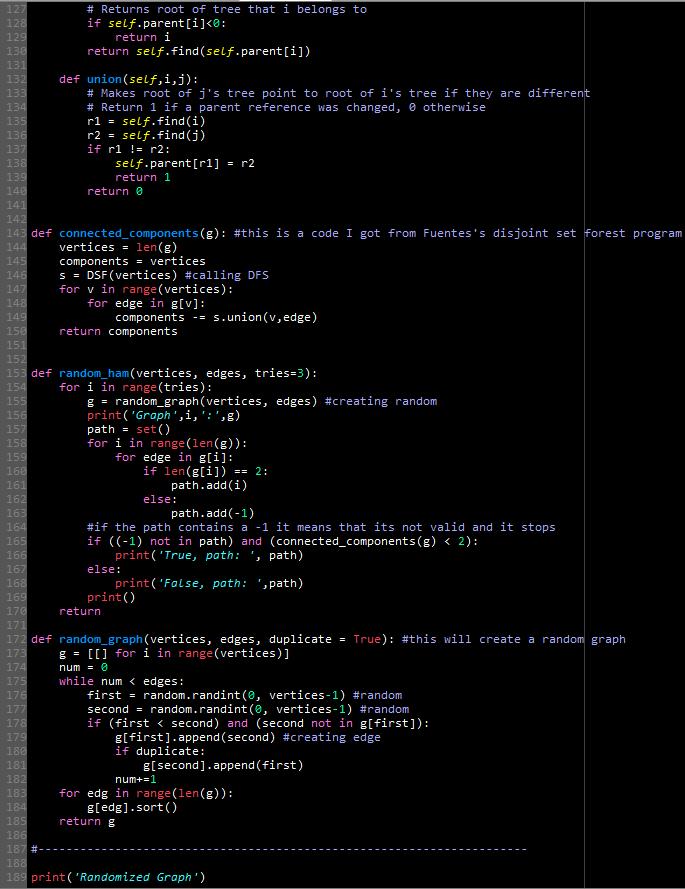
**Conclusion**

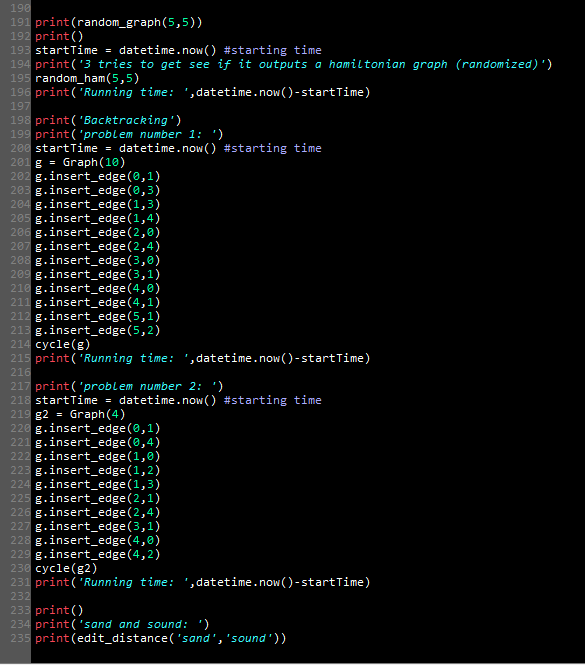
This lab helped me understand the material of what is a Hamiltonian Cycle and how to use a Backtracking algorithm. It also helped me understand how to edit distance. This lab was not as challenging as Lab 7 but it was still difficult to do. I learned how to trace backtracking and how to check if a graph has a Hamiltonian cycle. I also learned to deal with randomized numbers (I did not know how to implement this). This lab was very fun.

**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.