Lab 7 – Algorithm Design

Jonatan M. Contreras

University of Texas at El Paso

**Introduction**

In lab seven, we were tasked with implementing a randomized algorithm to find a Hamiltonian cycle in a graph; implementing a backtracking algorithm for the Hamiltonian cycle problem; and edit the distance function for strings to only allow replacements in the case where the characters are both vowels or consonants.

**Proposed Solution Design and Implementation**

*Part 1: Randomized Hamiltonian Implementation:*

My randomized Hamiltonian uses edge list representations of graphs. It also uses various helper methods. First, it takes the edge list graph and passes it to a helper method named g transform; the method takes the graph and returns a tuple, v and e, where v is the vertices and e is an edge list. The randomized Hamiltonian method passes this to a main randomized Hamiltonian method and returns what this method returns. This main randomized Hamiltonian method takes v, e and max trials, then runs the following process for the range max trials: It first creates a subset of edges of size v by passing v and e to a method named random subset. It then creates a potential graph out of this edge list and checks the connected components and in degree of this potential graph. If the connected component is and the in degree for every vertex is 2, it returns the edge list that was used to create this graph. Else, it continues with the max trials. If it exhausts the max trials without returning a successful edge list, it returns None.

*Part 2: Backtracking Hamiltonian Implementation:*

My backtracking Hamiltonian implementation is based on the backtracking algorithm for the subset sum problem. This method takes v vertices, an edge list e, and an initially empty edge list named Eh which will be the edge list with the Hamiltonian cycle. It has two base cases: if the length of Eh is equal to v, check for Hamiltonian cycles through a helper method named hc\_check; and if length of the edge list is less than 1, return None. Else, the rest of the method basically follows the algorithm of the subset sum problem by making two recursive calls where it either adds the first edge of e into Eh or it doesn’t. Hc\_check follows the algorithm from the randomized Hamiltonian algorithm but only runs once instead of max trials times.

*Part 3: Edit Distance Function Implementation:*

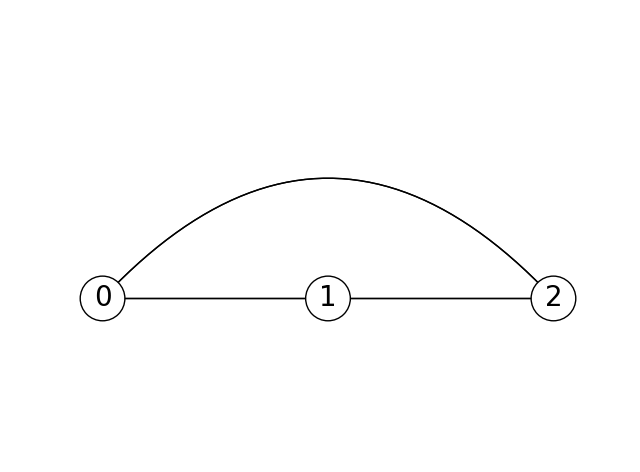
To edit the distance function, I found that, when the algorithm calculates cost by taking the min of the three cells adjacent, that taking the cost from the diagonal cell is a replacement, taking the cost of the cell above is a deletion, and taking the cost from the cell to the left is an insert. Thus, I created a vowels list and when the characters were not equal to each other, it would check if both characters are in the vowels list or both characters were not in the vowels list. If this condition is met, then the method only is allowed to take the min of the left and top cell.

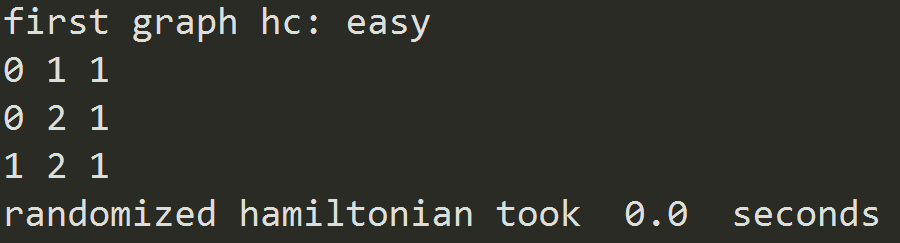
**Experimental Results**

*Part 1: Randomized Hamiltonian Implementation:*

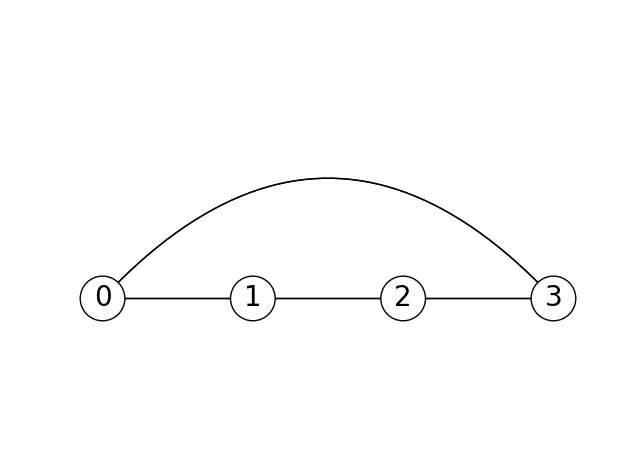
To test the randomized Hamiltonian algorithm, I created four graphs: the first three had Hamiltonian cycles and the last one does not. For the graphs 1-3, they increase in number of vertices. As mentioned earlier, the 4th graph has no Hamiltonian cycle. The max trials used was 1,000 for each run.

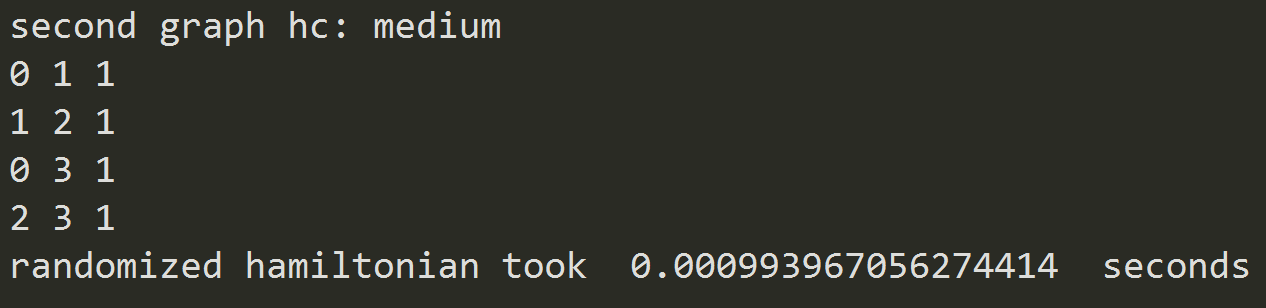
*1st Graph Output and Run Time:*





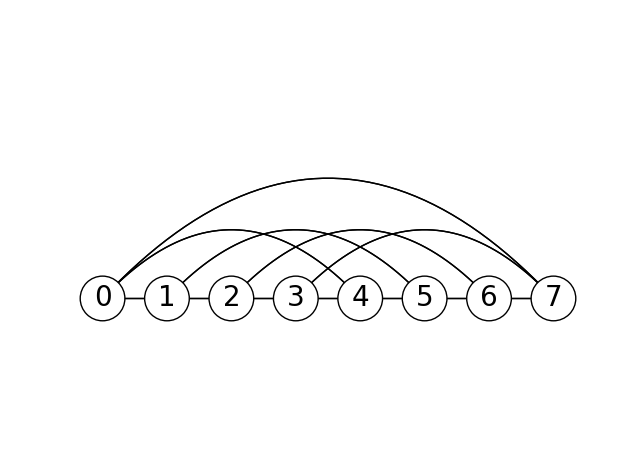
*2nd Graph Output and Run Time:*

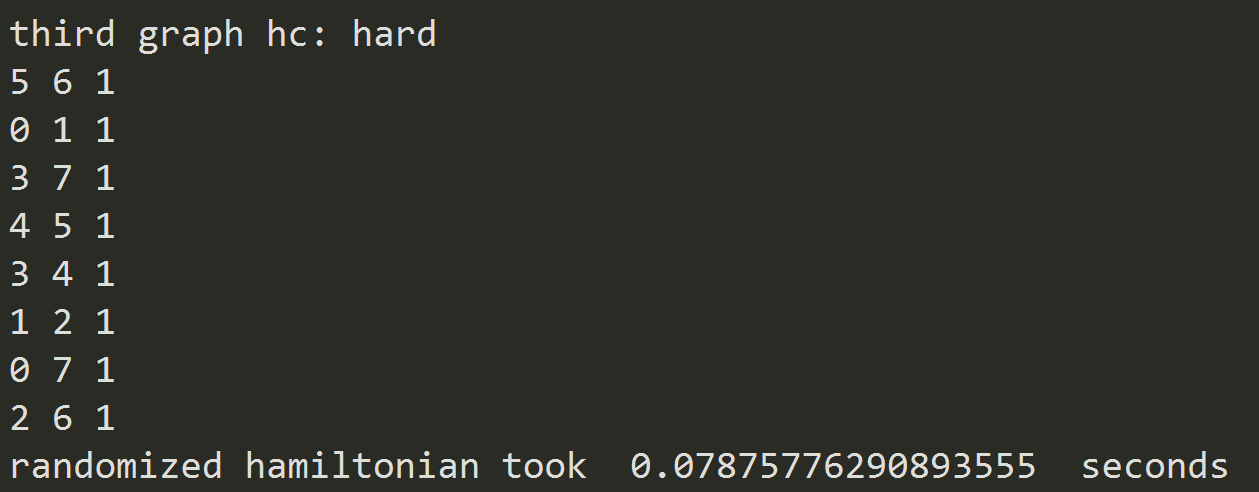
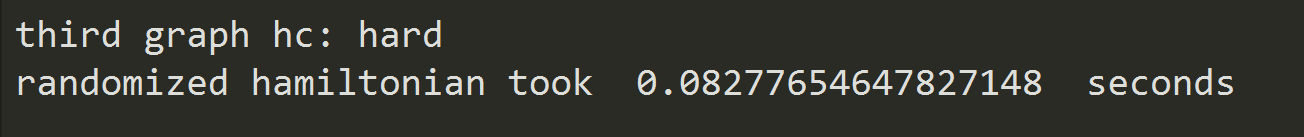
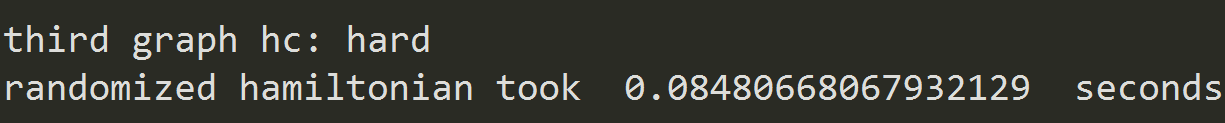
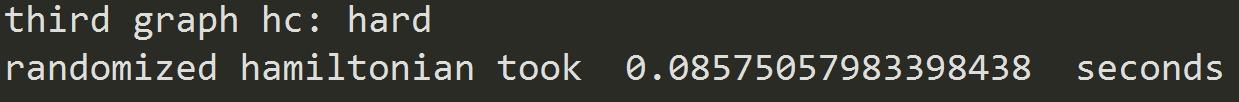




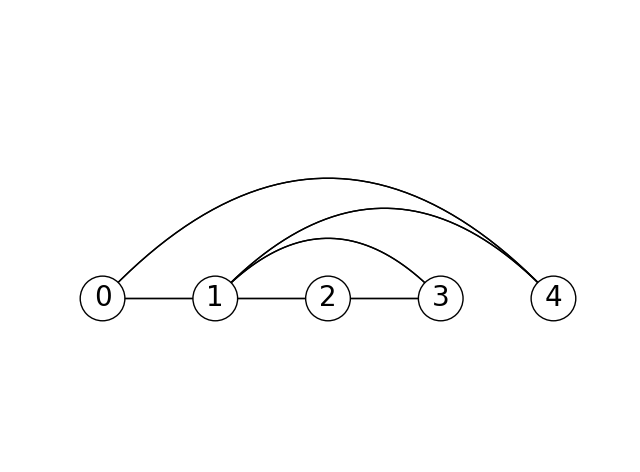
*3rd Graph Output and Run Time:*

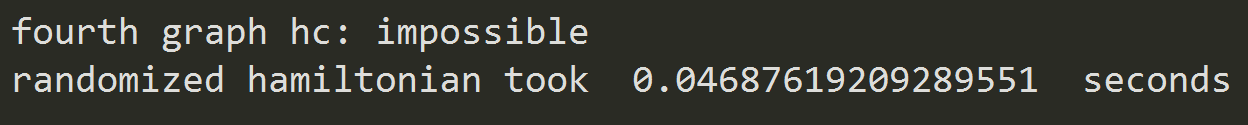
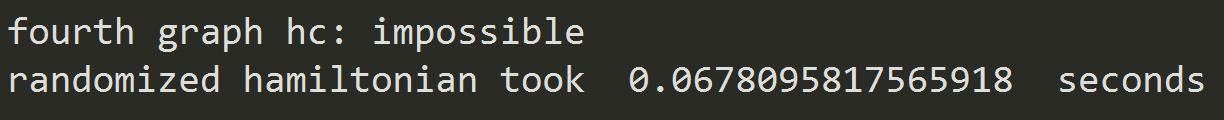
The third graph needed multiple runs in order to find the Hamiltonian cycle.





*4th Graph Output and Run Time:*

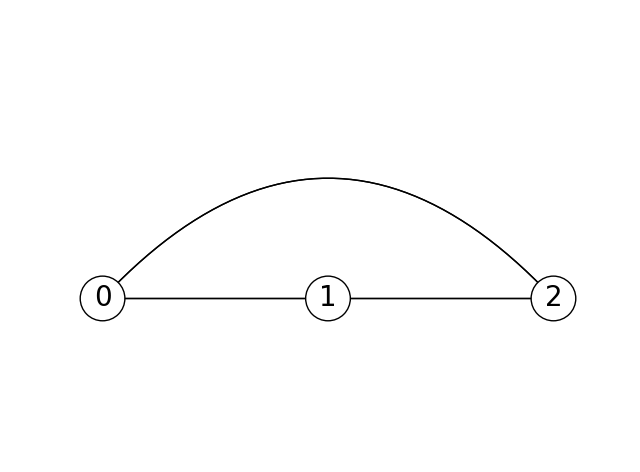


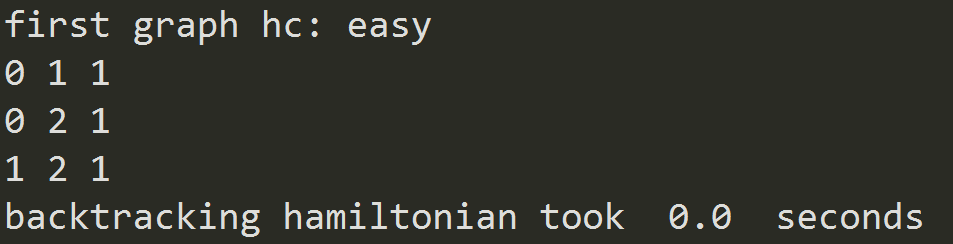


*Part 2: Backtracking Hamiltonian Implementation:*

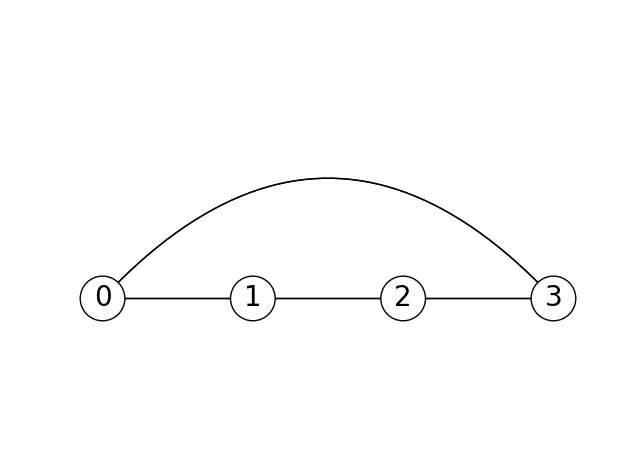
To test the backtracking Hamiltonian algorithm, I created four graphs: the first three had Hamiltonian cycles and the last one does not. For the graphs 1-3, they increase in number of vertices. As mentioned earlier, the 4th graph has no Hamiltonian cycle.

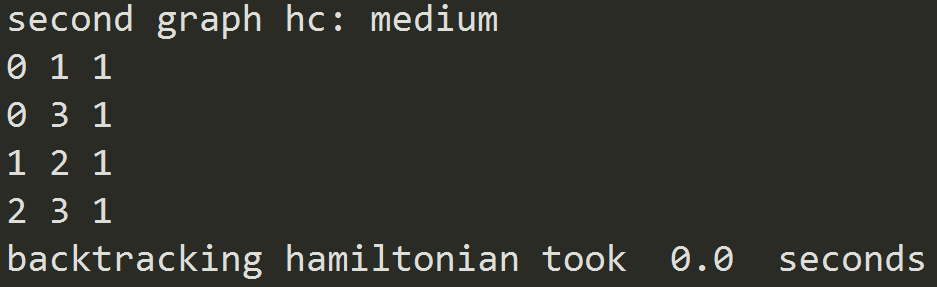
*1st Graph Output and Run Time:*



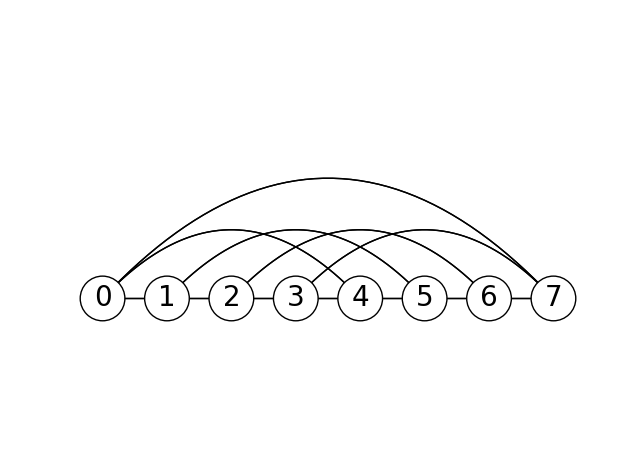
**

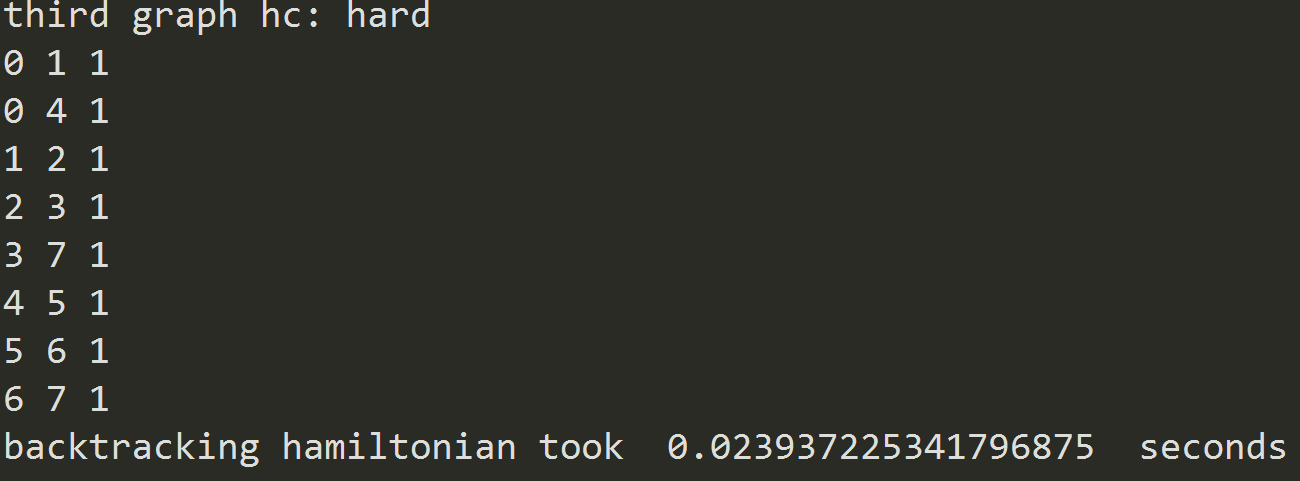
*2nd Graph Output and Run Time:*



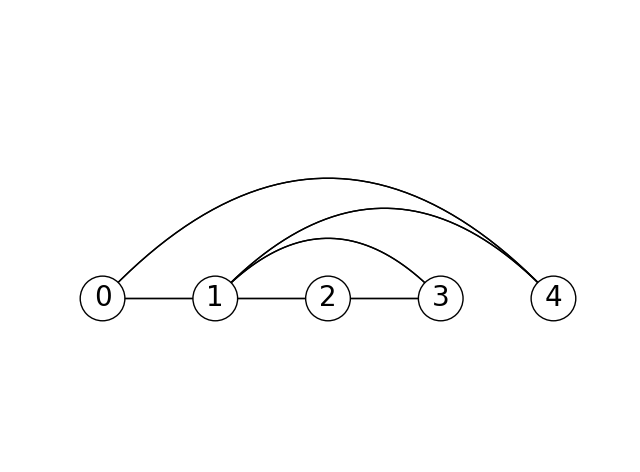
**

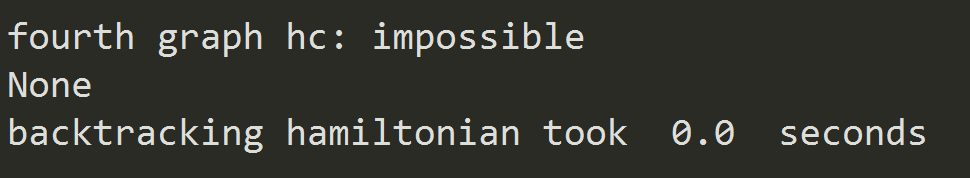
*3rd Graph Output and Run Time:*



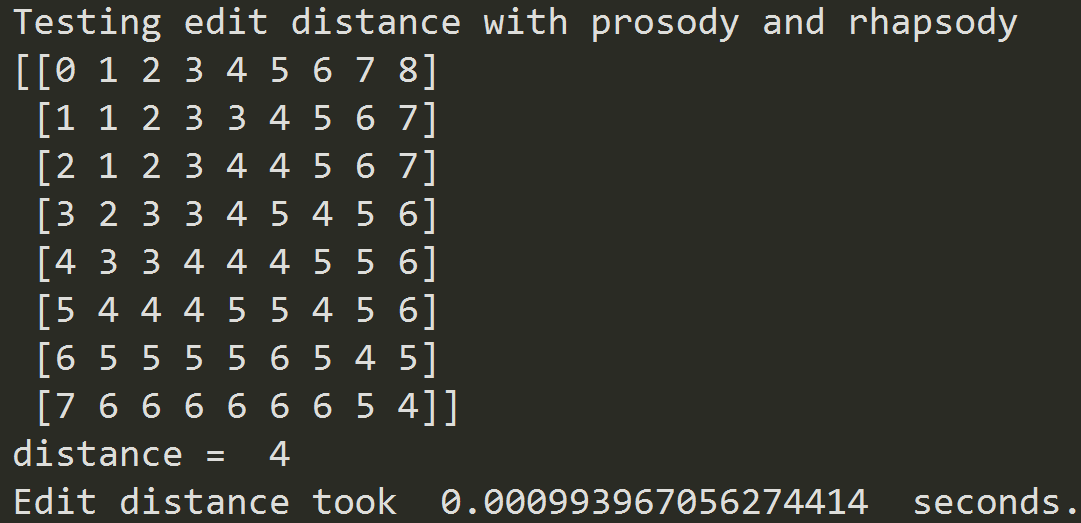
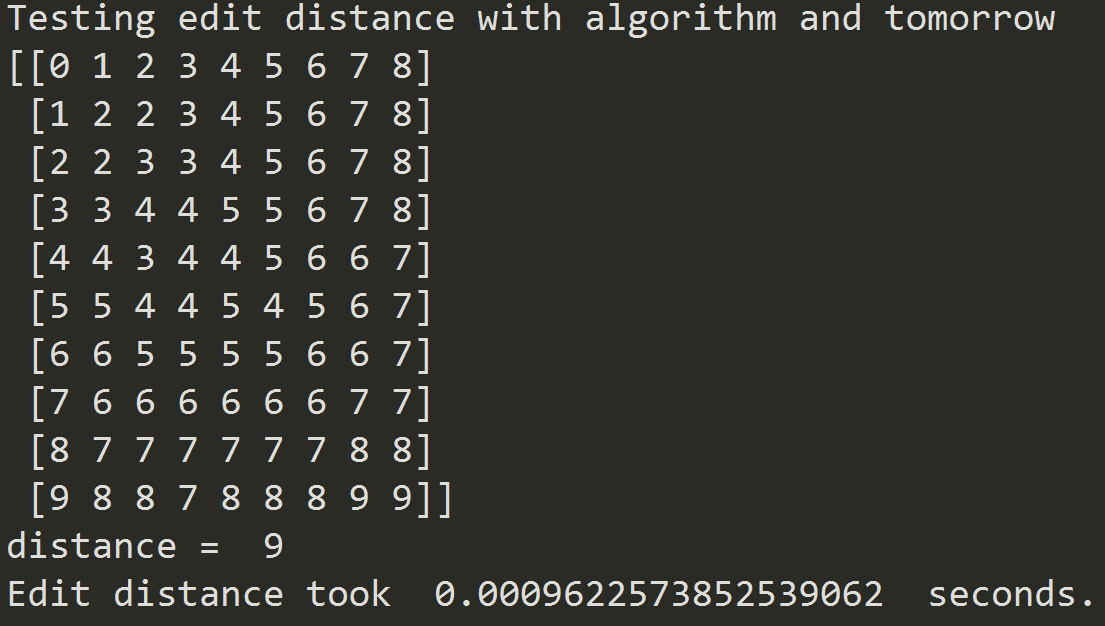
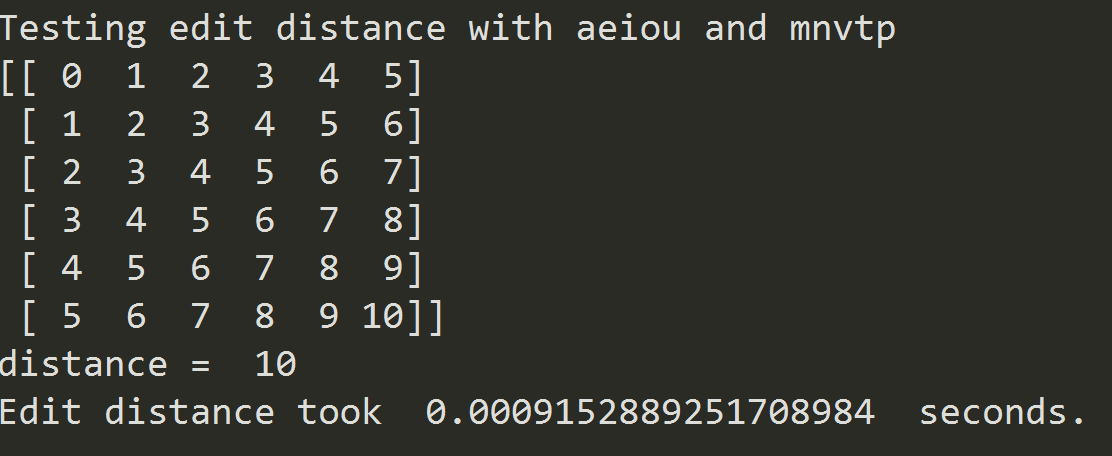
**

*4th Graph Output and Run Time:*



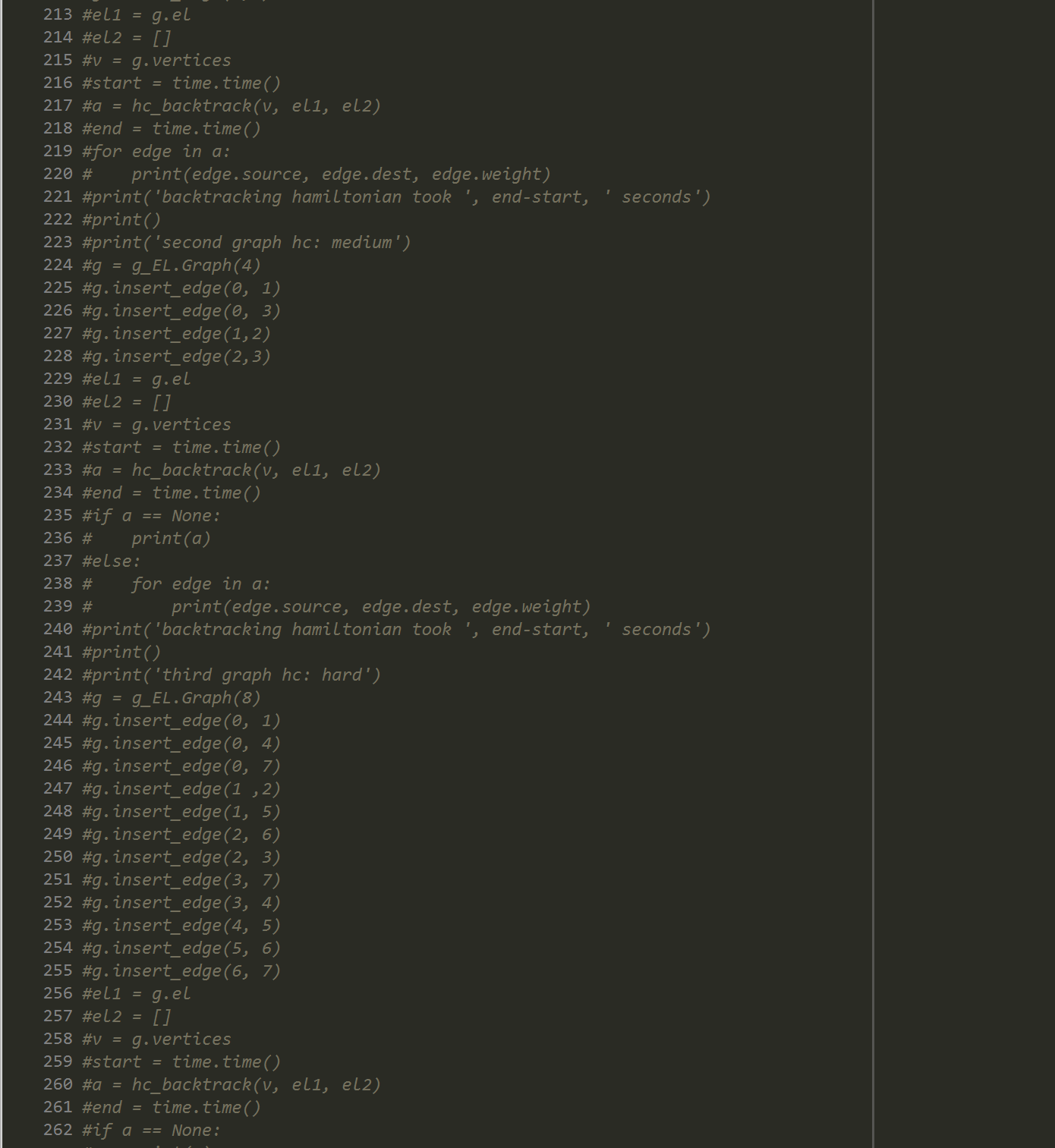
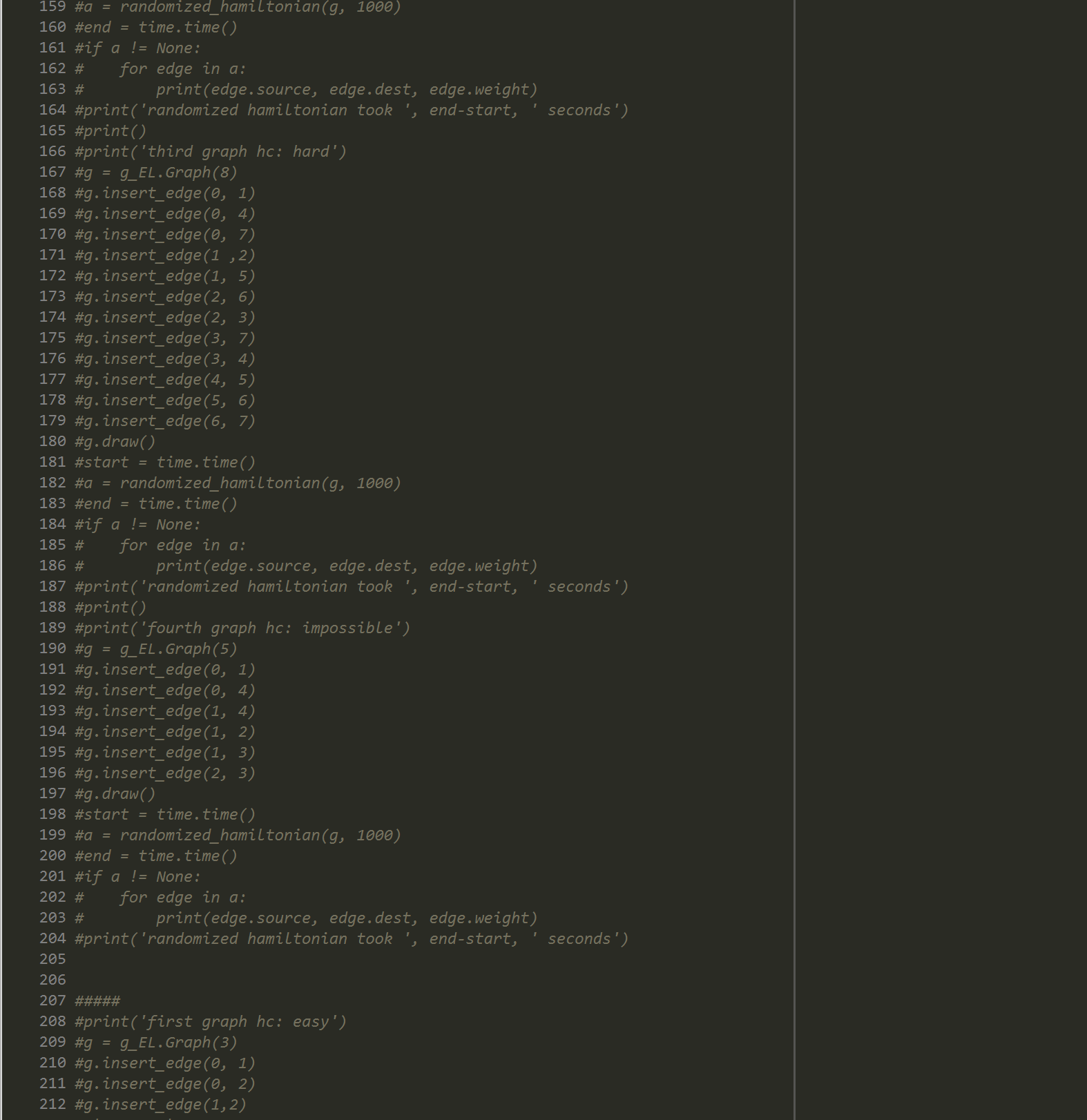
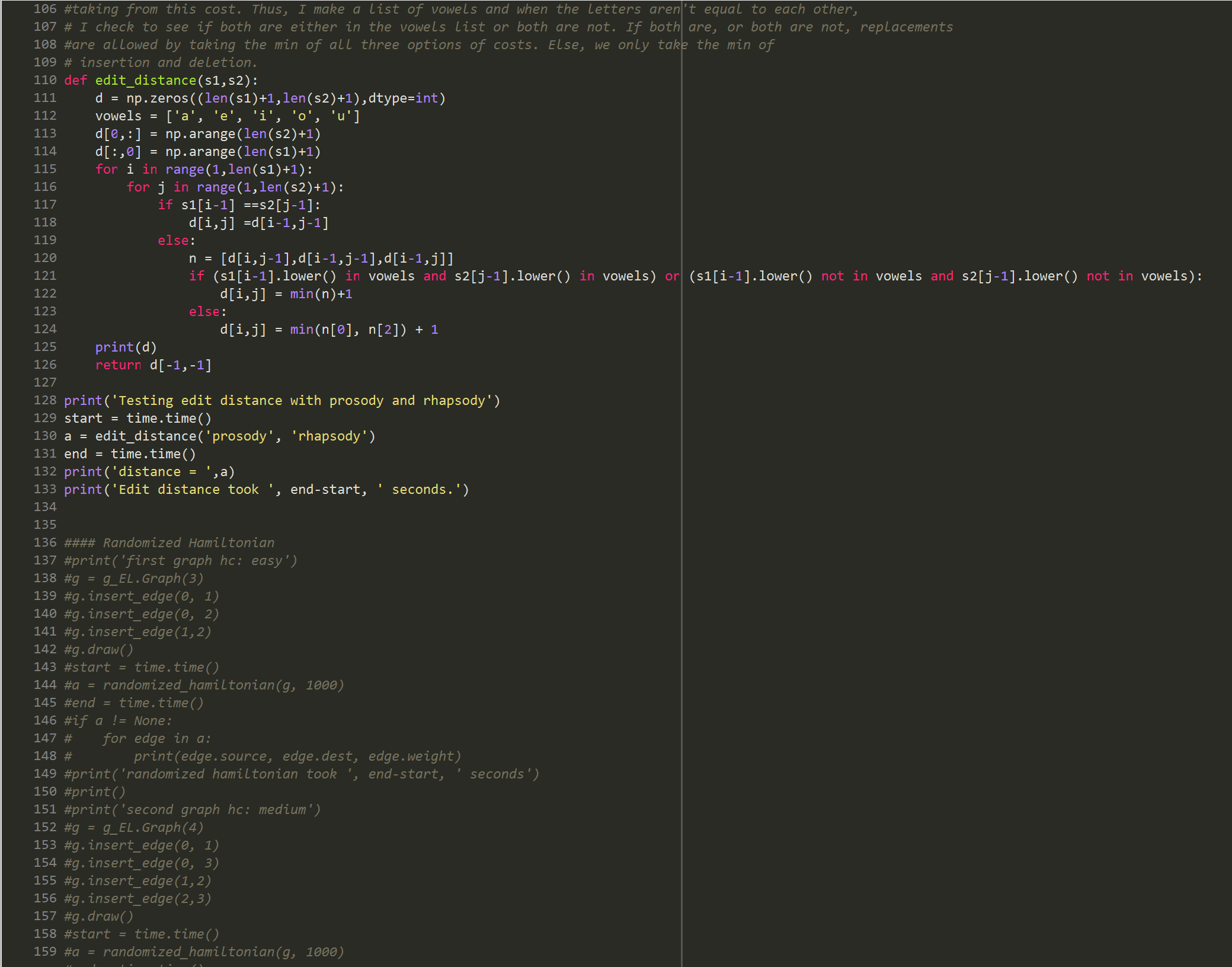
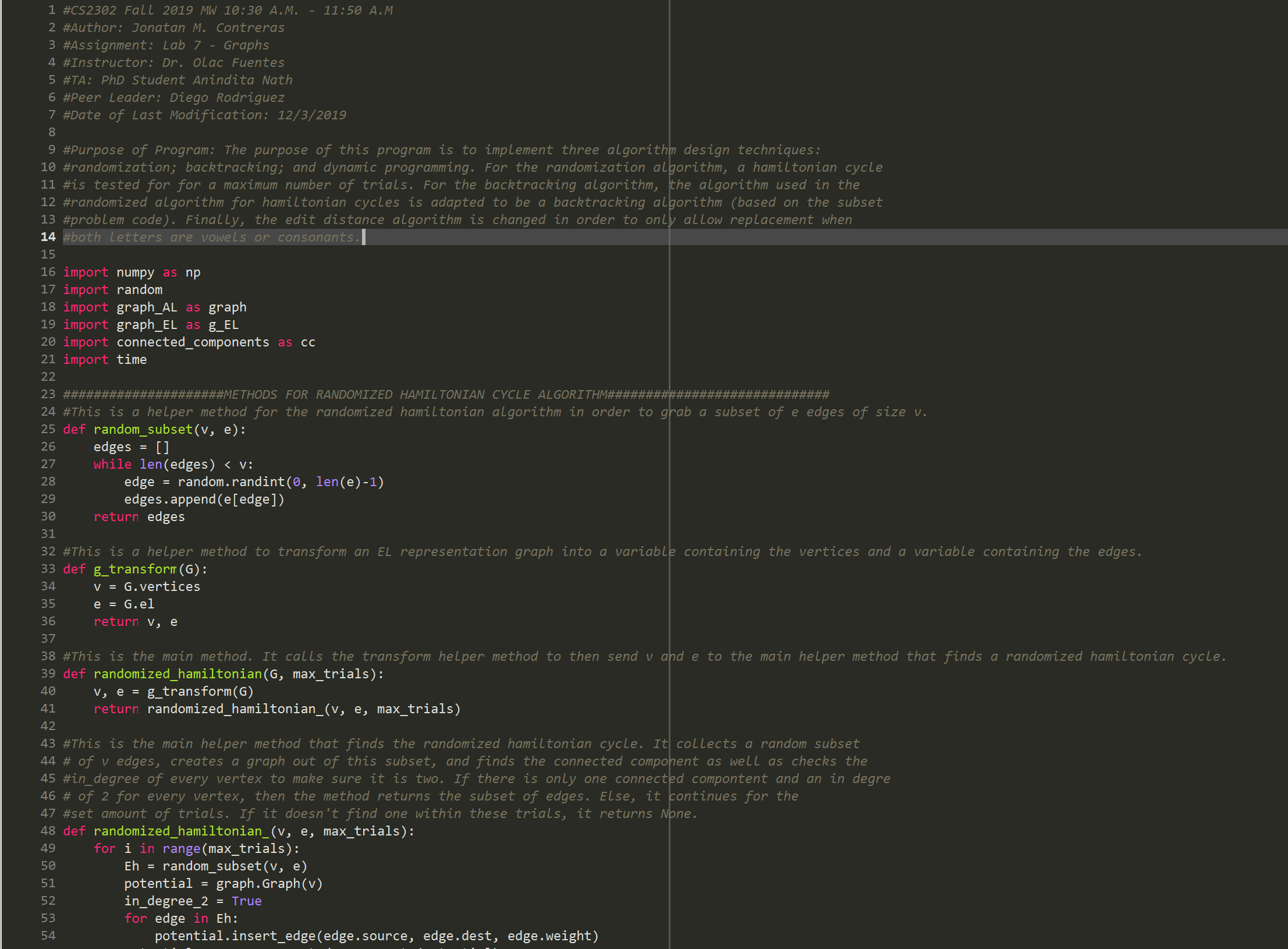
**

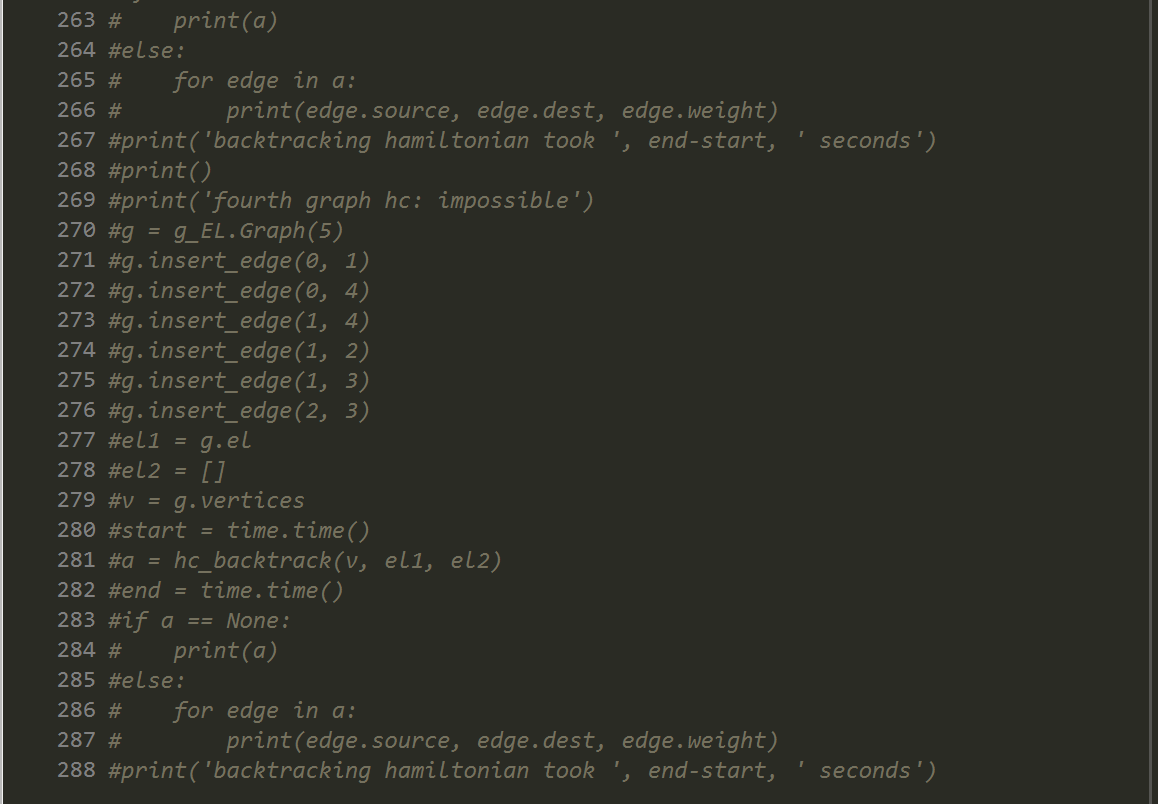
*Part 3: Edit Distance Function:*



**Conclusion**

For this lab, I learned how to use a randomized algorithm to deduce some results for NP-complete problems. I also learned how to create a backtracking algorithm. Finally, I acquired a more intimate understanding of the edit distance function.

**Appendix – Source Code**

****

**Academic Honesty Certification**

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.

Jonatan Contreras