This extra credit was inspired by the original hamiltonian path solver that I wrote for our first project. In project number one, I went into the assignment expecting to have to optimize my NP code at some point. To be quite honest it pained me a bit to see how slow my code ran as I added more and more vertices. So, for this extra credit, I took a revised version of the breadth first search which we learned for project 2, and implemented it within the concept of searching for a path within a specified graph. Instead of looking for a hamiltonian path, I created csv data files that specified a start point and goal point. I also defined the graphs in these csv files similar to in project one for continuity's sake. After my "load_graph_from_csv" populates the graph dictionary, the "bfs_recognizer" function begins a breadth first search on the graph, initializing a "visited" set and a "queue" to keep track of both visited nodes and storing nodes to be explored. The code checks each current node for the goal, returning True or False if the goal was found. I also added in new functionality which visualizes the graph and whether or not the path was found. This visualization utilizes networkx and matplotlib. This added visualization was my favorite new feature in the program, allowing me to more effectively understand the scope of the graph, its functionality, and the path itself. Implementing this new search method when exploring graphs also helped me distinguish between breadth first search and depth first, and why BFS can be more efficient solving NP problems. This extra credit also helped me better understand problems from later homeworks, such as homework 9 individual problem #2.

In my github repo: https://github.com/hokagejack/extra_credit_jkeller7

The code written is under the file main.py

I had lots of test files to test my code, these are labeled graph1.csv - graph8.csv