Teammates:
Alice Getmanchuk - aliceg3, Jerry Balan - agbalan2, Tim Vitkin - tvitkin2, Hassan Farooq - hfaroo9

Mentor:
Stav Ashur

Project Objective Summary

The intention of our project was to create a visual representation of ECE/CS classes and their prerequisites. When a certain class was inputted, the program would output the classes which are a prerequisite for it as well as classes that may be taken after the inputted class.

Discoveries

Direction: Initially, our plan was to make the directed graph have directed edges go from the prerequisite to the more specific class. We thought this direction was more proper since that goes with the flow of classes in course curriculums, but as we started coding our algorithms we discovered that having the graph directed the opposite way (ex. CS225 -> CS125) made our search algorithms work more efficiently.

Algorithm: Additionally, we found that since our graph was technically unweighted (all weights are 1 since the way classes work there is no need for weights), then the BFS algorithm and Djikstra's algorithm behave incredibly similar. Because of this, we decided to create another shortest-path algorithm, A*, to extend the scope of our project. This algorithm will help us find the shortest path between one vertex and all others in order to fulfill the prerequisites for the class the user intends to take (same with Djikstra's). But after some more research and implementation, we decided to go with the Floyd-Warshall algorithm INSTEAD of A* since it is unique in comparison to BFS and Djikstra's in finding the shortest path between all vertices. In addition, the heuristic algorithm for implementing in A* was not possible to make consistent with this university course prerequisite structure. This will allow us to run this algorithm once and be able to extract the shortest path we need (as opposed to running it each time for a single class). Incorporating this algorithm into our project takes our project further than we had originally planned in our GOALS document. We still kept in Djikstra's algorithm since it runs quicker than Floyd-Warshall.

Graph Implementation:
We decided to use a force-directed graph of all the classes in our database, but the initial graph did not group the classes well so that was something we had to figure out (*src/imgs/oldOutput.png* in repo). Using the psuedocode, many of the vertices were

being placed out of bounds of the image. Investigation into this issue showed us that the deltaX and deltaY for displacing our node were sometimes 0 (which when diving a value with this number led the program to not crash but just go out of bounds). We corrected this by changing the deltaX and deltaY to random numbers when they were 0 and also limiting the total displacement occurring on the vertex more aggressively than before. *src/imgs/fdgOutputFinal.png* in our repo contains our improved force directed graph.

Additionally, we created an output of a path when using Djikstra's/Floyd-Warshall's algorithm to show the shortest path from one class to another. Our algorithm was for some reason adding in an extra class at the end of the graph, but we fixed it now so it works correctly. An example output is *src/subsetGraph.png* in our repo for ECE 120 to ECE 391.

Overall

While developing our project, we realized that there is more room for advancing our project and many different applications. Given more time and more data, we could implement this project for all majors in the University of Illinois in order to help with class registration and 4-year plans. The graph concept is simple enough, but with the different algorithms and manipulations of data that we could implement, we could decrease the run time (by possibly using path compression to find classes but leaving our current graph in order to make output paths more reflective of class registration paths) and developing the user interface to be more visually pleasing as well as interactive.