Results:

For the first part of our project, we successfully implemented a Vertex and Graph class. In the Vertex class, we were successfully able to store the names of the articles, its neighbors, and map an index to the corresponding vertex. In the Graph class, our main execution occurs in the default constructor Graph(). This is where we parse the files and initialize our global variables.

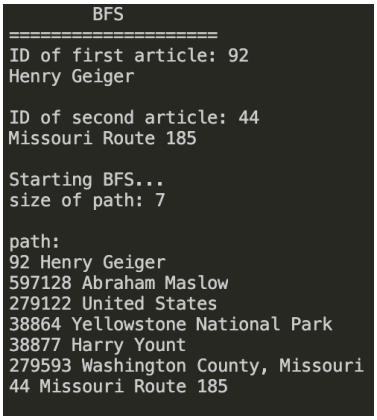
Here are pictures to demonstrate our functionality of our graph:

Print Neighbors

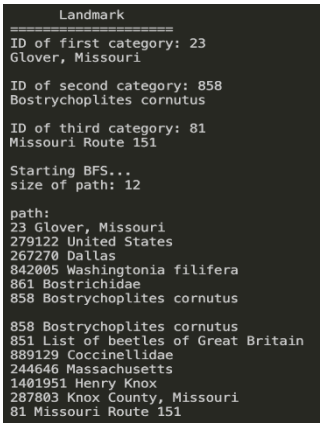
Regarding our algorithms, we have a functional BFS algorithm between two nodes. The BFS returns the shortest path of articles- represented by vertices- that are in between any two articles based on their edges. We used the BFS algorithm to implement Landmark's Path. Additionally, we have successfully implemented a function to detect cycles within our graphs, and Kosaraju's algorithm to detect strongly connected components.

Here is a picture of our main algorithms at work:

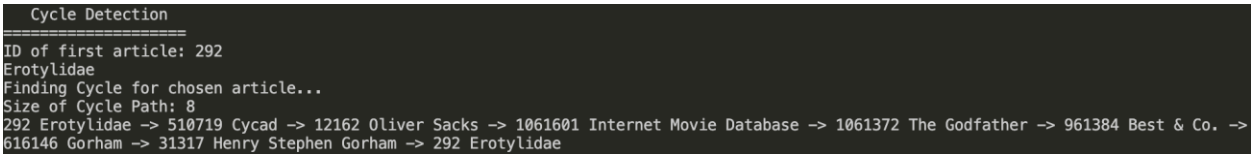
BFS



Landmark



Cycle Detection



Interestingly, while we applied these algorithms in the wiki dataset, we learned the graph is an extremely strongly connected graph. We discovered this by implementing Kosaraju's algorithm.