Natalie Poche
UF ID: 15339442

1. **Describe the data structure you used to implement the graph and why.**
    a. I used the maps and unordered maps to implement the graph because I liked how I could organize between in degree and out degree connections. Additionally, I through it would be easier to use maps and unordered maps to find the corresponding values to their links by directly calling on them instead of trying to figure out the position like I would when using arrays, and/or having it iterate multiple times to find matches to keep track of the connections.
2. **What is the computational complexity of each method in your implementation in the worst case in terms of Big O notation?**
    a. **getVertices() -** The worst case computational complexity of my getVertices() function in Big O notation would be **O(1)** as it is just returning the size of the unordered in degree map.
    b. **initializePageRank() -** The worst case computational complexity complexity of the initializePageRank() function in Big O notation is **O(V)** where V is the number of vertices in the graph, in other words, it is the total unique links in the graph. The reason for this is because it iterates through each vertex in the unordered degree map to create the page rank. When looking back at the in degree map, it is initialized using the insertEdge(string &from, string &to) function which adds to the in degree unordered_map if there is no value there. However, when it is used in the main function, it iterates through the entire list to get the to and from relationships of each link, so it adds to the independent unordered_map once per vertex, or unfound link.
    c. **printPageRanks() -** The worst case computational complexity of printPageRanks() in Big O notation is **O(V)** where V is the number of vertices in the graph, or the number of unique links used. The reason for this is because it uses a for loop that iterates through the entire pageRank map which holds all the unique links and their rank.
    d. **insertEdge(string &from, string &to) -** The worst case computational complexity of the insertEdge(string &from, string &to) function would be **O(1)** since all it does is use if statements to create the unordered maps with the in degree and out degree connections and since there's no for or while loops, each operation is O(1), so the overall complexity will simplify to O(1) in Big O notation.
    e. **pageRank(int power_iteration) -** The worst case computational complexity of pageRank(int power_iteration) in Big O notation is **O(P * (V + E))** where P is the number of power iterations, V is the number of vertices or unique links, and E is the number of edges. The reason for this is because at the start, the initializePageRank used is O(V), then going to the inner for loop of the inner for loop, it iterates through all the in degree links of the vertex. This is repeated for all the vertices in the in degree unordered_map as given by the inner for loop, which is V. Since the inner for loop of the inner for loop iterates through every in degree connection of a specific vertex given by the inner for loop, together they make (V + O), which is then repeated P times, which is due to the outer for loop that uses power iterations. So, the computational complexity would be (V + (P * (V + E))), but would instead be simplified down to O(P * (V + E)) since P * (V + E) is greater

than V and grows at a faster rate. The reason why the complexity is not P * V * E despite there being a for loop nested in a for loop nested in a for loop, is because the inner for loop of the inner for loop iterates through the in link connections of only the related vertex given by the inner for loop. If the inner for loop iterate through every edge regardless of its relationship to the specific vertex provided by the inner for loop, then the complexity would be O(P * V * E).

3. **What is the computational complexity of your main method in your implementation in your implementation in the worst case in terms of Big O notation?**
    a. The computational complexity of my main method would be O(P * (V + E)) where V is the power of iterations, V is the number of vertices or unique links, and E is the number of edges. The reason for this is because the initial for loop used to insert the edges from the input is iterated through based on the inputted number of lines. Since each line is a different relationship between the two links, each line is an edge, so together, the number of lines are E, or the total number of edges. Inside the for loop, the complexity is 1 + 1 from the established from and to variables and then another + 1 from the insert edge method as previously mentioned in the implementation method section, which is simplified to O(1), which makes the total complexity of the for loop O(E). Then going to the PageRank method which was previously established to be (P * (V + O)) in the implementation method explanation that would make the total complexity of the main method to be O(E + (P * (V + E))), which would simplify to O(P * (V + E)) since P * (V + E) is larger than E and grows at a faster rate.
4. **What did you learn from this assignment, and what would you do differently if you had to start over?**
    a. From this assignment I learned more about the difference between O(P * (V + E)) despite the fact that it was a for loop nested in a for loop nested in a for loop due to the discriminatory factor where the inner for loop of the inner for loop only iterated through edges based on the specific vertex provided by the inner for loop instead of indiscriminately iterating through every edge regardless of its relationship to the specific vertex. Before, I was under the impression that nested for loops automatically meant that the time complexity was multiplying variables, but it was interesting to learn I was wrong and why. Additionally, I got more practice using maps and unordered maps before I used to be somewhat familiar with them, but now I am definitely more comfortable using them. If I had to start over, I would probably redo the first 5 unique test cases I have since it was halfway through on my sixth test when I figured out a more efficient process where I just have to copy/paste the input and outputs instead of just copy/pasting the input and manually setting the requirements of the output to pass the test case. Additionally, instead of having a void page rank function, I would have made it return the map so i could control if the map printed or not instead of automatically printing by putting the printPageRank function inside the PageRank function. This would also mean I would make the printPageRank function have a

Natalie Poche
UF ID: 15339442

map input that it uses to print out the numbers. Additionally, I would adjust my getRoundedPageRanks to take in a map instead of an int, and use that.