JOHNSON NGUYEN

1. Describe the data structure you used to implement the graph and why. [2.5 points]

The graph is implemented using map, where each key is a string representing a vertex, or webpage, and each value is a pair. This pair contains a vector of pairs representing outgoing links and a float representing the PageRank value of the vertex. Each element in this inner vector is a pair, where the first component is the destination vertex and the second component is the edge weight. This structure was chosen because maps allow for efficient traversal and lookup, either through each vertex or a vertex's edges.

2. What is the computational complexity of each method in your implementation in the worst case in terms of Big O notation? [5 points]

insert: The complexity is O(log V), where V is the number of vertices in the graph. This is because I used the insertion() function of maps, which itself takes O(log V). Adding an outgoing edge link to the vector takes O(1).

pageRank: The complexity for each iteration is O(E + V * log V), where E is the number of edges in the graph. For each node with outgoing edge links, we distribute its rank across its neighbors, taking O(E). Updating the ranks involves iterating over each node, which takes O(V), and looking up vertices in map, which is O(log V) per iteration. Since pageRank performs p (power) iterations, the total complexity of pageRank is O(p * (E + V * log V)).

check: This method iterates through each vertex and its list of outgoing edges, resulting in a complexity of O(V + E), where V is the number of nodes and E is the number of edges.

3. What is the computational complexity of your main method in your implementation in the worst case in terms of Big O notation? [5 points]

The complexity of the main function is O(no_of_lines * log V + p * (E + V * logV)). The initial loop over no_of_lines edges calls insert, which takes O(log V) per insertion. So, inserting all edges has a complexity of O(no_of_lines * logV). Main eventually calls pageRank as well, which we know to be O(p * (E + V * log V)). Adding these time complexities together, you get the worst time complexity of the main function being O(no_of_lines * log V + p * (E + V * logV)).

4. What did you learn from this assignment, and what would you do differently if you had to start over? [2 points]

In this assignment, I learned how to implement an adjacent list and compute a simplified PageRank. I also gained experience with balancing data structures for both efficiency and readability, particularly when representing graphs with map, vector, and pair.

If I were to start over, I would consider implementing a separate data structure for incoming links, which would simplify and potentially optimize the PageRank computation. This approach could reduce complexity by allowing direct access to nodes with incoming edges, avoiding the need to iterate through all outgoing edges on each iteration.