Simplified PageRank – Documentation

Data Structure

Map Data Structure

```
map<string, vector<pair<string, double>>> adjacencyList;
```

I implemented a map data structure that stores the key-value as all the unique websites, which have a value of vectors of (website, rank) pairs. This way, in order to implement the project as an adjacency list, the "*from*" website links can point "to" a set of links. And each link that is pointed at has a rank that can be calculated afterwards.

Implementation Time Complexity

```
void insertVertex(string from, string to)
```

Best Case	Ω(1)	Operates in constant time where size of map is 1 or less to push back to a vector which is constant time
Average Case	Θ(log(n))	Where n is the size of the map and due to the implementation of a map as a red-black tree
Worst Case	O(log(n))	Where n is the size of the map and due to the implementation of a map as a red-black tree

```
void rankCalc()
```

Best Case	Ω(1)	Operates in constant time when the map size is 1 or less and has a vector size of 1 or less
Average Case	Θ(m*n)	Where m is the size of the link-map and n is the size of the vector of each link. Iterates through each vector times each link
Worst Case	O(m*n)	Where m is the size of the link-map and n is the size of the vector of each link. Iterates through each vector times each link

```
void PageRank(int power)
```

Best Case	Ω(1)	Where the size of the link-map, power iterations, and link vectors are all 1 or less
Average Case	Θ(m + p*m*n)	Where m is the size of the link-map, p is the number of power iterations, and n is the size of the vector of each link. Initializes an output map by going through the link-map, then iterates through each vector times each link for p iterations
Worst Case	O(m + p*m*n)	Where m is the size of the link-map, p is the number of power iterations, and n is the size of the vector of each link. Initializes an output map by going through the link-map, then iterates through each vector times each link for p iterations

Main Method Time Complexity

`int main()`

Best Case	Ω(1)	When the size of input is 1 or less
Average Case	Θ(n)	Where n is the number of lines
Worst Case	O(n)	Where n is the number of lines

Reflection

The biggest thing I learned from this project was the implementation of maps to simulate an adjacency list and how to set up structures to manipulate and return weights and probabilities. It was really interesting seeing how linear algebra could be incorporated into the project to get the Page Rank probabilities and was a nice exercise in matrix multiplication.

What I would do differently if I had to start over would be find a more optimized way to deal with the matrices and multiplication. In addition, storing the PageRank probabilities in a separate map increased the space complexity, so I would like to see if I could condense it into a single data structure that could handle all of the values and mathematical manipulation.