This project is designed to perform various graph operations, including calculating PageRank, finding minimum distances between vertices through BFS, identifying connected components, and determining the maximum degree of separation within a graph, if possible. The application reads graph data from a TSV file that represents 4.600 Wikipedia articles and how they are linked to one another. It constructs a graph with this data, and then allows the user to interact with it through a series of operations.

A key part of the graph struct that I use in this code is that it uses a hash function to map vertices to their string labels. This helps as the vertices are not in numerical form as with past graphs that I have worked with.

Here are some of the main functions and what each of them do:

- **read_file:** The graph data is read from a TSV file where each line represents an edge between two vertices.
- **page_rank**: Computes the PageRank of vertices to determine their importance based on the link structure.
- **min_distance**: Finds the shortest path between any two given vertices using BFS.
- **connected_components**: Identifies and lists all connected components of the graph.
- **max_degree_of_separation**: Determines the largest number of steps in the shortest path connecting any two vertices in the largest connected component of the graph.

To run the code, please use cargo run in the main.rs file. This will produce an output that includes the page rank results. Next, it will prompt you to enter the names of two articles, the program will then calculate the quickest path (if possible) from the first article to the second. Following this the code finds the connected components of the main graph. There is 1 main component of this graph, so the code then splits that component into its own subgraph. Sadly, this subgraph still is not strongly connected as the graph is directed. Finally, the code attempts to find the maximum degree of separation across all vertices, but fails as the component of index 1 is not connected to every vertex.