DEVELOPMENT JOURNAL

Week of 11/18 - 11/25

- **MakeFile (Arthur and Allan)**: We created a MakeFile and made sure it compiles.
- **General Structure (Team)**: After discussing, we decided to have 3 main cpp files. A graph file that generates graphs from data. A PageRank file to take in the graph(directed) and run the algorithm. Also need a BFS file to perform traversal on the graph, and perform the Dijkstra Algorithm to find the shortest path.
    - **Graph Generation (Arthur and Shouri)**: Randomly able to select a subset of file data to build the graph. Decided to use an adjacency-list and an adjacency matrix. Need to update weights for the adjacency matrix. We attempted to implement the PageRank algorithm using matrix multiplication with the adjacency matrix, but this could cause runtime problems, so we decided to use the adjacency-list for this implementation instead.
    - **Graph Functionality (Arthur and Shouri):** Implemented buildGraph to connect edges from data, and kept track of weights based on encountered edges. Able to properly connect source and destination vertices and update edges accordingly and tested for correctness.
    - **PageRank Algorithm (Henrik And Allan)**: We implemented the pagerank algorithm as a class which stores the weighted values/ranks of each vertex of the dataset after the class finished updating the weight.
        - GreatestThree: function that takes in a vector consisting of weighted vertices, and returns the 3 with the highest weight, because we don't want to return everything when it comes to a large dataset. Useful for testing and debugging.
        - Initialize Weight/UpdateWeight: function updates the weight using the adjacent list, running multiple iterations to get an accurate weight value before stopping. Implemented constants like decay value and epsilon for helping with number of iterations and the weighted value to add based on the adjacency list(explained more in comments of code).

- **Testing (Allan):** We use tests to demonstrate that our code is robust. The generateGraph executable will provide some visualization. Tested for different graphs and edge cases, covering PageRank.
- Note: The general graph data structure implementation is taken from lab_ml, so is the MakeFile.
- Algorithm followed(PageRank Algorithm guideline): https://courses.cs.washington.edu/courses/cse373/17au/project3/project3-3.html
- **Also worked from 11/25 to 11/29**

Week of 11/30 - 12/7

- **BFS Traversal (Shouri and Arthur)**
    - Continued adding to the buildGraph function so that it also builds a graph correctly with data with weights.
    - Implemented BFS to traverse all nodes in a graph based on pseudo code given in lectures.
    - Developed test cases(including simple ones and more complex ones) to test BFS traversal on a given graph.
        - The result returns the list of edges with labels “DISCOVERY” or “CROSS”.
- **Dijkstra’s Shortest Path (Henrik, Allan, Shouri)**
    - Found the dataset Oldenburg road network which has a list of edges and weights to run the Dijkstra’s shortest path algorithm on.
    - Implemented Dijkstra’s algorithm with a priority_queue with a custom comparator.
    - Developed test cases including a simple graph and a more complex graph which supported varied edge weights to comprehensively test the algorithm.
        - The result returns a list of distances(the shortest possible) to every other vertex from the starting vertex
    - Note: does not support self-loops, and negative edge weights.

Week of 12/7 - 12/11

- **Cleaning up code (Henrik):**
    - Tidied up code for readability.
    - Wrote code in Main to write to text files for easier visualization of results.
    - Deleted commented code and created spacing for readability.
- **Presentation Creation and Development Journal “Results” Work (Team):**
    - Making a 10 minute Youtube video demo-ing our code, test cases, and techniques used throughout the creation of the final project process.
    - Summarizing work and results discovered on project in “results” portion. Will be going over test case coverage and what we learned throughout working on the final project.