abiswas7, calebk3, kartikm3, lyuxing2

CS 225 FINAL PROJECT REPORT

Subreddit Hyperlink Network Analysis

Project recording: https://youtu.be/fpR6IwaQE3E

About the Dataset

The Dataset used for this project is the Reddit Hyperlink Network dataset, which came from the Stanford dataset collection. The hyperlink network represents the directed connections between two subreddits. This network is extracted from Reddit data of 2.5 years, from January 2014 to April 2017. This dataset is a tsv file (tab separated values file). We wrote our own parsing algorithm (as provided in read_tsv.cpp) which converts the tsv data to get columns in the dataset. The important columns that we are focusing on are: source subreddit and target subreddit, which will describe the hyperlinks between pages. There is also a “post label” column, which describes the sentiment toward the current subreddit, be it positive or negative. When creating our graph, the source and target subreddit hyperlinks are taken as the vertices, and the sentiments are taken as the weights of the edges — positive being 1, and negative being 3.

Project Goals

The main goal of our project is to provide an interactive tool for users that gives them useful information about subreddits stored within our dataset.

The “BFS” option asks the user to enter their starting subreddit, and it will output all the subsequent subreddits the user can go to in the order of Breadth First Search.

The “Dijkstra’s Shortest Path” option asks the user to enter their current visiting subreddit and their destination subreddit. It allows the user to find whether they can go from current subreddit to their desired subreddit, and, if that’s an yes, provide an estimation of how well they might feel about their desired subreddit. The lower the estimation is, the more enjoyable the user might feel about the destination subreddit.

The “Page Rank” option asks the user to enter an accuracy factor and the destination subreddit they want to check. It allows the user to find the probability in general of visiting each subreddit within the dataset. The higher the accuracy factor is, the more accurate the prediction will be. The higher the output Page Rank value is, the more likely a user might end up there. It also indicates the popularity/ significance of the certain subreddit is among all the subreddits within the dataset.

abiswas7, calebk3, kartikm3, lyuxing2

Algorithms

For our project, we implemented the Page Rank Algorithm, Dijkstra's Shortest Path Algorithm and the Breadth-First Search (BFS) Traversal. The BFS takes in a starting vertex (i.e. a starting subreddit) and uses a queue to traverse through the graph in a breadthward motion. Our BFS returns a vector of Unique IDs of all the traversed connected subreddits. The time complexity of the BFS algorithm turns out to be O(n).

The Page Rank algorithm helps find the most "important" vertex within our graph by determining the Page Rank value of each vertex. Each vertex is initially assigned to the same Page Rank value of $1/|V|$. At the next iteration, the page rank value of any vertex u is defined as $PR(u) = \Sigma_{v \in A(u)} PR'(v) / |B(v)|$, where PR(u) is the Page Rank value of currently-traversed vertex, A(u) is the set of incoming edges of vertex u, PR'(v) is the Page Rank value of u from the last iteration, and B(v) is the set of outcoming edges of vertex v. As the algorithm goes through more and more iterations, the Page Rank value of each vertex will normally converge to a certain value, and the vertex with the highest Page Rank value will be the most "important" vertex, just like a transport hub within a transportation network. The time complexity for each iteration turns out to be O(n+m) where m is the number of edges and n is the number of vertices/nodes.

The Dijkstra algorithm helps find the distance between an input vertex and the rest of the vertices in the graph. As introduced before, the weight of our graph is the sentiment (either 1 (positive feedback) or 3 (negative feedback)). The algorithm will return the distance vector, and the vertex with the shortest distance is the vertex that the user will like the most / have the best feeling. Building on this, we can also use the unique ID of a subreddit to index into the vector to find the 'distance' i.e. the number of minimum clicks needed to go between the two subreddits. The time complexity of the Dijkstra algorithm turns out to be O(n + mlog(n)).

We are using 3 classes in our project: Graph, Vertex, and Edge. As mentioned previously, the subreddits are treated as the vertices in our graph. The Vertex class stores the name of the subreddit as a string and a unique ID assigned to each subreddit as its private variables. The Edge class stores the sentiment (which acts as the weight) of each edge as its private variable. Finally, the Graph class has the three following private variables: (1) a vector of strings, which is used to store the subreddit names in the beginning during initialization; (2) a map which is used to map each subreddit to an Unique ID; and (3) an adjacency matrix which is a 2D vector of Edge pointers. The reason the adjacency matrix is Edge pointers, rather than just binary to indicate if an edge exists, is that our graph is weighted with the sentiment. Because we need to keep track of these weights, it is more efficient to just store the edges, rather than keeping track of weights within each vertex (since we would need to keep track of which vertices are connected, which is exactly what an Edge does anyways).

The Graph class also contains some noteworthy member functions, such as the Graph constructor, assignID, and findAdjacentVertices. AssignID, as the name suggests, assigns each subreddit an unique integer ID. This unique mapping is important, since it determines the entries

of our Adjacency Matrix so there are no repeated vertices. FindAdjacentVertices takes in a source vertex as the parameter and returns a vector of adjacent vertices. Since our graph is directed, the return vector depends on the flag passed into the function, which indicates whether to find the adjacent incoming vertices or outgoing.

Testing process

To test our algorithms separately in an independent manner, we created tests for each algorithm which is provided in the tests.cpp file in our repository. The instructions to run these tests are provided in the README.md file.
There are 9 tests with 37 assertions in total. We have 3 tests with 20 assertions for BFS. The tests check the traversal on graphs of varying sizes. 2 tests with 8 assertions for Djikstra's. The tests check Dijkstra's algorithm on graphs of varying sizes. 2 tests with 6 assertions for Page Rank. These tests check for page rank values of graphs of different complexity. 2 tests and 3 assertions for our graph constructor. This checks for an empty graph. I believe that our tests check any edge cases needed to be checked and do a thorough job in verifying the validity of our algorithms.

Outcomes

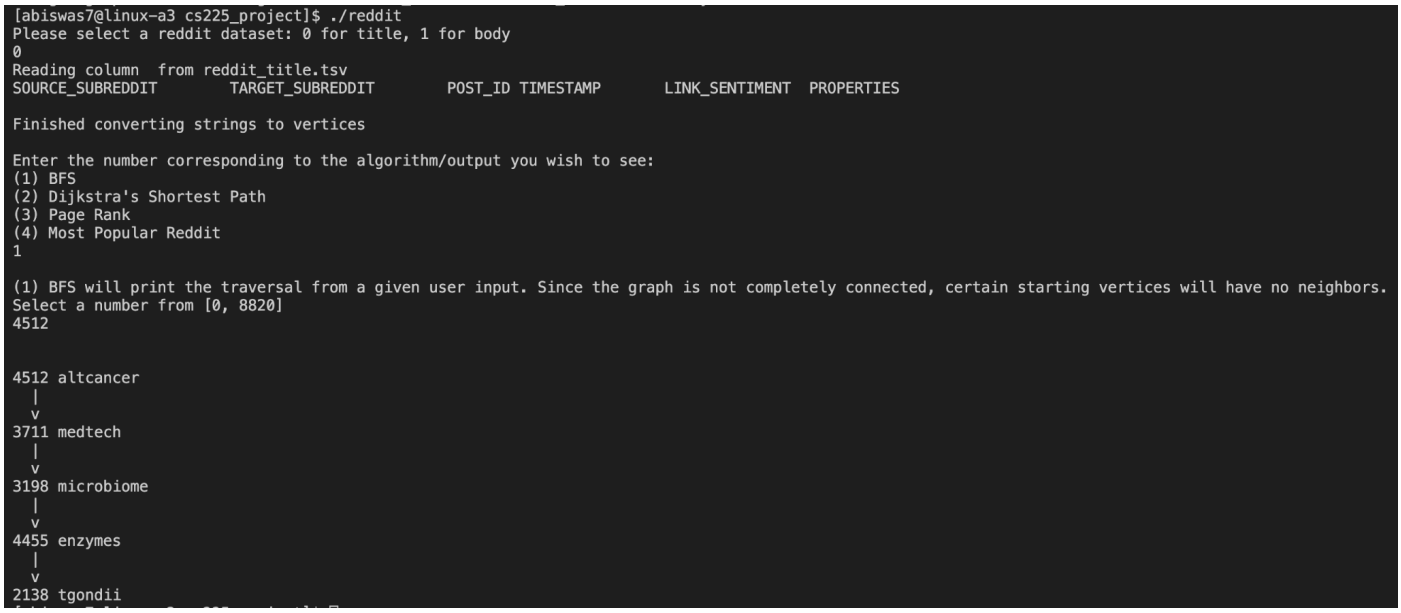
We are glad to say that we were able to achieve the goals that we set for ourselves in regards to how we wanted our project to turn out. All the members worked harmoniously together. We always brainstormed about any ideas together, ensuring that each member is able to express their opinions. We made sure to communicate every small issue with each other and also met up multiple times to work together as we found that to be the most efficient. We were successful in the implementation of all the algorithms mentioned in the proposal. There were moments when we were unsure of how to start working on a particular component of the project or were hesitant about the choices of our algorithms and data structures chosen but as a group were able to overcome any problems faced. Firstly, we were a bit confused on how to implement the graph using the adjacency list as we all had variations of it in our minds so we took the time to explain each of our ideas on a whiteboard and figure out which way would be the best. Secondly, we were not able to figure out how to get the make file working so we had to look at make files from previous assignments and through trial and error figure out what changes needed to be made for our case. Finally, we had an issue with how to coordinate all the code we write so we decided to split into pairs so we have fewer merge issues and branches. We still faced issues on git but we found resources online to figure out how to merge and keep the right code.

To carry this project further we could have used more algorithms like the betweenness centrality to get another perspective on which subreddits were more important according to one algorithm and another. This would have given us a more complete understanding of the network. With this knowledge about the network the real world application would be recommending the most apt subreddit links to the user and also make the network more efficient as we would know which nodes hold the most connections.

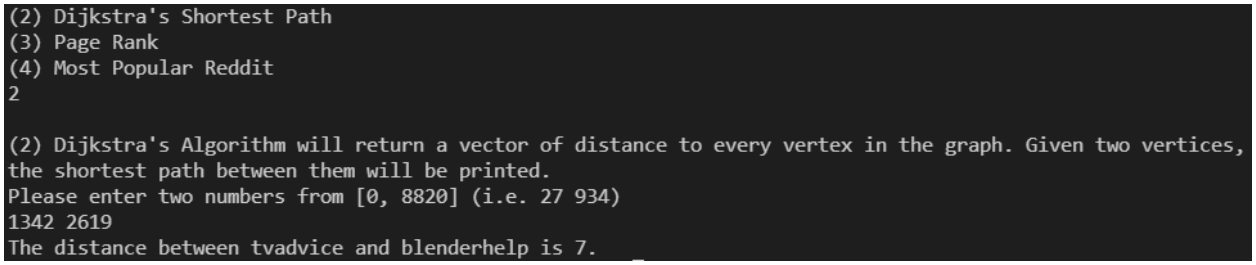
abiswas7, calebk3, kartikm3, lyuxing2

Results

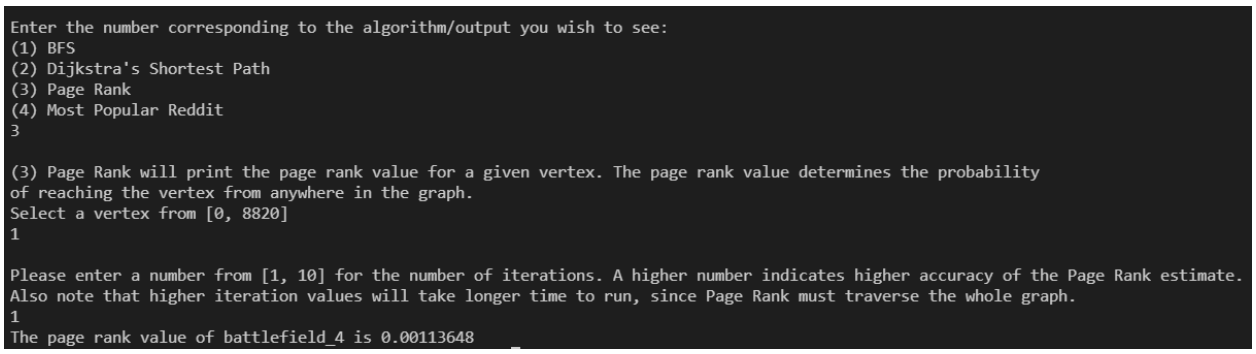
BFS output:



Djikstra's Shortest path algorithm:

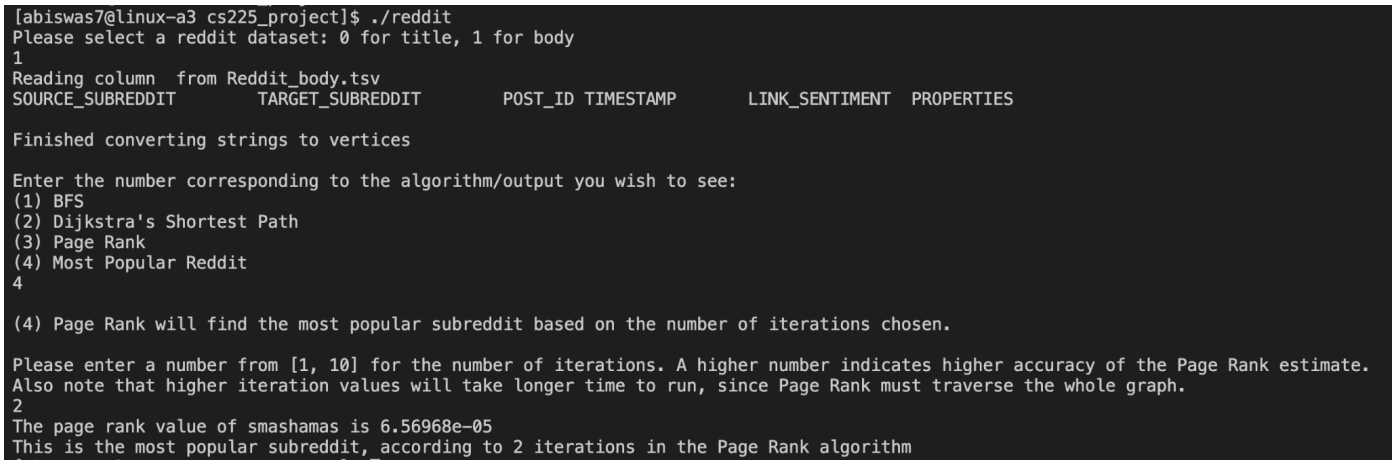


Page Rank algorithm output:



abiswas7, calebk3, kartikm3, lyuxing2

Most popular Reddit:



All test cases passed:

The following screenshot shows that our program is passing all the test cases and assertions