Final Project

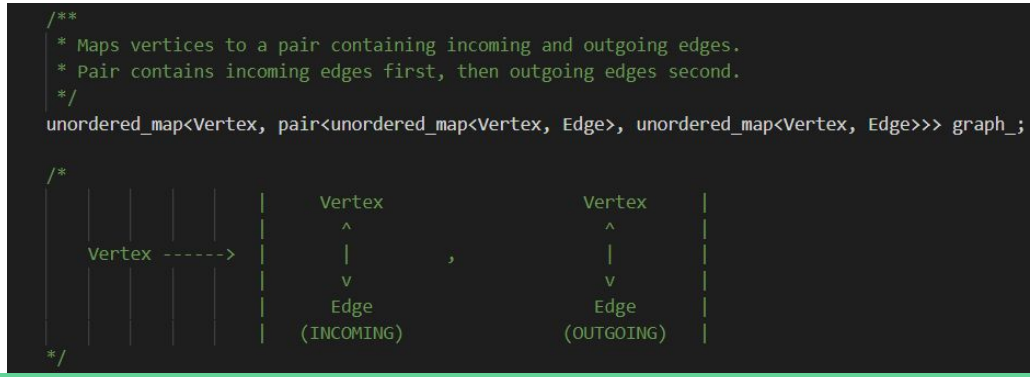
By: Francisco C, Abdi , Santiago V, Majeed

Goals

- Our main goal for this project was to find the answer to 2 questions:
  - The closest connection between 2 sellers.
  - To find the if there was a correlation between a seller's popularity and their likelihood a new user would interact with them.
- Our final deliverables are:
  - PageRank algorithm
  - BFS traversal
  - Dijkstra's algorithm
- With these, we are able to show a seller's PageRank and average rating as well as the path between them and another seller.

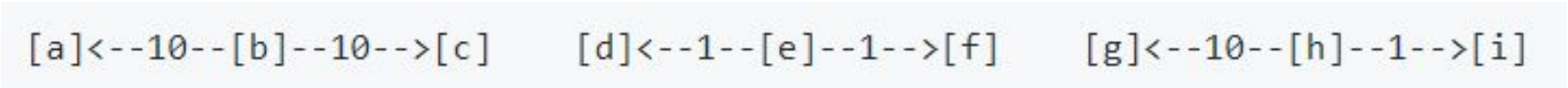
Development

- Wanted a way to store both incoming and outgoing edges of a node so they were easily accessible and were easy to add and remove from the graph.
- Chose to work with maps for our graph data structure. We used vertices as the keys and a pair of maps to separate the incoming and outgoing edges.
- Our edges are structs containing the incoming and outgoing edges as well as the weight, representing the rating, and a label which is used to label edges during our traversal.



PageRank Algorithm

- Our goal was to pair this with each user's average rating to compare them and see if they had any sort of correlation.
- Challenges: negative weights, sink nodes, when to assume convergence.
- The algorithm performs repeated iterations of a process that distributes each nodes pageRank (probability of going into that node) among it's outgoing edges factoring in the weights. Iterations stop once it is found that the difference in the probabilities between one iteration and the next is found to be less than a given margin of error.

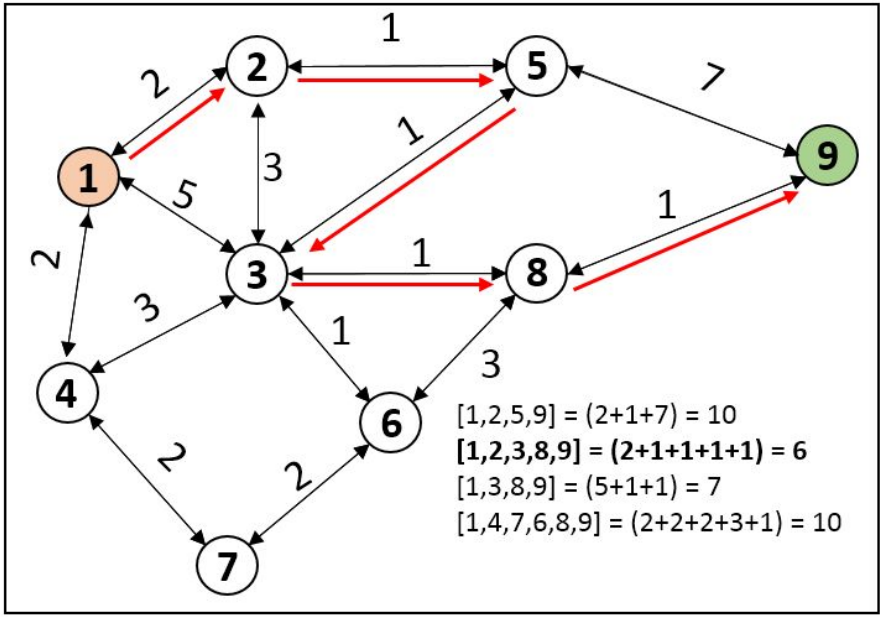


Dijkstra’s Algorithm

- We originally thought to solve the first problem through the use of Dijkstra’s algorithm, but we found out later that this could be solved through BFS.

- Challenges: having to account for negative ranking in implementation of algorithm.

- The algorithm starts at a source vertex and then finds the shortest path between said vertex and all other vertices on the graph. Keeping track of the currently known distances from each vertex to the source, it updates the values if a shorted path is found. This process is repeated until all the vertices in the graphs have been added to the path.



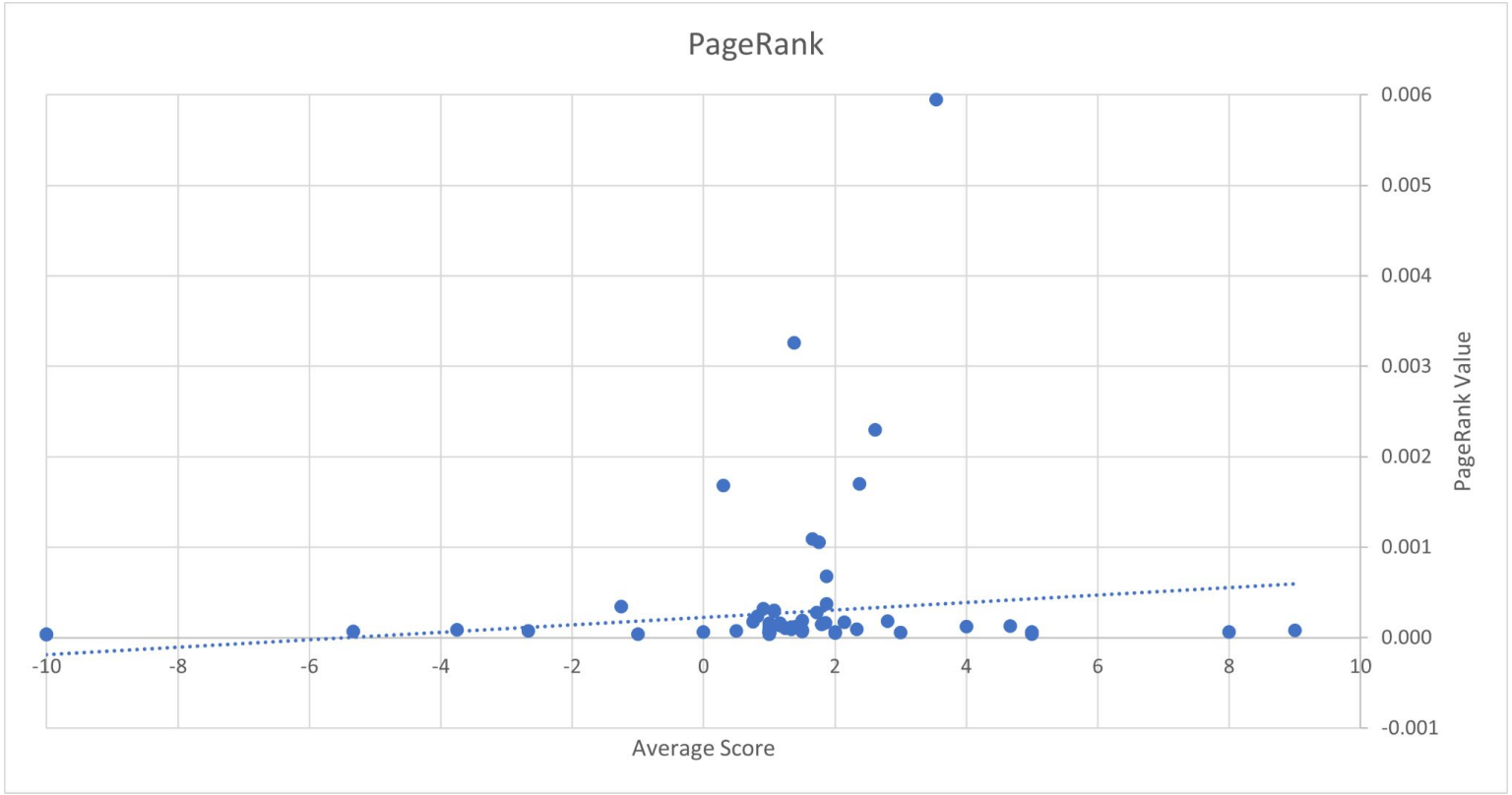
Traversal

- We traversed the graph using a BFS algorithm. Instead of creating an object called “BFS” for example, we included all of the code inside a object class called *vertexTraversal*, which has an *Iterator* class contained inside of it. In this class, the actual traversal occurs inside the ++operator.
- The traversal will only add a neighbor to the traversal is there is an outgoing edge from the source to the destination. So for example, if there is an edge from A to B, and the traversal reaches B, then A will not be added by B.
- We also included an *unordered_map <Vertex,Vertex>*, where the key was a vertex added to the traversal, and the value was the vertex responsible for adding it. This map allowed us to backtrack once we reached our endpoint, to find the shortest path.

Conclusion:

- Looking at our results from our traversal we were able to notice some interesting stuff about our dataset:
  - All vertices are part of the same “island”.
  - When putting in random start and end vertices, we never got a path longer than 4.
  - This shows our graph to be highly connected.
- Results from PageRank also tell us something interesting about our data:
  - We noticed that the PageRank of vertices was higher for traders who received higher reviews than other traders rated by the user, but the correlation between pageRank and a user’s raw average rating wasn’t very strong.
  - PageRank also runs an average of 25 iterations which is interesting when knowing that large datasets can run an amount of up to around 40 iterations.

Conclusion


PageRank
0.006
0.005
0.004
0.003
0.002
0.001
0.000
-0.001
PageRank Value
-10
-8
-6
-4
-2
0
2
4
6
8
10
Average Score