CS 225

Fall 2022

Creating a Safer Bitcoin Experience with BFS, Prim's, and Force-Directed Graph Drawing

David Rodriguez (davidr6), Alex Romo (aromo20), Carlos Perez (carlosp4)

“You should direct address your proposed leading question.”
In short, our leading question was whether we could create a safer platform for buying, selling, and/or trading Bitcoin due to the anonymity behind any transaction.

"How did you answer this question? “
Our solution was to create a ranking system that holds more trusted sources to a higher degree and less to a lower. We accomplished this by using the Bitcoin OTC trust-weighted signed network dataset and applied three different algorithms to aid us in its implementation.

BFS Traversal used in our makeAverageMap() function traversed through the data of our unordered map data structure that held our dataset and created a new unordered map that stored all Bitcoin members and their average rank. This was a huge factor in tackling our leading question as it created a data structure that we could directly connect to our main file where a user can search for a Bitcoin member and see their average ranking as long as that member is in the dataset.

Prim’s algorithm took our original unordered map and highlighted the members most prone to fraudulent and risky activity based on a starting member.

Finally, the Directed Graph Drawing plotted our original unordered map, with circles representing the vertices and lines the edges.

We believe that our project as a whole covers many bases for creating not only a safer platform, but a simpler one. It allows the users to see the most trusted sources, be aware of the most dangerous, and offers a simple and elegant way of being of visualizing it all.

What did you discover?

While implementing all the algorithms was difficult, the hardest part (as always) was debugging and testing them on our dataset. We realized that the best and most underappreciated aspect of a project is testing. Creating test cases for our algorithms was a long and grueling hassle, but in the end, removed a larger margin of error than we could’ve had.

BFS Traversal
Our graph implemented nodes ready with the information we needed during the data processing into our data storage stage, so no changes were made for input into our BFS traversal, as we used node input and tracking onto a queue. The output of our BFS traversal was a queue filled with our best-ranked Bitcoin member node at the front and worst at the

end, and when used we would start with the top of the queue as our searches would be based on best ranks. This made it possible to create the makeAverageMap() algorithm which created a new unordered map that stored all rated members and their average ranking from all received ratings.

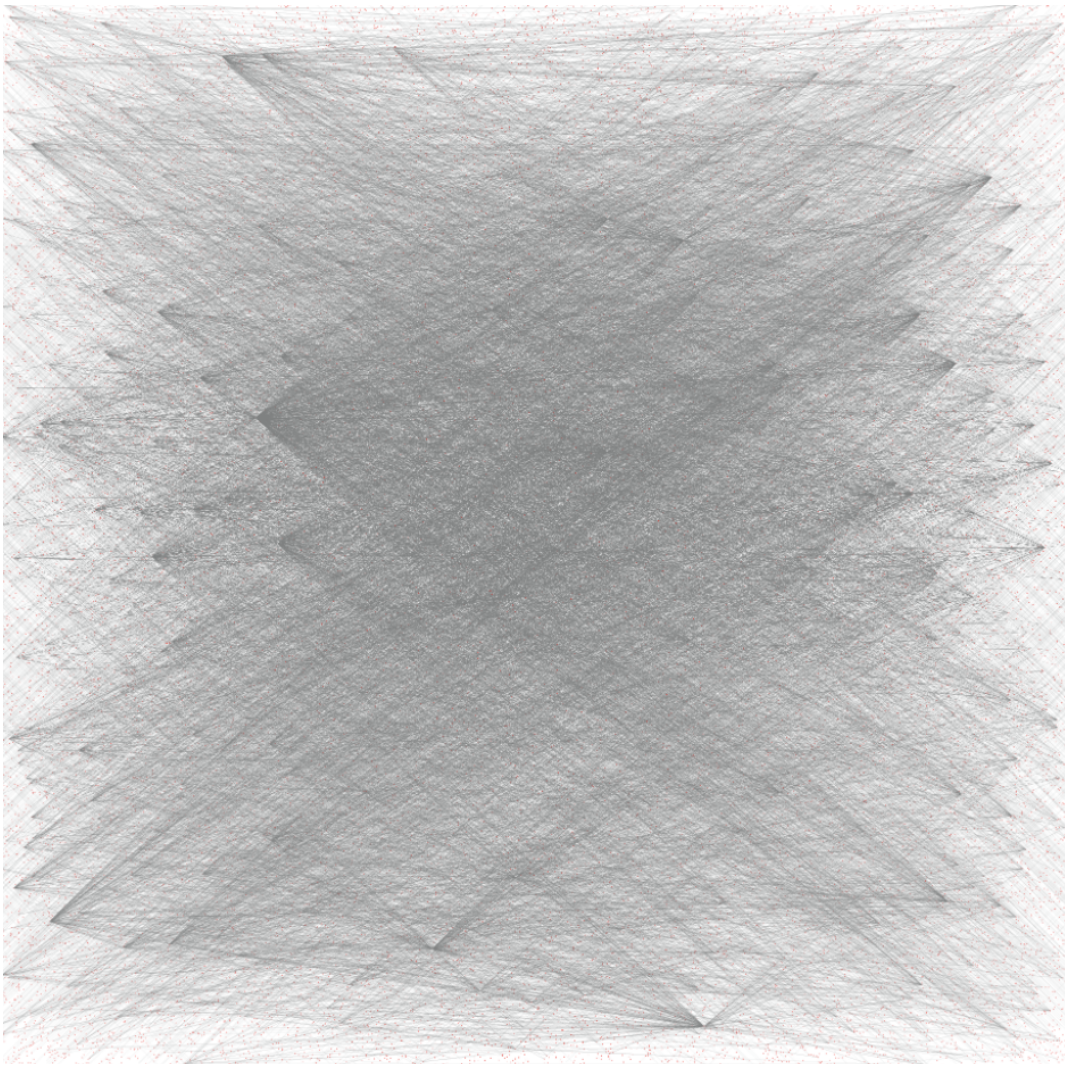
We each split off and looked through the CSV file for our project to create our test cases. We each chose some number of members and wrote down the rating given to them by other members. After that, we created test cases utilizing that data to test the BFS along with the averages of each of the members we selected.

Prim's Algorithm for Minimum Spanning Tree

Creating and running Prim's gave us a different perspective on visualizing our dataset, as we took the nodes and edges of the data set and found a tree made of the subtree of edges that depicts the minimized weight of all the edges. The weight in our dataset is the rankings, and this tree highlighted the Bitcoin members that might be the riskiest to engage in transactions. The creation of Prim's made it easier to apply the information and concept into our last algorithm, Force-Directed Graph Drawing.

To test the efficiency of our Prim's Algorithm for Minimum Spanning Tree, we inputted the data from the Unordered Map structure to produce a Prim output. From there, we took the prim output and ran test cases we made from the CSV file. Once we passed them all, we picked a vertex from which a small portion of the Prim graph was made. To make sure the newer, smaller output was correct, we manually compared it against that specific vertex according to the CSV data we collected.

Directed Graph Drawing



The image above displays our original unordered map, with circles representing the vertices and lines at the edges.

Testing this algorithm was a little difficult due to the scale of our CSV file. When we were first starting to test the way the drawing operated, we made small circles that would be filled with a vertex pointing to all the other users who had rated them. Our first showed our vertices but no lines connecting them. Upon inspection, we saw that our random plot was mapping points outside the range of what we wanted, so by adding some bounds we were able to see the connections between each user's rating. We also had a "floating" line that didn't connect to anything due to vertices that might have been outside the PNG, for this we went back and added a while look that would reselect random x and y values until they were in the PNG. Other bounds added were to tackle the issue of circles overlapping, to fix this we made sure that the x and y values chosen were not in bounds with existing pixels that were used to draw a vertices' circle.

Originally, we proposed a Force-Directed Graph Drawing algorithm but due to limited resources and time our algorithm created reflects a Directed Graph Drawing algorithm. Our algorithm includes the ratings of the forces that would potentially be used to develop a sub-algorithm that would attract or repel nodes from each other to turn our algorithm into a Force-Directed Graph Drawing which would be a point of further expansion and improvement for this project.

Further steps for improvement would be the design aspect of the graph, in which size, color, and location can indicate trustworthiness such that a bigger, green, centered vertex has a higher rating, than an small, red, outlying vertex thus incorporating Prim's algorithm such that the final image would show trustworthy ratee and raters as to inform the user if a rating is also reliable.