EE232E Project 5: Graph Algorithms

Group Member:
Fangyao Liu (204945018)
Chaojie Feng (505025111)
Haitao Wang (504294402)
Xiao PENG (005033608)

Question 1: Provide an upper and lower bound on ρij . Also, provide a justification for using log-normalized return (ri(t)) instead of regular return (qi(t)).

Obviously, the upper bound for rho is 1 and lower bound is 0. If rho is 0 then two stocks i, j are not related at all, if rho is 1 then two stocks i, j are completely accompanying with each other. The justification for using log-normalized return instead of regular return is due to the time-additive property of logarithm. The regular return over a period is multiplying and in logarithm scale it is adding, which keeps the distribution to be normal.

Question 2: Plot the degree distribution of the correlation graph and a histogram showing the un-normalized distribution of edge weights.

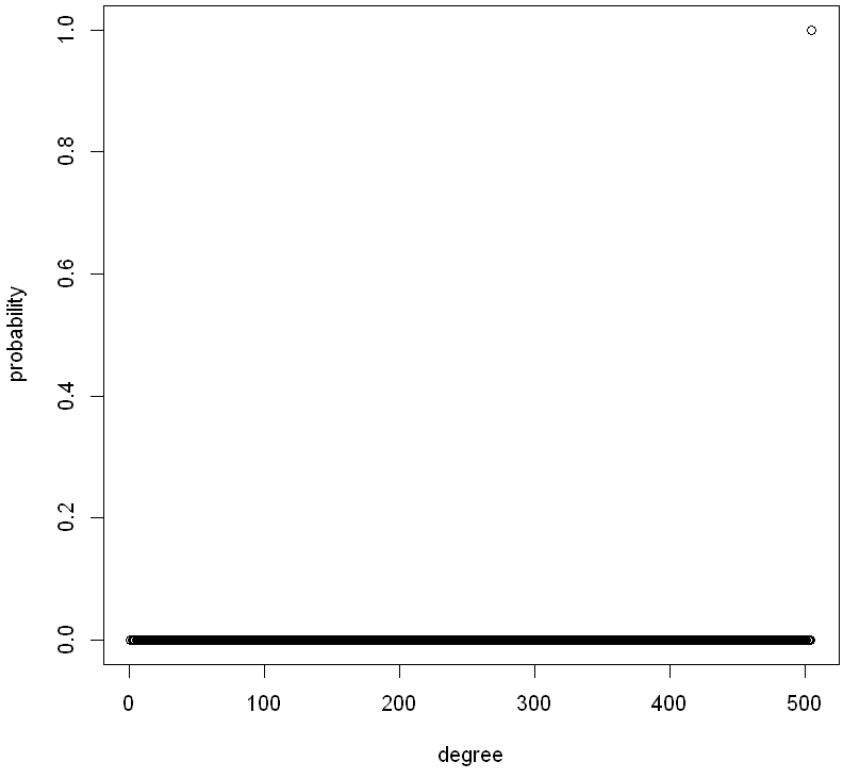


Figure 1. Degree distribution of correlation graph

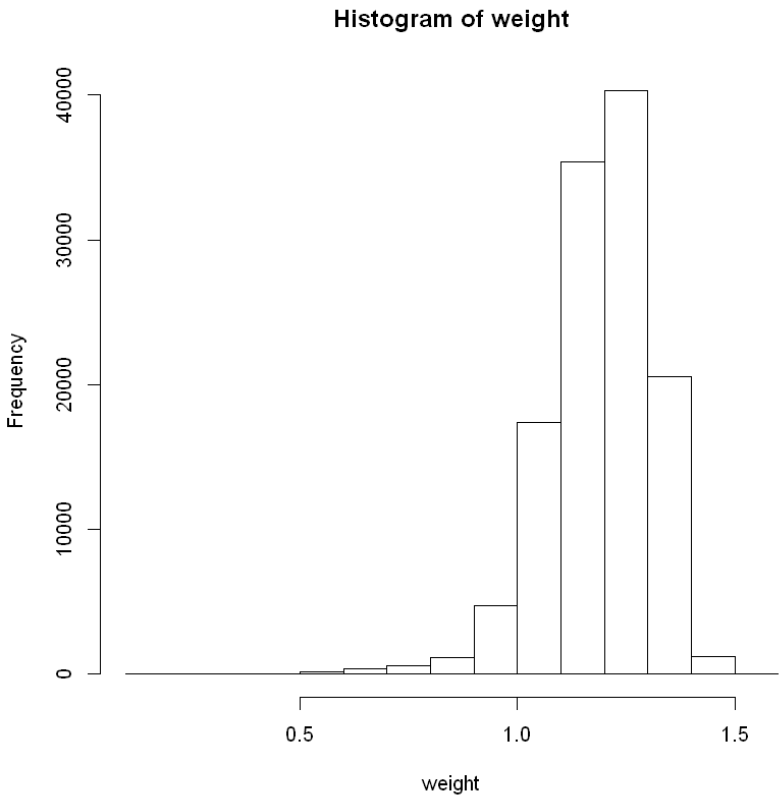


Figure 3. Histogram of weight

Question 3: Extract the MST of the correlation graph. Each stock can be categorized into a sector, which can be found in Name_sector.csv file. Plot the MST and color-code the nodes based on sectors. Do you see any pattern in the MST? The structures that you find in MST are called Vine clusters. Provide a detailed explanation about the pattern you observe.

It can be seen from the graph that the stocks belonging to the same sector tends to clusters together, which means they are more likely to be correlated to each other. Energy, Utilities and Consumer Staples tends to cluster in an one community, meaning that the sector is more independent, and other sectors tend to cluster but spread throughout the graph, meaning that the sector is affected by other sectors. It also makes sense that the correlation for different sectors. If one sector community is close to the other, these two sectors are more correlated. For example, Energy and Industries are strongly correlated, Utilities and Real Estate are also strongly correlated.

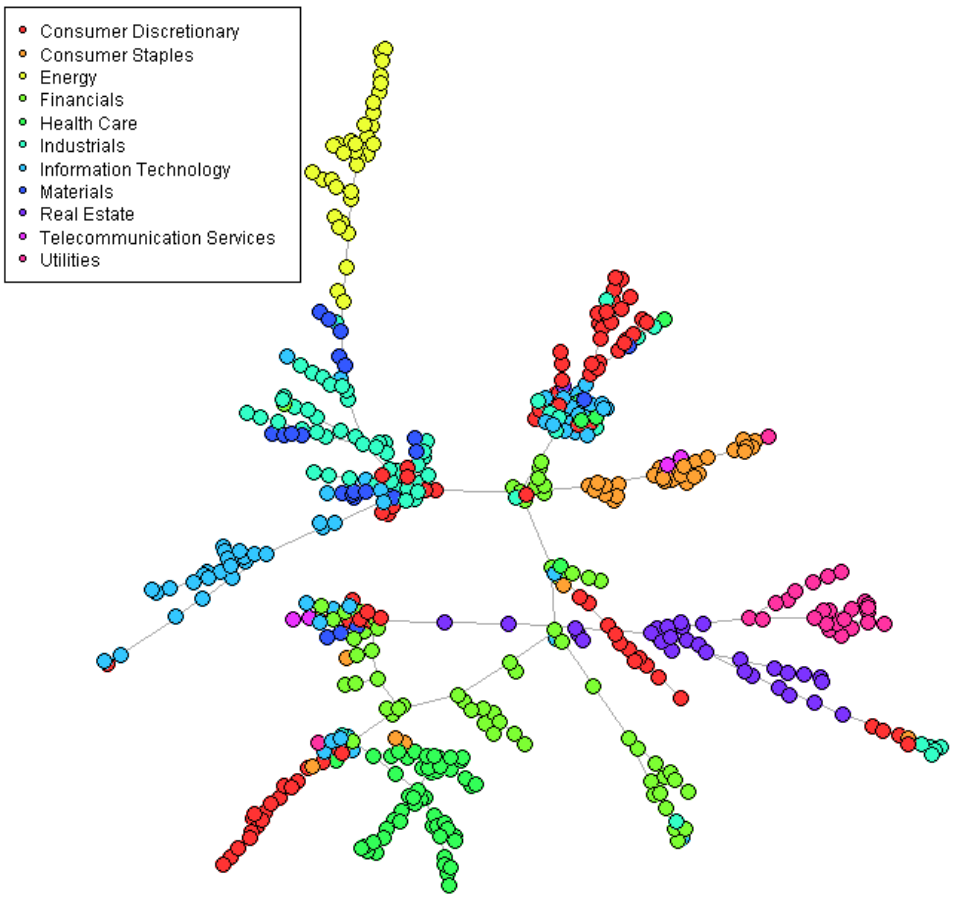


Figure 4. MST of stock correlation graph

Question 4:

According to the question, we have two definitions for the probability of a vertex belonging to a certain sector. The first one uses the quotient of the amount of the neighbors belonging to the same sector and the amount of the neighbors, as the estimation for the probability.

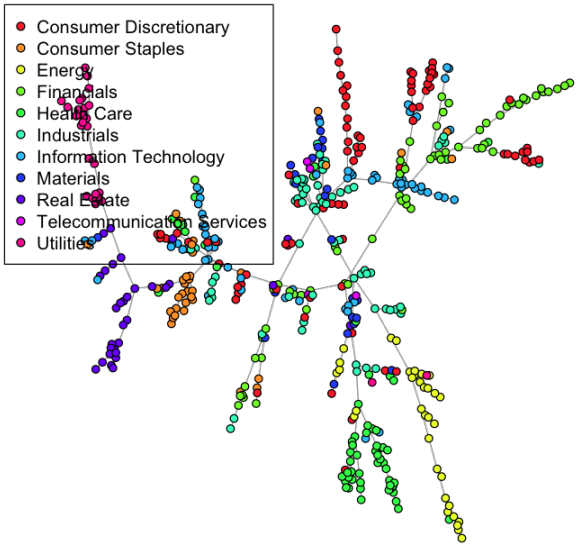
The alpha metric is the sum of all the probabilities divided by the amount of the vertices. Thus we have two alpha results due to the different definitions for the probability. We obtain 0.8289301 and 0.1141881 as the two alpha, respectively.

Interpretation: alpha1 is higher than alpha2, because alpha is a measurement for the performance of the MST algorithm while alpha2 is irrelevant to MST. Alpha2 is only related with the sectors distribution itself.

Question 5

We apply the "weekdays" and "as.Date" function to extract all the data on Monday. After that, we calculate the Pearson correlation coefficient matrix similarly as what we do in questions above. Also, we extract the MST from the correlation graph as shown in the figures. From the figure, we can come to the conclusion that, the pattern showing in the figure is basically same with what we observe in the MST based on the daily data. However, the pattern is less obvious to be figure out since the vertices in the same sectors (colors) are not divided as well as they are in Question 3.

This reasonableness of this observation can be boosted by applying the alpha metric defined in Question 4, to measure the performance of predicting of the market sector for an unknown stock. We can find that we only get 0.7429696 and 0.1141881, respectively, for the MST based on the weekly data, which are relatively lower than what we got for the daily data.



Part 2 Let's Help Santa

In this part, we are going to take advantage of "Uber Movement" data to help Santa find an optimal path to traverse all the places in shortest time.

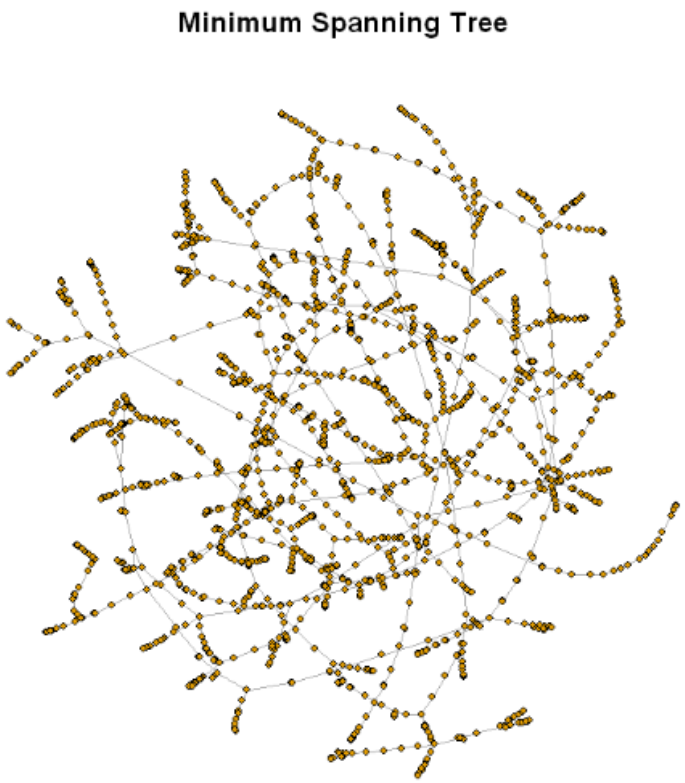
Question 6:

We will first utilize csv file to build a graph of San Francisco area and weights are set by mean_travel_time. Then we process the json file to add attributes like street address and coordinates to vertices. Following, we extract giant connected component of this network and treat it as the graph G where we are going to find the optimal path.

The number of nodes is 1880 and the number of edges is 311802

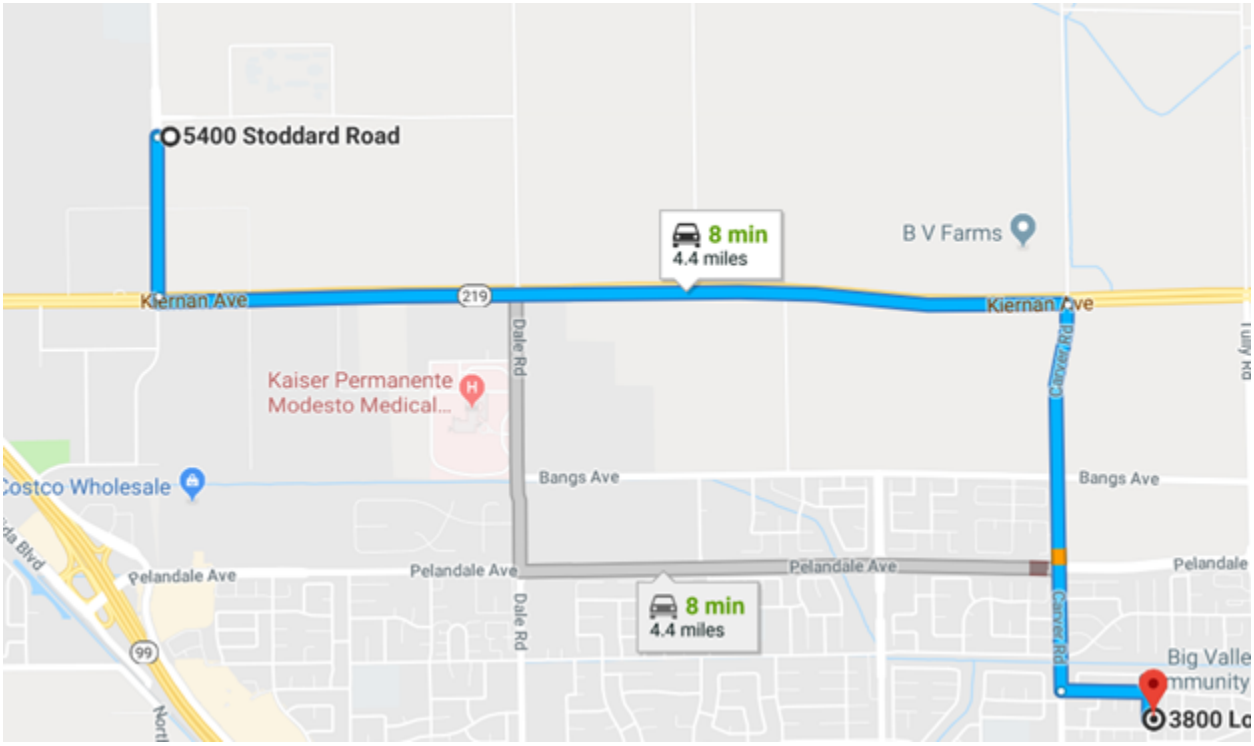
Question 7:

Second, we will build a minimum spanning of graph G. We call mst function on G and get the plot below

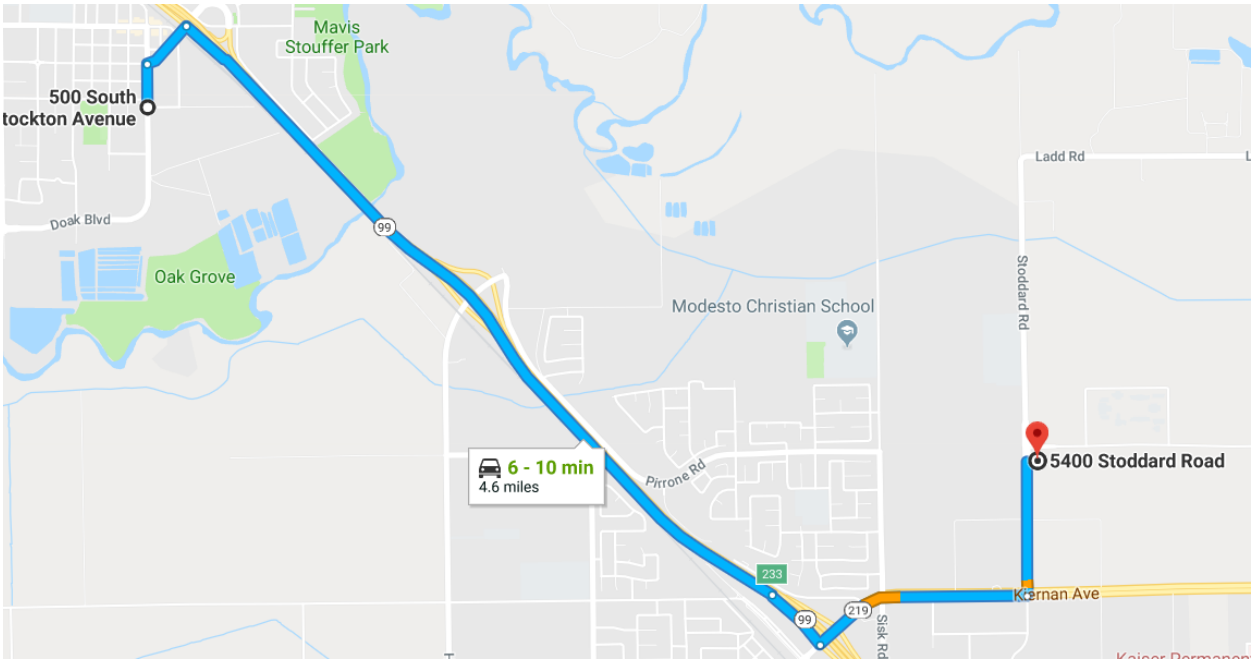


From the plot, it seems all the components are connected. We check the vertex number and edge number. It has 1880 vertices and 1879 edges, which satisfy our expectation of a spanning tree. Then we will check whether this tree is “intuitively minimum”. We select three edges as an example. We input addresses into google map and get their travelling time. Since google map doesn’t provide us an average travelling time option, we will present the current travelling time(around 8 PM) of these three edges:

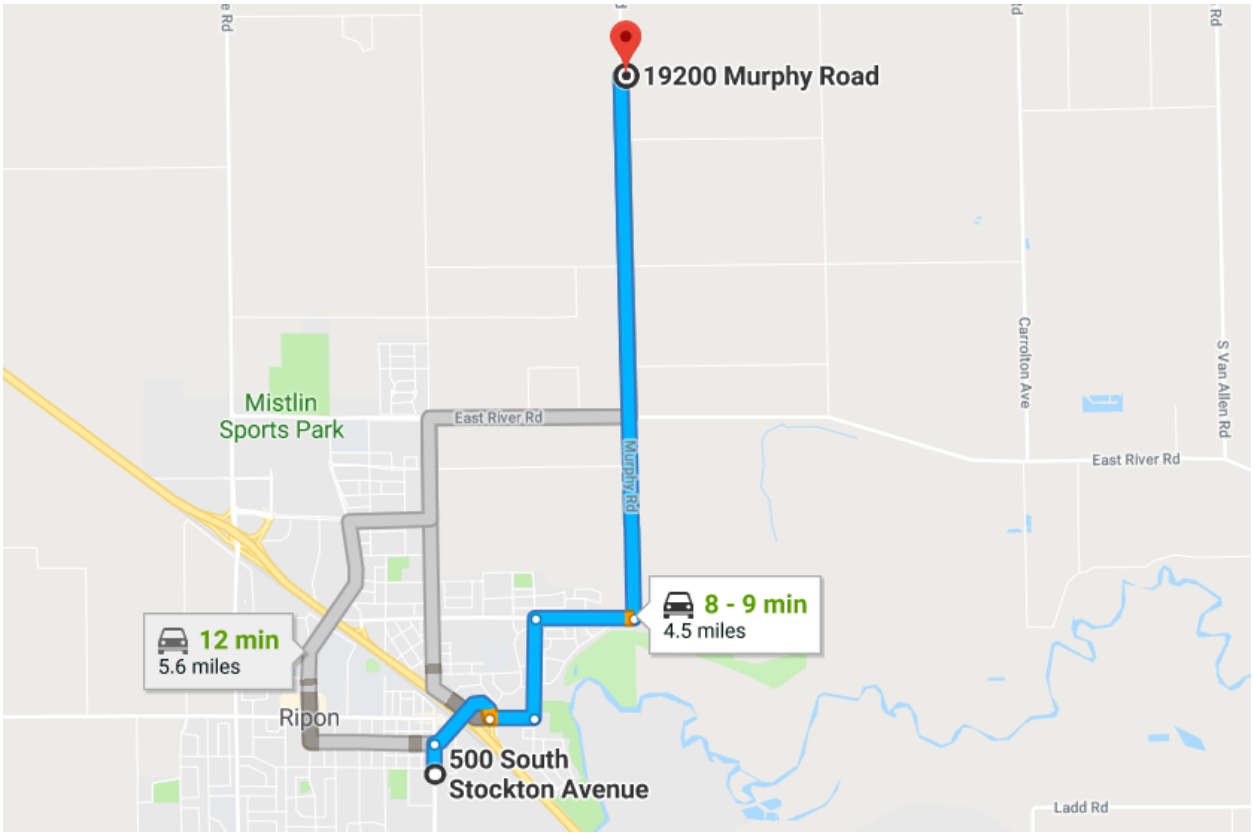
374(3800 Longbridge Drive, Modesto) to 2076(5400 Stoddard Road, Modesto)



2076(5400 Stoddard Road, Modesto) to 2446(500 South Stockton Avenue, Ripon)



2446(500 South Stockton Avenue, Ripon) to 2328(19200 Murphy Road, Ripon)



As we can see here, the travelling time are all around 300~500 seconds. This is a pretty small value in our dataset. Therefore, intuitively, we can believe our result is a valid minimum spanning tree.

Question 8:

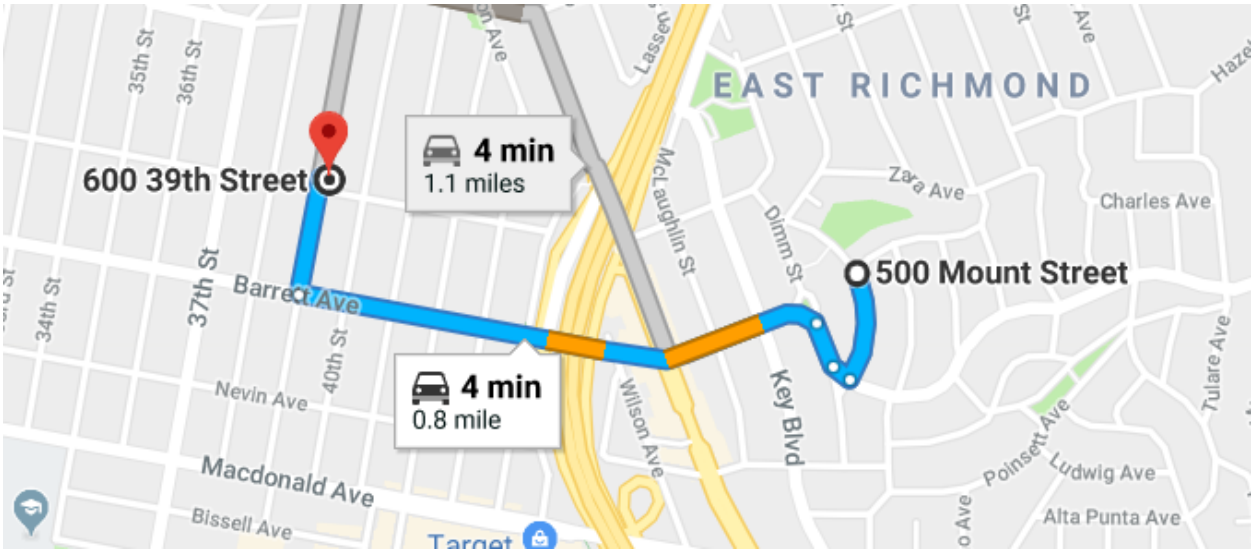
As a prerequisite of using travelling salesman approximation algorithm, triangle inequality must be satisfied. Here we sample 1000 triangles and test whether we can apply approximation algorithm on the graph. The result shows that 94.1% triangles satisfy triangle inequality. Therefore, with this percentile, we can ensure that approximation algorithm will have a good performance on our network.

Question 9:

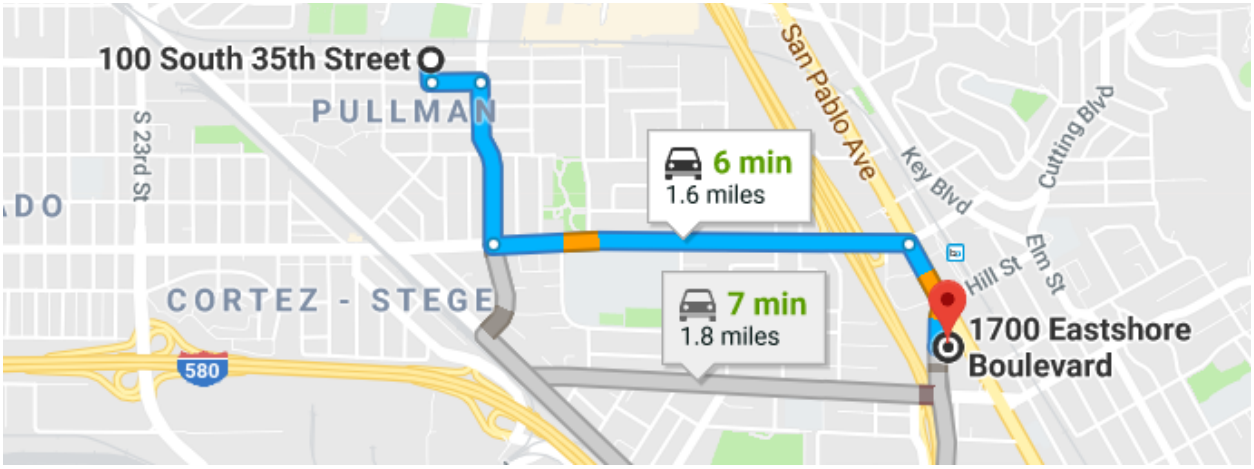
Here we will apply deep first search algorithm on the minimum spanning tree to find a tour. DFS is going to return us a sequence of vertices and we will compute the cost of this sequence. This result is an approximated solution of travelling salesman problem. This is called 2-approximate algorithm. Then, we regard the sum of weights of our MST as an approximation of "optimal TSP cost". After computing both of them, we get $\rho = 1.65558116444551$. This result satisfies the theorem that approximated solution cost will be greater than sum of weights of MST and less than twice the sum of weights of MST.

Then we choose two edges to see whether this solution is intuitive or not.

102 (500 Mount Street, East Richmond, Richmond) to 1651 (600 39th Street, North & East, Richmond)



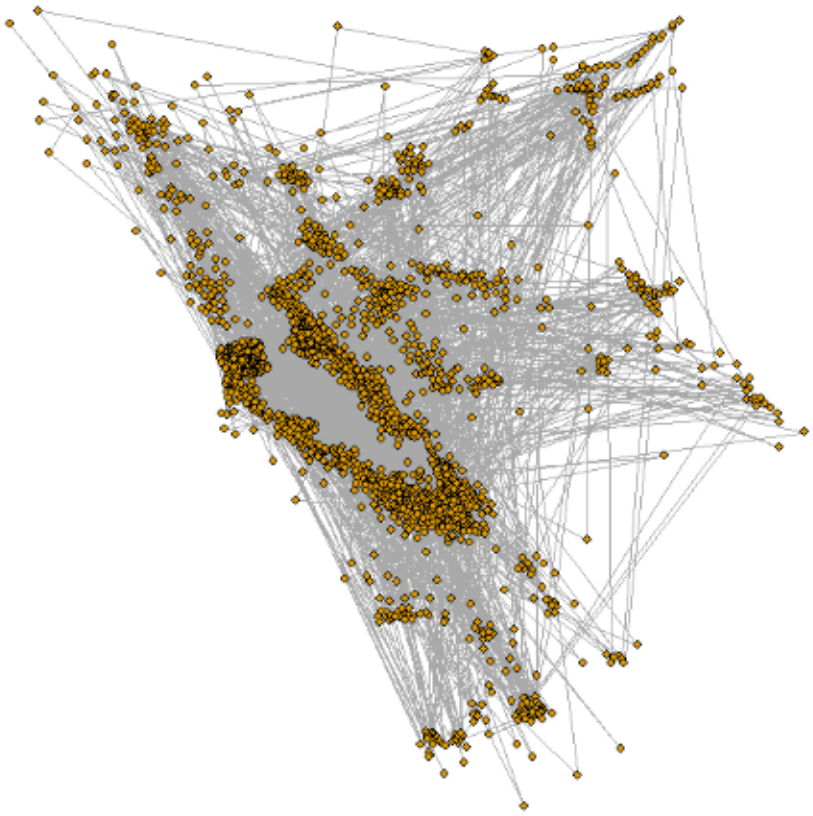
1655(100 South 35th Street, Pullman, Richmond) to 1047(1700 Eastshore Boulevard, El Cerrito)



As we can see here, the weight is pretty small and therefore our solution is intuitive.

Question 10:

In this question, we visualize the our approximate solution. We plot the graph with vertices' coordinators as layout and get the plot below.



Since it involves too many edges, we plot another graph without coordinator layout:

3. Analyzing the Traffic Flow

Section 3 is continuation of section 2, so we only use the nodes existing in the greatest connected component as the nodes we are considering in this section.

Question 11

First, we want to build a road mesh based on the nodes we derive from section 2. We simply extract the IDs for nodes in the gcc, and find the corresponding coordinate of each node using the node ID and put all coordinated into a list. Then, apply Delaunay Triangulation algorithm to the list of coordinates to generate a new graph. The graph is what we are looking for, which is illustrated in Figure 3.11.1. According to Delaunay Triangulation algorithm, no point in the triangulation graph is inside the circumcircle of any triangle. The Delaunay triangulation also maximizes the minimum angle of all the angles of the triangles in the triangulation. In Figure 3.11.1, we can see that there is no node inner any triangle, as proved by the algorithm. The subgraph GΔ induced by the edges produced by the triangulation is shown in Figure 3.11.2. The graph GΔ is simplified by removing the repeated edges.

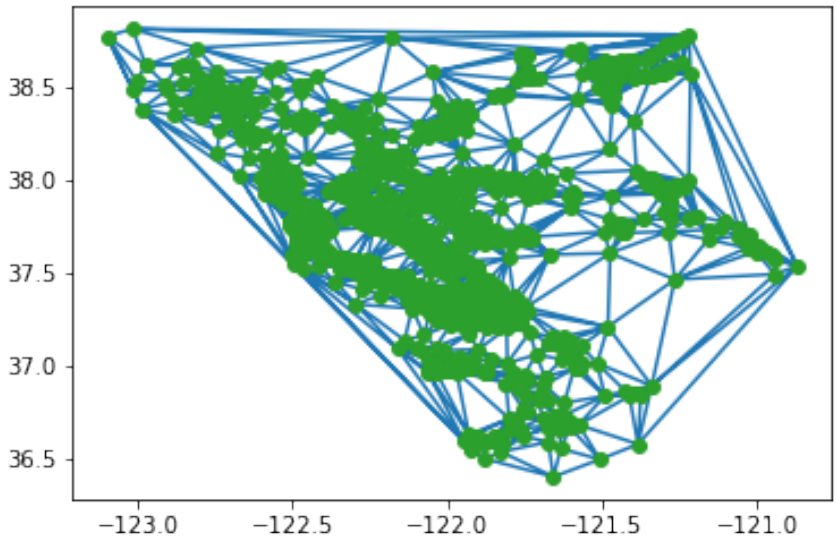


Figure 3.11.1: Road mesh

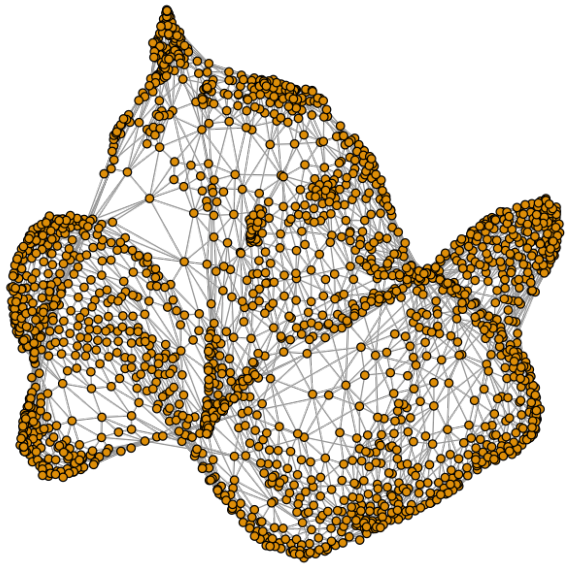


Figure 3.11.2: Subgraph GΔ

Question 12

Here, we want to find the road traffic maxflow in Question 13, so we need to assign weight for each edge in the graph GΔ. The edge is the created road, while the weight is maximum number of cars driving though the road in an hour. Here are some assumptions which we are following:

- Each degree of latitude and longitude ≈ 69 miles

- Carlength≈5m=0.003mile
- Cars maintain a safety distance of 2 seconds to the next car
- Each road has 2 lanes in each direction

First, we can compute the mean car speed (in mile/sec) of each road by dividing the road length (in mile) by mean travel time (in second). The mean car flow (in cars/hour) is computed as:

$$Car\ Flow = \frac{speed \times 3600}{(0.003\ +\ (speed\ \times\ 2))}$$

Question 13

Since the weight represents car flow, we can directly use the function "max_flow()" to compute the maximum number of cars that can commute from Stanford to UCSC. The maxflow is 15014. There are, in total, five edge-disjoint paths between Stanford and UCSC. In the road map in Figure 3.13.1, we can see that there are clearly 5 edge-disjoint paths from Stanford to UCSC.

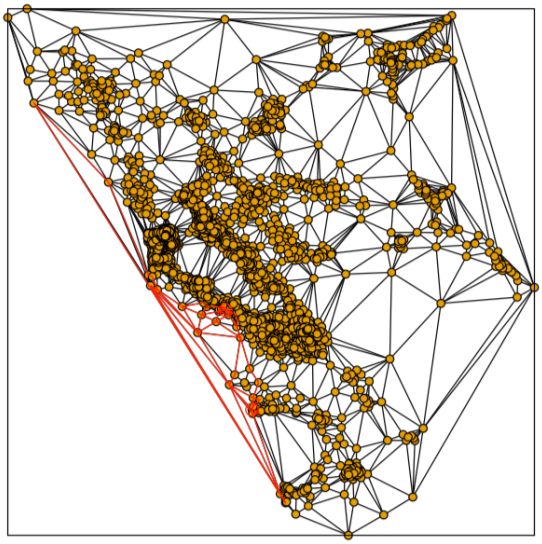


Figure 3.13.1: Road map

Question 14

Comparing with the real map of the San Francisco, we can find many fake edges in the plot of G_Δ, such as the edges through the sea, and the edges across the mountains. In order to trim those fake edges, we come up with a threshold for the travel time of the roads. We delete all the edges with the weight higher than the threshold. Then we use the "layout.norm" to plot the $\tilde{G}_\Delta$ on real map coordinates.

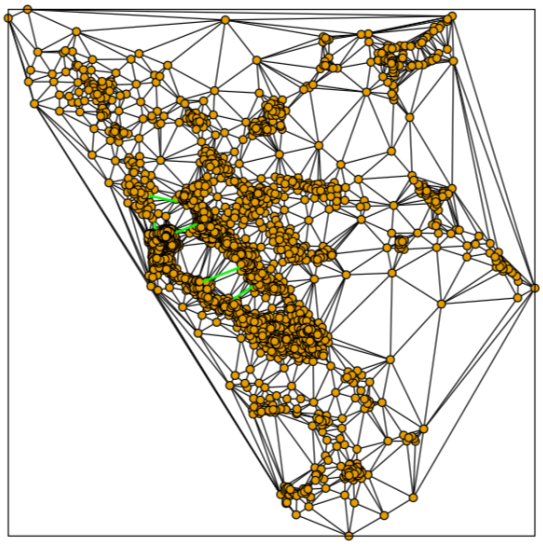


Figure 3.14.1: The G_Δ on the real coordinates

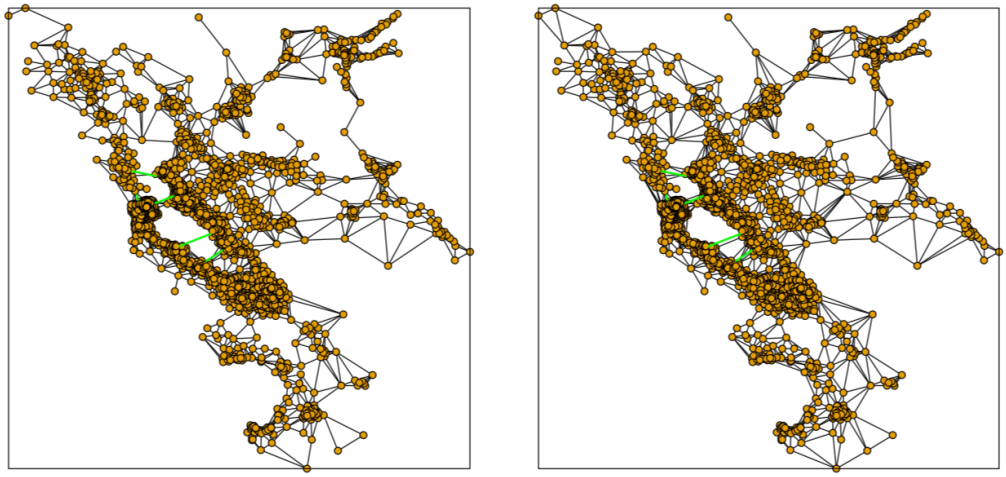


Figure 3.14.2: The $\tilde{G}_\Delta$ on the real coordinates when threshold is set as 1700 and 2250, respectively

At the beginning, we use the distribution of the weight to set the threshold, however, the effect is not satisfying. Then we choose the threshold so that the $\tilde{G}_\Delta$ won't become disconnected after the thresholding. This rule of the thumb helps us to narrow down the choices among all the threshold. In addition, to evaluate the performance of the thresholding method, we can check some interest edges, like real bridges, roads across the mountain, on the plot for $\tilde{G}_\Delta$ .In this question, we use green lines to highlight five real bridges, Golden Gate Bridge, Richmond, San Rafael Bridge, San Mateo Bridge, Dumbarton Bridge, San Francisco - Oakland Bay Bridge. In order to preserve these real bridges on the plot, we sweep the threshold. And according to the performance, we select two typical thresholds as representatives here as shown in Figure 3.14.2. Comparing them with the original G_Δ, we can find a trade-off here. The threshold of 1700 can eliminate most fake edges while the San Mateo Bridge has been delete too. And the threshold of 2250 preserves all the real bridges with an imperfect filtering result.

We also would like to note that each green line stands for no edges in G_Δ but a group of the near edges. In other words, we may not find the bridges in the G_Δ, but the edges near each bridge certainly will go through this bridge.

Question 15

Since the weight represents car flow, we can directly use the function "max_flow()" to compute the maximum number of cars that can commute from Stanford to UCSC. The maxflow is 3221. There are, in total, five edge-disjoint paths between Stanford and UCSC.