Project 4 - Graph Algorithms

Aryaman Gokarn **UID:506303588**

Mugdha Bhagwat **UID: 606297799**

Tania Rajabally **UID: 806153219**

QUESTION 1:

What are upper and lower bounds on ρij? Provide a justification for using log- normalized return (ri(t)) instead of regular return (qi(t)).

We know that the log- normalized stock return $r_i(t)$ of stock i over a period [*t-1, t*] of is given by:

$$r_i(t) = \log(1 + q_i(t))$$

where

$$q_i(t) = \frac{p_i(t) - p_i(t-1)}{p_i(t-1)}$$

$q_i(t)$ = regular return of stock i over a period [*t-1, t]*
$p_i(t)$ = closing price of stock *i* on the t^{th} *day*

Correlation Calculation Explanation

The correlation formula given is:

$$\rho_{ij} = \frac{\langle r_i(t) r_j(t) \rangle - \langle r_i(t) \rangle \langle r_j(t) \rangle}{\sqrt{(\langle r_i(t)^2 \rangle - \langle r_i(t) \rangle^2)(\langle r_j(t)^2 \rangle - \langle r_j(t) \rangle^2)}}$$

where ($\langle \cdot \rangle$) denotes the temporal average over the investigated time period.

- ($\langle r_i(t) r_j(t) \rangle$) is the average product of the log-normalized returns of stocks (i) and (j).
- ($\langle r_i(t) \rangle$) and ($\langle r_j(t) \rangle$) are the average log-normalized returns of stocks (i) and (j), respectively.
- ($\langle r_i(t)^2 \rangle$) and ($\langle r_j(t)^2 \rangle$) are the average squared log-normalized returns of stocks (i) and (j), respectively.
- The denominator represents the product of the standard deviations of the log-normalized returns for stocks (i) and (j).

By using log-normalized returns, this correlation measure benefits from the properties of log returns, such as normality and additivity, making it more robust and reliable for financial analysis.

Upper and Lower Bounds on ρ_{ij}

By observing the numerator and denominator of the given defined correlation function ρ_{ij}, we can see the following relation:

$$\rho_{ij} = \frac{\langle r_i(t) r_j(t) \rangle - \langle r_i(t) \rangle \langle r_j(t) \rangle}{\sqrt{(\langle r_i(t)^2 \rangle - \langle r_i(t) \rangle^2)(\langle r_j(t)^2 \rangle - \langle r_j(t) \rangle^2)}} = \frac{COV(r_i(t), r_j(t))}{\sqrt{VAR(r_i(t)) VAR(r_j(t))}}$$

, which resemble the form of standard correlation coefficient $COR(r_i(t), r_j(t))$. Therefore, the upper bound of ρ_{ij} is +1 and lower bound of ρ_{ij} is -1 since the values of correlation coefficient $COR(r_i(t), r_j(t))$ can range from -1 to +1.

The correlation coefficient (ρ_{ij}) between two variables is a measure of their linear relationship and is identical to the Pearson's correlation coefficient, essentially providing a normalized measurement of the covariance.

The value of (ρ_{ij}) always lies within the range of -1 to 1, inclusive based on the Cauchy-Schwarz Inequality.

- **Upper Bound:** The upper bound of (ρ_{ij}) is 1.
- **Lower Bound:** The lower bound of (ρ_{ij}) is -1.

These bounds are a fundamental property of the correlation coefficient, which can be derived from the definition and properties of covariance and variance.

When (ρ_{ij} = 1), it indicates a perfect positive linear relationship between the two variable, with their prices altering in the same direction (though the scale of growth may be different).

Conversely, (ρ_{ij} = -1) indicates a perfect negative linear relationship. It means that the two stocks and are strongly negatively or inversely correlated, with the stock prices moving in opposite directions.

When (ρ_{ij} = 0), it indicates no linear relationship between the variables.

Justification for Using Log-Normalized Return (ri(t)) Instead of Regular Return (qi(t))

- **Data:** Log-normalized returns reduce the effects of outliers and variance in the data, constraining values within finite bounds. They also diminish any skewness present, providing a more normalized distribution. This method better represents relative changes in stock prices over absolute numbers, which is practical as percentage changes are consistent for stocks treated similarly. Additionally, log normalization amplifies the scale of smaller stocks and shrinks the scale of larger stocks, creating a homogeneous scale that allows for the visualization of relative changes across all stocks, regardless of their prices.

- **Normality and Statistical Properties:** Log-normalized returns often approximate a normal distribution better than regular returns. Many statistical methods, including those used in finance, assume normality. This makes log-normalized returns more suitable for further analysis and modeling.

- **Additivity Over Time:** Log returns are additive over time, whereas regular returns are not. For example, if you have daily log returns, the log return over a week is simply the sum of the daily log returns. This additive property simplifies the aggregation of returns over different time periods and is useful for portfolio management and risk assessment.

  If $ri(t)$=log(1+$qi(t)$), then the return over multiple periods is$\sum k$=1$Tri(t+k)$

- **Comparative and Consistent Scale:** Log-normalized returns provide a consistent scale for comparing different stocks, regardless of their price levels. Since the log function compresses the range of returns, it reduces the impact of extreme values and makes the returns of different stocks more comparable.

- **Handling of Small and Large Values:** The log transformation can handle a wide range of values more effectively. For instance, large changes in price are scaled down, and small changes are scaled up, which helps to manage heteroscedasticity (non-constant variance) in financial time series data.

- **Multiplicative Processes:** Financial prices typically evolve according to multiplicative processes. The log transformation converts multiplicative changes into additive changes, which are easier to analyze and interpret statistically.

- **Proportionality:** Log returns reflect proportional changes in price. For instance, a 1% increase in price has the same log return regardless of whether the price increases from $100 to$ 101 or from $10 to$ 10.10. This is not the case with regular returns, which are influenced by the absolute price level.

QUESTION 2:

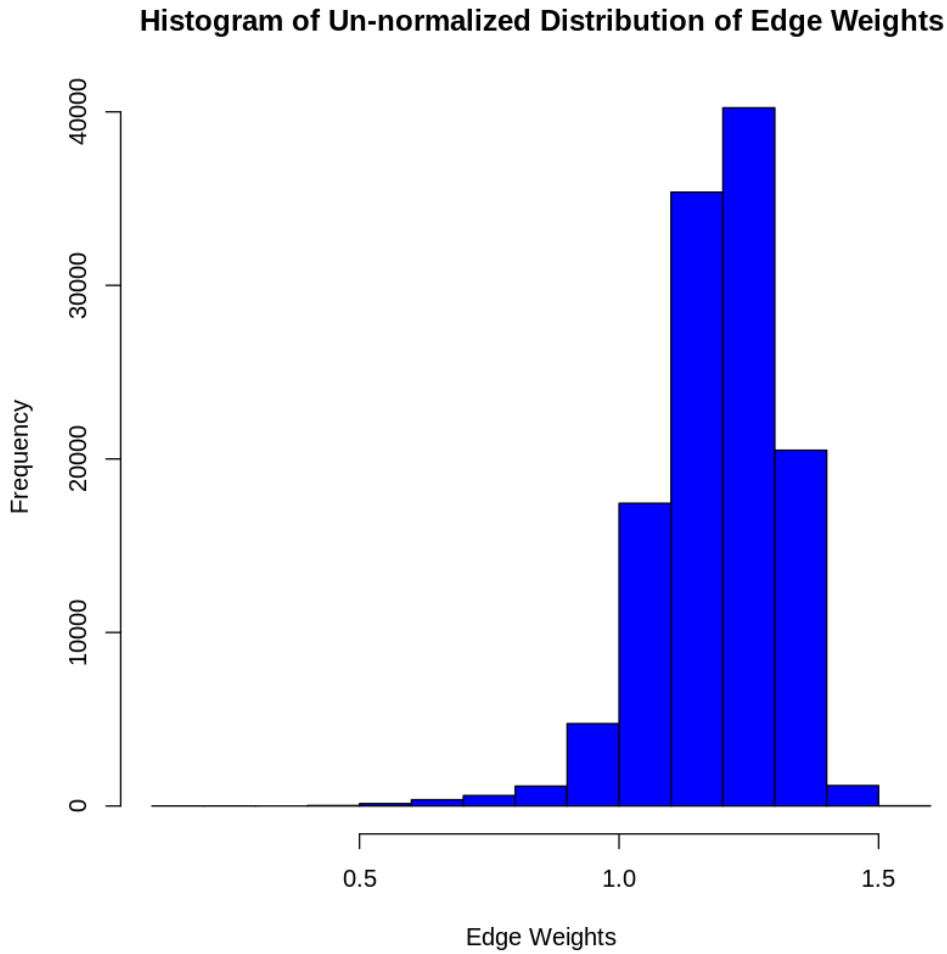
Plot a histogram showing the un-normalized distribution of edge weights.

As described - the correlation graph consists of the stocks as vertices and the edge weights as follows:

$$w_{ij} = \sqrt{2(1 - \rho_{ij})}$$

- If ρ_{ij} is +1 (strong positive and perfect correlation), $w_{ij} = 0$.
- If ρ_{ij} is -1 (strong negative correlation), $w_{ij} = 2$.

Thus, w_{ij} should be between 0 and 2. To construct the graph, we first omit the stocks with missing data (NaN values) and only keep those stocks with 765 entries. We then calculate vector of $r_i(t)$, containing log-normalized return for each stock *i*. Afterwards, we create a file to store the w_{ij} (calculated using the formula shown above) for each stock *i* and *j*, where stock *i* is the source node and stock *j* is the sink node. We use the igraph library in R for generating the graph. Figure 1 shows the histogram plot.

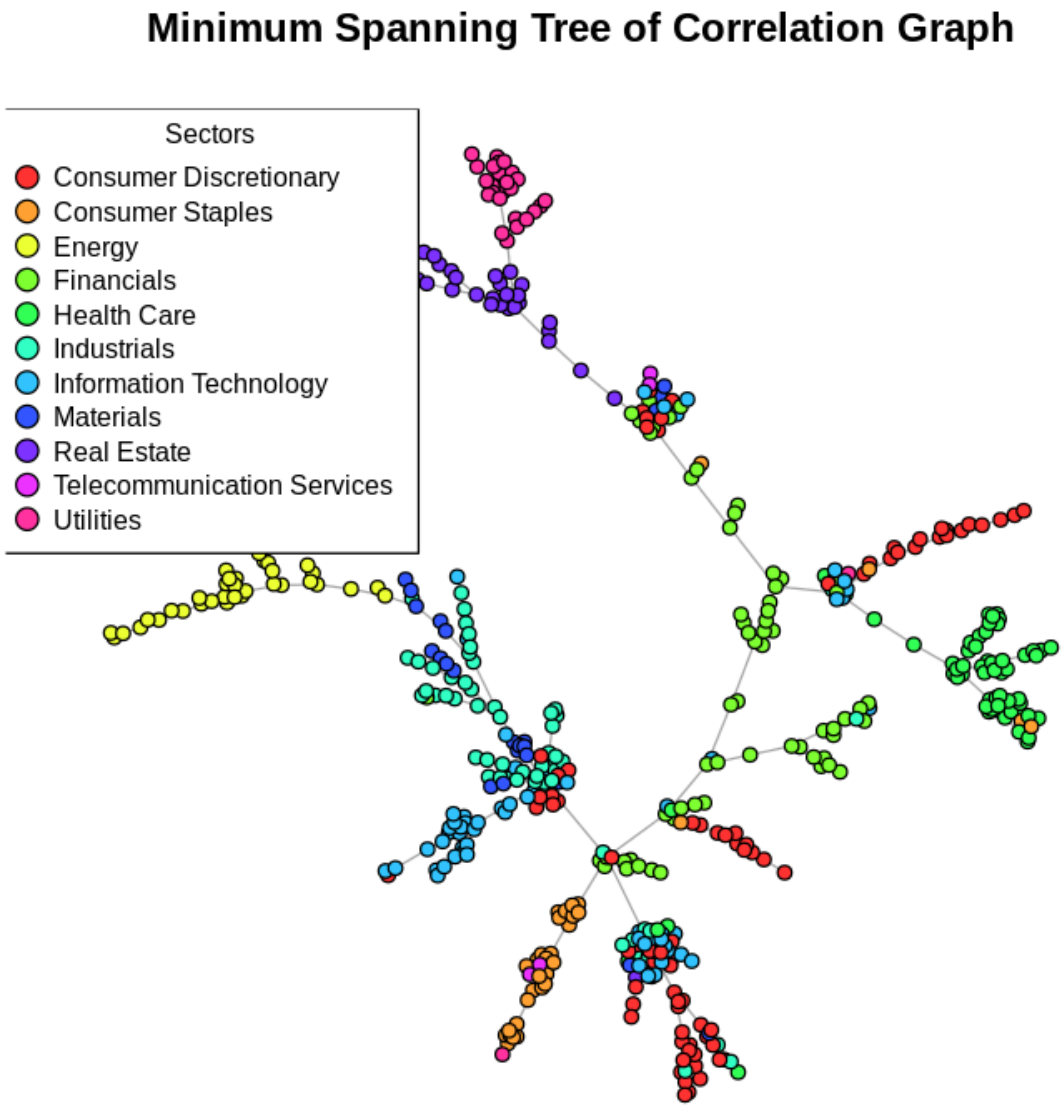


From the above figure we observe, we observe that most of the edge weights are between 1.0 and 1.4. As expected, all the weights lie between 0 and 2, however, we observe that the lower bound for edge weights is 0.5 and not 0. This means that there are no stocks with perfect positive correlation, as this would require some of the edge weights to be 0. Most of the stocks are weakly correlated with each other, and majority of the stocks have negative correlation with each other.

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.

We know that the MST connects all the nodes of a connected and edge-weighted undirected graph without any cycles and with the lowest possible cumulative edge weights. We use Prim's algorithm to create the MST from the correlation graph. The below figure shows the MST for the correlation graph, with stocks belonging to the same sector color coded the same.



Based on the correlation graph depicted above, several patterns can be observed as follows:

1. Firstly, stocks within the same sector, represented by the same color, tend to exhibit a tendency to cluster together in the MST. This is in line with expectations, given their high correlation, resulting in connections with minimal edge weights on the graph. Secondly, stocks belonging to distinct sectors, indicated by different colors, are generally not connected to one another and exhibit significant edge weights.

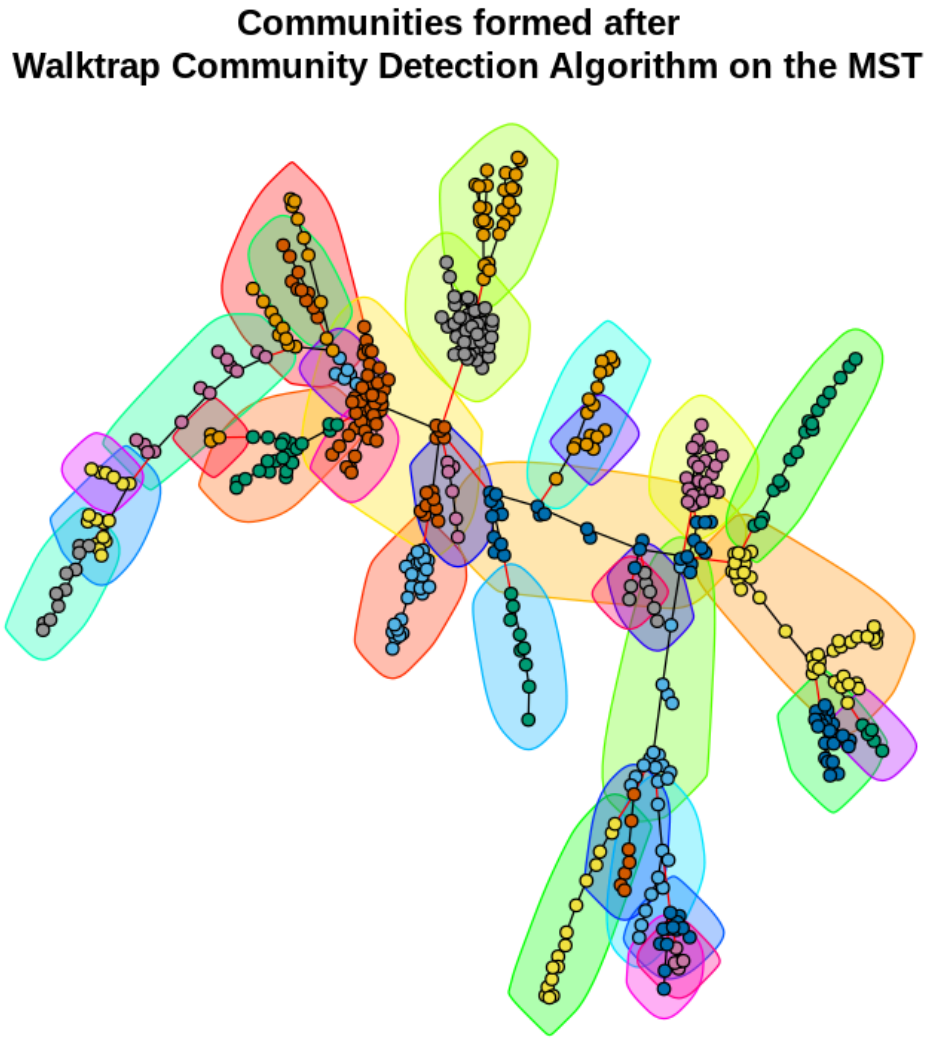
2. In other words, stocks that are highly correlated tend to be connected on the correlation graph with the least possible edge weights (the higher the correlation ρ_{ij} between stocks, the lower the edge weights w_{ij}, as $w_{ij} = \sqrt{2(1 - \rho_{ij})}$. On the other hand, the stocks which are not strongly correlated have large edge weights.

3. The goal of a Minimum Spanning Tree (MST) is to connect all nodes using the edges with the lowest cumulative weights. This process naturally groups highly correlated stocks together into clusters, which are often referred to as **vine clusters** because they resemble grapes hanging from a vine. These clusters typically represent distinct sectors of the market. Stocks within the same cluster tend to move in the same direction, suggesting they may require similar investment strategies. Vine clusters offer a snapshot of the stock market over longer time periods, such as daily data, highlighting the relationships and sectoral connections among different stocks.

QUESTION 4:

Run a community detection algorithm (for example walktrap) on the MST obtained above. Plot the communities formed. Compute the homogeneity and completeness of the clustering. (you can use the 'clevr' library in r to compute homogeneity and completeness).

We ran the Walktrap algorithm on the MST obtained above to get the following results:



The Homogenity and Completeness scores of the clustering are:

Number of clusters : 33
Homogenity : 0.682645
Completeness : 0.479284

QUESTION 5:

Report the value of α for the above two cases and provide an interpretation for the difference.

We evaluate the two sector clustering techniques of an unknown stock vi in the MST using the methods:

1. $$P(v_i \in S_i) = \frac{|Q_i|}{|N_i|}$$
2. $$P(v_i \in S_i) = \frac{|S_i|}{|V|}$$

where, Si is the sector of node i, Qi is the set of neighbors of node i that belong to the same sector as node i, Ni is the set of all neighbors of node i, V is the set of all vertices of the MST. The performance metric for each of the methods is as follows:

$$\alpha = \frac{1}{|V|} \sum_{v_i \ inV} P(v_i \in S_i)$$

The higher the value of , the more effective is the clustering technique.The values of α for the two cases are:

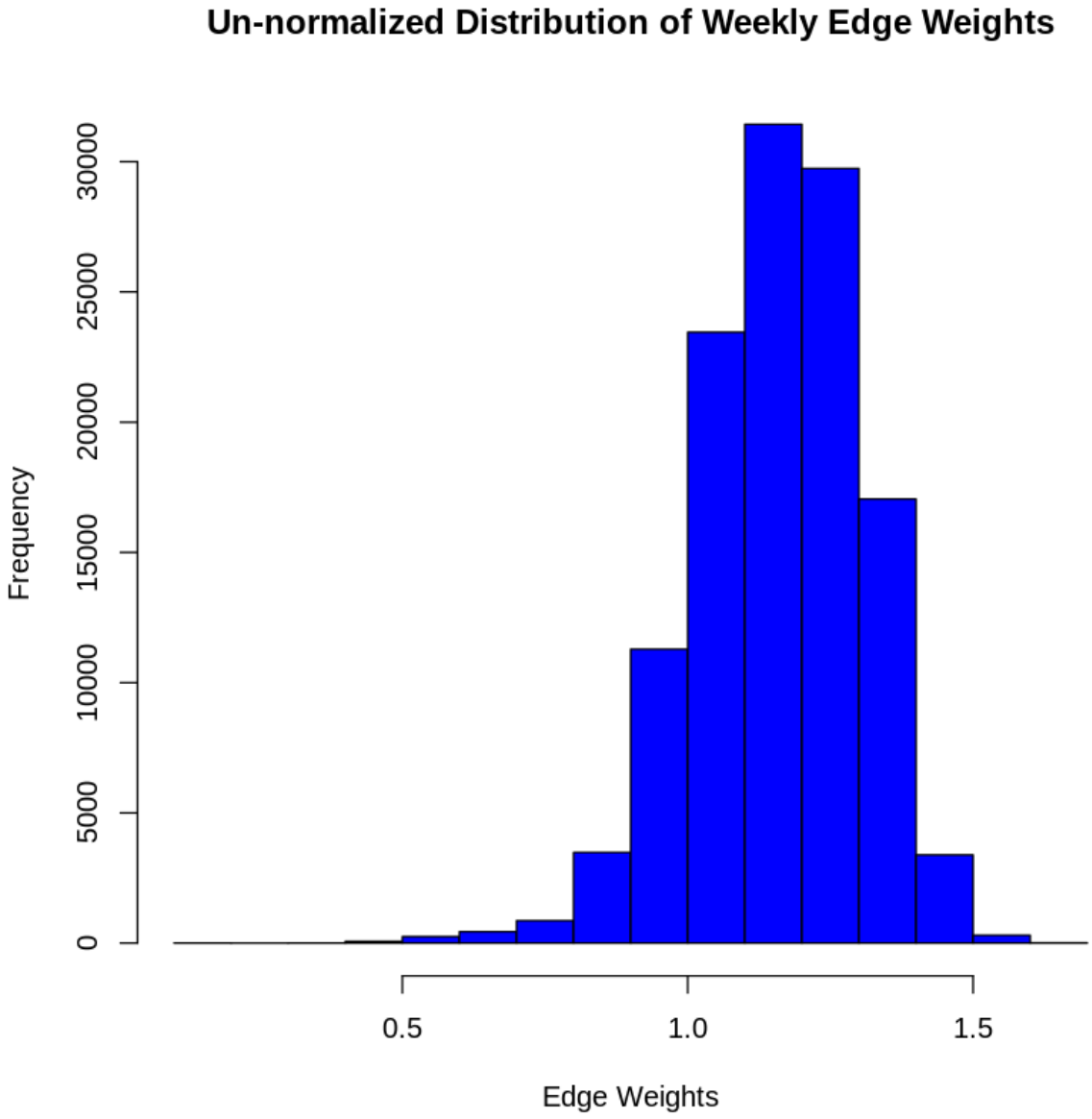
Values of first alpha based on Qi, Ni : 0.828930
Values of second alpha based on Si, V : 0.114188

$\alpha 1$ is around 0.83 and $\alpha 2$ is around 0.11. One can see that $\alpha 1$ is much higher than $\alpha 2$, meaning that the first technique based on $P(vi \in Si) = |Qi||Ni|$ is way better compared to the second technique.

This outcome is anticipated because Case 1 leverages the vine cluster structures identified by the Minimum Spanning Tree (MST) within the correlation graph, focusing on nodes (stocks) that are highly correlated and cluster together. By utilizing the local connectivity among neighboring nodes, Case 1 makes precise decisions based on these strong, localized relationships. In contrast, Case 2 examines all nodes within a sector as a whole, overlooking specific local spatial connections or cluster formations. This global approach fails to differentiate between local clusters and treats the entire sector uniformly, resulting in only a broad, generalized probability estimate. Consequently, Case 2 lacks the precision provided by the detailed local analysis in Case 1, which accurately captures the behavior and relationships within the market.

QUESTION 6:
Repeat questions 2,3,4,5 on the WEEKLY data.

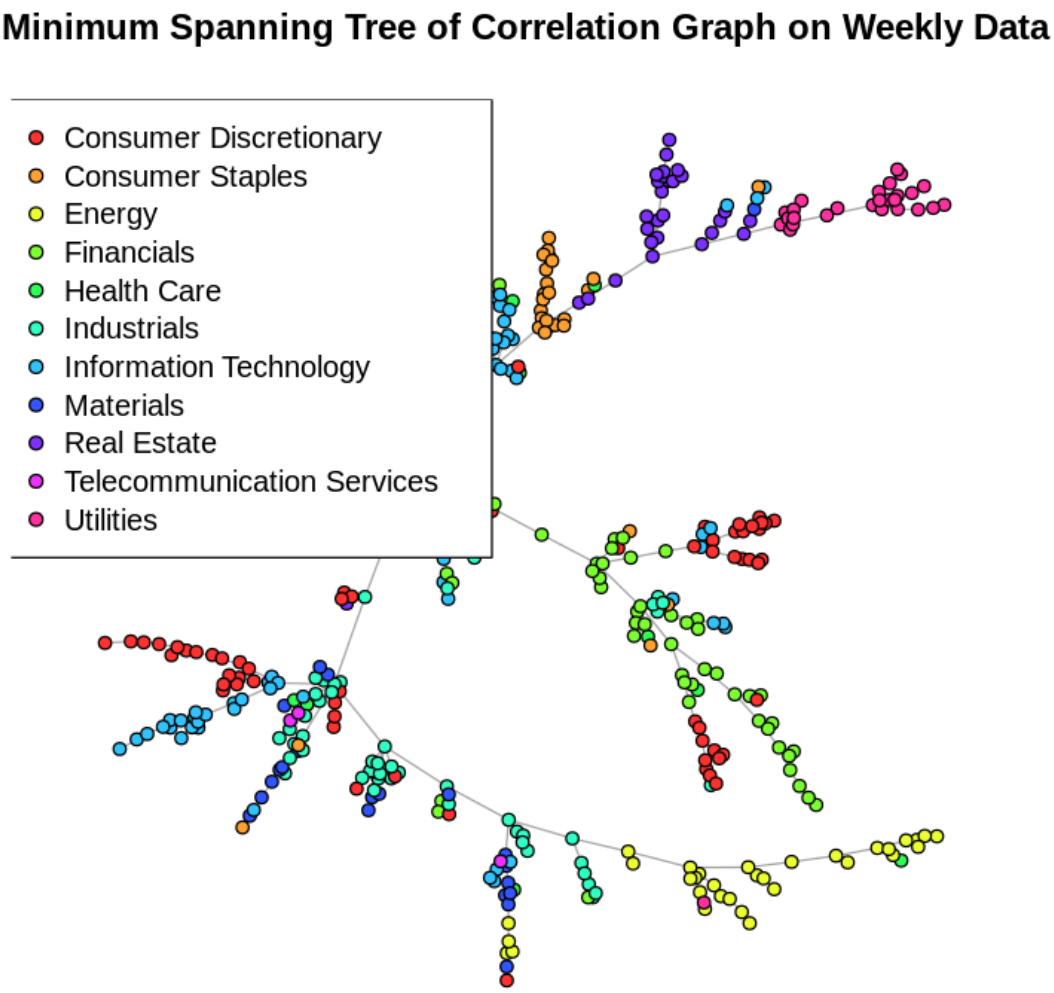
1. Histogram showing the un-normalized distribution of edge weights on Weekly data.



From the above figure we observe, we observe that most of the edge weights are between 1.0 and 1.4. As expected, all the weights lie between 0 and 2, however, we observe that the lower bound for edge weights is 0.5 and not 0. This means that there are no stocks with perfect positive correlation, as this would require some of the edge weights to be 0. Most of the stocks are weakly correlated with each other, and majority of the stocks have negative correlation with each other.

2. Minimum Spanning Tree (MST) of the correlation graph

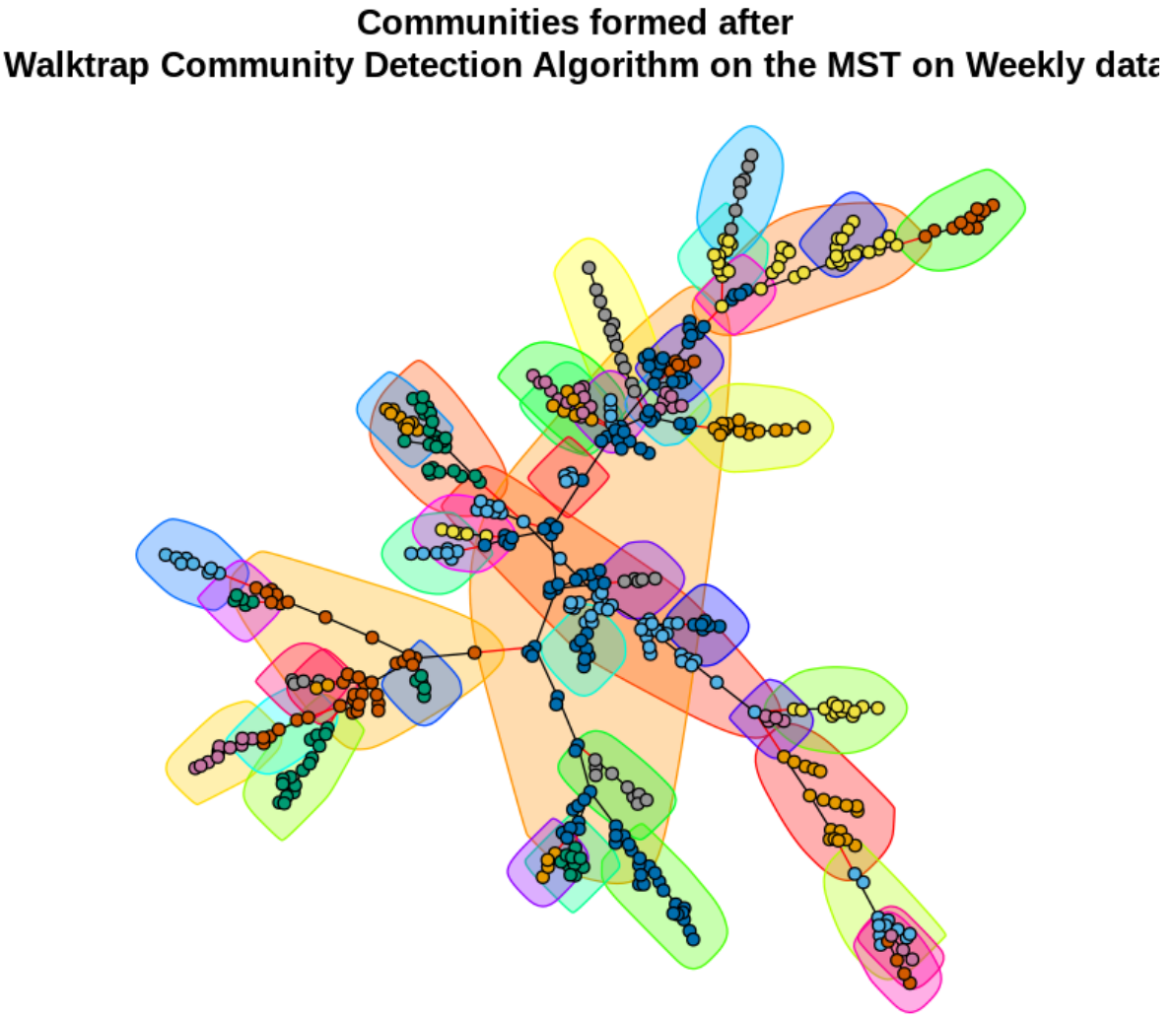
We know that the MST connects all the nodes of a connected and edge-weighted undirected graph without any cycles and with the lowest possible cumulative edge weights. We use Prim's algorithm to create the MST from the correlation graph. The below figure shows the MST for the correlation graph, with stocks belonging to the same sector color coded the same for the weekly data.



Based on the correlation graph depicted above, several patterns can be observed as follows:

Firstly, stocks within the same sector, represented by the same color, tend to exhibit a tendency to cluster together in the MST. This is in line with expectations, given their high correlation, resulting in connections with minimal edge weights on the graph. Secondly, stocks belonging to distinct sectors, indicated by different colors, are generally not connected to one another and exhibit significant edge weights. In other words, stocks that are highly correlated tend to be connected on the correlation graph with the least possible edge weights (the higher the correlation ρ_{ij} between stocks, the lower the edge weights w_{ij}, as $w_{ij} = \sqrt{2(1-\rho_{ij})}$. On the other hand, the stocks which are not strongly correlated have large edge weights. The goal of a Minimum Spanning Tree (MST) is to connect all nodes using the edges with the lowest cumulative weights. This process naturally groups highly correlated stocks together into clusters, which are often referred to as **vine clusters** because they resemble grapes hanging from a vine. These clusters typically represent distinct sectors of the market. Stocks within the same cluster tend to move in the same direction, suggesting they may require similar investment strategies. Vine clusters offer a snapshot of the stock market over longer time periods, such as daily data, highlighting the relationships and sectoral connections among different stocks.

3. Community Detection Algorithm on the MST with Homogenity and Completeness scores.
We ran the Walktrap algorithm on the MST obtained above to get the following results for the Weekly data:



The Homogenity and Completeness scores of the clustering are:
Number of clusters : 42
Homogenity : 0.582007
Completeness : 0.390771

4. Values of Alpha for the 2 cases

The values of α for the two cases are:

Values of first alpha based on Qi, Ni : 0.742970

Values of second alpha based on Si, V : 0.114188

$\alpha 1$ is around 0.74 and $\alpha 2$ is around 0.11. One can see that $\alpha 1$ is much higher than $\alpha 2$, meaning that the first technique based on $P(vi \in Si) = |Qi||Ni|$ is way better compared to the second technique.

This outcome is anticipated because Case 1 leverages the vine cluster structures identified by the Minimum Spanning Tree (MST) within the correlation graph, focusing on nodes (stocks) that are highly correlated and cluster together. By utilizing the local connectivity among neighboring nodes, Case 1 makes precise decisions based on these strong, localized relationships. In contrast, Case 2 examines all nodes within a sector as a whole, overlooking specific local spatial connections or cluster formations. This global approach fails to differentiate between local clusters and treats the entire sector uniformly, resulting in only a broad, generalized probability estimate. Consequently, Case 2 lacks the precision provided by the detailed local analysis in Case 1, which accurately captures the behavior and relationships within the market.

Observations:

The results based on weekly data are shown above. Compared to daily data, the total number of clusters increases from 33 to 42, and both the homogeneity and completeness values decrease by about 0.1. This indicates that the stocks are not as clearly clustered as before, which can be observed in the MST correlation graph. Stocks within the same sector, represented by the same color, do not group together as distinctly as they do in the correlation graph based on daily data. In fact, there is a mix of nodes from different sectors near some nodes, indicating that the stocks are not as highly correlated in the weekly data. Consequently, the α1 value, which relates to the local connectivity among neighboring nodes, decreases from 0.83 to 0.75. However, the α2 value remains unchanged because it always considers all nodes.

Explanations:

The analysis of weekly data shows a shift in the clustering pattern of stocks when compared to daily data. The increase in the total number of clusters from 33 to 42 suggests that the stocks form more, but smaller, groups when observed over a weekly period. This fragmentation is accompanied by a reduction in the values of homogeneity and completeness by around 0.1, indicating that the clarity and coherence of the clusters have diminished.

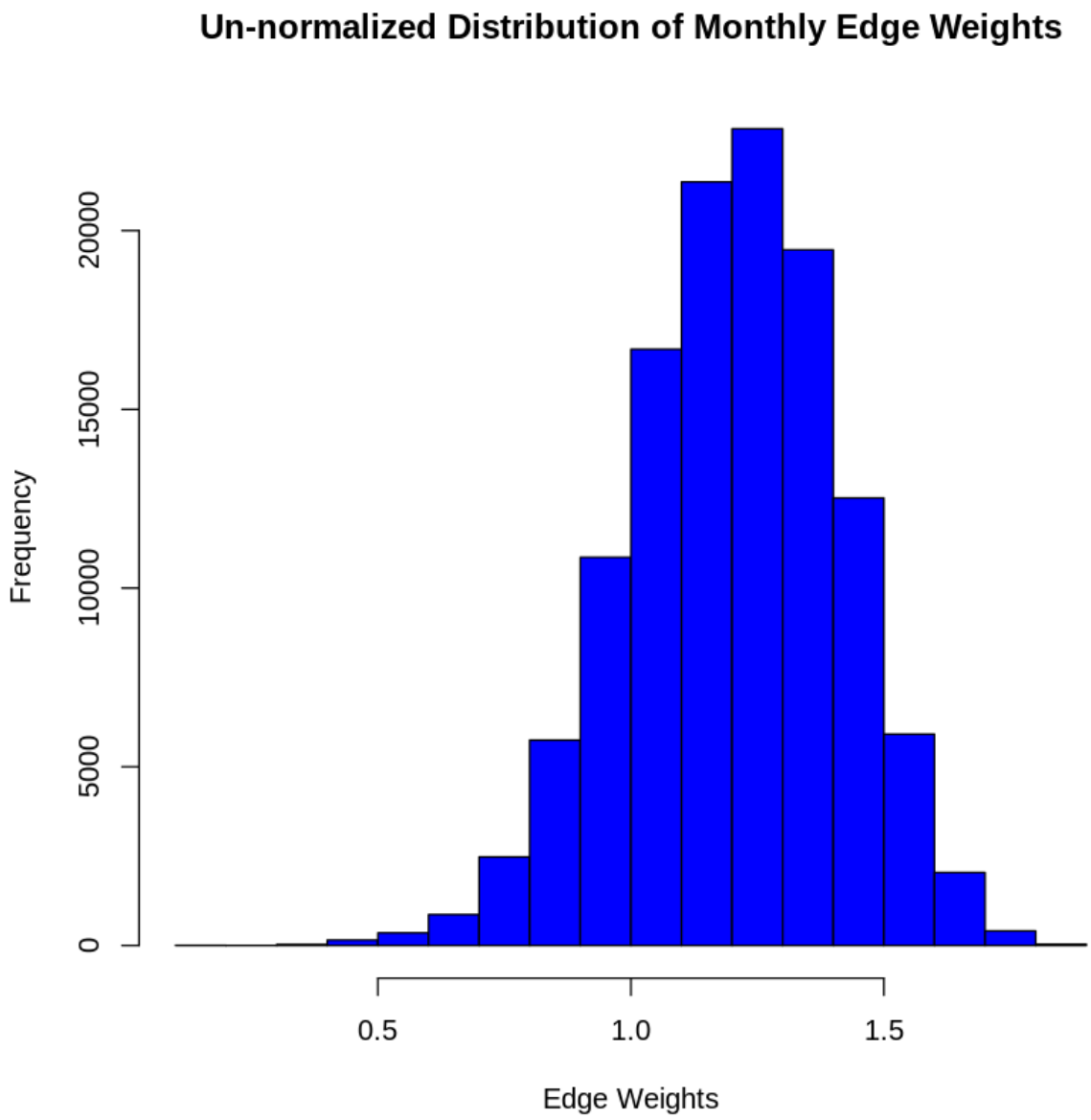
In the MST correlation graph, stocks that belong to the same sector and are marked with the same color are not as tightly grouped as they are in the daily data correlation graph. This suggests that the correlation between stocks within the same sector is weaker on a weekly basis. There is noticeable mixing of nodes from different sectors around certain nodes, showing a decrease in sector-specific clustering.

The value, which measures the local connectivity among neighboring nodes, also decreases from 0.83 to 0.75. This α1 reduction reflects the weaker local correlations in the weekly data. In contrast, the α2 value remains unchanged, as it considers the overall structure of the graph and includes all nodes, maintaining its global perspective regardless of the time scale.

QUESTION 7:

Repeat questions 2,3,4,5 on the MONTHLY data.

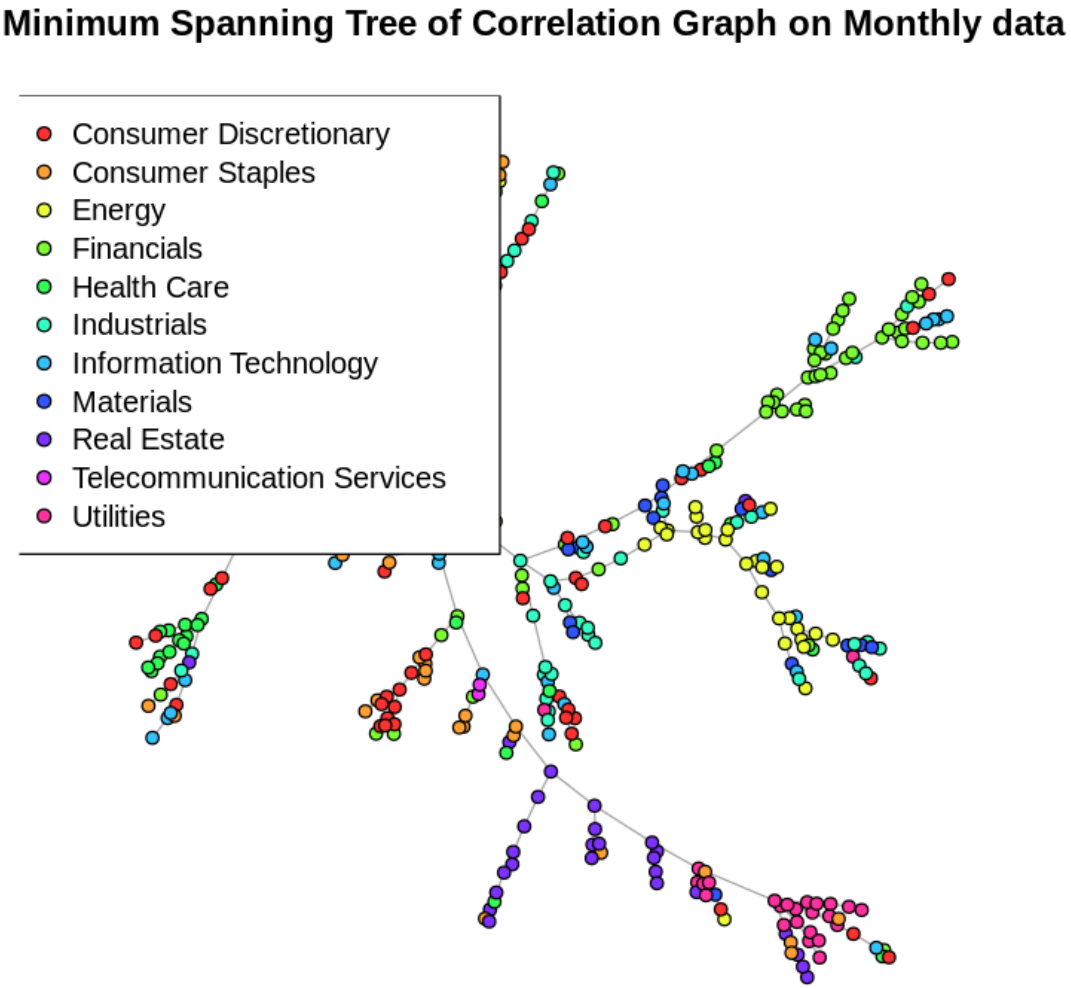
1. Histogram showing the un-normalized distribution of edge weights on Monthly data.



From the above figure we observe, we observe that most of the edge weights are between 1.0 and 1.4. As expected, all the weights lie between 0 and 2, however, we observe that the lower bound for edge weights is 0.5 and not 0. This means that there are no stocks with perfect positive correlation, as this would require some of the edge weights to be 0. Most of the stocks are weakly correlated with each other, and majority of the stocks have negative correlation with each other.

2. Minimum Spanning Tree (MST) of the correlation graph

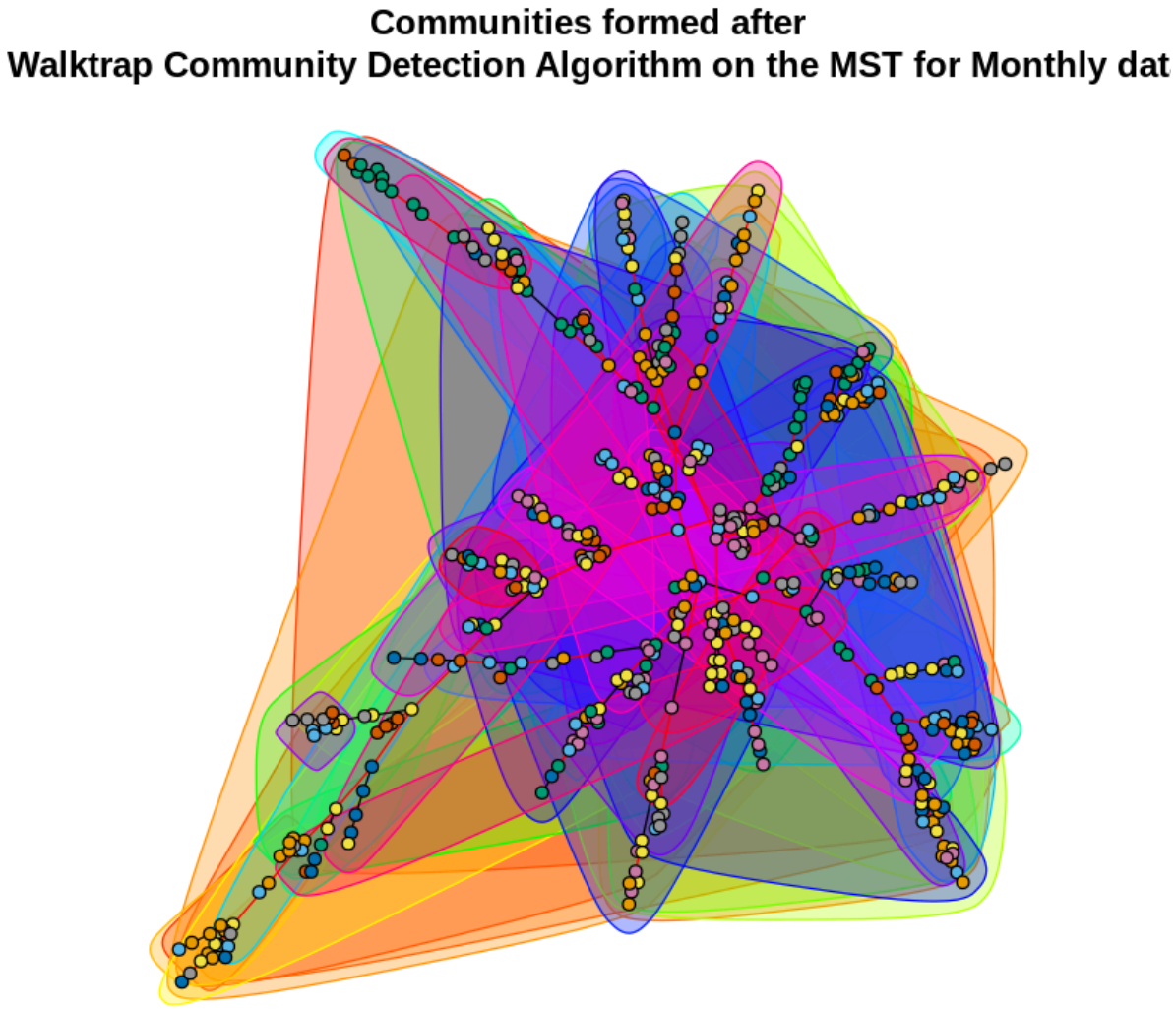
We know that the MST connects all the nodes of a connected and edge-weighted undirected graph without any cycles and with the lowest possible cumulative edge weights. We use Prim's algorithm to create the MST from the correlation graph. The below figure shows the MST for the correlation graph, with stocks belonging to the same sector color coded the same for the monthly data.



Based on the correlation graph depicted above, several patterns can be observed as follows:

Firstly, stocks within the same sector, represented by the same color, tend to exhibit a tendency to cluster together in the MST. This is in line with expectations, given their high correlation, resulting in connections with minimal edge weights on the graph. Secondly, stocks belonging to distinct sectors, indicated by different colors, are generally not connected to one another and exhibit significant edge weights. In other words, stocks that are highly correlated tend to be connected on the correlation graph with the least possible edge weights (the higher the correlation ρ_{ij} between stocks, the lower the edge weights w_{ij}, as $w_{ij} = \sqrt{2(1-\rho_{ij})}$. On the other hand, the stocks which are not strongly correlated have large edge weights. The goal of a Minimum Spanning Tree (MST) is to connect all nodes using the edges with the lowest cumulative weights. This process naturally groups highly correlated stocks together into clusters, which are often referred to as **vine clusters** because they resemble grapes hanging from a vine. These clusters typically represent distinct sectors of the market. Stocks within the same cluster tend to move in the same direction, suggesting they may require similar investment strategies. Vine clusters offer a snapshot of the stock market over longer time periods, such as daily data, highlighting the relationships and sectoral connections among different stocks.

3. Community Detection Algorithm on the MST with Homogeneity and Completeness scores.
We ran the Walktrap algorithm on the MST obtained above to get the following results for the Monthly data:



The Homogeneity and Completeness scores of the clustering are:

Number of clusters : 74
Homogeneity : 0.509509
Completeness : 0.282338

4. Values of Alpha for the 2 cases

The values of α for the two cases are:

Based on monthly data:
Values of first alpha based on Qi, Ni : 0.483468
Values of second alpha based on Si, V : 0.114188

$\alpha 1$ is around 0.48 and $\alpha 2$ is around 0.11. One can see that $\alpha 1$ is higher than $\alpha 2$, meaning that the first technique based on $P(vi \in Si) = |Qi||Ni|$ is way better compared to the second technique.

This outcome is anticipated because Case 1 leverages the vine cluster structures identified by the Minimum Spanning Tree (MST) within the correlation graph, focusing on nodes (stocks) that are highly correlated and cluster together. By utilizing the local connectivity among neighboring nodes, Case 1 makes precise decisions based on these strong, localized relationships. In contrast, Case 2 examines all nodes within a sector as a whole, overlooking specific local spatial connections or cluster formations. This global approach fails to differentiate between local clusters and treats the entire sector uniformly, resulting in only a broad, generalized probability estimate. Consequently, Case 2 lacks the precision provided by the detailed local analysis in Case 1, which accurately captures the behavior and relationships within the market.

Observations:
The results based on monthly data are shown above, yielding similar outcomes to those in Q6. The overall performance decreases when using monthly data. For example, both the homogeneity and completeness values drop by approximately 0.18, indicating less clarity among different sectors in the correlation graph. Consequently, more clusters are needed, resulting in 74 clusters, which is more than double the number compared to daily data. Additionally, the $\alpha 1$ value drops significantly to 0.48. This reduction in performance is primarily due to the limited amount of data available. Since daily data provides significantly more information for each node than monthly data, it is expected that the graph algorithm performs better with daily data.

Explanation:
The analysis of monthly data shows a further decline in clustering performance compared to daily data. Both homogeneity and completeness values decrease by around 0.18, suggesting that the clarity and distinctness of the clusters have diminished even more. This lack of clear sector separation in the correlation graph necessitates a greater number of clusters—74 in this case, which is more than double the number required for daily data.
The $\alpha 1$ value, which reflects local connectivity among neighboring nodes, also drops sharply to 0.48. This significant reduction indicates that the local correlations are much weaker when using monthly data.
The primary reason for this decrease in performance is the reduced data size. Daily data offers much more information per node compared to monthly data, providing a richer dataset for the graph algorithm to analyze. With more data points, the algorithm can identify and leverage stronger and more numerous correlations, leading to better clustering performance. Conversely, the scarcity of data in monthly observations limits the algorithm's effectiveness, resulting in poorer performance metrics and a need for more clusters to achieve the same level of detail.

QUESTION 8:

Compare and analyze all the results of daily data vs weekly data vs monthly data. What trends do you find? What changes? What remains similar? Give reason for your observations. Which granularity gives the best results when predicting the sector of an unknown stock and why?

Analysis and Comparison of Results: Daily Data vs. Weekly Data vs. Monthly Data

Parameters	**Daily data**	**Weekly data**	**Monthly data**
Number of Clusters	33	42	74
Homogeneity Score	0.682645	0.582007	0.509509
Completeness Score	0.479284	0.390771	0.282338
First alpha α1 based on Qi, Ni	0.828930	0.742970	0.483468
Second alpha α2 based on Si, V	0.114188	0.114188	0.114188

Trends and Observations:

1. Cluster Count:
The number of clusters increases significantly as the data granularity shifts from daily to monthly. Daily data has 33 clusters, weekly data has 42, and monthly data has 74 clusters. This increase indicates that as the time scale extends, the clustering algorithm requires more clusters to represent the data adequately.

2. Homogeneity and Completeness:
Both homogeneity and completeness scores decrease as the data granularity shifts from daily to monthly. Daily data achieves the highest homogeneity (0.682645) and completeness (0.479284), indicating clearer and more cohesive clusters. These scores drop to 0.582007 and 0.390771 for weekly data, and further to 0.509509 and 0.282338 for monthly data. This trend suggests that with less frequent data, the clarity and cohesion of clusters diminish.

3. First Alpha α1 based on Qi, Ni:
The first alpha value, which measures local connectivity among neighboring nodes, decreases from 0.828930 (daily) to 0.742970 (weekly), and significantly to 0.483468 (monthly). This reduction highlights that the local correlations among stocks weaken as the data frequency decreases.

4. Second Alpha α2 based on Si, V:
The second alpha value remains constant at 0.114188 across all time scales. This consistency indicates that α2, which considers the global sector connectivity, is unaffected by the change in data granularity.

Reasons for Observations:

Data Density and Detail:
Daily data provides more detailed and dense information about stock price fluctuations, allowing for more precise correlation and clustering. With more data points, the algorithm can identify stronger and more numerous correlations, leading to better-defined clusters. Weekly and monthly data reduce the amount of available information, leading to less accurate clustering. The extended time scale dilutes the specific fluctuations, making it harder to capture strong correlations among stocks.

Local vs. Global Connectivity:
The decrease in $\alpha 1$ values with less frequent data reflects the loss of detailed local correlations. Daily data captures short-term fluctuations more effectively, which is crucial for local connectivity. The constant $\alpha 2$ value shows that the overall sector connectivity remains the same because it considers all nodes collectively, regardless of the time scale.

Analyzing the data across different time scales, it is evident that using daily data results in superior performance and clustering accuracy compared to weekly and monthly data. Daily data provides a comprehensive and detailed view of stock price fluctuations, allowing for more accurate correlation analysis. When the input data is reduced to weekly or monthly intervals, the quality of the results deteriorates due to the reduced amount of data available for analysis. As the time scale extends from daily to monthly, the correlation between stocks within the same sector weakens, leading to higher edge weights in the correlation graph for stocks that are part of the same sector. This makes it more difficult to accurately assign a sector to an unknown stock, as evidenced by the significant decrease in the $\alpha 1$ value from 0.83 (daily) to 0.48 (monthly). The $\alpha 1$ value measures local connectivity among neighboring nodes, and its decline indicates a loss of detailed local correlation information.

On the other hand, the $\alpha 2$ value remains constant because it considers all sectors collectively without considering individual stock price fluctuations. This metric is unaffected by the change in time scale and data granularity. Overall, the best results for predicting the sector of an unknown stock are achieved using daily data with the first technique to calculate α. This method provides detailed information necessary for generating a clear correlation graph and takes into account local spatial connectivity, ensuring higher accuracy in sector prediction. Therefore, the summarized results and their interpretation appear to be correct, indicating that daily data is most effective for capturing the intricate relationships between stocks and their sectors.

Best Granularity for Sector Prediction:
Daily Data: Daily data provides the best results for predicting the sector of an unknown stock. The higher homogeneity and completeness scores indicate clearer and more accurate clustering. The higher $\alpha 1$ value suggests stronger local correlations, which are essential for precise sector identification. Daily data captures the intricate and immediate relationships between stocks, making it more effective for detailed analysis and prediction.

In conclusion, daily data is the most effective granularity for predicting the sector of an unknown stock due to its detailed information, higher clustering accuracy, and stronger local correlations. Reducing the data frequency to weekly or monthly intervals significantly diminishes the performance, clarity, and precision of the clustering results.

QUESTION 9:

Report the number of nodes and edges in G.

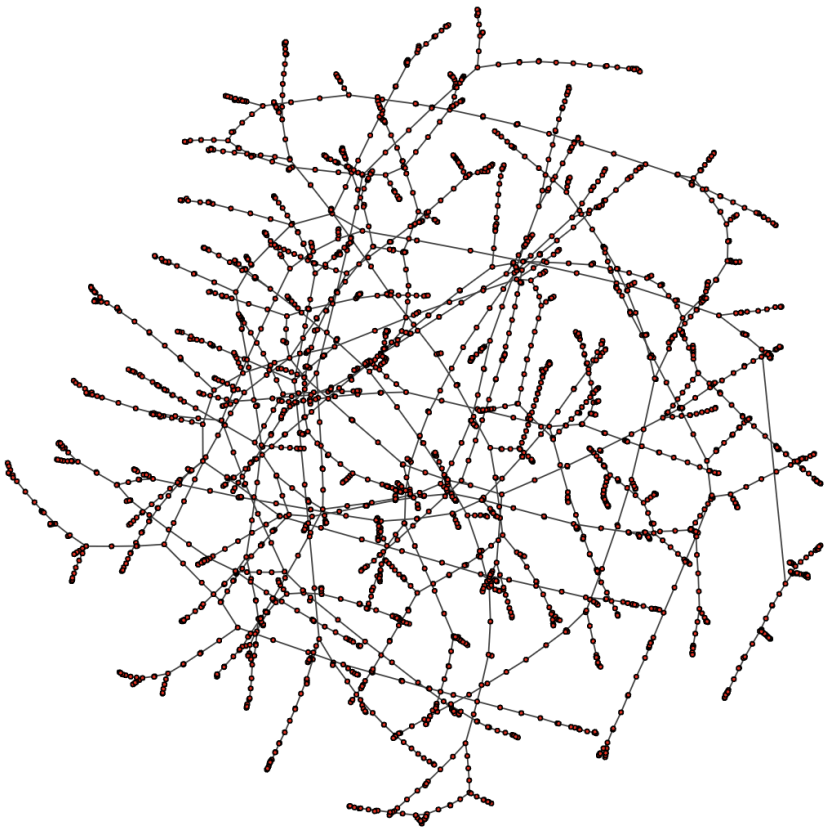
The nodes represent geographic locations (latitude and longitude), and the edges denote average travel times between these locations. To simplify the graph, we remove isolated nodes, merge duplicate edges by averaging their weights, and retain only the largest connected component. We use the 'los_angeles-censustracts-2019-4-All-MonthlyAggregate.csv' file to import the source ID, destination ID, and mean travel times for constructing the graph.

The number of nodes in G are: 2649
The number of edges in G are: 1004955

QUESTION 10:

Build a minimum spanning tree (MST) of graph G. Report the street addresses near the two endpoints (the centroid locations) of a few edges. Are the results intuitive?

The minimum spanning tree of graph G can be found below:

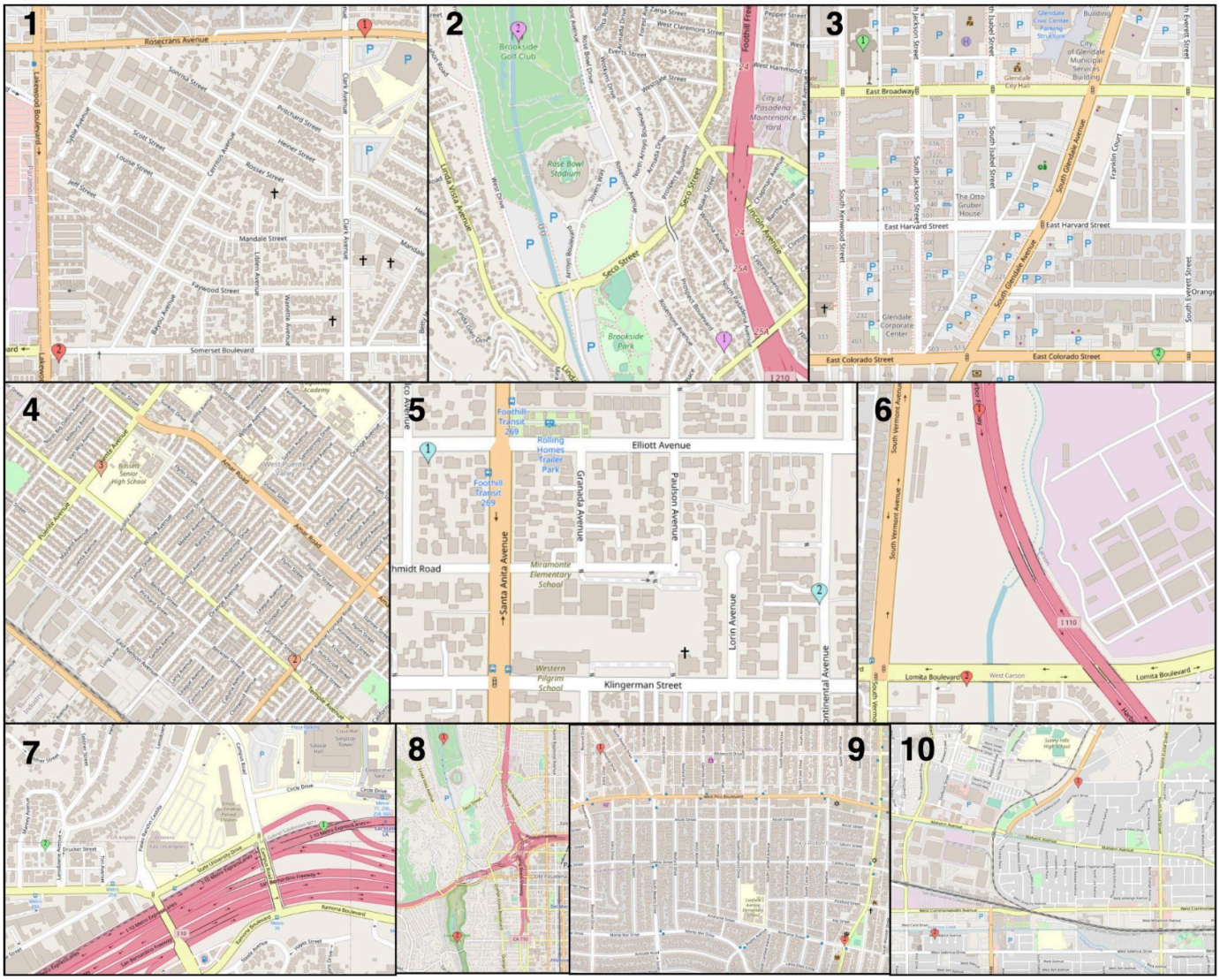


The street addresses near the two endpoints of a few edges can be found in the table below:

ID	Census Tract (a)	Coordinates (a)	Census Tract (b)	Coordinates (b)	Street Address (a)	Street Address (b)
1	554001	[-118.133298 33.904119]	554002	[-118.141448 33.896526]	9421, Rosecrans Avenue, Bellower, California, 90706, United States of America	9020, Somerset Boulevard, Bellower, Los Angeles County, California, United States of America
2	461700	[-118.159325 34.153604]	460800	[-118.172425 34.180286]	500, Prospect Boulevard, Pasadena, California, 91103, United States of America	Brookside Golf Club, Arrovo Woods, Pasadena, CA 91109, United States of America
3	302201	[-118.251349	302202	[-118.248811	First United	East Colorado

		34.146334]		34.142665]	Methodist Church of Glendale, 134 North Kenwood Street, Glendale, CA 91206, United States of America	Street, Glendale, CA 91205, United States of America
4	407101	[-117.967696 34.04101]	407002	[-117.980572 34.051745]	619 North Sunset Avenue, La Puente, CA 91744, United States of America	652 Puente Avenue, El Monte, CA 91746, United States of America
5	433401	[-118.043316 34.058606]	433402	[-118.038157 34.056957]	10468 Fern Street, South El Monte, CA 91733, United States of America	2433 Continental Avenue, El Monte, CA 91733, United States of America
6	543603	[-118.28871 33.803659]	294410	[-118.289861 33.797764]	Harbor Freeway, West Carson, CA 90710, United States of America	828 Lomita Boulevard, West Carson, CA 90710, United States of America
7	482001	[-118.165951 34.062293]	530700	[-118.176633 34.062274]	I-10 Metro ExpressLanes, Los Angeles, CA 90032, United States of America	4225 Drucker Street, East Los Angeles, CA 90032, United States of America
8	460800	[-118.172425 34.180286]	463800	[-118.167761 34.141469]	Brookside Golf Club, Arrovo Woods, Pasadena, CA 91109, United States of America	Lower Arroyo Park, 415 Arroyo Boulevard, Pasadena, CA 91105, United States of America
9	269100	[-118.398461 34.057095]	217001	[-118.385502 34.04886]	475 Smithwood Drive, Beverly Hills, CA 90212, United States of America	Robertson & Airdrome, Robertson Boulevard, Los Angeles, CA 90035-4232, United States of America
10	011000	[-117.950433 33.881015]	001901	[-117.965225 33.868368]	1382 West Valley View Drive, Fullerton, CA 92833, United States of America	2069 West Walnut Avenue, Fullerton, CA 92833, United States of America

Below are the maps for the above points



We can see that the street addresses of the endpoints are very close to each other. This is intuitive because the Minimum Spanning Tree (MST) aims to connect all addresses with the lowest cumulative mean travel time, or the lowest weight of edges. A straightforward way to achieve this is by connecting nodes that are close to each other, as the mean travel time between nearby nodes will also be smaller. Thus, Prim's algorithm selects edges with the smallest weights (mean travel times), ultimately connecting nodes with small pairwise distances. In fact, we observed that the average pairwise distance between the endpoints is around 0.7 miles.

QUESTION 11:

Determine what percentage of triangles in the graph (sets of 3 points on the map) satisfy the triangle inequality. You do not need to inspect all triangles, you can just estimate by random sampling of 1000 triangles.

The triangle inequality states that the sum of any two sides of a triangle must always be greater than the third side:

$$a + b > c, \quad b + c > a, \quad a + c > b$$

In the case of our graph, it means the mean travel time between two addresses will always be smaller than the mean travel time between those addresses via a third address.

After randomly sampling 1000 triangles (3 nodes each), we found out that 91.5% of the triangles satisfy the triangle inequality.

QUESTION 12:

Find an upper bound on the empirical performance of the approximate algorithm:

$$\rho = \frac{\text{Approximate TSP Cost}}{\text{Optimal TSP Cost}}$$

The Traveling Salesman Problem (TSP) involves an undirected weighted graph where the nodes represent cities and the edges represent roads or streets. The weights of the edges indicate the distances between cities. The objective is to find the shortest route starting from a city, visiting all other cities exactly once, and returning to the starting city. This route is known as a Hamiltonian cycle, which must satisfy the triangle inequality. Since the TSP is NP-Hard, approximation methods are valuable for reducing computation time. We employ a 1-approximation algorithm for solving the TSP, which involves the following steps:

- Find the Minimum Spanning Tree (MST).
- Create a multi-graph by replacing each edge of the MST with two directed edges.
- Identify the Euler cycle in the multi-graph to establish the tour sequence. An Euler cycle covers each edge exactly once.
- For each edge in the tour sequence, check its existence in the MST. If an edge does not exist, use Dijkstra's algorithm to find the shortest path between the corresponding nodes.
- Calculate the approximate TSP cost as the sum of the weights of the edges in the tour sequence, or the shortest path cost if the edge is absent in the MST. The MST cost provides a lower bound, as the optimal TSP cost (best possible route) is always greater than or equal to the MST cost.

Actually,

Length of the tour returned by the algorithm ≤ Distance covered by Euler cycle ≤ 2 X Weight of minimum spanning tree (Length of Euler tour) ≤ 2 X Weight of minimum tour length ☰ MST cost ≤ Optimal TSP cost ≤ Approximate TSP cost ≤ 2 X MST cost

Our MST cost was 269084.545000000166

Approximate TSP cost was 421489.3149999998 (which was less than 2 X MST cost = 538169.09)

Normalizing all the costs by dividing everything with MST cost, we get the following inequality:

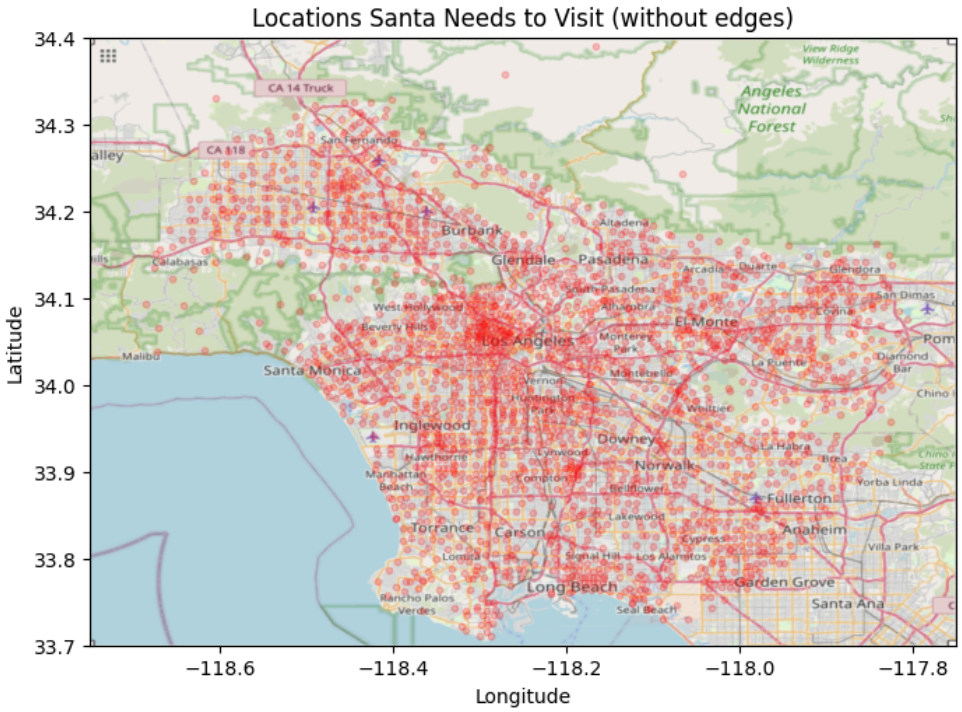
$$1 \leq \frac{\text{Optimal TSP cost}}{\text{MST cost}} \leq \rho \leq 2$$

Thus, the value of ρ we obtained was 1.5663824728395292. This satisfies the inequality we obtained above. We observe that the upper bound of ρ is 2, and the lower bound of ρ is 1.

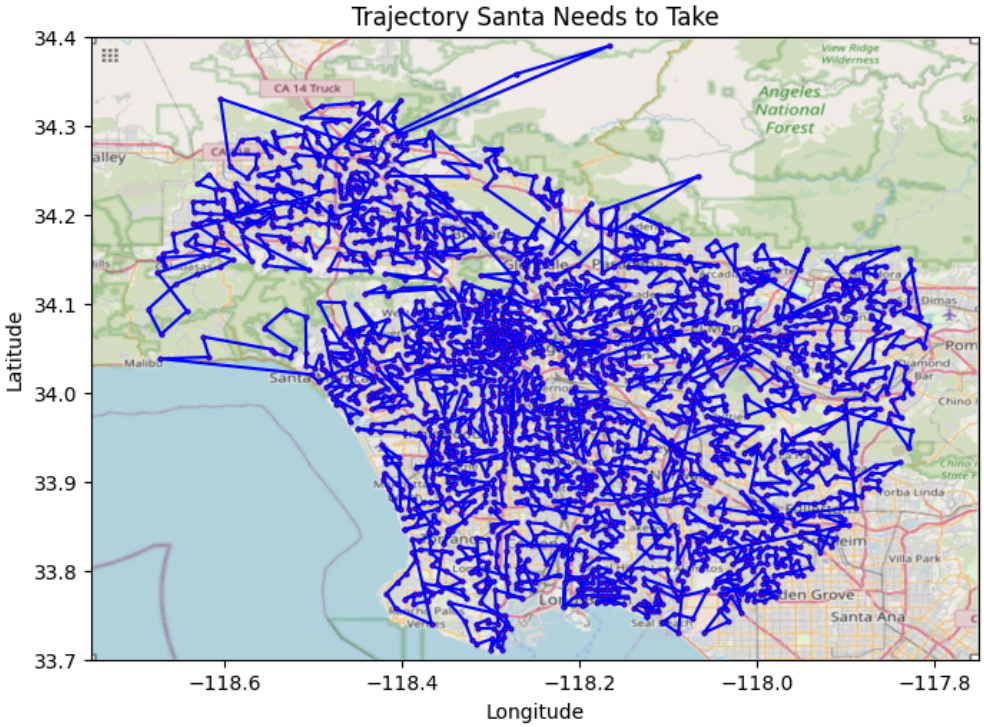
QUESTION 13:

Plot the trajectory that Santa has to travel!

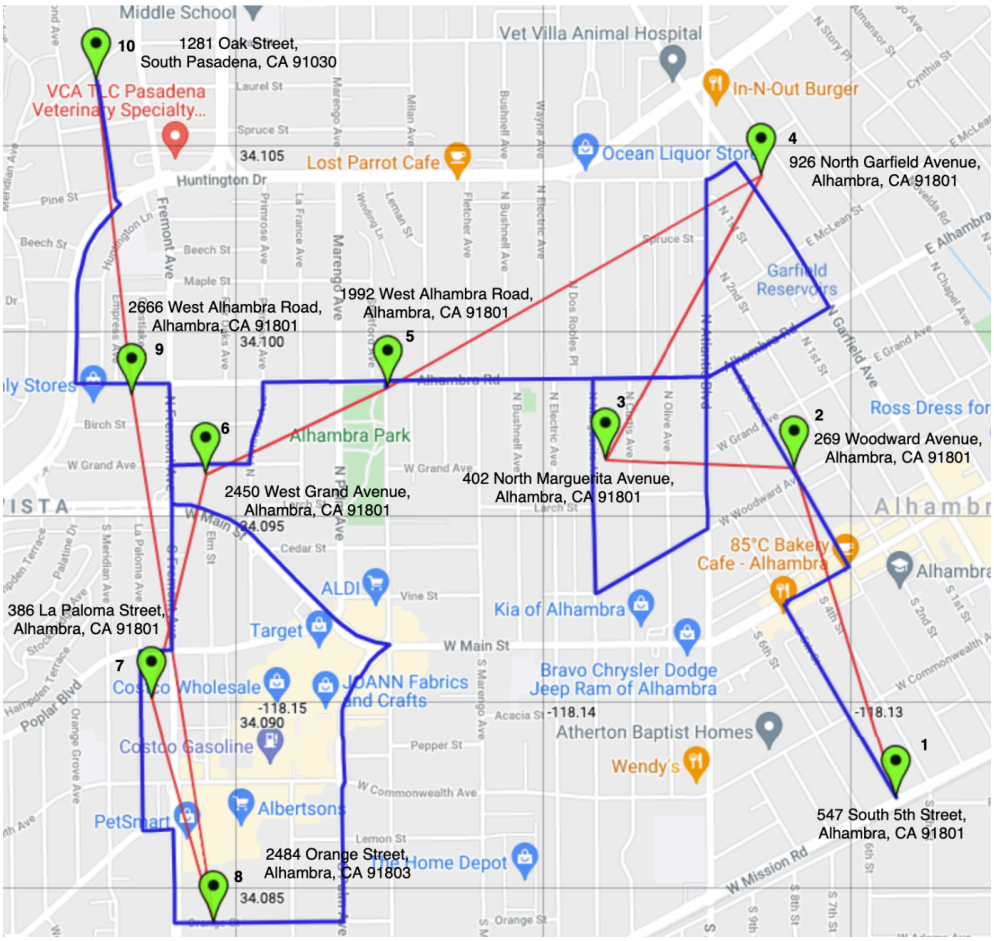
The plot below shows all the nodes Santa needs to visit on the Los Angeles map



The plot below shows the trajectory Santa needs to take on the Los Angeles map



To see if the results make intuitive sense, we plot the first 10 addresses Santa needs to visit in the order the 1-approximation TSP algorithm suggested. The plot is below:



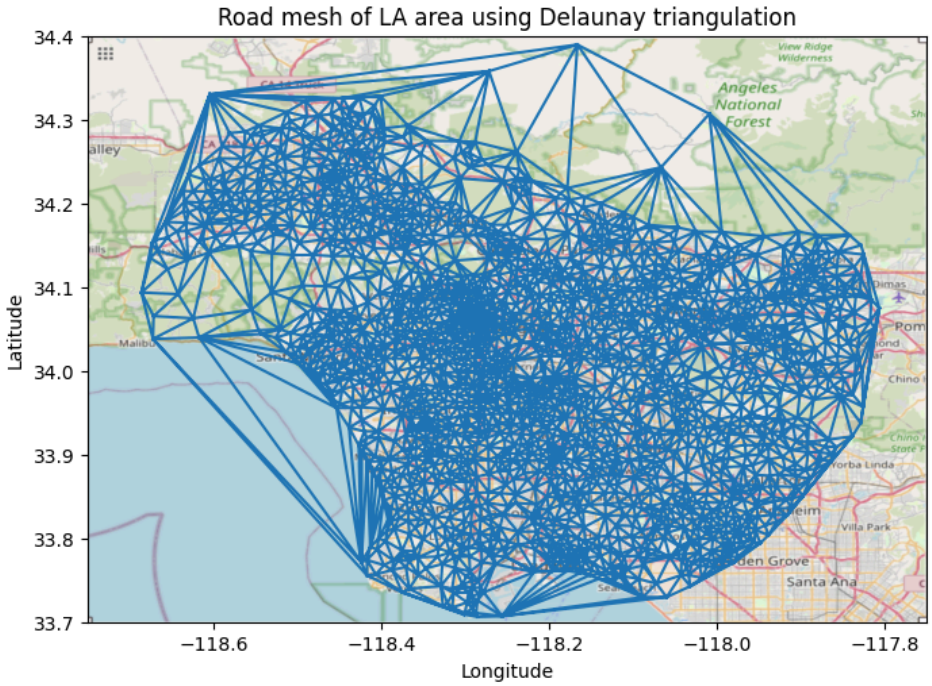
The results are logically consistent. Most edges do not overlap, but some do because the graph is not fully connected. This necessitates using Dijkstra's shortest path algorithm in some cases, which does not adhere strictly to the Eulerian circuit's rule that an edge can only be traversed once. Consequently, some edges overlap, requiring Santa to backtrack to other nodes. Additionally, since the TSP does not incorporate actual road information, some edges cross mountains, oceans, or buildings without any real roads, as seen in the figures above.

We can also see that the TSP algorithm selects the closest addresses (or those that minimize the overall mean travel time) for each traversal, with no addresses being visited more than once. We also calculated that the average distance between successive points in the figure above is approximately 0.6 miles.

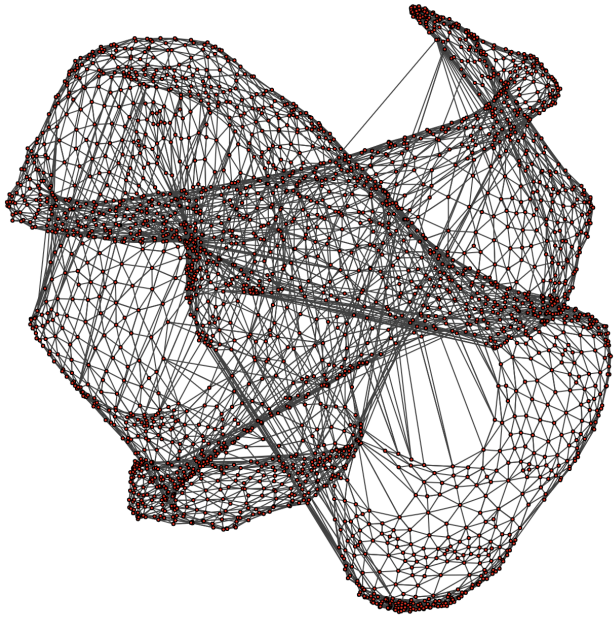
QUESTION 14:

Plot the road mesh that you obtain and explain the result. Create a graph GΔ whose nodes are different locations and its edges are produced by triangulation.

Delaunay Triangulation (DT) is an algorithm for a given set of discrete points that ensures no point lies inside the circumcircle of any triangle in the triangulation. Essentially, DT aims to maximize the minimum angle of all the angles in the triangles, thus avoiding very narrow, or "sliver," triangles. This property is useful for deriving realistic and practical road structures from a set of locations, as real-world road networks tend to follow geometric patterns similar to those produced by DT.
The figure below shows the road mesh of Los Angeles found using Delaunay triangulation.



Below is the graph produced from the triangulation edges.



We observe that the triangulation algorithm has successfully extracted most of the road structures in Los Angeles, especially in the downtown area. However, it also shows roads extending over oceans and mountains, which do not exist in reality. This occurs due to the nature of the Delaunay Triangulation (DT) algorithm, which aims to avoid creating very narrow, or "sliver," triangles by ensuring no point in the set lies within the circumcircle of any triangle. In other words, the DT algorithm tries to fit each triangle inside a circumcircle. This is evident in the figure above, where the nodes represent locations and the edges are generated through triangulation. Most of the resulting polygons do not have extremely small acute angles.

QUESTION 15:

Using simple math, calculate the traffic flow for each road in terms of cars/hour.
Report your derivation.
Hint: Consider the following assumptions:
• Each degree of latitude and longitude ≈ 69 miles
• Car length ≈ 5 m = 0.003 mile
• Cars maintain a safety distance of 2 seconds to the next car
• Each road has 2 lanes in each direction
Assuming no traffic jam, consider the calculated traffic flow as the max capacity of each road.

The derivation is below:

- $$\text{Total distance} = \frac{\text{Velocity of car} \times \text{Mean Travel time}}{60 \times 60}$$ (divide by 3600 to convert data from seconds to hours)
- $$\text{Gap} = 0.003 + \frac{2 \times \text{Velocity of car}}{60 \times 60}$$ (to accommodate for length of each car and the distance between the cars)
- $$\text{Total number of cars on the road} = \frac{2 \times \text{Total distance}}{\text{Gap}}$$ (multiplied by 2 to accommodate ow in both directions)
- $$\text{Traffic Flow (cars/hour)} = \frac{60 \times 60}{\text{Mean Travel Time}} \times \text{Total number of cars on the road}$$

Simplifying the derivation, we get:

$$\boxed{\text{Traffic Flow (cars/hour)} = \frac{3600 \times \text{Velocity of car}}{5.4 + \text{Velocity of car}}}$$

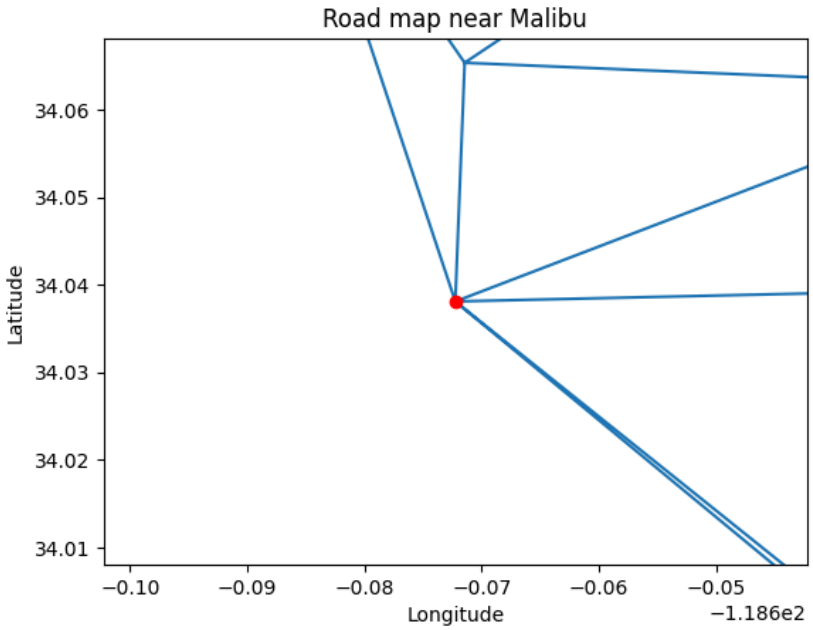
The unit of velocity is in miles per hour. From Pythagoras Theorem, the velocity of car between two coordinates is given as:

$$\text{Velocity of car} = \frac{69}{\text{Mean Travel Time}} \times \sqrt{(\text{Lattitude}_{\text{point 1}} - \text{Lattitude}_{\text{point 2}})^2 + (\text{Longitude}_{\text{point 2}} - \text{Longitude}_{\text{point 2}})^2}$$

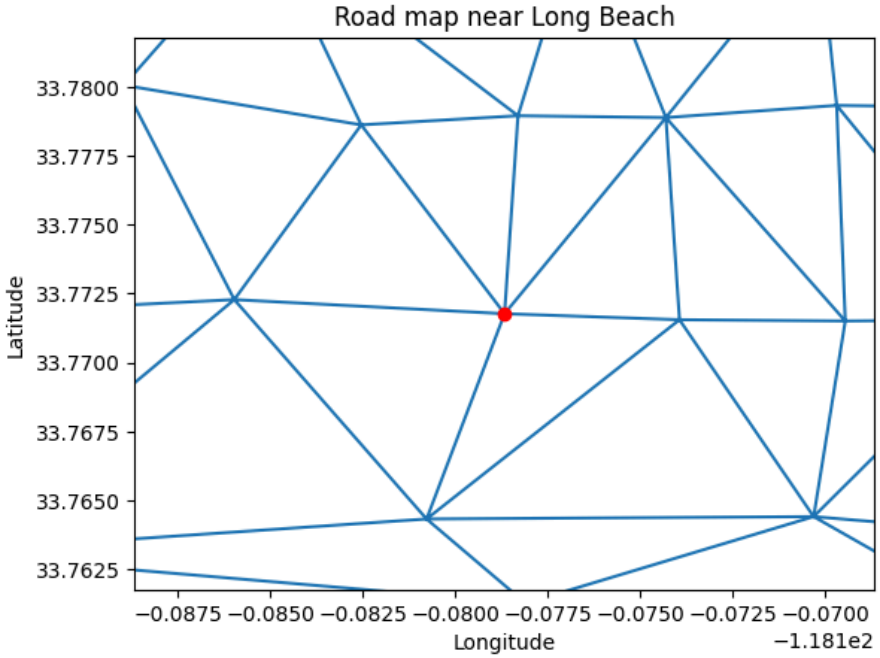
QUESTION 16:

Calculate the maximum number of cars that can commute per hour from Malibu to Long Beach. Also calculate the number of edge-disjoint paths between the two spots. Does the number of edge-disjoint paths match what you see on your road map?

The maximum number of cars from Malibu to Long Beach: 3285
Below is the roadmap near Malibu:



Roadmap near Long Beach:

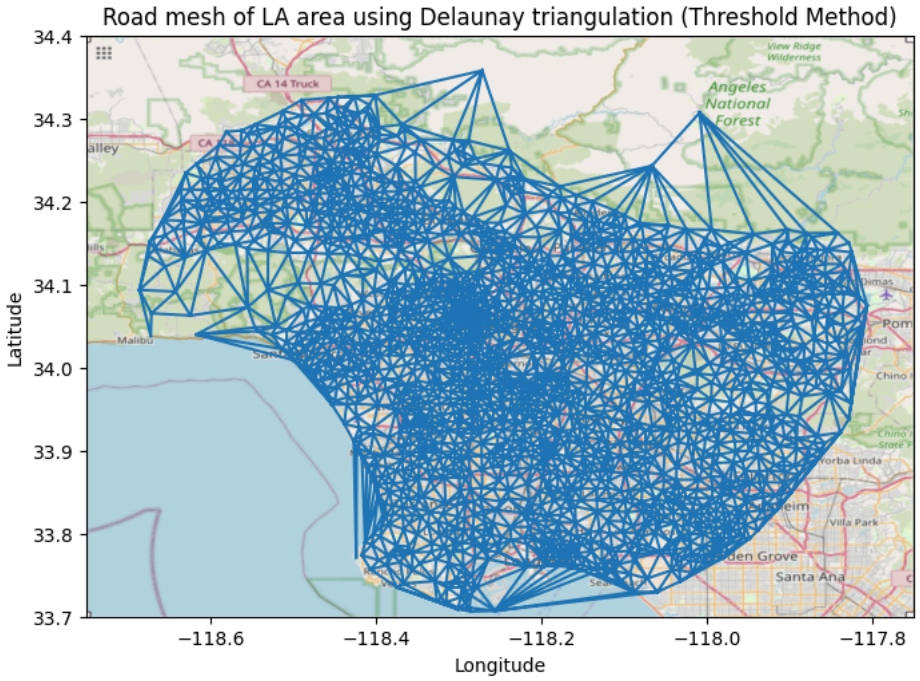


The plots in the figure above show that both Malibu and Long Beach have 6 outgoing and incoming edges, respectively. Additionally, we can see that the degree of each node is the same. For two nodes in a graph, the minimum number of outgoing and incoming edges at each node determines the number of edge-disjoint paths between them. In this case, that number is 6.

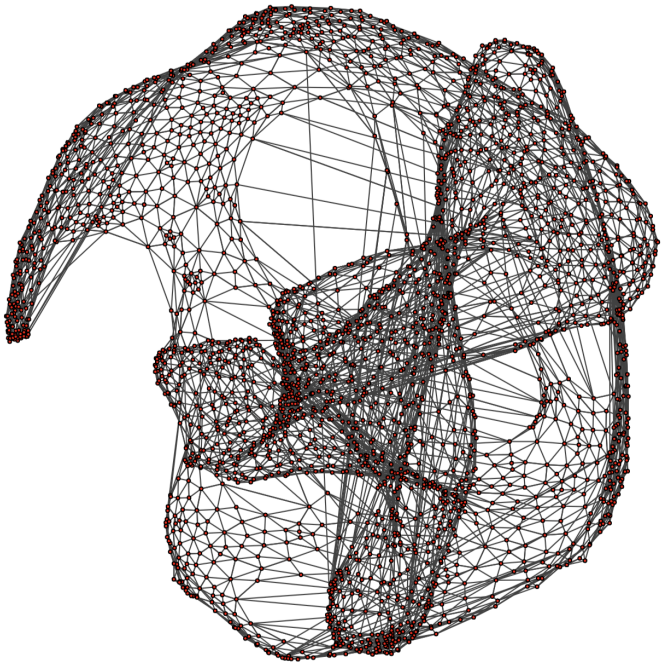
QUESTION 17:

Plot G˜∆ on actual coordinates. Do you think the thresholding method worked?

We apply a threshold to the travel time of roads. Theoretically, this thresholding method removes long edges that the DT algorithm might produce to fit triangles within the circumcircle. The mean travel time for these edges would be higher than for other edges, as they span several nodes at smaller distances. We improve the threshold by incorporating distance information, so that points which are close but have a large travel time are excluded. This ensures that we do not eliminate roads between points that are actually far apart. By incorporating distance information, we effectively set a speed threshold, which we establish at 19.2 miles per hour. The figure below shows the resulting road mesh map.



We can see the graph produced from the triangulation edges (after thresholding) below:



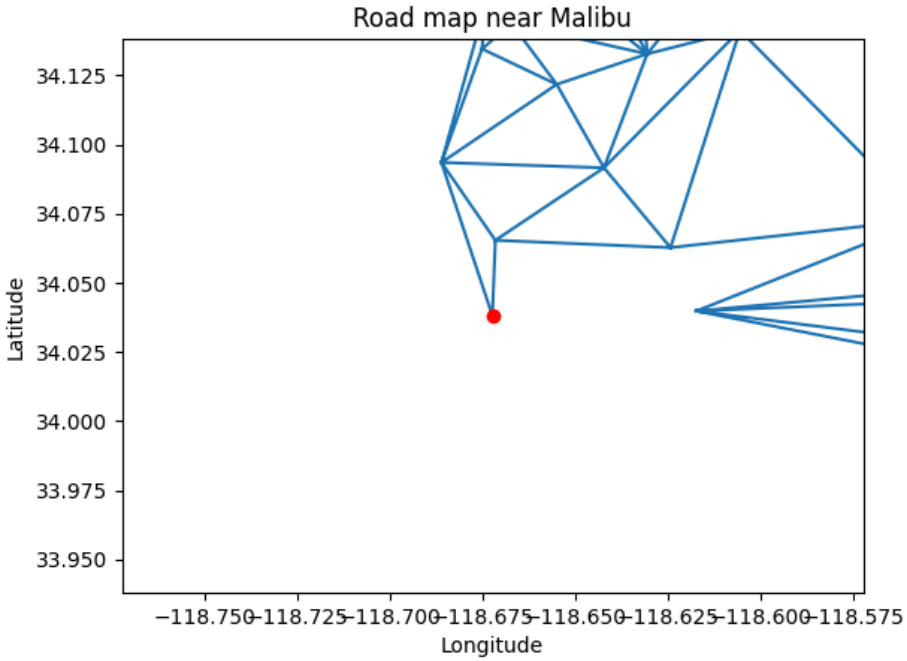
From the figure above, we can observe that the long, non-existent routes over the ocean and the routes over the Topanga mountain ranges have disappeared, indicating the effectiveness of the thresholding method. The road mesh network now more closely resembles the actual Los Angeles road map, with most roads staying within the coastline and avoiding beaches and mountains. Additionally, the figure above shows fewer long edges among neighboring nodes compared to the figure from Question 14, demonstrating that the speed threshold successfully removed only those long edges that connected neighboring nodes.

QUESTION 18:

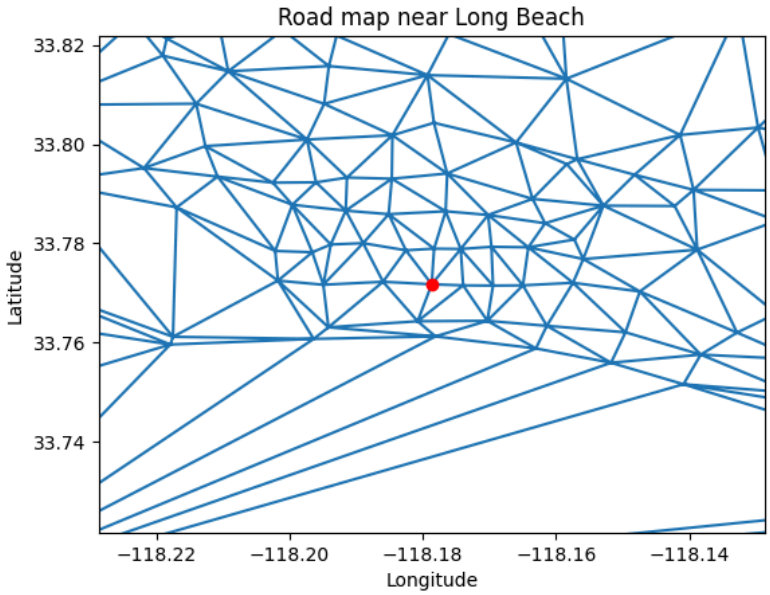
Now, repeat question 13 for G˜∆ and report the results. Do you see any changes? Why?

The maximum number of cars from Malibu to Long Beach: 3300
After removing the non-existent paths, the number of disjoint paths have now decreased from 6 to 3.

The road map near Malibu is below:



The roadmap near Long beach is below:



From the figure above, we see that there has not been any significant change to the road map near Long Beach. However, several non-existent paths to Malibu have been pruned. For two nodes in a graph, the minimum number of outgoing and incoming edges at each node determines the number of edge-disjoint paths, which in this case is 3. Due to the removal of some paths to Malibu, the number of possible edge-disjoint paths between Malibu and Long Beach has decreased.

However, the maximum flow remains at 13,095 cars per hour. This is because there are still numerous paths available for cars to travel from Malibu to Long Beach. The total capacity of all roads connecting Malibu to Long Beach still exceeds 13,095 cars. Additionally, the way the max-flow algorithm functions contributes to this outcome. Maximum flow is calculated by summing the flow across the minimum cuts (edges with the lowest weights). It is unlikely that the non-existent roads had the lowest weights, and therefore they did not impact the maximum flow. As a result, there is no significant difference in the maximum flow.

QUESTION 19:

Strategy 1 (geo distance, static): Reduce the maximum extra travelling dis- tance. The extra travelling distance between 2 locations is the difference of the shortest traveling distance and the straight line distance. i.e

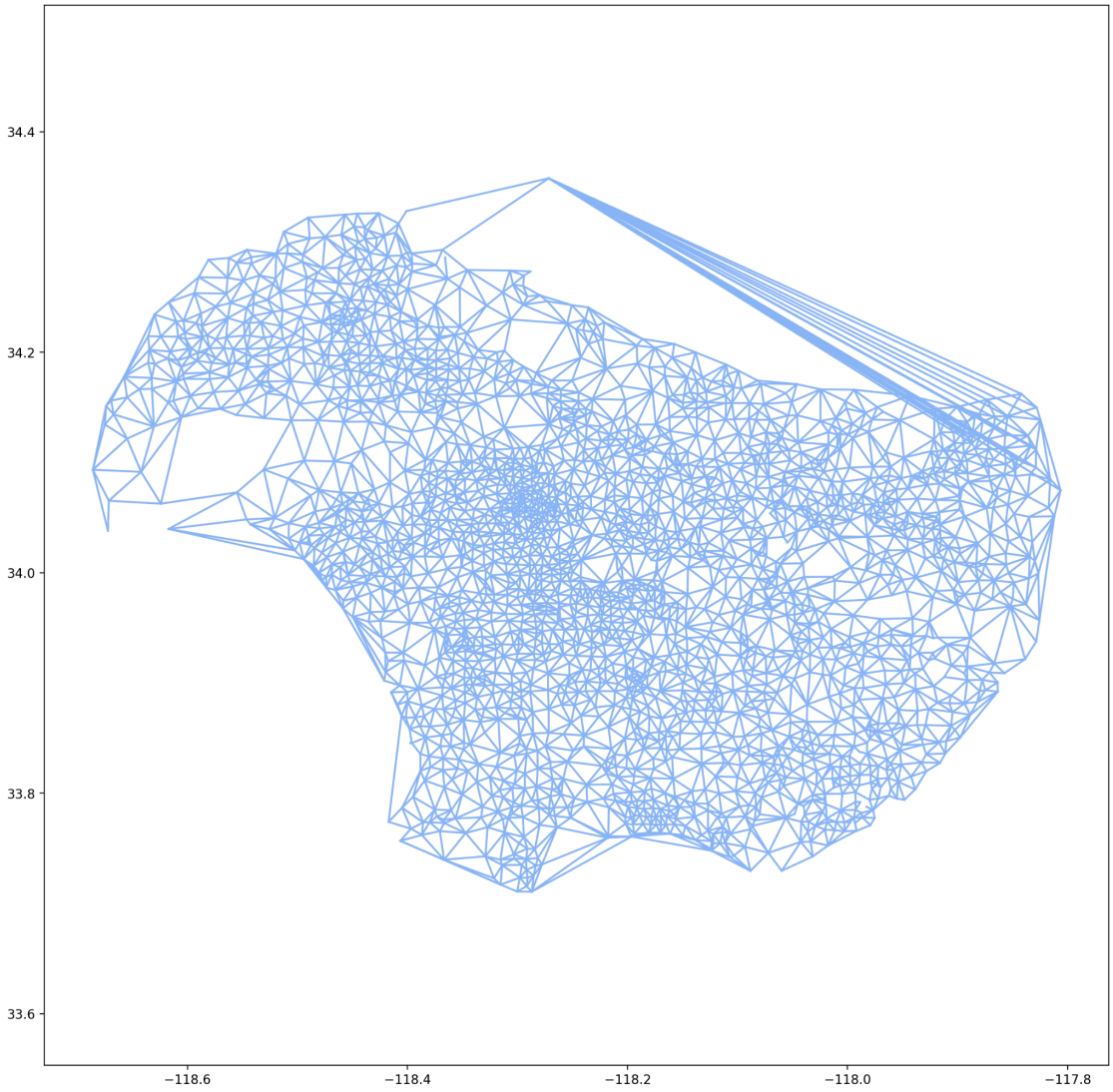
extra distance(v,s) = distance of shortest path(v,s) - euclidean distance(v,s)

Use the coordinates of v and s to get the euclidean distance between them. Calculate the ex- tra distance between all pairs of points. The top 20 pairs with highest extra distance will be the new edges we suggest. Print the source and destination of these pairs and write them in your report. Create these new edges in the graph, and plot the resultant graph on actual coordinates. What is the time complexity of the strategy?

The resultant graph on actual coordinates is illustrated below and the time complexity of this strategy is around O(E.logV) where V is the number of vertices and E is the total number of edges because the main time-consuming part at here is the Dijkstra's shortest path algorithm, which has a time complexity of O(E.logV). For the Subtraction of receiving the final distance difference, the time complexity is supposed to be $O(V^2)$. However, since it's implemented with numpy, which probably is programmed in C with time complexity of O(V) or even faster. Therefore, it is ignorable compared to O(E.logV).

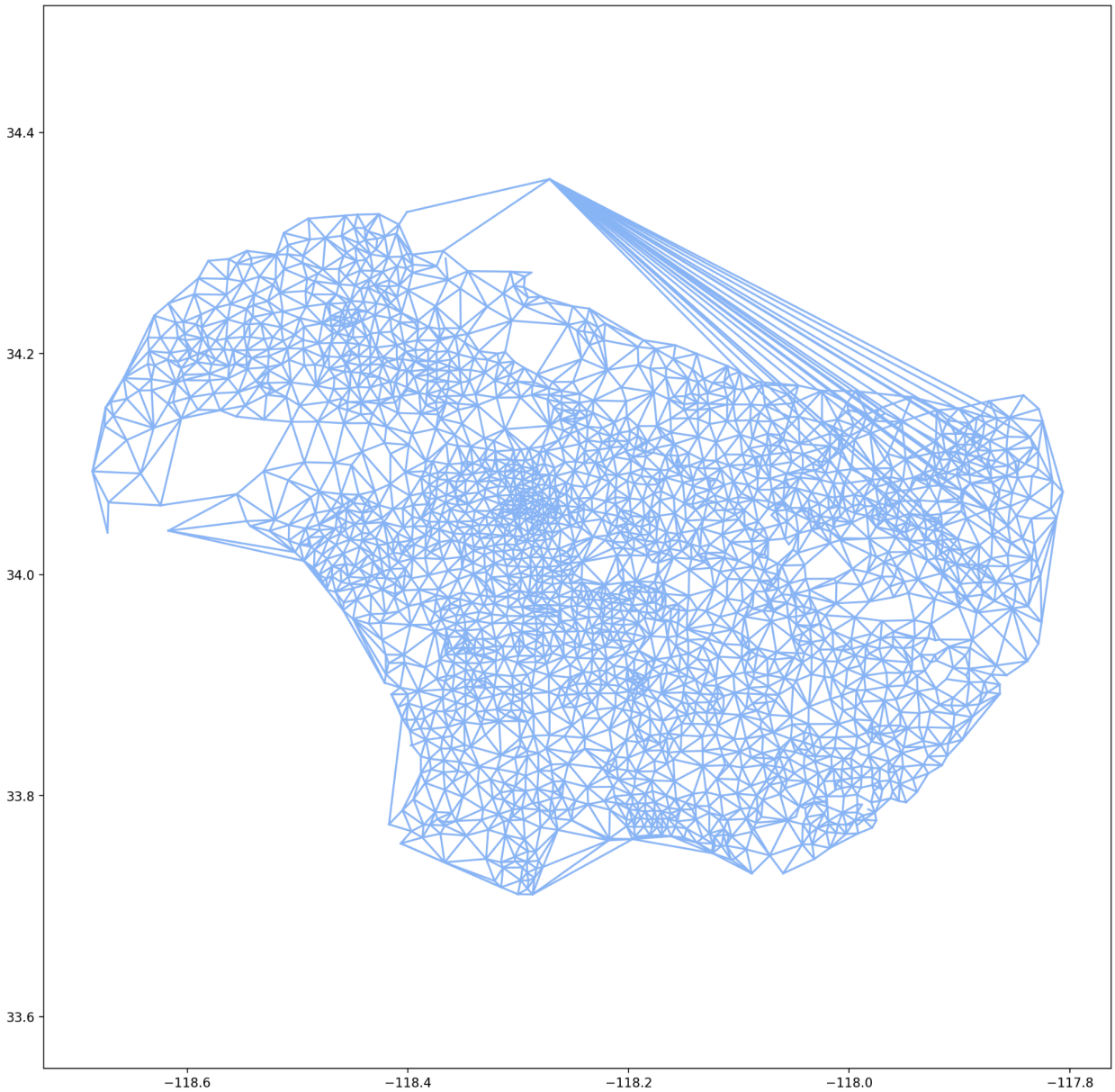
The top 20 pairs with highest extra distance are the following:

- Source, Destination: (2140, 2418), Extra Distance: 0.20474
- Source, Destination: (382, 2418), Extra Distance: 0.20195
- Source, Destination: (383, 2418), Extra Distance: 0.19896
- Source, Destination: (391, 2418), Extra Distance: 0.19730
- Source, Destination: (1901, 2418), Extra Distance: 0.19669
- Source, Destination: (381, 2418), Extra Distance: 0.19652
- Source, Destination: (2163, 2418), Extra Distance: 0.19515
- Source, Destination: (386, 2418), Extra Distance: 0.19462
- Source, Destination: (390, 2418), Extra Distance: 0.19458
- Source, Destination: (1904, 2418), Extra Distance: 0.19407
- Source, Destination: (45, 2418), Extra Distance: 0.19392
- Source, Destination: (1906, 2418), Extra Distance: 0.19391
- Source, Destination: (2158, 2418), Extra Distance: 0.19390
- Source, Destination: (1902, 2418), Extra Distance: 0.19386
- Source, Destination: (392, 2418), Extra Distance: 0.19370
- Source, Destination: (393, 2418), Extra Distance: 0.19369
- Source, Destination: (2159, 2418), Extra Distance: 0.19362
- Source, Destination: (48, 2418), Extra Distance: 0.19355
- Source, Destination: (2162, 2418), Extra Distance: 0.19349
- Source, Destination: (47, 2418), Extra Distance: 0.19336


34.4
34.2
34.0
33.8
33.6
−118.6
−118.4
−118.2
−118.0
−117.8

QUESTION 20:

Strategy 2 (geo distance, static, with frequency): In strategy1, we are using the geographical distance between points to decide which new road to create. However, in the real world, some pairs are more in demand than other pairs. Assume that you know the frequency of travel between every pair. Use the frequency as the weight to multiply the difference. i.e
weighted extra distance(v,s) = extra distance(v,s) * frequency(v,s)
where frequency(v,s) is a random integer between [1,1000]. Use the coordinates of v and s to get the euclidean distance between them. Calculate the weighted extra distance between all pairs of points. The top 20 pairs with highest weighted extra distance will be the new edges we suggest. Print the source and destination of these pairs and write them in your report. Create these new edges in the graph, and plot the resultant graph on actual coordinates. What is the time complexity of the strategy?



The resultant graph on actual coordinates is illustrated above and similar to Q19, the time complexity of Dijkstra's shortest path algorithm is O(E.logV). However, this time, we have to multiply the distance difference

matrix with the frequency matrix, which has a time complexity of $O(V^2)$. Therefore, the total time complexity is $O(E.logV) + O(V^2)$.

The top 20 pairs with highest extra distance are the following:

- Source, Destination: (47, 2418), Weighted Extra Distance: 192.39530
- Source, Destination: (393, 2418), Weighted Extra Distance: 190.39247
- Source, Destination: (394, 2418), Weighted Extra Distance: 187.01942
- Source, Destination: (452, 2418), Weighted Extra Distance: 184.07887
- Source, Destination: (2171, 2418), Weighted Extra Distance: 183.52048
- Source, Destination: (2173, 2418), Weighted Extra Distance: 182.83703
- Source, Destination: (384, 2418), Weighted Extra Distance: 182.47690
- Source, Destination: (2156, 2418), Weighted Extra Distance: 179.17012
- Source, Destination: (2053, 2418), Weighted Extra Distance: 178.42716
- Source, Destination: (2146, 2418), Weighted Extra Distance: 177.57198
- Source, Destination: (2079, 2418), Weighted Extra Distance: 177.34854
- Source, Destination: (2222, 2418), Weighted Extra Distance: 177.18152
- Source, Destination: (386, 2418), Weighted Extra Distance: 176.91278
- Source, Destination: (2155, 2418), Weighted Extra Distance: 175.34277
- Source, Destination: (2191, 2418), Weighted Extra Distance: 175.01971
- Source, Destination: (2226, 2418), Weighted Extra Distance: 174.65783
- Source, Destination: (2418, 2471), Weighted Extra Distance: 173.87486
- Source, Destination: (53, 2418), Weighted Extra Distance: 173.45512
- Source, Destination: (390, 2418), Weighted Extra Distance: 172.78550
- Source, Destination: (1956, 2418), Weighted Extra Distance: 172.23827

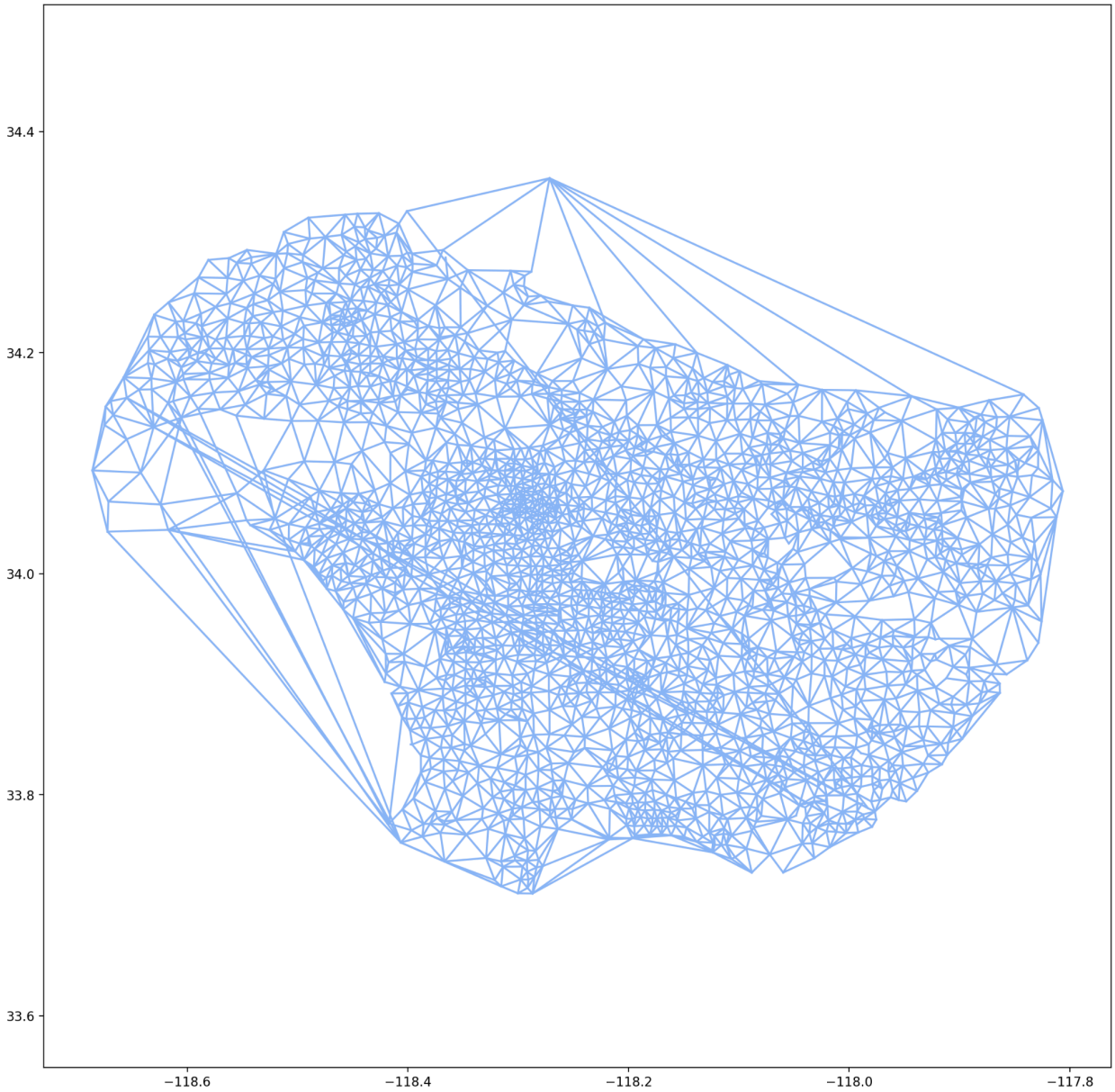
QUESTION 21:

Strategy 3 (geo distance, dynamic): In the above strategy, we are creating all roads at the same time. This time, we will create the roads one by one.

Repeat 20 times: i) Compute extra distance between all the pairs in the graph.

ii) Create a road between the pair with highest extra distance and update the graph. iii) Print the source and destination of this new edge.

Report the source and destination of the 20 new edges. plot the final graph on actual coordinates. What is the time complexity of the strategy?



The final graph on actual coordinates is shown above. The time complexity of this algorithm is around O(N.ElogV), where N is the number of repeats.

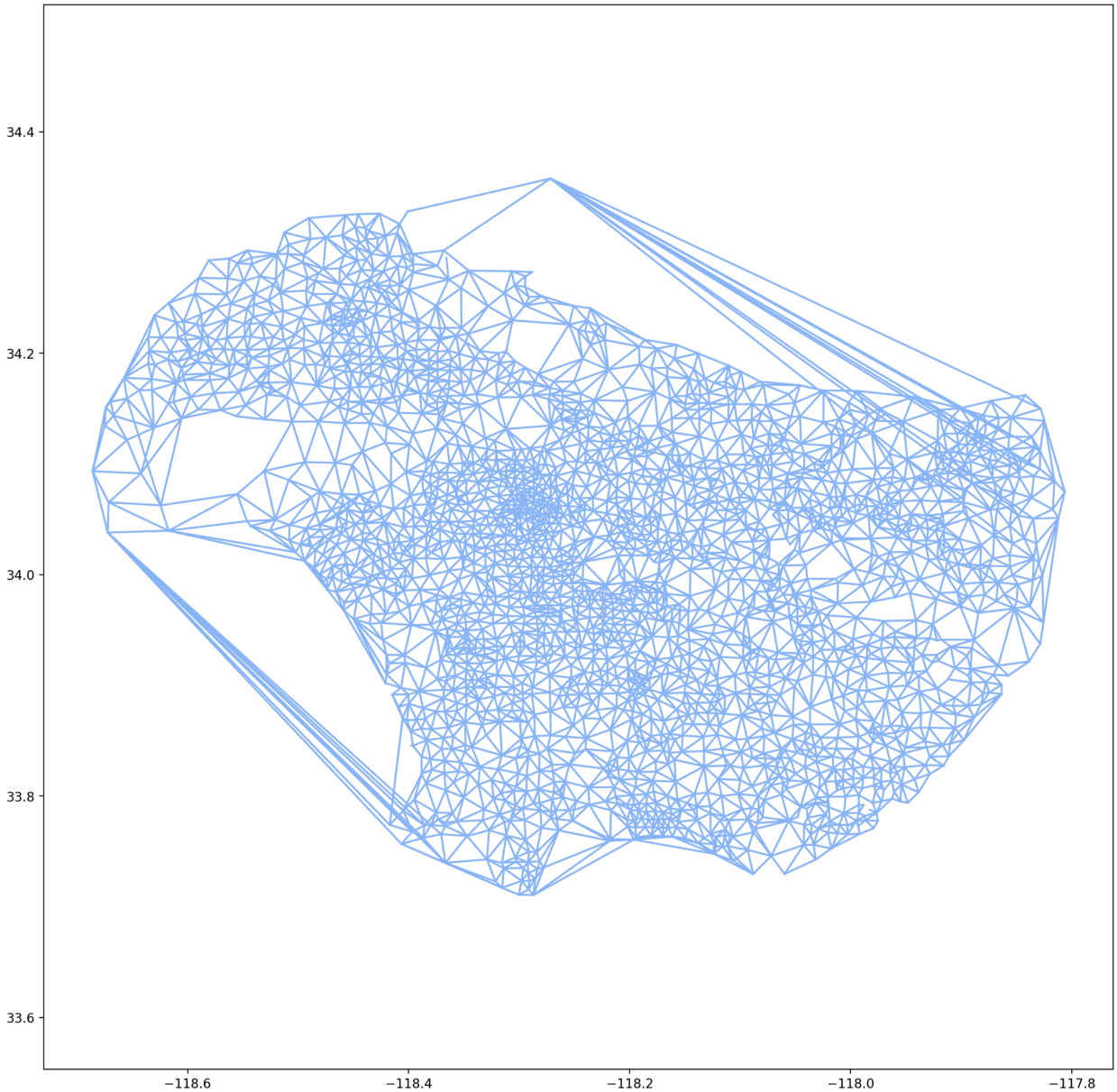
The top 20 pairs with highest extra distance are the following:

- Source, Destination: (2140, 2418), Extra Distance: 0.20474
- Source, Destination: (285, 2418), Extra Distance: 0.19004
- Source, Destination: (1956, 2418), Extra Distance: 0.17665
- Source, Destination: (2241, 2418), Extra Distance: 0.16490

- Source, Destination: (1699, 1700), Extra Distance: 0.15982
- Source, Destination: (509, 2418), Extra Distance: 0.14485
- Source, Destination: (1684, 2418), Extra Distance: 0.11692
- Source, Destination: (1700, 2417), Extra Distance: 0.10567
- Source, Destination: (985, 1860), Extra Distance: 0.09794
- Source, Destination: (1699, 1860), Extra Distance: 0.09577
- Source, Destination: (1783, 2411), Extra Distance: 0.08951
- Source, Destination: (989, 1510), Extra Distance: 0.08380
- Source, Destination: (430, 2618), Extra Distance: 0.07732
- Source, Destination: (1860, 2417), Extra Distance: 0.07624
- Source, Destination: (144, 2618), Extra Distance: 0.07009
- Source, Destination: (1700, 1783), Extra Distance: 0.06700
- Source, Destination: (430, 2558), Extra Distance: 0.06657
- Source, Destination: (144, 2558), Extra Distance: 0.06551
- Source, Destination: (2128, 2618), Extra Distance: 0.06456
- Source, Destination: (1005, 1860), Extra Distance: 0.06455

QUESTION 22:

Strategy 4 (Travel time, static): We want to optimize to reduce the maximum extra travelling time. The extra travelling distance between 2 locations is the difference of the shortest traveling time and the straight line travel time. i.e
extra time(v,s) = travel time of shortest path(v,s) - euclidean distance(v,s)/travel speed(v,s)
travel speed(v,s) = distance of shortest path(v,s) / travel time of shortest path(v,s)
Use the coordinates of v and s to get the euclidean distance between them. Calculate the extra time between all pairs of points. The top 20 pairs with highest extra time will be the new edges we suggest. Print the source and destination of these pairs and write them in your report. Create these new edges in the graph, and plot the resultant graph on actual coordinates. What is the time complexity of the strategy? What is the time complexity of the strategy?



The resultant graph on actual coordinates is illustrated above and the time complexity of this strategy is around $O(2.ElogV) + O(V^2)$ where V is the number of vertices and E is the total number of edges. The reason for this is that we have to apply Dijkstra's shortest path algorithm 2 times to find the shortest path and shortest_time,

which results in a time complexity of O(2.ElogV), and we also have to compute the travel speed by dividing the shortest path by the shortest_time, which has a time complexity of $O(V^2)$.
The top 20 pairs with highest extra distance are the following:

- Source, Destination: (1699, 1700), Extra Time: 0.61056 hr
- Source, Destination: (1699, 1860), Extra Time: 0.53256 hr
- Source, Destination: (1699, 1783), Extra Time: 0.49601 hr
- Source, Destination: (1699, 1859), Extra Time: 0.48273 hr
- Source, Destination: (1699, 1782), Extra Time: 0.46548 hr
- Source, Destination: (1700, 2416), Extra Time: 0.45792 hr
- Source, Destination: (1699, 1882), Extra Time: 0.44822 hr
- Source, Destination: (1699, 1857), Extra Time: 0.43570 hr
- Source, Destination: (2144, 2418), Extra Time: 0.43050 hr
- Source, Destination: (2052, 2418), Extra Time: 0.42573 hr
- Source, Destination: (2051, 2418), Extra Time: 0.42348 hr
- Source, Destination: (2150, 2418), Extra Time: 0.42276 hr
- Source, Destination: (1699, 1861), Extra Time: 0.42198 hr
- Source, Destination: (1901, 2418), Extra Time: 0.42049 hr
- Source, Destination: (1904, 2418), Extra Time: 0.42004 hr
- Source, Destination: (382, 2418), Extra Time: 0.41971 hr
- Source, Destination: (430, 1700), Extra Time: 0.41959 hr
- Source, Destination: (389, 2418), Extra Time: 0.41952 hr
- Source, Destination: (1699, 1881), Extra Time: 0.41942 hr
- Source, Destination: (225, 2418), Extra Time: 0.41930 hr

QUESTION 23:

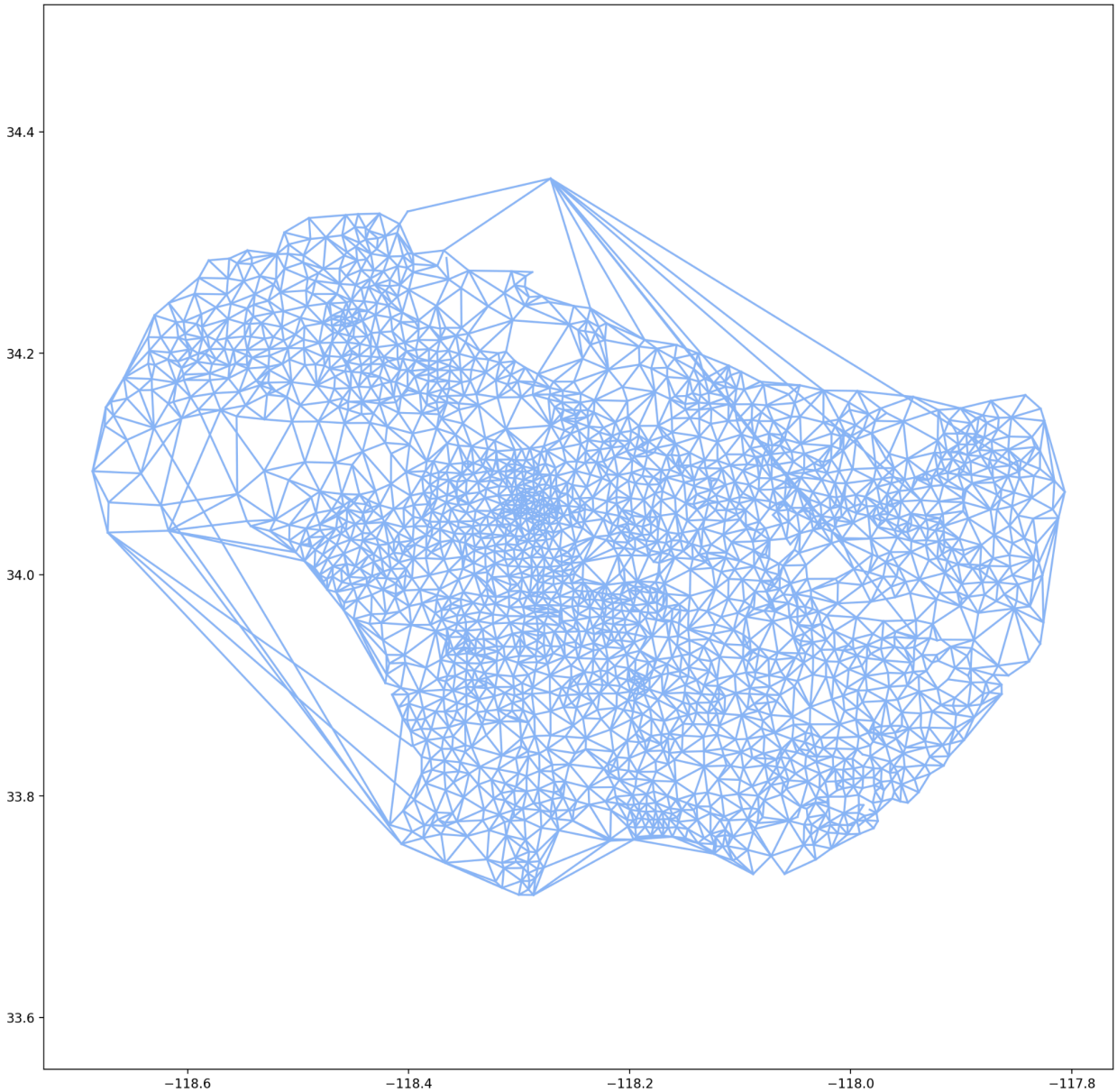
Strategy 5 (Travel time, dynamic): Similar to strategy 3, this time we will create roads one by one while optimizing the travel time.

Repeat 20 times:

i) Compute extra time between all the pairs in the graph.

ii) Create a road between the pair with highest extra time and update the graph. iii) Print the source and destination of this new edge.

Report the source and destination of the 20 new edges. plot the final graph on actual coordinates. What is the time complexity of the strategy?



The final graph on actual coordinates is shown above. The time complexity of this algorithm is around $O(N.2.ElogV) + O(N.V^2)$, where N is the number of repeats.

The top 20 pairs with highest extra distance are the following:

- Source, Destination: (1699, 1700), Extra Time: 0.61056 hr
- Source, Destination: (2144, 2418), Extra Time: 0.43050 hr
- Source, Destination: (2418, 2634), Extra Time: 0.40086 hr

- Source, Destination: (2072, 2418), Extra Time: 0.39170 hr
- Source, Destination: (1699, 1860), Extra Time: 0.37998 hr
- Source, Destination: (311, 2418), Extra Time: 0.36396 hr
- Source, Destination: (22, 2418), Extra Time: 0.32931 hr
- Source, Destination: (988, 1700), Extra Time: 0.32361 hr
- Source, Destination: (2248, 2418), Extra Time: 0.30973 hr
- Source, Destination: (1700, 2417), Extra Time: 0.30452 hr
- Source, Destination: (989, 1510), Extra Time: 0.27611 hr
- Source, Destination: (507, 2418), Extra Time: 0.27074 hr
- Source, Destination: (1699, 1781), Extra Time: 0.25922 hr
- Source, Destination: (1860, 2417), Extra Time: 0.25646 hr
- Source, Destination: (144, 1783), Extra Time: 0.21558 hr
- Source, Destination: (1700, 1783), Extra Time: 0.21119 hr
- Source, Destination: (988, 1678), Extra Time: 0.20903 hr
- Source, Destination: (2135, 2418), Extra Time: 0.19896 hr
- Source, Destination: (1699, 2402), Extra Time: 0.19655 hr
- Source, Destination: (1005, 1510), Extra Time: 0.18877 hr

QUESTION 24:

Strategy comparison. Compare the following strategies:

a) 1 vs 2 - compare and analyze the results. which is better? why?

b) 1 vs 3 - compare and analyze the results. which is better? why?

c) 1 vs 4 - compare and analyze the results. which is better? why?

d) statics vs dynamic - Considering we want to improve the overall road network by constructing new roads, is either of them the optimal strategy?

i) if yes, which one and why?

ii) if not, what would be a better strategy to construct roads and is it optimal? What is the time complexity?

(assume that you want to reduce travelling distances and you know before hand that we have to construct 20 new roads).

e) Open ended question : Come up with new strategy. You are free to make any assumptions. You don't have to necessarily optimize traveling time or distance, you can come up with any constraints. justify your strategy.

a) There isn't a huge difference between the result of strategy 1 and 2 according to the plots shown above. Basically, most of the new edges we suggest are related to the node 2418 and connects to other nodes, which is mostly on the other side across the mountain area. However, after a closer look, one can notice that with strategy 2, those nodes besides the node 2418 spread wider to different areas/cities. Hence, we can say that strategy 2 is better compared to strategy 1 as the 20 new edges generated through this method spread to more area, which is in general better than having 20 new edges connecting the same area.

b) It is obvious that strategy 3 is hugely better than strategy 1. From the plots, one can see that with strategy 3, the 20 new generated roads successfully connect various different areas and cities, unlike strategy 1 which mostly connects to 2418 in the mountain area. We can see that there are new roads like highways that directly connect cities across the ocean, mountain area, and north-west side of LA to the south-east side of LA. This high variety of connections will definitely benefit the city more than the suggestion roads generated by strategy 1. Moreover, from the reported value of extra distance generated at the end of each algorithm, we can see that after strategy 3, the maximal extra distance is only 0.06455 X 69 =4.45395 miles, which is way smaller than the maximal extra distance after strategy 1, which is 0.19336 X 69 = 13.34184 miles. Therefore, strategy 3 is better.

c) Based on the plots, we can see that the strategy 4 is better than the strategy 1 as it recognizes not only the need of new roads to directly connect the mountain area (node 2418) to other cities on the other side of the mountain but also the better direct solution to travel from Malibu to other cities by building highways across the ocean to reduce the total time to travel from any point in LA to another point.

d) Generally speaking, dynamic strategy is better than statics strategy if we considering the improvement of the overall road network by constructing new roads, which can be proven based on both maximal extra distance (dynamic vs. statics: 4.45 miles vs. 13.34 miles) or maximal extra time (dynamic vs. statics: 0.19 hr vs. 0.42 hr) after 20 new roads construction. However, even though the dynamic strategy is better, it is not the optimal strategy because it doesn't consider the reality of such construction such as the cost, safety, feasibility, time, etc. A better strategy would be to take all of the aforementioned reality factors into consideration. However, this will results in a very high time complexity at least O(N.M.ElogV) where N is the number of new road to construct and M is the number of reality factors that we need to use Dijkstra's shortest path algorithm O(ElogV) to find the optimal value. V is the number of vertices and E is the total number of edges. If there is further calculation needed such as computing the weights between each factor or other mathematical formulation, then the time

complexity will even increase to a higher value. For example, adding matrix multiplication at the end of each road construction decision will result in increasing the time complexity by $O(N.V^2)$.

e) An intuitive idea is a dynamic strategy to optimize between the pros and cons of constructing the new road. The pros are, for instance, less time and distance to travel and the cons are the cost of such roads such as time to construct and money. In order to optimize this, we have to come up with an objective function that combines both pros and cons in some mathematical formulation or some weight factors to balance the importance among each factor. Other constraints might also need to be introduced to make such a decision of construction feasible. With such an idea and algorithm, I think we can find the optimal road to construct.