Algorithms Term Project Documentation

A TRAVEL GUIDE

-By Team C

Team Contributions

“Eighteen dedicated minds, guided by Prof. David, sculpted this project through sheer hard work and determination”.			
Team Person Name	**Email Address**	**Contribution**	**Contribution on a Scale of 10**
Sirish	schejerl@pnw.edu	Sirish demonstrated exceptional algorithmic proficiency, contributing significantly to all three milestones and serving as the coding team lead. His role included maintaining the GitHub repository and skillfully integrating codes for Milestones 1 and 2, ensuring seamless collaboration and progress.	10
Shoaib	moham124@pnw.edu	Shoaib played a pivotal role in developing and implementing algorithms across all three milestones, leading the coding team with expertise. His dedication extended to maintaining the GitHub repository and devising a code for collecting distance data from Google Maps API, enhancing the accuracy of city distances crucial for our travel maps.	10
SriRam	samineni@pnw.edu	Sriram's expertise in graph visualization was instrumental in Milestones 1, 2 and 3, bringing our travel maps to life. His commitment to creating clear and insightful visual representations of city names and distances significantly enriched the user interface of our project.	10
Samhitha	kasim@pnw.edu	Samhitha focused on vital data acquisition, specializing in fetching weather data from AccuWeather API and contributing to data collection in all project milestones. Her efforts enhanced the depth and relevance of our travel maps, incorporating real-time weather conditions	10

		for a comprehensive user experience.	
Nitish	vemuri8@pnw.edu	Nitish excelled in visualizing graphs for the shortest distances, contributing to Milestones 2 and 3. His work not only enhanced the aesthetic appeal of our travel maps but also played a crucial role in presenting concise and informative data to users.	10
Abhigna	vchiluka@pnw.edu	Abhigna's rigorous testing of algorithms, including Dijkstra's, Bellman-Ford, and A*, significantly contributed to the accuracy of our shorter path calculations. Her dedication to ensuring the reliability and efficiency of our algorithms bolstered the overall quality of the project.	10
Sunil	skalagar@pnw.edu	As the team lead, I have managed both coding, testing and documentation teams. I not only contributed to flowcharts in documentation but also presented all milestones to the class and professor, showcasing effective communication and organizational abilities.	10
Lokesh	ssanise@pnw.edu	Lokesh's contributions encompassed creating sequence diagrams, class diagrams, and leading the Verification and Validation aspect of our project. His attention to detail and systematic approach ensured the project's structural integrity and adherence to quality standards.	10
Sai Kiran	smittap@pnw.edu	Sai Kiran played a crucial role in developing flowcharts and led the Verification and Validation efforts for our algorithms. His meticulous approach to ensuring the accuracy and reliability of our algorithms contributed to the overall robustness of the project.	10
Suraj	sannepal@pnw.edu	Suraj took on the role of a meticulous testing expert for the visualization part, ensuring that our travel maps met the highest standards of functionality and accuracy. His efforts in quality assurance played a crucial role in delivering a polished and reliable user experience.	10
Sumanth	bontha@pnw.edu	Sumanth's expertise in class diagrams and sequence diagrams, coupled with thorough testing of the code, significantly contributed to the project's documentation and structural clarity. His commitment to quality control added depth to our project's technical foundation.	10

Sri Latha	skalava@pnw.edu	SriLatha, as the Sub Team Lead, provided invaluable leadership, focusing on understanding the pros and challenges of the project. Her strategic insights and analytical skills enhanced our project's overall understanding, contributing to its success.	10
Shalini	skandlam@pnw.edu	Shalini's dedication to documentation and dataset testing showcased her commitment to project integrity. Her meticulous testing of datasets and contribution to documentation ensured the accuracy and reliability of the information presented in our travel maps.	10
Sree Sai	adusumi1@pnw.edu	As the Team Lead for Documentation, Sree Sai played a pivotal role in documenting datasets and creating readme files for Milestone 3. H organizational skills and attention to detail ensured that the project's documentation was comprehensive and accessible.	10
Sumana	snalubol@pnw.edu	Sumana's content creation skills and integration of various documentation components showcased her ability to present a cohesive narrative. Her work in integrating all parts of documentation added depth to the project's comprehensive understanding.	10
Sarayu	skethidi@pnw.edu	Sarayu's focus on testing codes and datasets related to weather data and distances contributed significantly to the reliability of our travel maps. Her efforts in quality assurance ensured that users could trust the accuracy of the information presented.	10
Sravya	gadde0@pnw.edu	Sravya's contributions in testing codes and datasets, particularly in evaluating the visualization of graphs, played a crucial role in ensuring the project's visual and functional integrity. Her commitment to results testing added a layer of quality control to the project.	10
Vakula	vbhima@pnw.edu	Vakula's expertise in the algorithmic part of documentation and flowchart creation contributed to the project's clarity and understanding. Her work in creating flowcharts added a visual dimension to the documentation, aiding in comprehension and analysis.	10

Abstract

This project represents a pioneering initiative aimed at transforming the landscape of inter-city travel planning through a crafted fusion of cutting-edge algorithms and integrated datasets. Centered on the core objective of uncovering the shortest paths within intricate graph structures, this endeavor combines crucial datasets, including inter-city distances, real-time weather dynamics spanning 48 hours, and curated sea-level information. A highly sophisticated computational framework built upon these datasets, aggregated and validated via a specialized Load Data Program to ensure integrity and accuracy.

The project showcases an array of established and cutting-edge algorithms tailored to unravel the most efficient pathways between graph nodes. This amalgamation of algorithms not only delineates shortest distances but also factors in real-world conditions such as weather influences and geographical features, elevating the precision of computed routes.

Beyond conventional route planning, the project introduces multi-modal pathfinding, accommodating various transportation modes and offering a holistic perspective on inter-city travel. Real-time adaptability, graph optimization strategies, and user-centric features further enrich the system and personalized travel recommendations.

The project's contributions extend to advanced graph visualization techniques, facilitating a deeper understanding of complex networks, and the integration of user-defined constraints and preferences for tailored route recommendations. This visualization not only simplifies the comprehension of computed paths but also serves as an educational tool, demystifying intricate graph-based algorithms for a broader audience. Additionally, the integration of user-defined constraints and preferences further enhances the system's utility, allowing users to tailor their route recommendations based on personalized criteria such as scenic routes, time optimization, or specific weather conditions.

Through its multifaceted approach, this project not only redefines benchmarks in route optimization but also establishes a blueprint for leveraging disparate datasets to enhance decision-making processes in travel planning. It stands as a testament to the potential of data integration and algorithmic innovation in reshaping the domain of shortestpath and optimal routing.

Introduction

The shortest path refers to the most direct or efficient route between two points in a network, graph, or any space with interconnected nodes or locations. Finding the shortest path is a fundamental problem in computer science and has diverse applications in various fields, including transportation, logistics, telecommunications, and more.

The concept of the shortest path is essential in scenarios where efficiency, time, or cost minimization is crucial. There are two primary types of shortest path problems

Single-Source Shortest Path, this involves finding the shortest path from a single source node to all other nodes in the graph or network. Classic algorithms for this problem include Dijkstra's algorithm and the Bellman-Ford algorithm.

The applications of finding the shortest path are vast and varied, ranging from GPS navigation systems and ride-sharing apps to network routing, logistics planning, circuit design, and more. The ability to compute the most efficient path between locations or nodes is crucial in optimizing resource allocation, time management, and decision-making in numerous real-world situations.

The shortest paths within graph structures are identified by a comprehensive set of algorithms in this project, reinventing route optimization. This project, which is primarily geared toward inter-city travel, combines 48-hour real-time weather data, intercity distances, and meticulously curated sea-level data. A specialized Program, which serves as the foundation of a sophisticated computational framework, aggregates, verifies, and harnesses these crucial datasets. Within this repository lie an array of well-known algorithms adept at swiftly uncovering the most efficient pathways between graph nodes. What sets this project apart is the harmonious fusion of these algorithms, showcasing remarkable accuracy and efficacy in validation mechanisms.

At its core, this implementation features an exhaustive compilation of distances originating from a primary city to various destination cities. Augmenting this dataset are real-time weather insights, sourced meticulously from AccuWeather, providing a 48-hour forecast for each city. Moreover, a comprehensive repository of sea-level data for all cities involved enriches the analysis, enabling potential correlations and geographical optimizations in routing strategies.

The crux of this project lies in its ability to compute optimal gasoline consumption for intercity travel, leveraging the synergy of critical datasets—comprising the distance matrix, real-time weather updates, and sea-level information. Orchestrated through Java, the code seamlessly integrates these datasets, culminating in a meticulously organized JSON-formatted output that simplifies data comprehension and accessibility.

Ensuring the integrity and accuracy of accessed datasets stands as a pivotal facet, facilitated by a specialized Loading Data Validation Program. This program rigorously scrutinizes acquired data, validating its authenticity against expected criteria. By fortifying the foundation of computations and analyses, this validation process instills unwavering confidence in the reliability of project outcomes.

The culmination of these efforts not only empowers the computation of optimal travel routes but also presents an opportunity to visualize inter-city data, fostering a deeper understanding of geographical correlations and enhancing decision-making processes.

This documentation aims to elucidate the multifaceted approach, methodologies employed, and the significance of integrating disparate datasets in the domain of path finding and route optimization.

Beyond the initial amalgamation of critical datasets, this project pioneers an advanced approach in data fusion, entwining inter-city distances, real-time weather dynamics, and intricate sea-level statistics. Through sophisticated algorithms and correlation analyses, the project unearths nuanced relationships between these datasets, revealing hidden patterns that significantly impact route optimization. This approach not only identifies optimal pathways but also considers weather-influenced variables and geographical features, elevating the precision of the computed routes.

The project is not confined to static analyses; rather, it embraces dynamism through predictive modeling. By leveraging historical weather patterns and sea-level fluctuations, the system forecasts potential changes in route efficiency. This predictive modeling ensures adaptability, enabling the system to proactively suggest alternate routes in response to anticipated weather variations or evolving geographical conditions, fostering resilience and efficiency in travel planning.

A pivotal aspect of this project lies in its user-centric approach to data visualization. Leveraging intuitive graphical representations and interactive interfaces, the system offers a seamless visual portrayal of inter-city data. Through visually intuitive maps, users gain comprehensive insights into route variations based on weather patterns, elevation changes, and optimized fuel consumption, empowering informed decision-making in travel planning.

The project's architecture is designed with scalability and extensibility in mind. With the potential for further expansion, the framework accommodates additional datasets and algorithms, fostering continual advancements in route optimization strategies. This scalable architecture ensures adaptability to diverse scenarios, catering not only to inter-city travel but potentially extending its applications to diverse domains such as logistics, urban planning, and transportation management.

As part of its ethos, this project thrives on collaboration and open-source contributions. Encouraging a collaborative ecosystem, it invites expertise from diverse domains—be it data science, transportation engineering, or geographic information systems (GIS). Open-source accessibility fosters innovation and continual refinement, driving the evolution of route optimization paradigms while nurturing a community dedicated to pushing the boundaries of computational route planning.

In summary, this documentation serves as a comprehensive guide to the intricacies of this pioneering project. It elucidates the meticulous data integration strategies, advanced computational methodologies, and the profound impact on revolutionizing pathfinding and route

optimization. Through its multifaceted approach, this project not only redefines the benchmarks in route planning but also establishes a blueprint for leveraging diverse datasets in decision-making processes, igniting a new era in the domain of optimal routing.

Data Collection and Data Pre-processing:

In our project, we have meticulously gathered comprehensive data for 120 cities spanning six states - Nevada, Oregon, Washington, California, Wyoming, and Arizona. This dataset encapsulates crucial details, encompassing the dynamic spectrum of environmental factors. Firstly, we obtained **real-time weather conditions** for each city, meticulously sourced from the AccuWeather API, capturing the nuances of atmospheric phenomena on an hourly basis. Additionally, we meticulously calculated the **distances between every city and the remaining 119 cities**, leveraging advanced mapping technologies to unveil intricate spatial relationships. Moreover, our dataset is enriched with **the sea level information for each city**, providing a vertical dimension to our understanding of these diverse urban landscapes.

Weather details data collection:

- The data includes information on weather conditions (e.g., cloudy, sunny, rainy) and temperature recorded every hour for a total duration of 48 hours.

I. Introduction:

- In this project, the collection and preprocessing of weather data from AccuWeather are crucial for understanding weather patterns across multiple states.
- AccuWeather has been chosen for its reliable and extensive weather data.

II. Data Collection:

- To begin, create a developer account on AccuWeather and obtain the API key necessary for data retrieval.
- Identify in total 120 cities from the following 15 states: Nevada, Oregon, Washington, California, Wyoming, Arizona, Wyoming, New Mexico, Utah, Oregon, North Dakota, Nebraska, Texas, Kansas, Idaho.

- The AccuWeather API endpoint for current conditions is: http://dataservice.accuweather.com/currentconditions/v1/.
- Parameters like city key, language, and details level are essential for accurate data retrieval.
- Example API request: GET http://dataservice.accuweather.com/currentconditions/v1/{cityKey}?apikey={yourApiKey}&language={language}&details={detailsLevel}.

III. Google Sheets Integration:

- Create a project on Google Cloud Platform, enable Google Sheets API, and generate credentials.
- Structure the Google Sheet with columns for city, state, timestamp, weather condition, and temperature.

Sea Level Data Collection:

I. Introduction:

- To capture the vertical dimension of our dataset, we employed **whatismyelevation.com** for the calculation of sea level information for each city. This online platform proved invaluable, ensuring precise elevation calculations.

II. Data Collection:

- The process involved manually recording the sea level data for every city considered in our study. The meticulous use of Whatismyelevation.com reflects our commitment to accuracy and thoroughness in documenting the environmental characteristics of each city.

III. Data Storage in Google Sheets:

- Subsequently, these diligently obtained sea level values were thoughtfully integrated into our Google Sheets, seamlessly merging with other dimensions of data to create a comprehensive dataset for analysis and exploration.

City-to-City Distances Data Collection:

I. Introduction:

- For an exhaustive understanding of spatial relationships, we systematically collected city-to-city distance data for each of the 120 cities, resulting in a comprehensive 120x120 matrix.

II. Data Collection:

- To achieve this, we leveraged the capabilities of Google Maps by creating a Google Cloud account and obtaining a dedicated API key.
- Subsequently, a Java code was meticulously crafted to interface with the Google Maps API, enabling the retrieval of accurate distance information between each city and the remaining 119 cities.
- This intricate process underscores our commitment to spatial precision in elucidating the intricate interconnections within our dataset.

III. Data Storage in Google Sheets:

- Automates data transfer to Google Sheets, enhancing efficiency and accuracy.
- Provides a complete and thorough dataset for geographical analysis.
- Facilitates real-time collaboration and easy data manipulation in a user-friendly environment.

IV. Overview of the code written to collect distances from google sheets:

- **Automation of Distance Calculation:** The script uses the Google Maps API to automate the calculation of distances. This reduces the possibility of human error and saves a great deal of time by doing away with the necessity for manual computations or lookups.
- **Batch Processing:** The script carries out batch computations by working with a list of cities. This increases efficiency, particularly for large datasets, as it may compute numerous city-to-city distance estimates in a single operation.
- **Distance Conversion Feature:** The script has the ability to convert distances between kilometers and miles. It does this by default. The data is adaptable to a variety of use cases and user preferences thanks to its dual measurement output.
- **Data Structuring and Organization:** An accessible and well-organized dataset is produced by using a dictionary structure to link each city with its corresponding distance

to every other city. The retrieval and processing of data is made easier by this arrangement.

- **Integration with Pandas for Data Handling:** Pandas is a robust data manipulation library that may be used to handle the calculated data in an effective manner. These include functions that are necessary for analysis, such as data sorting, filtering, and aggregation.
- **Export to Excel Format:** The script makes the data easily shareable and compatible with popular data analysis tools by transforming the data into a DataFrame and exporting it to an Excel file. This makes it easier to integrate or do additional analysis for other projects.
- **Scalability and Adaptability:** The script's architecture may be modified to fit various geographical areas or distance measures, and it is adaptable to support a greater number of cities.
- **Potential for Integration:** The script's structured output can be effortlessly incorporated into more extensive systems or initiatives, including apps for trip planning, logistics planning, or geographic analysis.

Algorithms

Dijkstra's Algorithm to find the shortest path

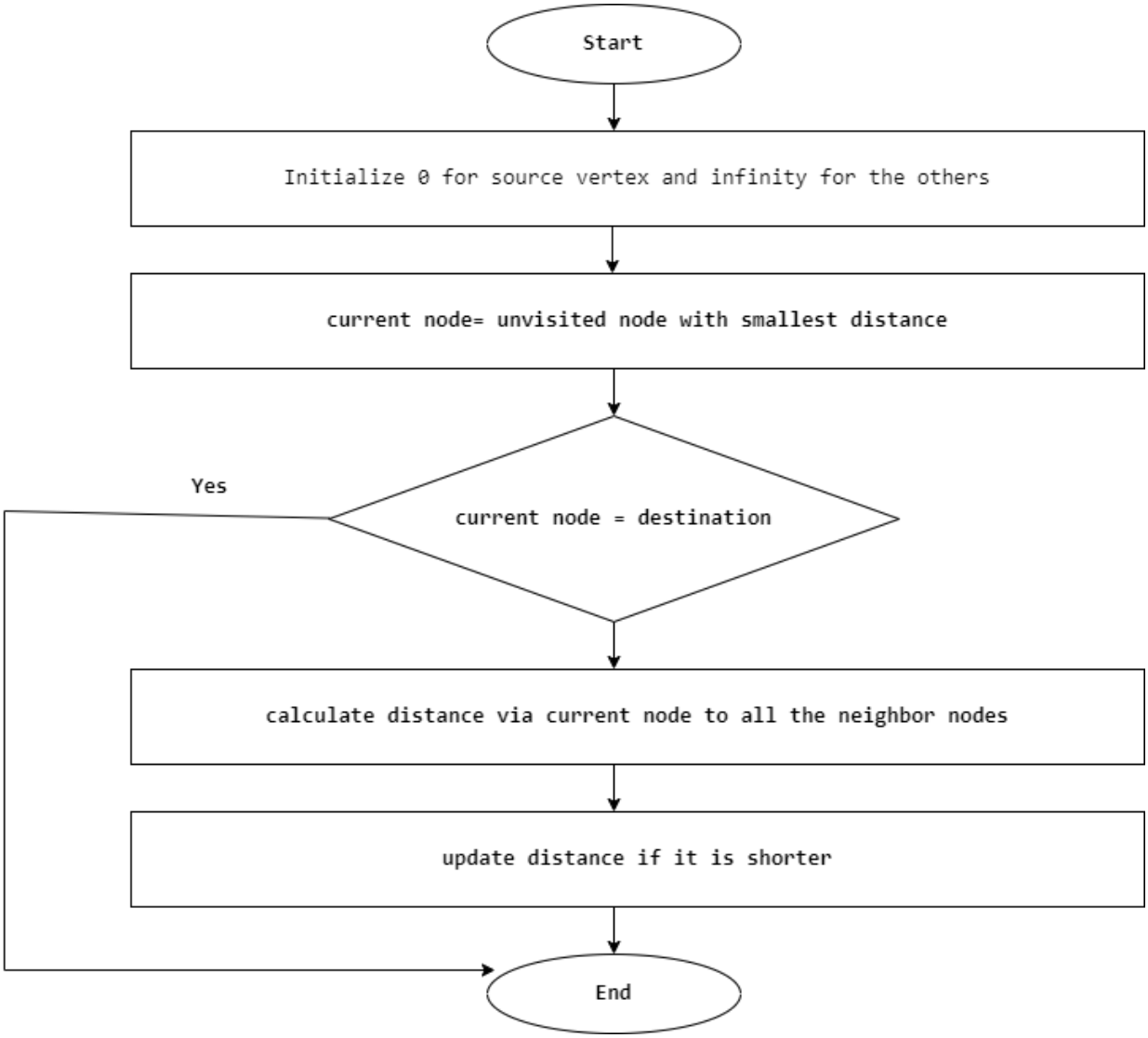
Dijkstra's algorithm is used to find the shortest distance between the nodes(cities). Since there are no negative values (Distance is never negative) we choose Dijkstra to find the shortest route from the source to the destination with the minimum distance possible. The cities and their connections are represented in an **adjacency matrix – adjacencyMatrix[],** where each cell holds the distance between two cities. The '**dijkstra**' method determines the shortest route between the source and destination cities employing Dijkstra's algorithm. It traverses the vertices, identifying the briefest path by evaluating the weights (distances) of connections among them.

Pseudo Code

```
function Dijkstra(adjacencyMatrix, startVertex, endVertex):
     numVertices = size of adjacencyMatrix
     visited[numVertices] = false
     dis[numVertices] =
      // Distance from startVertex to itself is zero
     dis[startVertex] = 0
     for count = 0 to numVertices - 1:
```

```
        minDistance =
        minVertex = -1
        // Find the vertex with the minimum distance
        for v = 0 to numVertices - 1:
           if not visited[v] and dis[v] < minDistance:
           minDistance = dis[v]
           minVertex = v
        visited[minVertex] = true
        // Updating distances of neighboring vertices
        for v = 0 to numVertices - 1:
           if not visited[v] and adjacencyMatrix[minVertex][v] != 0 and
           dis[minVertex] != infinity
           dis[minVertex] + adjacencyMatrix[minVertex][v] < dis[v]:
               dis[v] = dis[minVertex] + adjacencyMatrix[minVertex][v]
return dis[endVertex]
```

Flow Chart



Prims and Kruskal's algorithm for MST

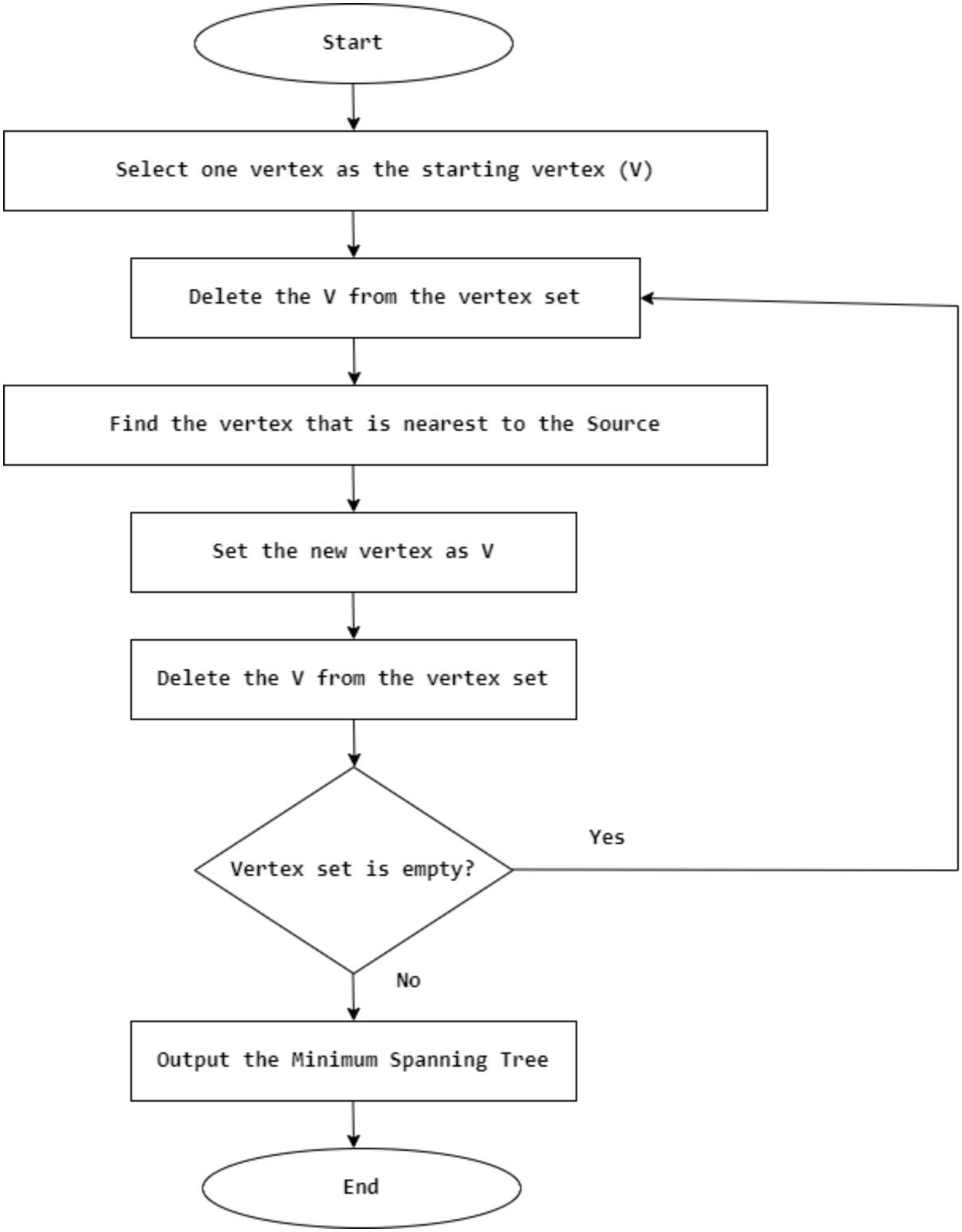
Prim's algorithm finds the minimum spanning tree of a graph, ensuring connectivity with the least total edge weight.

The file "distance_matrix.csv" contains distance information between 120 cities. Starting from an arbitrary city as the initial point(start_node), a priority queue is set up to manage the distances between cities. The algorithm proceeds by selecting the closest neighbouring cities (nodes with the shortest valid distances) from each city(node) successively. By doing this, it constructs a Minimum Spanning Tree (MST) by continually adding nodes based on the shortest available paths.

Pseudo Code

```
Q ← ∅
for each u ∈ V
     do key[u] ← ∞
           π[u] ← NIL
            INSERT(Q, u)
DECREASE- KEY(Q, r, 0)              key[r] ← 0
While Q ≠∅
       do u ← EXTRACT-MIN(Q)
            for each v ∈ Adj[u]
               do if v ∈ Q and w(u, v) < key[v]
                   then π[v] ← u
                         DECREASE-KEY(Q, v, w(u, v))
```

Flow Chart



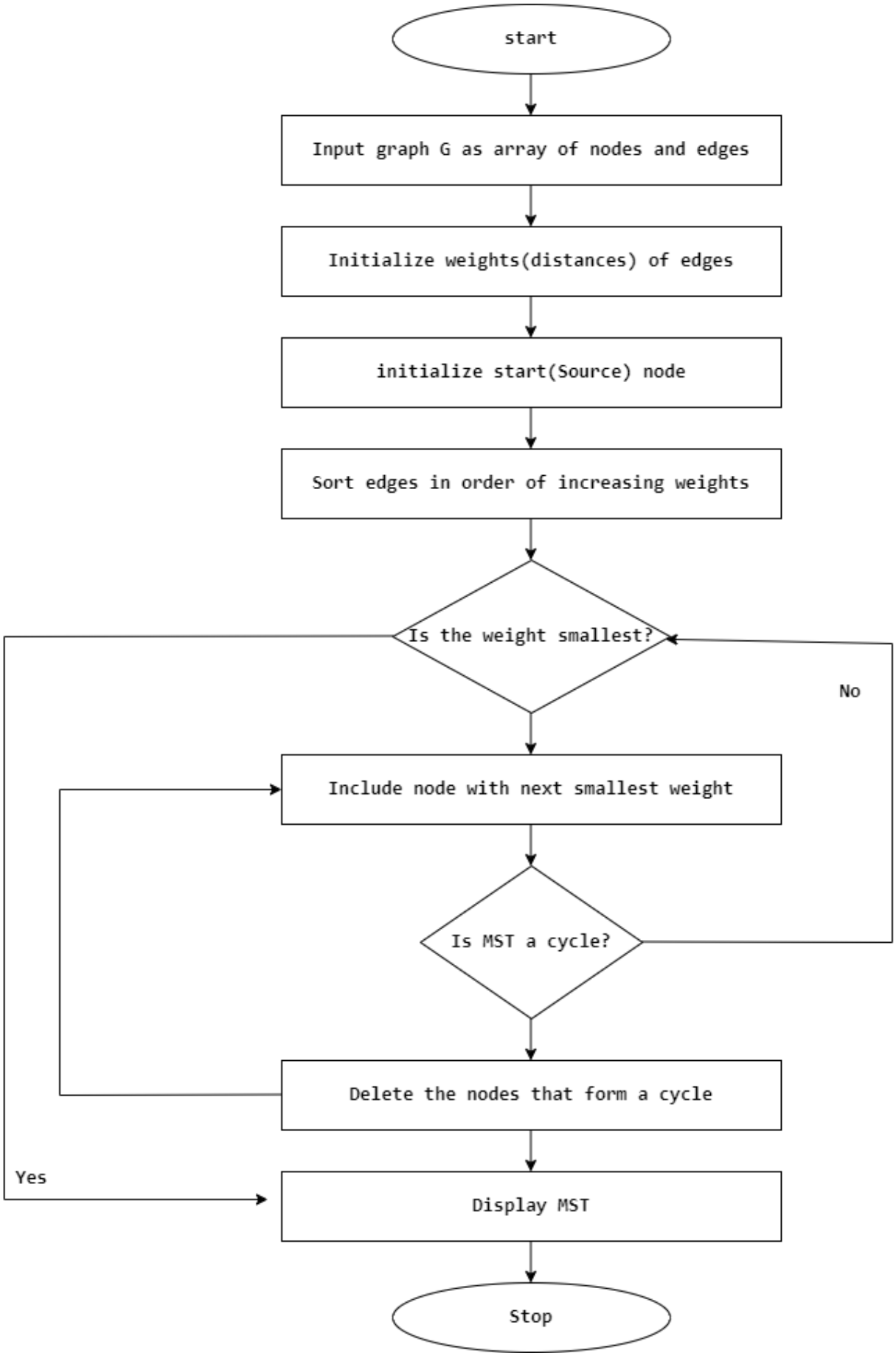
Kruskal's algorithm is a greedy algorithm that finds the minimum spanning tree of a connected, undirected graph by iteratively adding the smallest edge that doesn't create a cycle.

The file "distance_matrix.csv" contains distance information between 120 cities. Initially, for Kruskal's algorithm, the algorithm will consider each city as an individual node. The edges in the graph represent distances between pairs of cities. Kruskal's algorithm for Minimum Spanning Tree (MST) begins with all nodes as separate components and progressively adds edges based on their weights while ensuring that no cycles are formed. This process constructs an MST that connects all cities with the shortest possible total distance.

Pseudo Code

```
KRUSKAL(V, E, w)
A← Æ
for each vertex v Î V
      do MAKE-SET(v)
sort E into non-decreasing order by w
for each (u, v) taken from the sorted list
      do if FIND-SET(u) ¹ FIND-SET(v)
            then A ← A È {(u, v)}
                        UNION(u, v)
return A
```

Flow Chart


start
Input graph G as array of nodes and edges
Initialize weights(distances) of edges
initialize start(Source) node
Sort edges in order of increasing weights
Is the weight smallest?
No
Include node with next smallest weight
Is MST a cycle?
Delete the nodes that form a cycle
Yes
Display MST
Stop

Belman Ford and Dijkstra's Algorithm for shortest path

Bellman- Ford Algorithm aims to find the shortest path from a source node ('src) to a destination node ('dest') in a graph represented by an edge list ('edges'). The arrays are initialized for distances ('dest') in a graph represented by an edge list('edges'). The negative weights are detected and handled as needed.

Pseudo Code

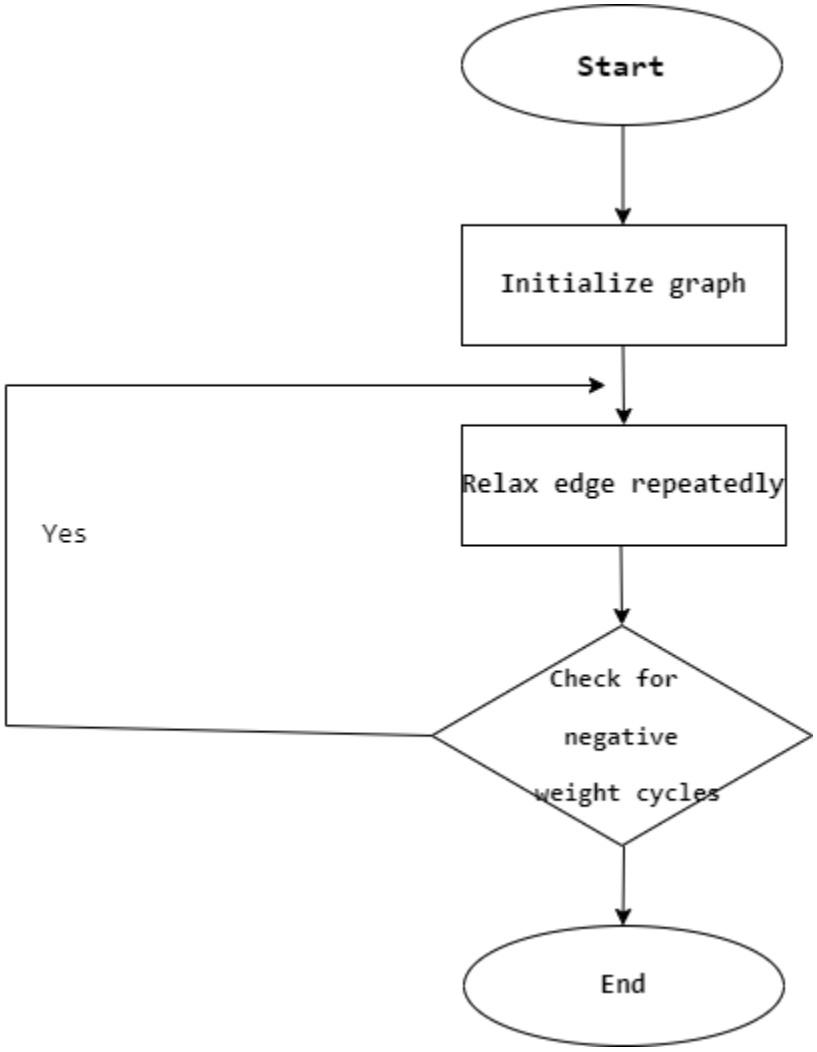
```
BellmanFord(graph, src, dest):
     dist[] = array with distances initialized to , except dist[src] = 0
     pred[] = array to store predecessors initialized to -1

     // Relax all edges |V| - 1 times
     for i = 1 to V-1:
     for each edge in edges:
          u = edge.start
          v = edge.end
          weight = edge.weight
          if dist[u] + weight < dist[v]:
               dist[v] = dist[u] + weight
               pred[v] = u

     // Check for negative weight cycles
     for each edge in edges:
     u = edge.start
     v = edge.end
     weight = edge.weight
     if dist[u] + weight < dist[v]:
          print "Graph contains negative weight cycle"
          return

     // Print the shortest paths
     print "Shortest paths from the source vertex:"
     printPath(src, dest, pred)
     print " (Distance: " + dist[dest] + ")"
```

Flow Chart

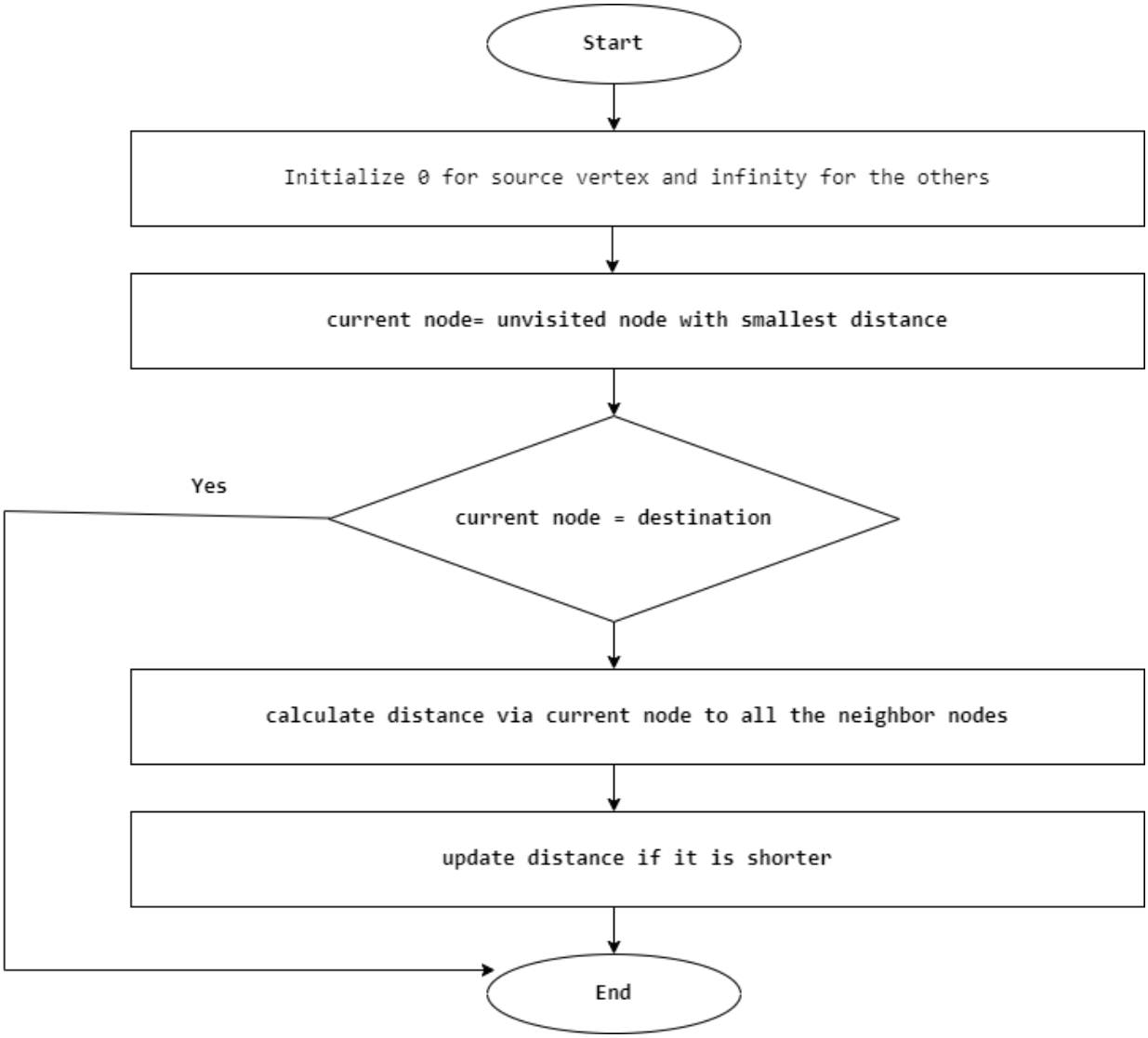


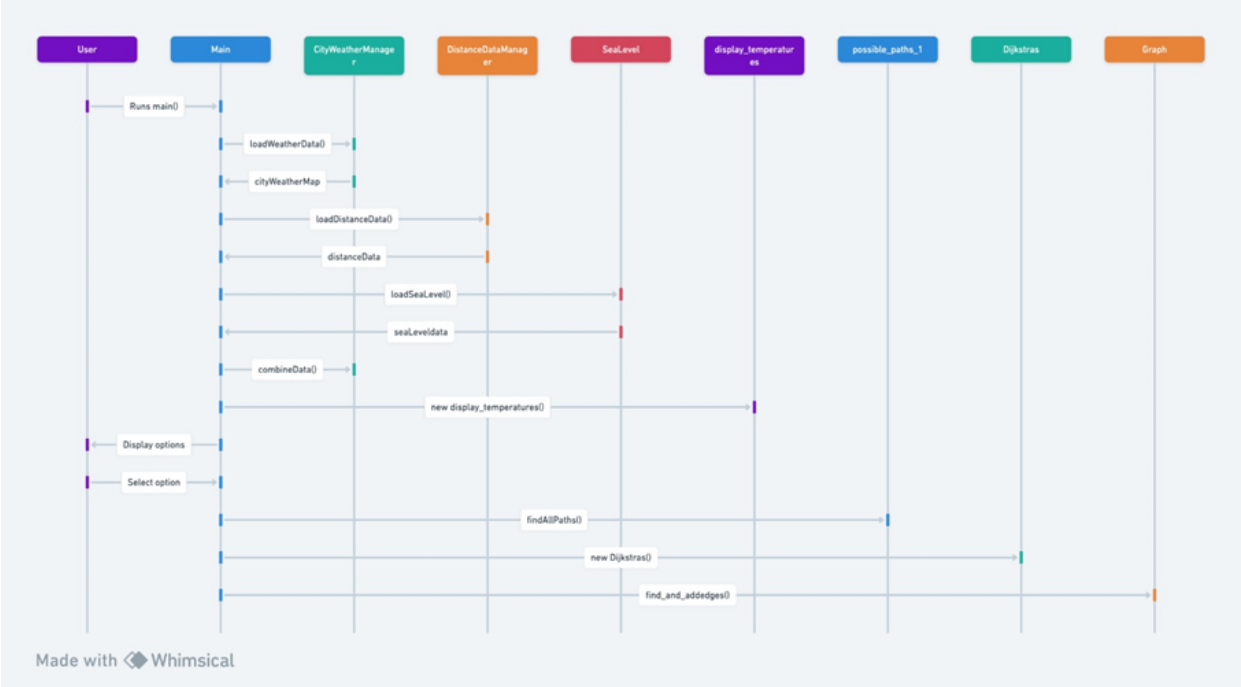
Dijkstra's algorithm is used to find the shortest distance between the nodes(cities). Since there are no negative values (Distance is never negative) we choose Dijkstra to find the shortest route from the source to the destination with the minimum distance possible. The cities and their connections are represented in an **adjacency matrix – adjacencyMatrix[],** where each cell holds the distance between two cities. The '**dijkstra**' method determines the shortest route between the source and destination cities employing Dijkstra's algorithm. It traverses the vertices, identifying the briefest path by evaluating the weights (distances) of connections among them.

Pseudo Code

```
function Dijkstra(adjacencyMatrix, startVertex, endVertex):
     numVertices = size of adjacencyMatrix
     visited[numVertices] = false
     dis[numVertices] =
     // Distance from startVertex to itself is zero
     dis[startVertex] = 0
     for count = 0 to numVertices - 1:
        minDistance =
        minVertex = -1
        // Find the vertex with the minimum distance
        for v = 0 to numVertices - 1:
           if not visited[v] and dis[v] < minDistance:
           minDistance = dis[v]
           minVertex = v
        visited[minVertex] = true
        // Updating distances of neighboring vertices
        for v = 0 to numVertices - 1:
           if not visited[v] and adjacencyMatrix[minVertex][v] != 0 and
           dis[minVertex] != infinity
           dis[minVertex] + adjacencyMatrix[minVertex][v] < dis[v]:
               dis[v] = dis[minVertex] + adjacencyMatrix[minVertex][v]
return dis[endVertex]
```

Flow Chart


Start
Initialize 0 for source vertex and infinity for the others
current node= unvisited node with smallest distance
current node = destination
Yes
calculate distance via current node to all the neighbor nodes
update distance if it is shorter
End



Sequential Diagram

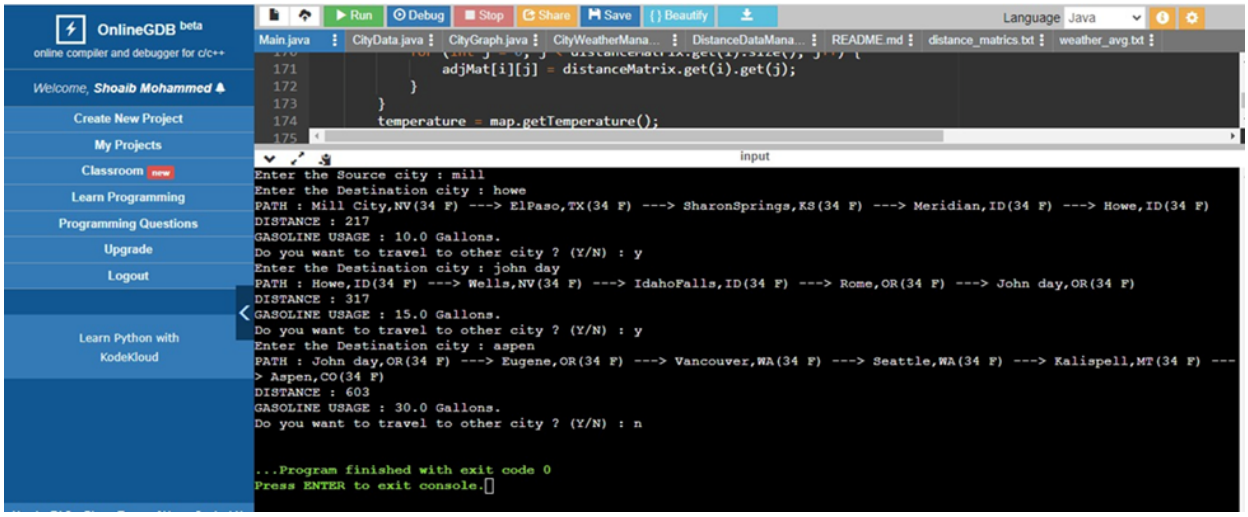
Results

The provided milestone 1 code reads temperature and distance data from CSV files and prints information about cities, including temperature and distance data. The output of the code depends on the actual data in the CSV files.

For each city, the code prints:

- City name
- Temperature
- Distance data

Here's an example of what the output look like:



Minimum Spanning Tree algorithm is implemented which includes both Kruskal and Prim algorithm. User is provided with two algorithms to choose between to obtain the minimum distance to travel between the edges of the cities in miles.

```
Enter the which algorithm to apply
(Kruskal / Prims) [press n to terminate.] ?
Prims
Executing Prims Algorithm
============================
Edge 0:(Mill City-NV, Rome-OR) Distance : 171
Edge 1:(JordanValley-OR, Rome-OR) Distance : 33
Edge 2:(JordanValley-OR, Caldwell-ID) Distance : 61
Edge 3:(Nampa-ID, Caldwell-ID) Distance : 11
Edge 4:(Nampa-ID, Meridian-ID) Distance : 11
Edge 5:(Boise-ID, Meridian-ID) Distance : 12
Edge 6:(Adrian-OR, Caldwell-ID) Distance : 25
Edge 7:(Adrian-OR, Owyhee-OR) Distance : 4
Edge 8:(Nyssa-OR, Owyhee-OR) Distance : 9
Edge 9:(Baker city-OR, Nyssa-OR) Distance : 87
Edge 10:(John day-OR, Baker city-OR) Distance : 80
Edge 11:(Donnelly-ID, Boise-ID) Distance : 94
Edge 12:(Riley-OR, John day-OR) Distance : 97
Edge 13:(Bend-OR, Riley-OR) Distance : 104
Edge 14:(Boise-ID, Buhl-ID) Distance : 120
```

Input: Enter which algorithm to apply and used to run prims or kruskal's algorithm.

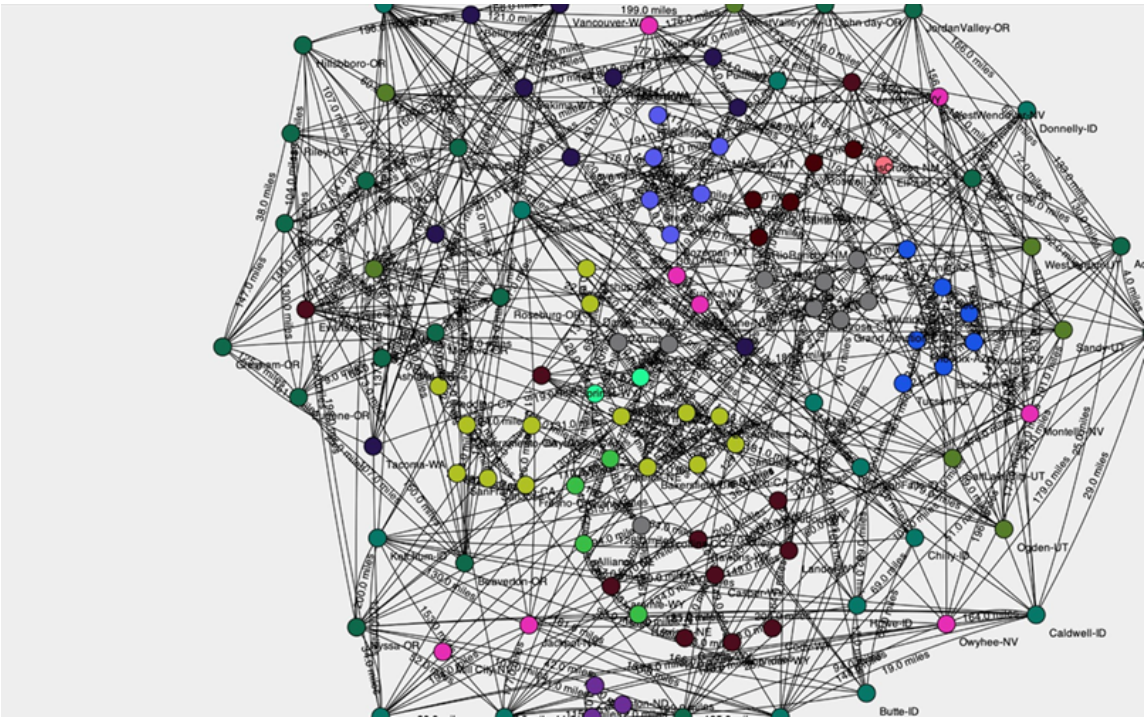
```
Edge 106:(SantaFe-NM, Roswell-NM) Distance : 192
Edge 107:(Roswell-NM, LasCruces-NM) Distance : 184
Edge 108:(LasCruces-NM, ElPaso-TX) Distance : 47
Edge 109:(Chinle-AZ, Sedona-AZ) Distance : 197
Edge 110:(Sedona-AZ, Prescott-AZ) Distance : 67
Edge 111:(Phoenix-AZ, Prescott-AZ) Distance : 100
Edge 112:(Phoenix-AZ, Buckeye-AZ) Distance : 40
Edge 113:(Tucson-AZ, Phoenix-AZ) Distance : 112
Edge 114:(Prescott-AZ, Kingman-AZ) Distance : 148
Edge 115:(Gillette-WY, Bowman-ND) Distance : 219
Edge 116:(Dickinson-ND, Bowman-ND) Distance : 74
Edge 117:(Williston-ND, Dickinson-ND) Distance : 132

 Minimum Distance  = 9909 miles
Do you want to continue(y/n)?
y
Enter the which algorithm to apply
(Kruskal / Prims) [press n to terminate.] ?
krushal
Invalid Input!
Enter the which algorithm to apply
(Kruskal / Prims) [press n to terminate.] ?
n


...Program finished with exit code 0
Press ENTER to exit console.
```

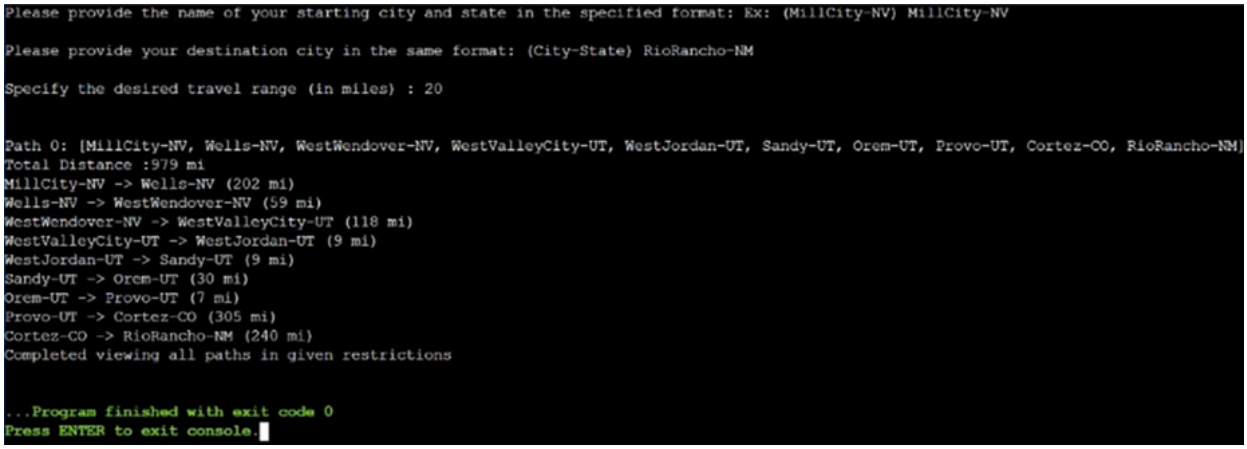
Output: Gives the minimum distance to travel between the edges of the cities.

A visualization map is created showing city names and distances between all the cities by taking distance matrix and weather data as the input.



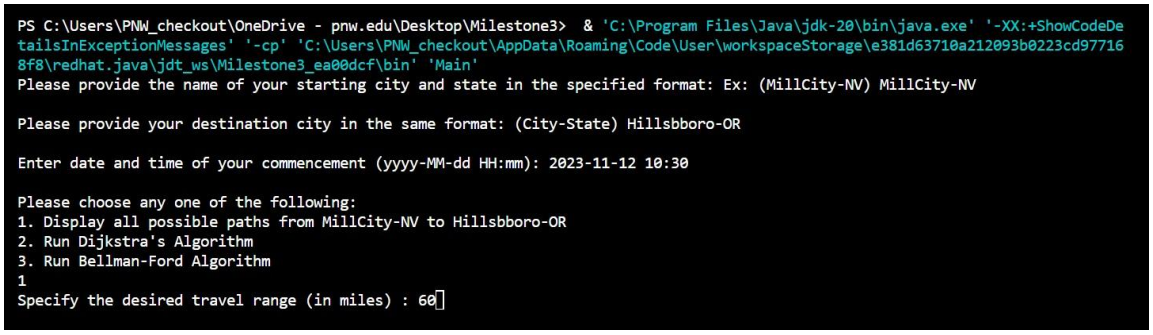
Visualization map showing cities names and distances between all cities.

By taking a starting city in the format of city_name – state_name and destination city, specifying the desired travel range (in miles), a list of all the nearby cities that can be covered within the given range is given.



List of all the nearby cities that can be covered within the given range.

By taking the source city and state[City-State] and Destination city and state: [City-State], Date and time of commencement: yyyy-MM-dd HH:mm, the Travel Planner provides various output formats depending on the user's selection.



The program will show all paths that are available within the specified distance after you specify the desired travel range. A list of paths with each path represented as a series of cities will be the output. Each path's total length will also be shown.

Discussion

Advantages:

1. **Real-Time Weather Data Integration:** The dataset must be both comprehensive and up-to-date by utilizing AccuWeather‘s API for instantaneous weather data. In addition, this advantage makes it suitable in such applications as disaster management and travel planning, which need actual time data.
2. **Vertical Dimension with Sea Level Data:** Including sea levels to any data is a vital point to consider while doing any environmental impact studies, especially that you are in context about future consequences of global warming and rising ocean levels for coastal areas and low lying cities.
3. **Automation and Efficiency:** Automated scripts for data collection and integration into Google Sheets and Excel improve accuracy while allowing easy manipulation and distribution of data. Such an automation is essential in handling big sets of data so as to guarantee proper data.
4. **Scalability and Flexibility:** Organizational chart ensures flexibility in terms of scaling upwards with additional cities, states and/or various information types. This allows for a very versatile project, which can be applicable in many different upcoming needs or uses.
5. **Educational and Academic Applications:** The dataset is not only useful but also a rich source of case study for student learning in geography, environmental science, data science and urban planning. Additionally, it holds potential at a study level on issues concerning climate change, urbanization, and transport.
6. **Support for Environmental Impact Assessments:** Weather data including sea levels plays a key role in carrying out environmental impact assessments. Such study may be crucial, especially about the impact of climate change on various cities and being ready for possible upcoming situations.
7. **Foundation for Machine Learning and AI Projects:** Structured and detailed dataset provides a great starting point for machine learning and AI ventures that reveal deep meanings and forecasts about the data.
8. **Resource for Disaster Management and Response:** Real-time weather information and Geospatial data will help with planning emergency measures and mitigating effects of a natural catastrophe.
9. **Support for Tourism and Travel Industry:** These data can be very helpful for tourism and hospitality industries. For example they can help plan for the high seasons and also give updates on the current weather.
10. **Enabling Smart City Initiatives:** Smart city projects can leverage these different kinds of datasets so as to design cities that are more sensitive, greener, and smarter.
11. **Facilitating Climate Research:** The intersection of climate, topography, and weather can go a long way in shaping our understanding about why climate is

experienced differently among certain locations and why there exists varying responses to climate change across different parts of the world.

12. **Promotion of Sustainable Development:** This project will enable cities to plan for growth using environment-friendly approaches with specific insights into environmental and urban changes.
13. **Potential for Commercial Applications:** The dataset is crucial to businesses operating in different industries such as real estate, transport and logistics for market study, route planning, site identification, and strategy development.

Challenges:

1. **Data Accuracy and Consistency:** The reliability of such API is dependent upon the nature and the updating rates of these other sources such as AccuWeather and Google maps. Errors in their data directly impact on the outcome of the analysis and subsequently, the success of study.
2. **API Limitations and Costs:** The use of AccuWeather and Google maps APIs are also subject to limitation on the number of free requests and exceeding the number is expensive as it may cost millions. Moreover, API policy changes may also result in long term project sustainability and price model modification.
3. **Complexity in Data Integration:** The process of merging information from different points of departure, such as weather, sea levels, and city-to-city distances into a single database, proves challenging. It is difficult to get every individual point straight at times with regard to both chronological significance and position on the map.
4. **Handling Large Datasets:** Large sets of data could result from working with 120 cities having hourly weather information and a 120×120 distance matrix.
5. **Dependency on Third-Party Platforms:** Many services are from third parties such as Google Cloud, Google Sheets etc. Project's functioning can directly depend on any minor down time or change in their service.
6. **Data Privacy and Security:** There should be a strong focus on information confidentiality, particularly where cloud-based applications are concerned, if you're working with data that requires protection of privacy and security attributes.
7. **Scalability Challenges:** There can be a major need for reengineering of an entire project as you scale up data collection in respect to more cities or complex data.

8. **Time-Intensive Processes:** It appears that it's a manual process to collect sea level data for this project. They are time consuming and may restrict the rate of updating or enlargement of the dataset.
9. **Data Interpretation and Analysis:** Such a massive data with so many variants makes it hard to come up with a few significant observations. While it is tedious, it may demand complex statistical or machine learning techniques.
10. **Technical Complexity:** The team would have to be experts in using diverse APIs and programming languages since they may need Java for running Google Maps and possibly Python for analysis purposes.
11. **Maintenance and Updating:** The process is ongoing and, therefore, requires periodic updates concerning the data, code, and APIs or else it could become very costly in resources.
12. **Risk of Data Overload:** Data overload occurs when there is more data that can be effectively analyzed within a given period or subject, therefore, valuable insights become difficult to discern.

Applications:

1. **Urban Area Planning:** Planners can use data to understand the impact of the environment on urban growth. Climate modeling and geographic data are critical for planning infrastructure that can withstand climate change and natural disasters.
2. **Environmental assessment:** Dataset can be used to assess the impact of various projects on the environment, particularly to understand how various climate and ocean impacts can affect different regions.
3. **Transportation and logistics management**: Companies can use real-time city-to-city distance data and weather data to better plan routes, reduce fuel consumption and have improved lead times.
4. **Disaster management and preparedness:** Government agencies and NGOs can use real-time weather data to better prepare for and respond to natural disasters such as floods, hurricanes, or wildfires.
5. **Climate change research:** Researchers can use this data to study the spatial impacts of climate change, primarily by analyzing long-term weather and sea level changes.
6. **Tourism Services:** The tourism industry can benefit from this project by using weather data to plan events and activities, and inform tourists about the best times to visit certain destinations.

7. **Agriculture and water management:** Farmers and water management managers can use climate data to plan irrigation, crop production and cropping seasons, thus providing inputs and crops the best results.
8. **Real Estate Development:** Real estate developers can use geoenvironmental data to assess the feasibility of building in different locations, taking into account factors such as flood risk or frequency of severe weather events.
9. **Energy Sector:** The energy industry, especially renewable energy companies, can use weather data to forecast power generation from wind and solar power and design new energy systems that are more efficient.
10. **Public health and epidemiology:** Public health professionals can use meteorological and environmental data to understand and predict weather-affected diseases such as infectious diseases.
11. **Educational tools:** Dataset can be used as an instructional tool in academic settings, providing students with real-world data for projects in geography, environmental science, data analysis, and more around.
12. **Smart City Projects:** Aspiring ‘smart cities’ can incorporate data into their plans to improve city services such as public transport, waste management and emergency services.
13. **Sales management:** Retailers can analyze data to predict changes in customer behavior due to weather, and help with inventory management and marketing strategies.
14. **Insurance and risk management:** Insurance companies can use detailed environmental data for risk assessment, policy pricing, understanding and understanding of potential issues related to weather.
15. **Scientific research:** The dataset can be a valuable resource for scientific research in a variety of fields, from studying the impact of urbanization on climate models to the impact of seawater increases on coastal cities.

Conclusion

In conclusion, this pioneering project represents a paradigm shift in inter-city travel planning, seamlessly integrating cutting-edge algorithms and diverse datasets. The meticulous fusion of real-time weather data, city-to-city distances, and sea-level information has redefined route optimization, providing users with not only the shortest paths but also factoring in real-world conditions. The system's adaptability, advanced algorithms, and user-centric features elevate its utility beyond conventional route planning, offering personalized travel recommendations. With a scalable architecture designed for continual advancements, collaboration, and open-source contributions, this project serves as a blueprint

for leveraging disparate datasets in decision-making processes. It not only sets new benchmarks in optimal routing but also exemplifies the potential of data integration and algorithmic innovation in reshaping the landscape of travel planning.

Acknowledgement

We express our sincere gratitude to all individuals and entities who have contributed to the successful completion of this project. Their support, expertise, and collaboration have been instrumental in bringing this pioneering initiative to fruition.

- **Prof. Wei Dai (David) - Our Mentor:** For providing guidance, feedback, and expertise shaping the project.
- **AccuWeather API**: We extend our appreciation to AccuWeather for providing reliable and extensive real-time weather data through their API. The accuracy of weather information has been crucial in enhancing the precision of our route optimization strategies.
- **GoogleCloud Platform:** Our sincere thanks to Google Cloud Platform for providing the infrastructure and services that facilitated the integration of Google Sheets API. This platform has been integral to the organization, storage, and retrieval of crucial data for our project.
- **Whatismyelevation.com:** We acknowledge the invaluable contribution of Whatismyelevation.com for providing accurate sea-level information for each city in our study. This data has added a vertical dimension to our understanding of diverse urban landscapes.
- **Open Source Community (GitHub):** Our project thrives on collaboration and open-source contributions. We extend gratitude to the open-source community for fostering innovation, sharing knowledge, and contributing to the continual refinement of route optimization paradigms.

References:

- **AccuWeather API:** *AccuWeather APIs | home*. (n.d.). Developer.accuweather.com. https://developer.accuweather.com/
- **Google Cloud Platform:** Overview | Distance Matrix API. (n.d.). Google Developers. https://developers.google.com/maps/documentation/distance-matrix/overview
- **Whatismyelevation.com:** What is my elevation? (n.d.). Whatismyelevation.com. https://whatismyelevation.com/
- **GitHub:** GitHub - Mohammed-Shoaib124/Shortest-Path at Milestone3. (n.d.). GitHub. Retrieved December 2, 2023, from https://github.com/Mohammed-Shoaib124/Shortest-Path/tree/Milestone3
- **Algorithm References:** Introduction to Algorithms by MIT (Third Edition)