ROAD NETWORK DESIGN

USING MINIMUM SPANNING TREE ALGORITHMS

AGENDA

01 PROBLEM STATEMENT

02 INTRODUCTION

03 OBJECTIVES

04 ALGORITHMS OVERVIEW

05 DATA COLLECTION

06 GRAPH CONSTRUCTION

07 COMPARISION

08 CHALLENGES FACED

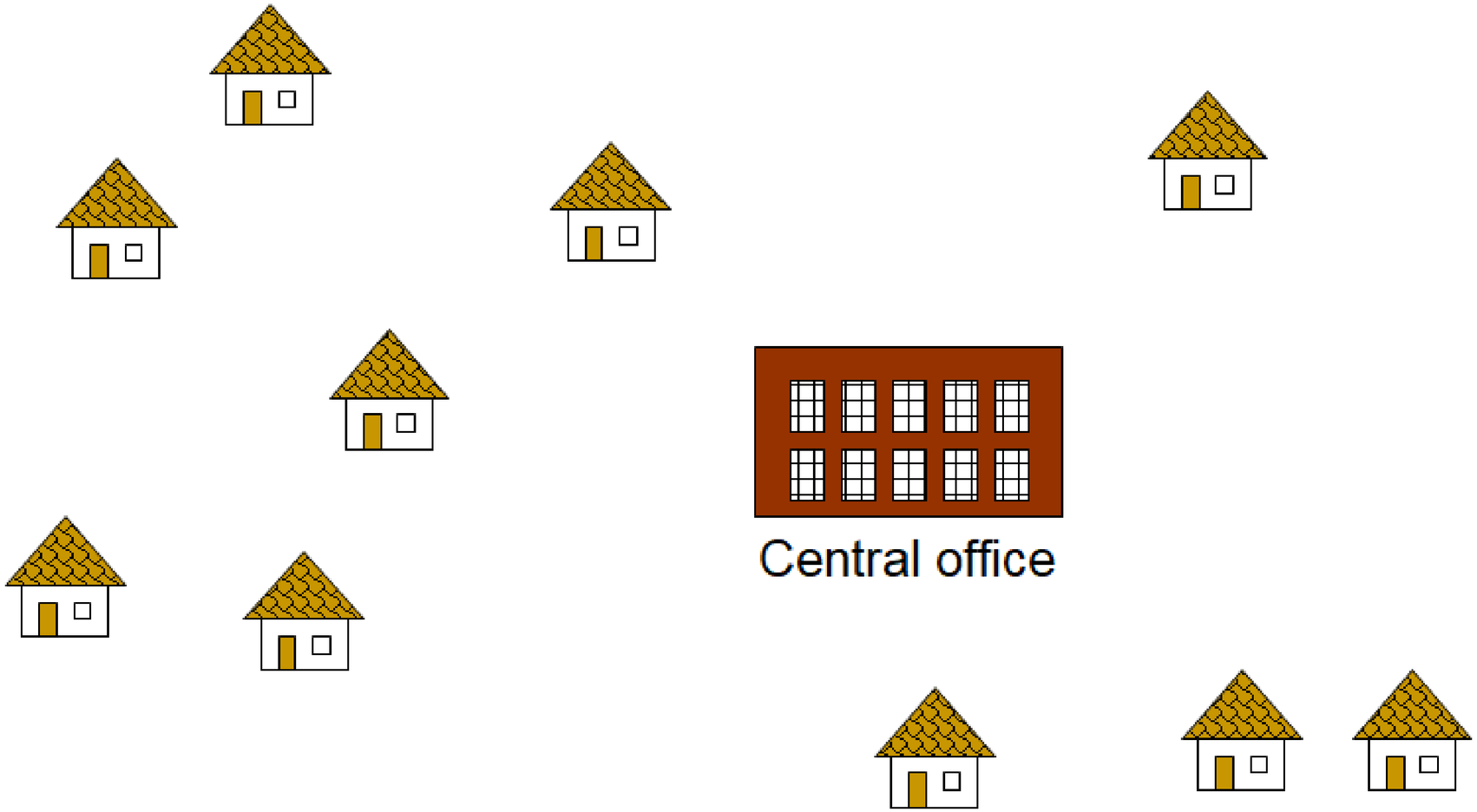
09 FUTURE ENHANCEMENT

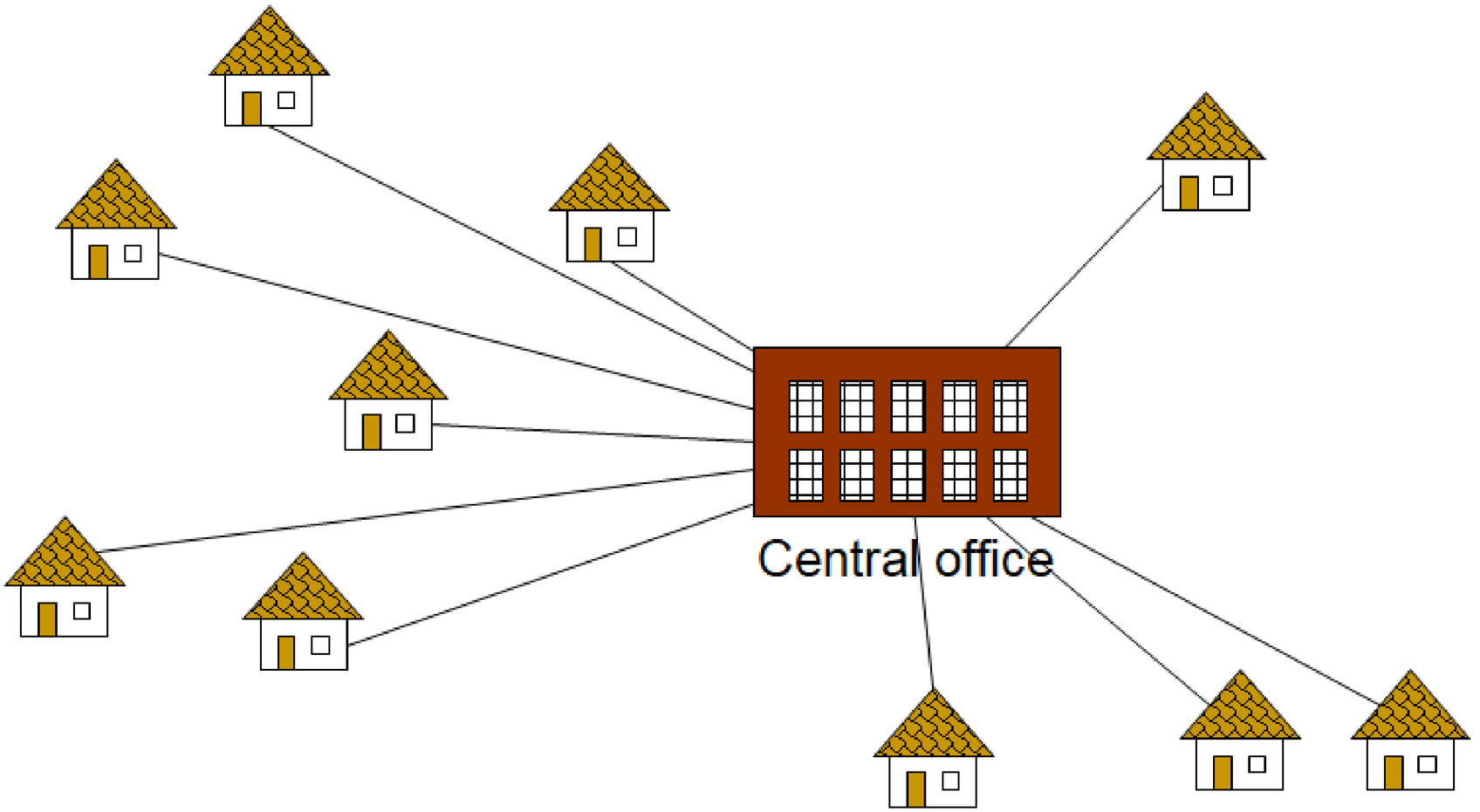
10 CONCLUSION

PROBLEM STATEMENT

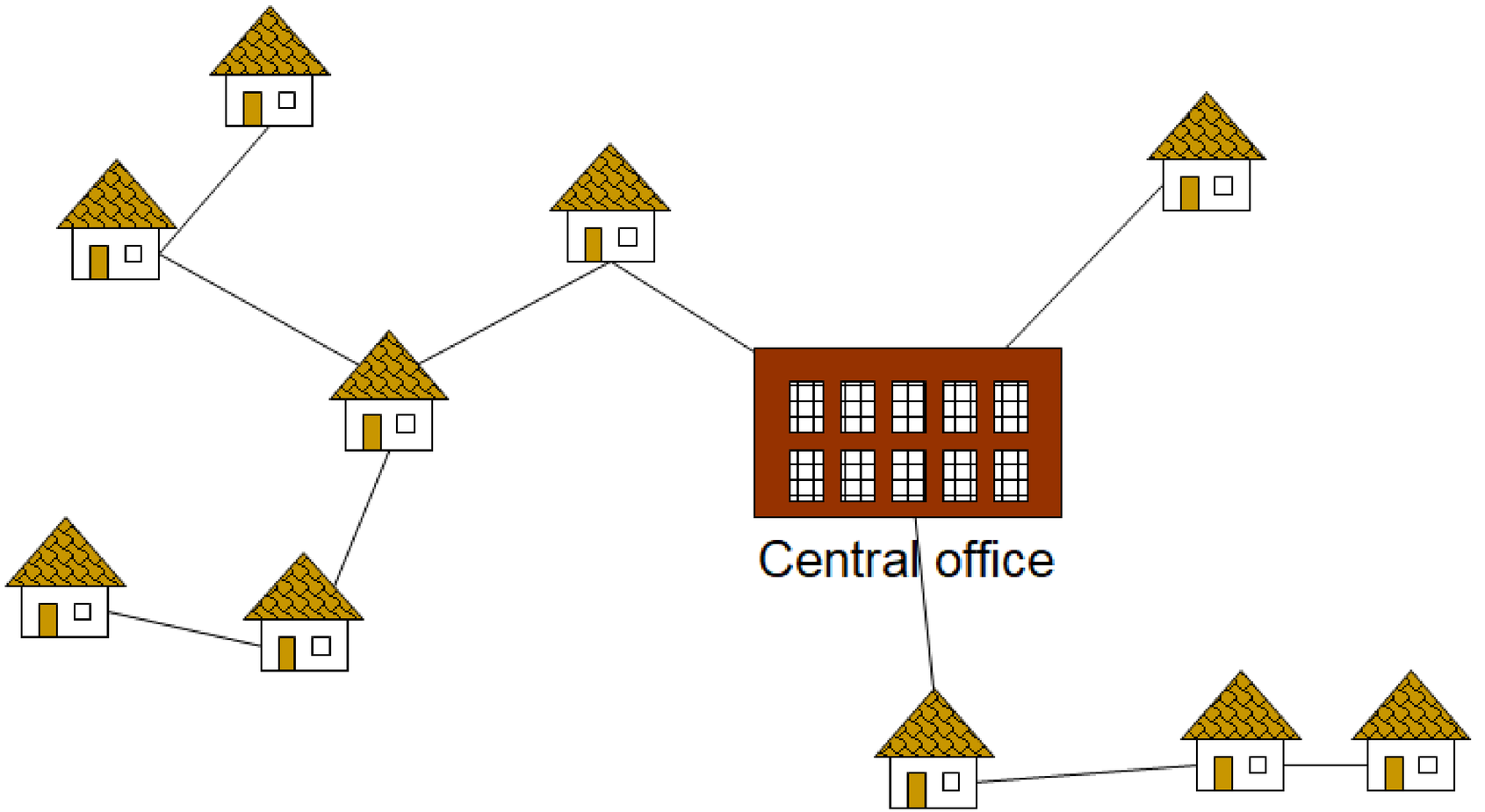
Expanding road networks efficiently is essential for improving connectivity and reducing construction costs. The challenge is to connect multiple cities or regions with the least cost while ensuring accessibility. This can be modeled as a Minimum Spanning Tree (MST) problem, where the goal is to connect all nodes (cities) with the minimal total road length. Using algorithms like Prim’s or Kruskal’s, we aim to design an optimal road network that minimizes cost while ensuring effective connectivity.

PROBLEM: LAYING TELEPHONE WIRE





Expensive!



Minimize the total length of wire connecting the customers

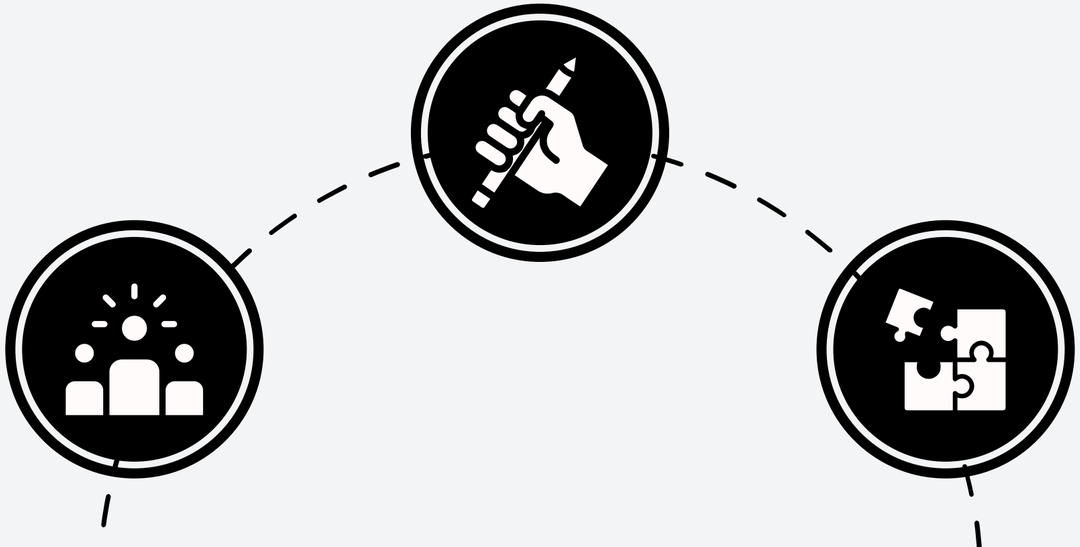
INTRODUCTION

Expanding road networks is crucial for economic growth, urbanization, and improving transportation efficiency. The process involves connecting multiple cities or regions while minimizing construction costs. To achieve this, Minimum Spanning Tree algorithms such as Prim's and Kruskal's play a key role in ensuring optimal connectivity. These algorithms help in designing road networks that minimize the total distance or cost while ensuring all cities are accessible. MST can also be used in optimizing road networks, electric grids, water pipelines, sewage systems, telecommunication networks, and computer networks.

MST-based road planning enhances transportation, reduces costs, and ensures equitable access, transforming regional infrastructure.

OBJECTIVES

- **Optimize Road Network:** Minimize construction cost and distance using MST algorithms
- **Ensure Connectivity:** Connect all cities with the minimum total distance
- **Algorithm Comparison:** Evaluate Prim’s and Kruskal’s in terms of performance
- **Real-time Applications:** Apply Minimum Spanning Tree to road networks
- **Visual Representation:** Showcase optimized road networks through graphs



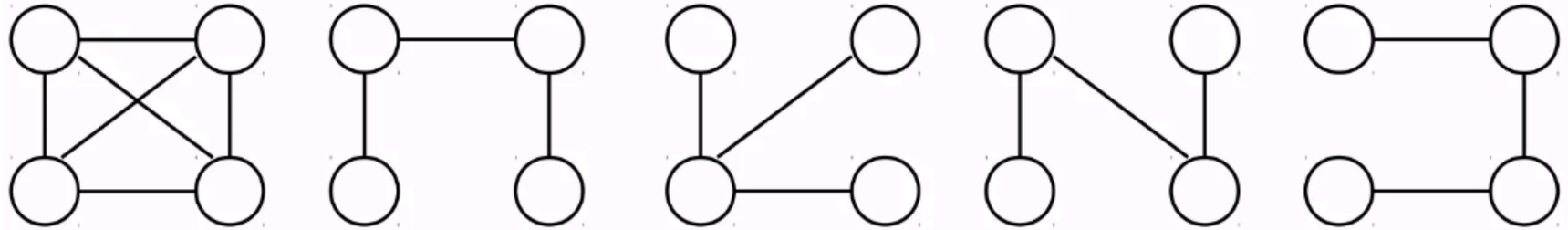
ALGORITHMS OVERVIEW

MINIMUM SPANNING TREE:

- A **Minimum Spanning Tree** is a subgraph that connects all vertices of a graph with the minimum total edge weight, without forming any cycles.
- **Key Algorithms:** Prim's and Kruskal's algorithms.

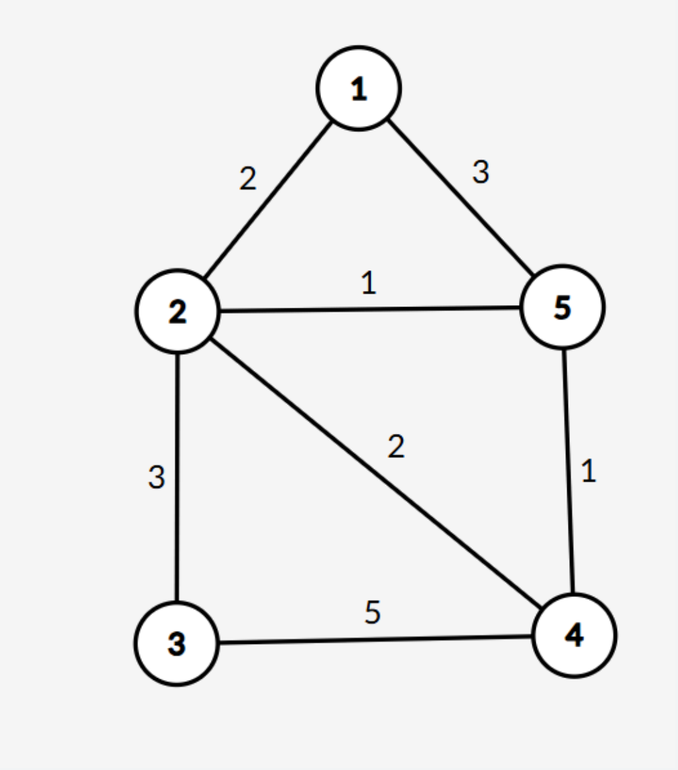
Importance of MST:

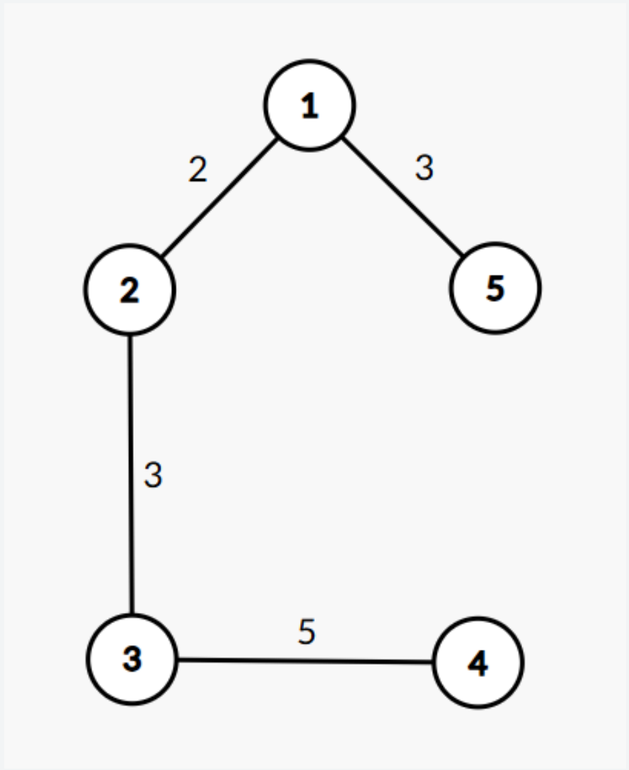
- **Cost-Effective:** Minimizes the cost of constructing roads, telecommunication networks, etc.
- **Optimized Connectivity:** Ensures all points (cities, stations) are connected efficiently



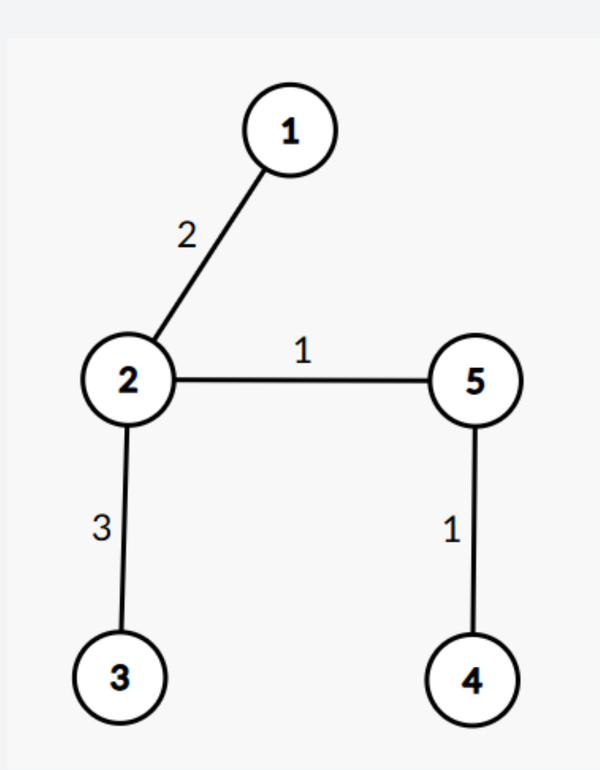
Graph **Spanning Trees**

EXAMPLE OF MINIMUM SPANNING TREE:





Spanning Tree:
3+2+3+5 = 13



Minimum Spanning Tree:
2+3+1+1 = 7

KRUSKAL'S ALGORITHM

- An algorithm used to find the Minimum Spanning Tree of a graph by selecting edges in increasing order of weight.

Key Steps:

- Sort all edges by weight in ascending order.
- Start adding edges to the MST, ensuring no cycles are formed.
- Stop when all vertices are connected.

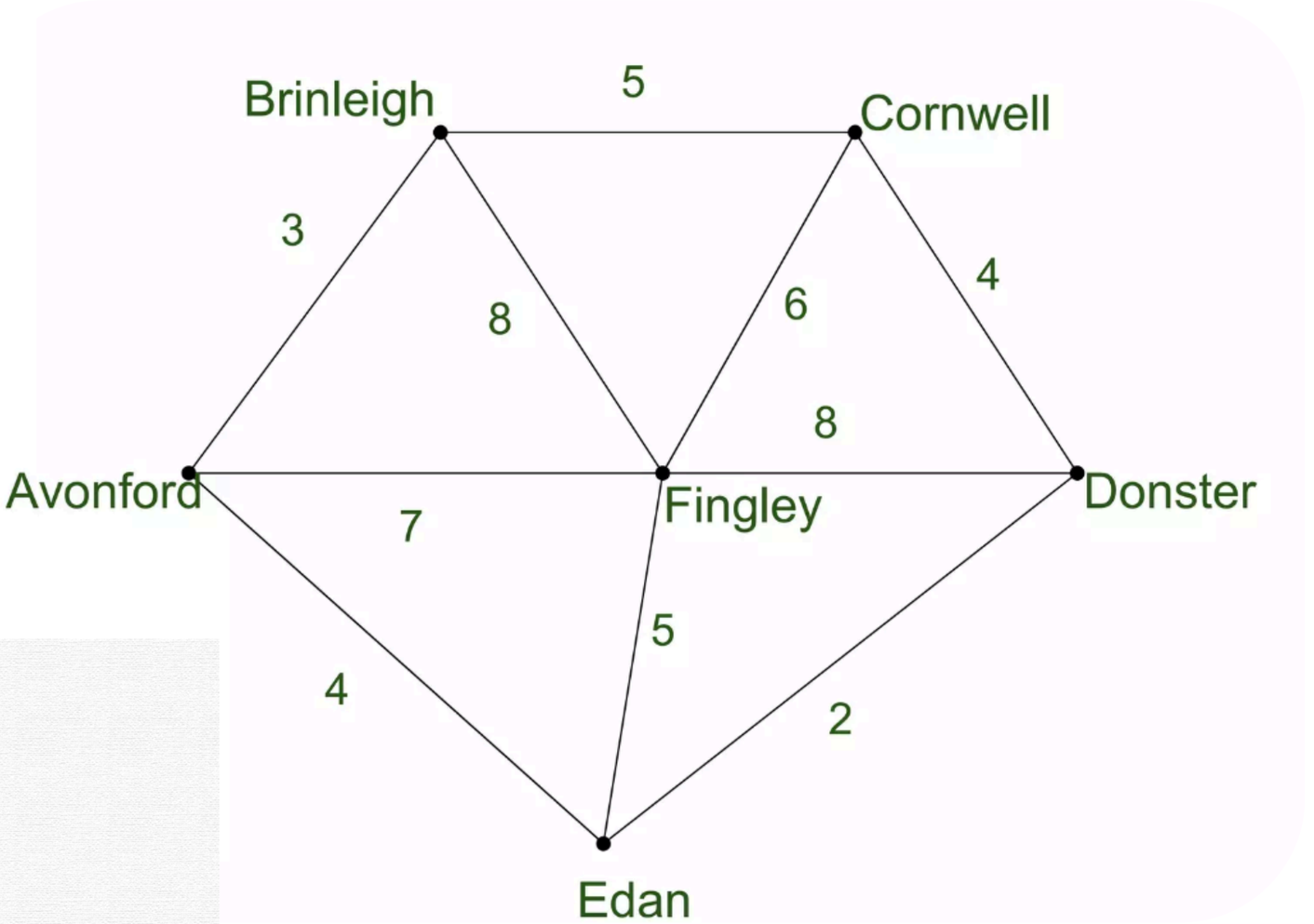
PRIM'S ALGORITHM

- An algorithm that constructs the Minimum Spanning Tree by adding the smallest edge connecting a vertex in the tree to a vertex outside the tree.

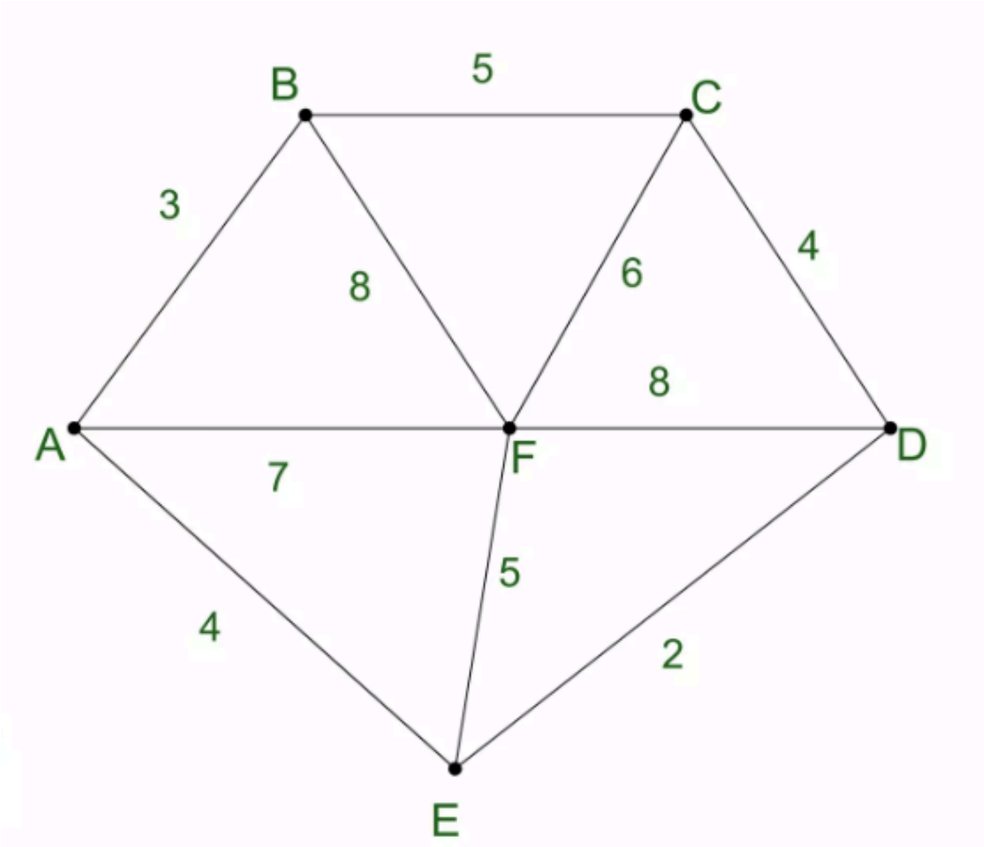
Key Steps:

- Begin with any vertex.
- Add the smallest edge that connects the current tree to a new vertex.
- Repeat until all vertices are connected.
- Ensure no cycles are formed while adding edges.

A cable company want to connect five villages to their network which currently extends to the market town of Avonford. What is the **minimum length** of cable needed?

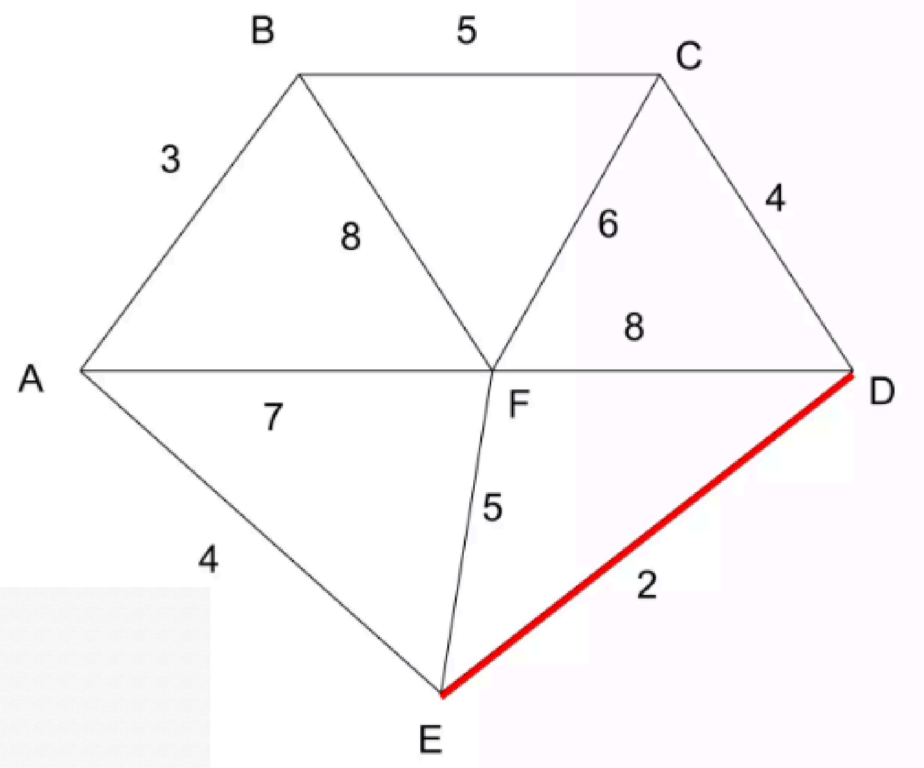


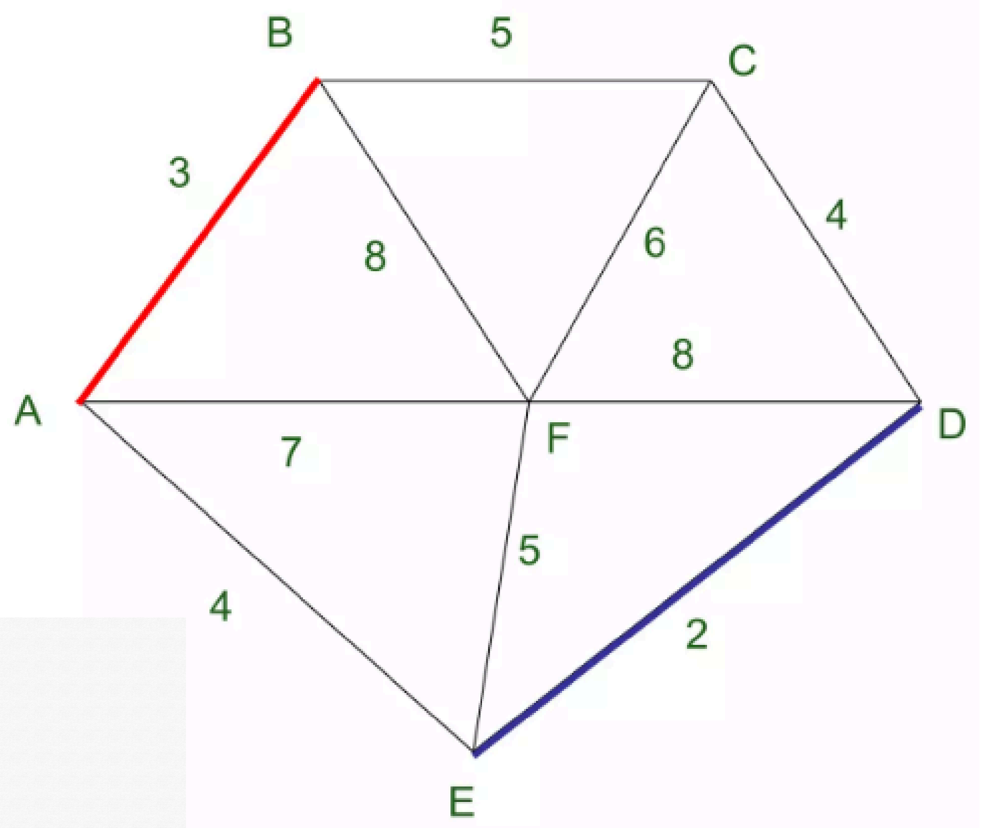
KRUSKAL'S ALGORITHM

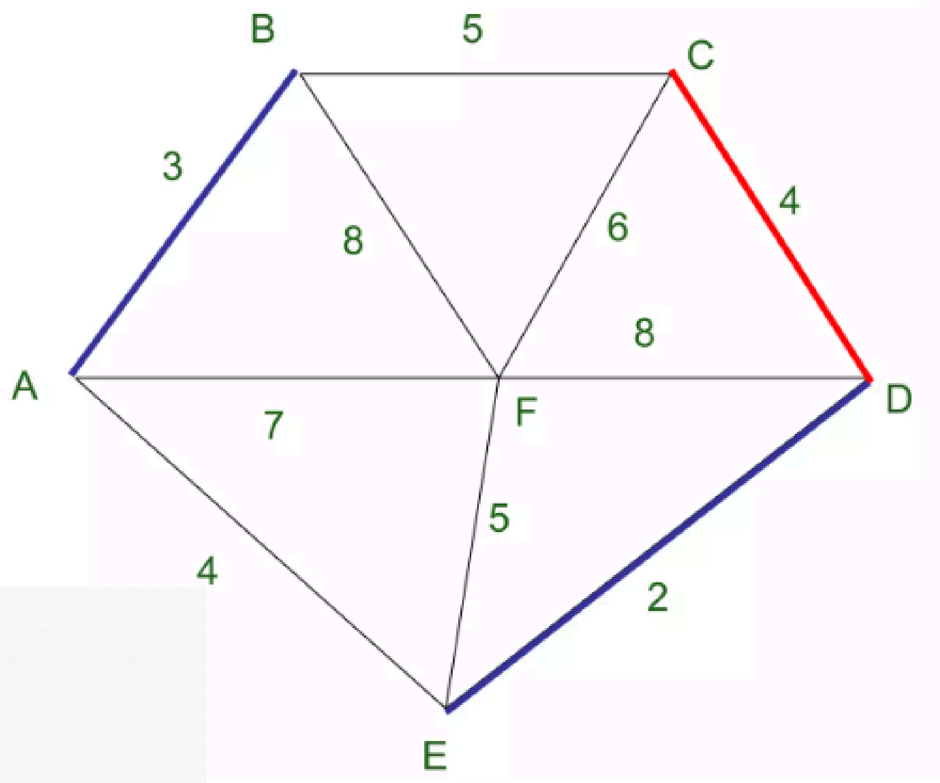


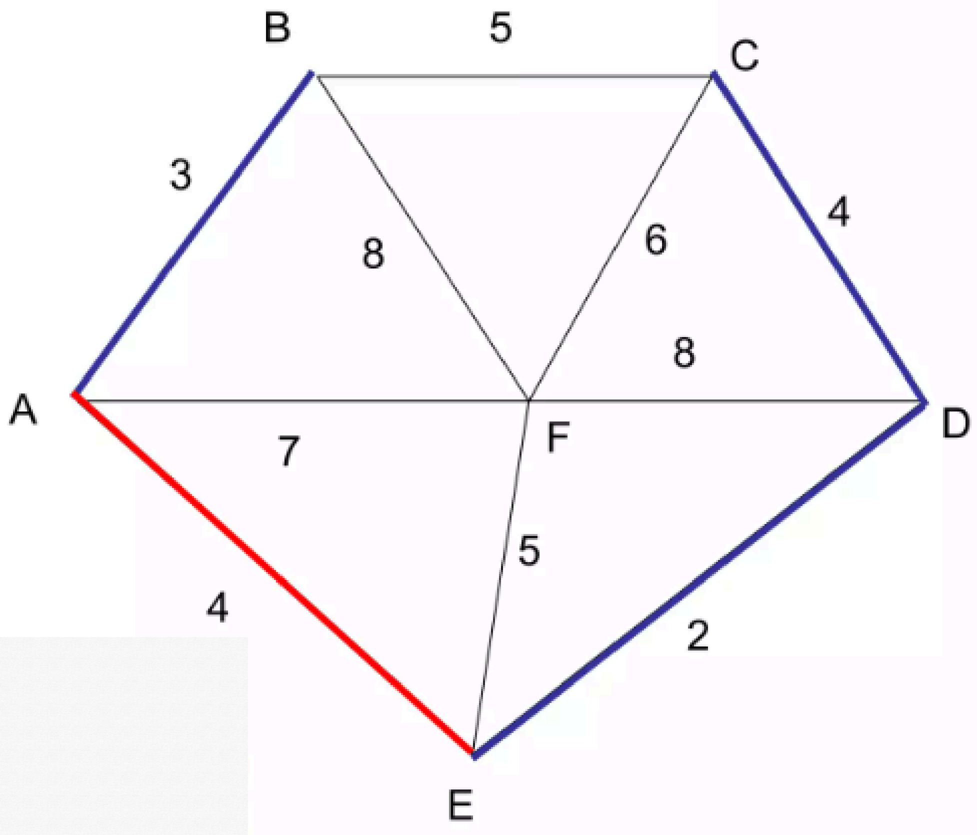
List of Edges:

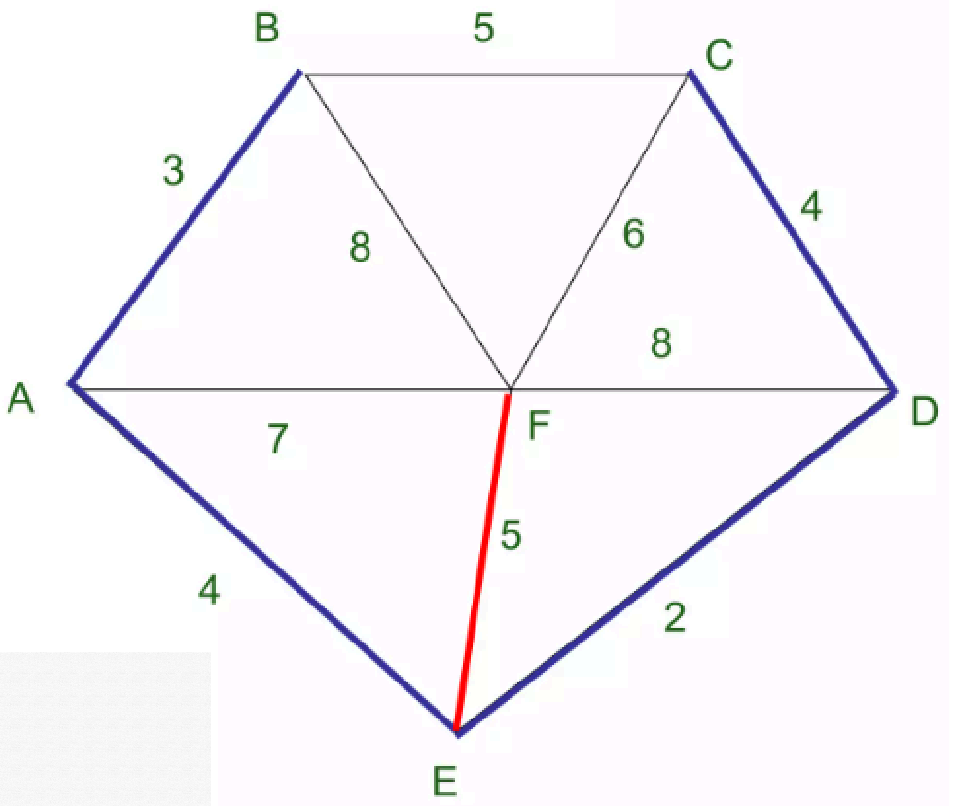
ED 2
AB 3
CD 4
AE 4
BC 5
EF 5
CF 6
AF 7
BF 8
CF 8

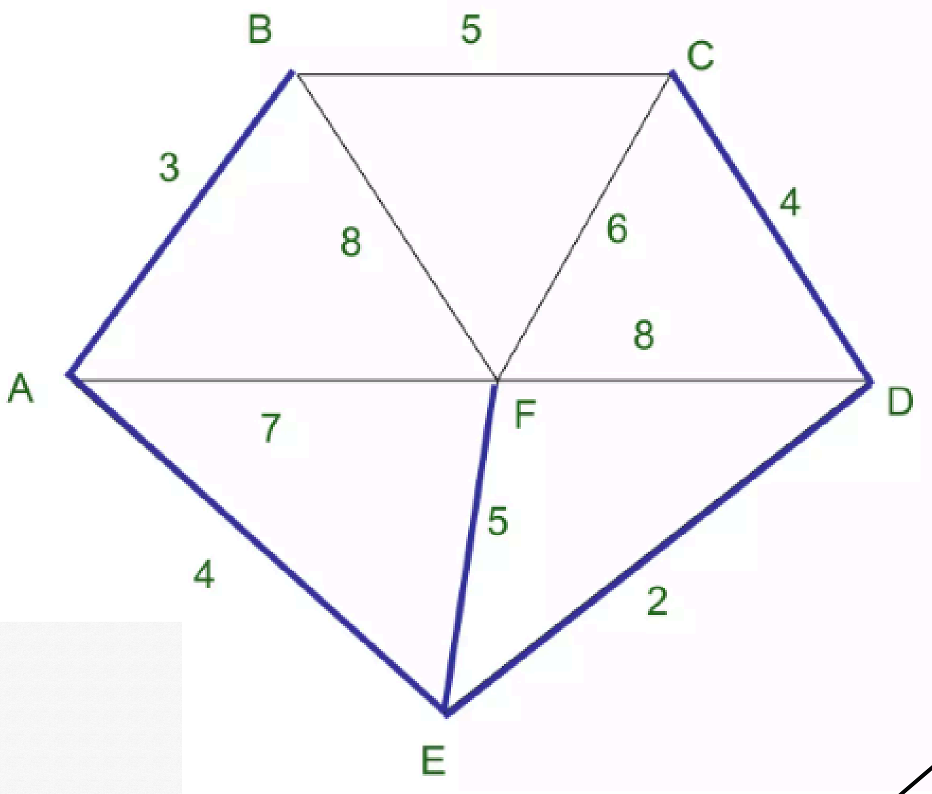

B
5
C
3
8
6
4
8
A
7
F
D
5
4
2
E

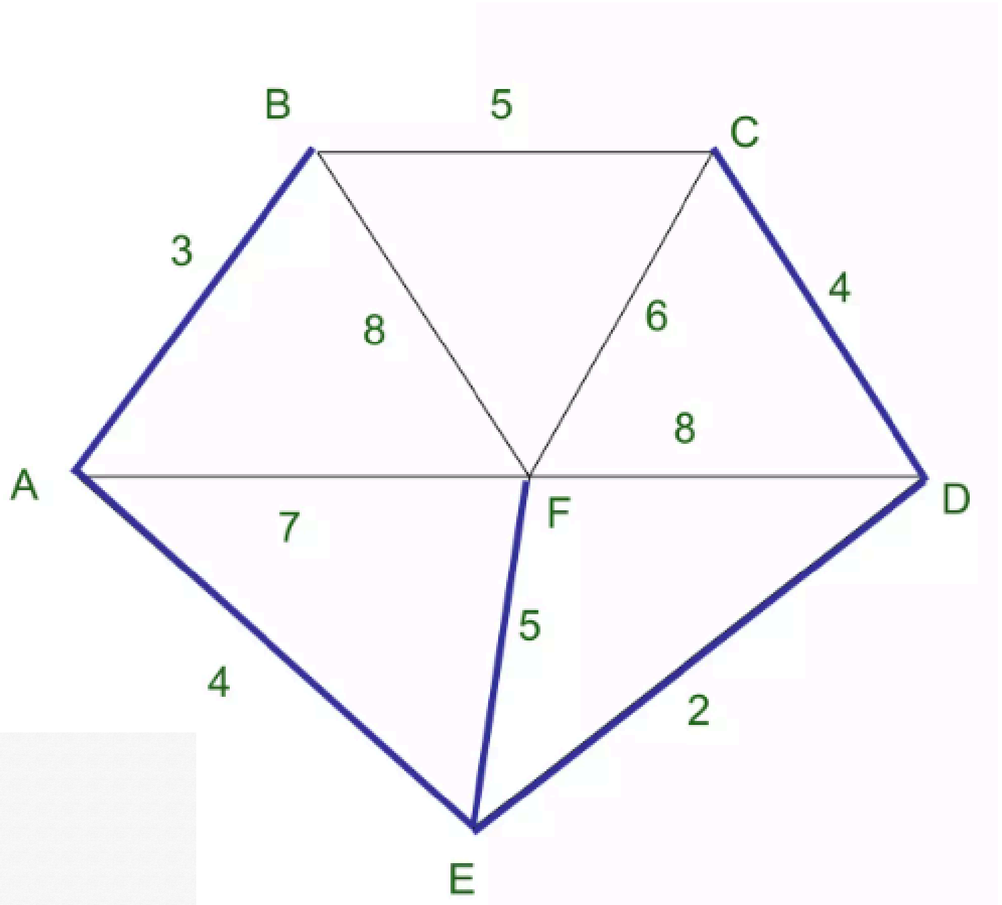

B
5
C
3
8
6
4
8
A
7
F
D
5
4
2
E


B
5
C
3
8
6
4
8
A
7
F
D
5
4
2
E


B
5
C
3
8
6
4
8
A
7
F
D
5
4
2
E


B
5
C
3
8
6
4
8
A
7
F
D
5
4
2
E


B
5
C
3
8
6
4
8
A
7
F
D
5
4
2
E

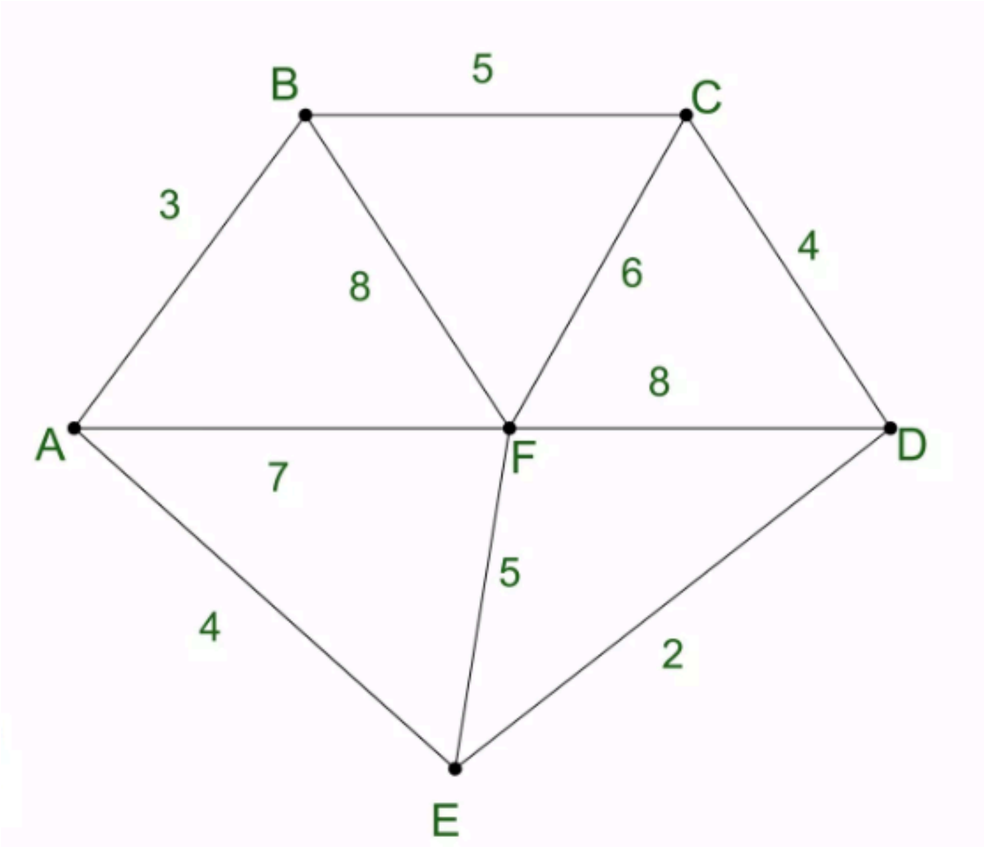


The solution is:

ED 2
AB 3
CD 4
AE 4
EF 5

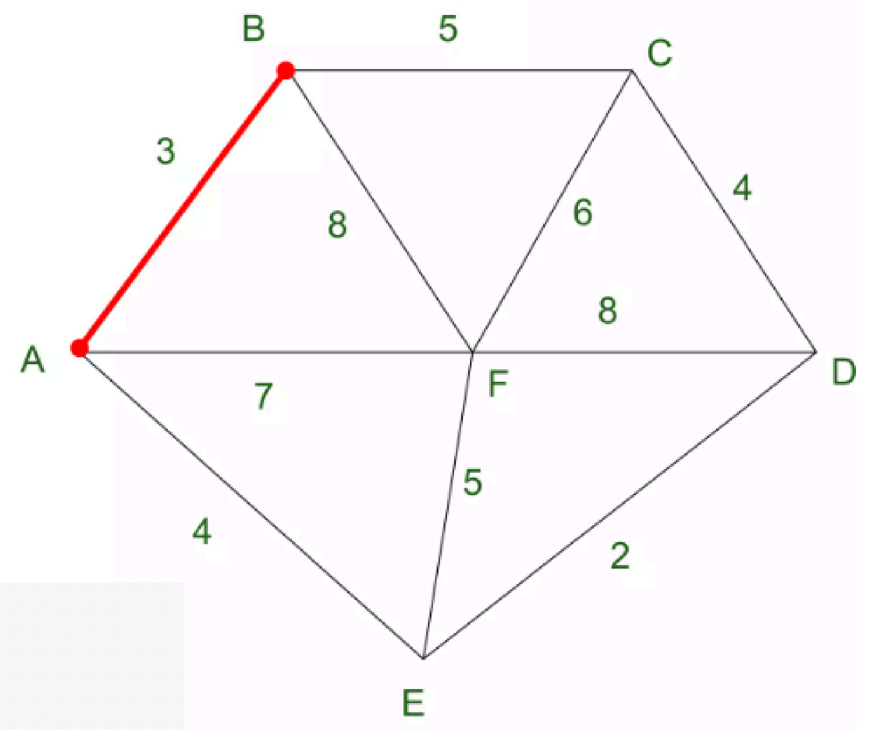
Total weight of the tree: 18

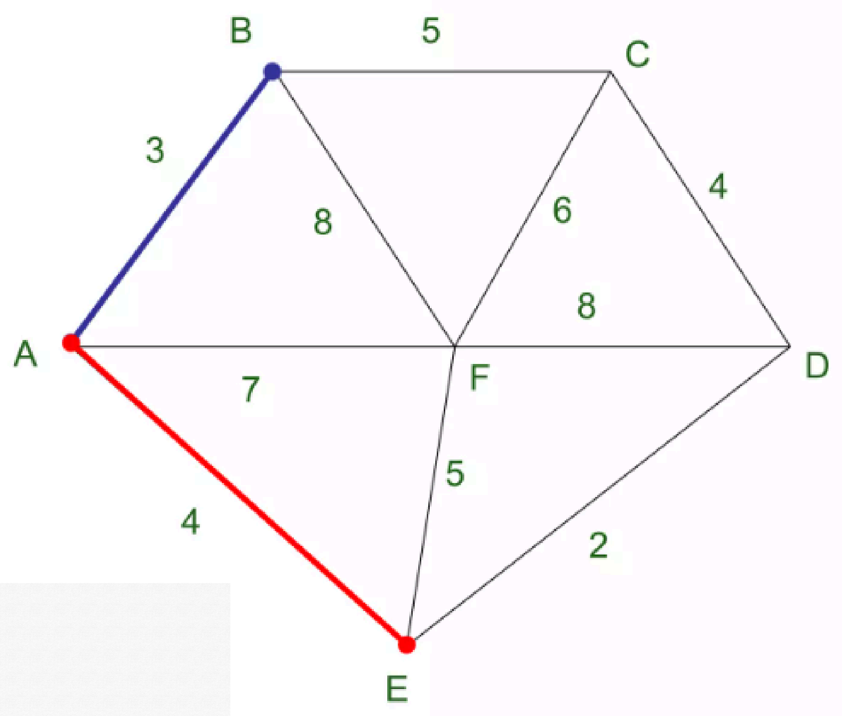
PRIM'S ALGORITHM

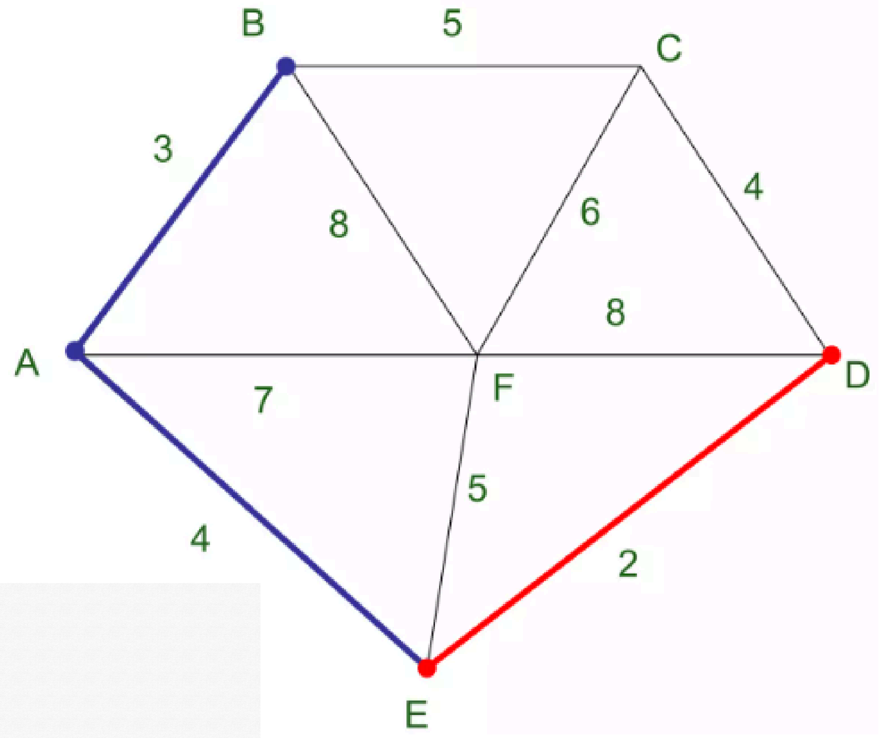


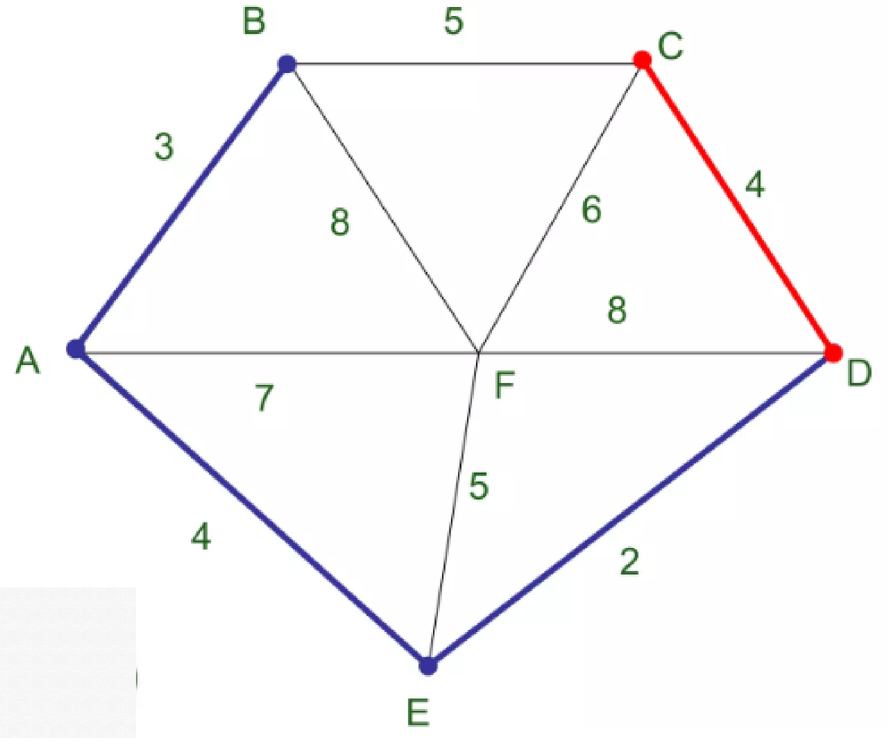
Starting vertex: A

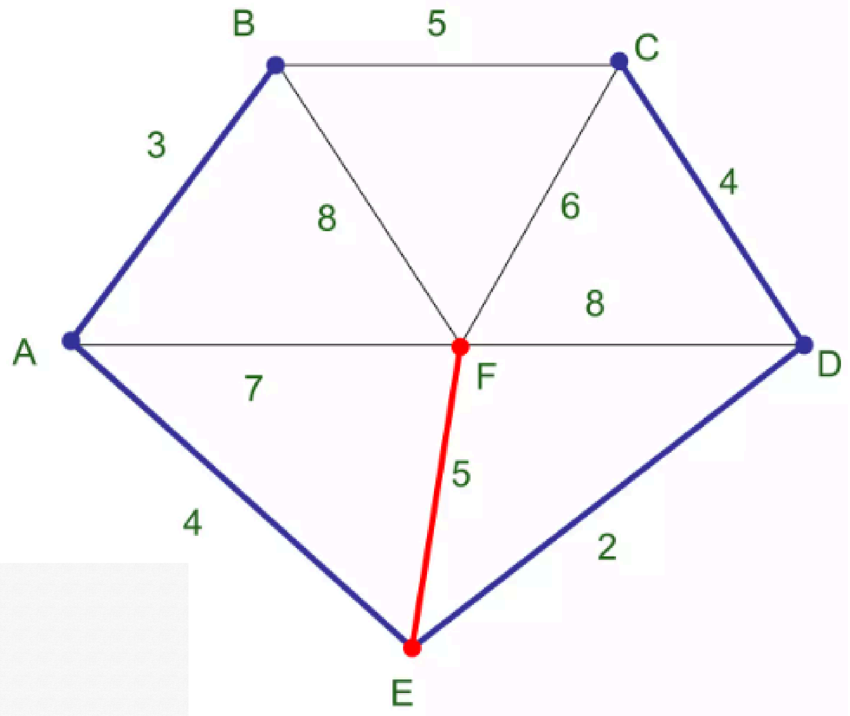
Select the shortest edge connected to that vertex

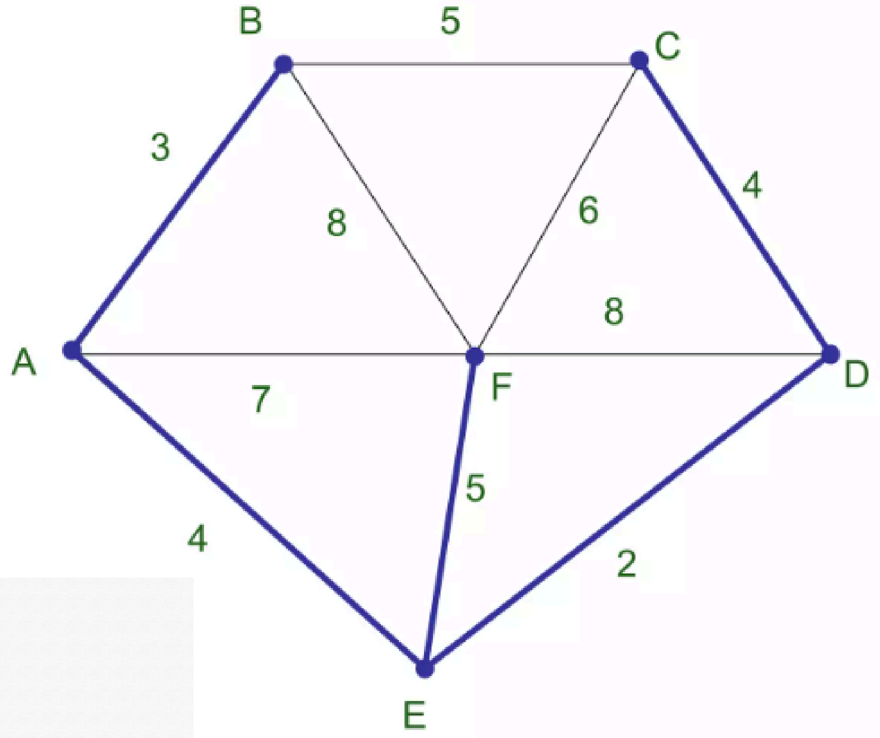

B
5
C
3
8
6
4
8
A
D
7
F
5
4
2
E

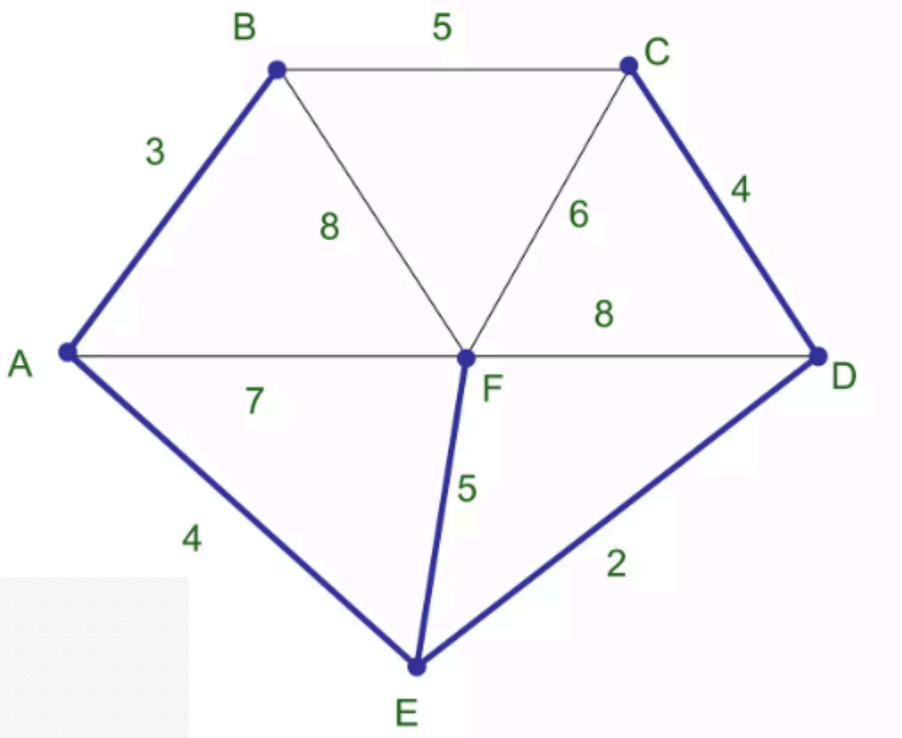

B
5
C
3
8
6
4
8
A
D
7
F
5
4
2
E


B
5
C
3
8
6
4
8
A
D
7
F
5
4
2
E


B
5
C
3
8
6
4
8
A
D
7
F
5
4
2
E


B
5
C
3
8
6
4
8
A
D
7
F
5
4
2
E


B
5
C
3
8
6
4
8
A
D
7
F
5
4
2
E



All the vertices have been connected.

The solution is:

ED 2
AB 3
CD 4
AE 4
EF 5

Total weight of the tree: 18

COMPARISON

TIME COMPLEXITY

- Prim's Algorithm: $O(V^2)$
- Kruskal's Algorithm: $O(ElogE)$

USE CASES

- Prim's Algorithm: Ideal for dense graphs with many edges between vertices.
- Kruskal's Algorithm: Suitable for sparse graphs with fewer edges.

OBSERVATIONS

- Both algorithms minimized road network construction costs effectively.
- Prim's Algorithm performed better for closely connected cities.
- Kruskal's Algorithm worked well for scattered city connections.

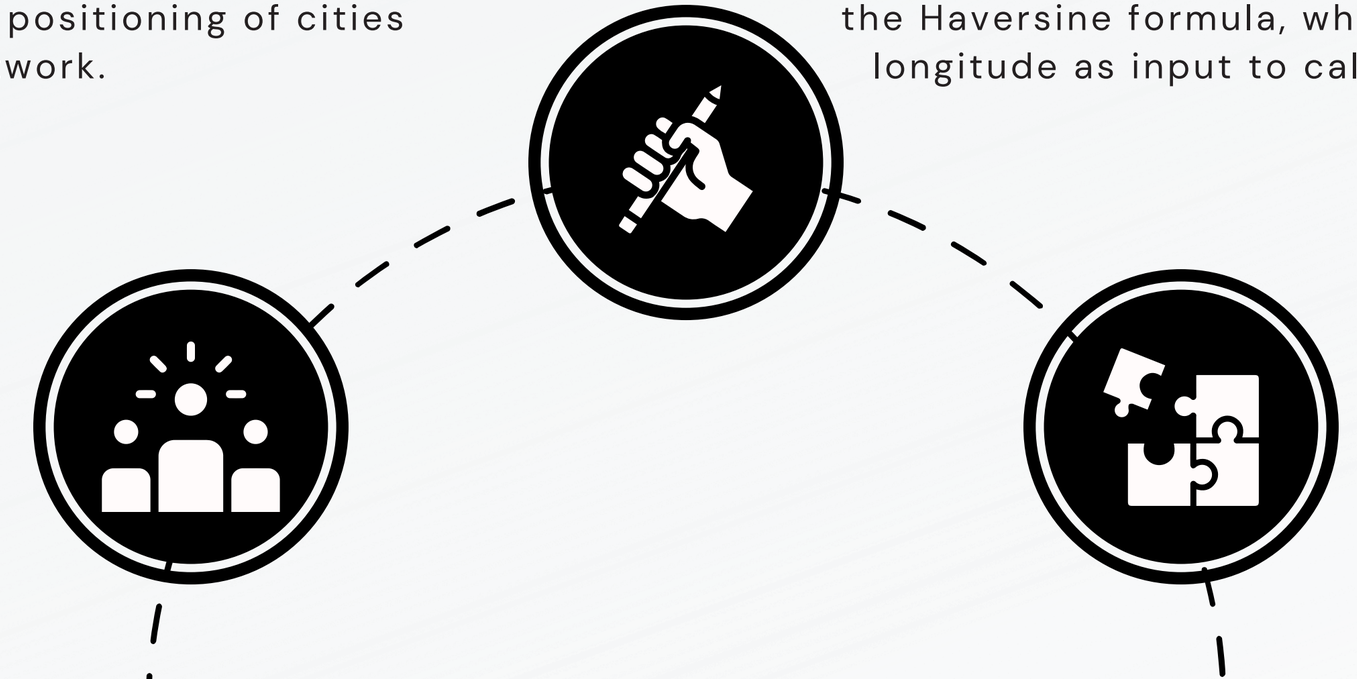
DATA COLLECTION

List of Cities

A comprehensive list of cities, along with their geographic coordinates, serves as the foundational data for the analysis. These coordinates are crucial for accurately determining the relative positioning of cities in the network.

Distance Calculation

The great-circle distance is used to calculate the shortest distance between two points on the Earth's surface, based on their coordinates. The formula for calculating the great-circle distance involves using the Haversine formula, which takes latitude and longitude as input to calculate the distance.



GRAPH CONSTRUCTION

- The cities were represented as vertices in the graph, each corresponding to a specific city
- The edges between the vertices represent the distances between pairs of cities, with the edge weights corresponding to the calculated distances
- The complete graph is fully connected, meaning every city is connected to every other city
- The graph visually displays all the cities and the corresponding connections, providing an overall structure for applying MST algorithms

CHALLENGES FACED

- Ensuring the accurate calculation of distances between cities, particularly when considering real-world geographical data
- For graphs with numerous nodes and edges, maintaining efficiency in terms of time and space complexity was challenging
- Overcoming issues like edge cases and ensuring all cities are included in the spanning tree
- Dealing with disconnected components and ensuring the algorithms work properly under these conditions was a key challenge.

FUTURE ENHANCEMENT

01

Create mobile apps for road network visualization and updates.

02

Extend to other infrastructure like water pipelines, electricity grids, and communication networks.

03

Adapt algorithms to accommodate real-time changes like road closures or new city additions

04

Use AI to predict traffic trends, enabling better route planning and optimized road usage during peak and off-peak hours.

CONCLUSION

This project focused on optimizing road networks by using Minimum Spanning Tree algorithms, specifically Prim's and Kruskal's algorithms, to minimize construction costs while ensuring connectivity between cities. By applying these algorithms, we were able to find the most efficient routes for road construction, significantly reducing both distance and costs. The outcomes demonstrate the practical application of MST in real-world road network planning, providing a cost-effective solution for infrastructure development. These algorithms can also be extended to other areas like power grids, telecommunications, and water pipelines for broader optimization.

THANK YOU