***Water Distribution Planning***

***Submitted by***

ABHYUD KRISHNA [BL.EN.U4AIE23103]

BANDI AMITH SREENIVASA REDDY [BL.EN.U4AIE23103]

SHREEHARI SM [BL.EN.U4AIE23103]

CHANDRAKIRAN KOPPARAPPU [BL.EN.U4AIE23103]

***in partial fulfillment for the award of the degree***

***of***

**BACHELOR OF TECHNOLOGY**

**IN**

**COMPUTER SCIENCE AND ARTIFICIAL INTELLIGENCE**



AMRITA SCHOOL OF COMPUTING

AMRITA VISHWA VIDYAPEETHAM

**BENGALURU 560 035**

May- 2024

**AMRITA VISHWA VIDYAPEETHAM**

**AMRITA SCHOOL OF COMPUTING, BENGALURU**



**BONAFIDE CERTIFICATE**

This is to certify that the Report entitled **“Water Distribution Planning ”** submitted by “Abhyud Krishna [BL.EN.U4AIE23103], Bandi Amith Sreenivasa Reddy [BL.EN.U4AIE23104], Shreehari SM [BL.EN.U4AIE23128] , Chandrakiran Kopparappu [BL.EN.U4AIE23141]” in partial fulfillment of the requirements for the End semester project of Object Oriented Paradigmin “**COMPUTER SCIENCE AND ARTIFICIAL INTELLIGENCE** “ is a bonafide record of the work carried out under my guidance and supervision at Amrita School of Engineering, Bengaluru.

SIGNATURE SIGNATURE

Ms. Vineetha Jain K. V & Mr. Ullas S. Dr. Gopalakrishnan E. A.

SUPERVISORS Principal & Chairperson

Asst. Professor Sr. Gr. , Dept. of CS&AI

Dept. of CSAI, ASC, Bengaluru. ASC, Bengaluru.

This project report was evaluated by us on 23/5/24

EXAMINER I EXAMINER II

# **ACKNOWLEDGEMENTS**

The satisfaction that accompanies successful completion of any task would be

incomplete without mention of people who made it possible, and whose constant

encouragement and guidance have been source of inspiration throughout the course of this project work. We offer our sincere pranams at the lotus feet of “AMMA”, MATA AMRITANANDAMAYI DEVI who showered her blessing upon us throughout the course of this project work. We owe my gratitude to Prof. Manoj P., Director, Amrita School of Engineering, Bangalore. We also thank Dr. Gopalakrishnan E. A., Principal & Chairperson, ASC & ASAI, , Bangalore for his supervision and motivation

It is a great pleasure to express our gratitude and indebtedness to our project guide Ms. Vineetha K.V, Assistant Professor (Sr. Gr.), & Mr. Ullas S. Assistant Professor (Sr. Gr.) Department of Computer Science and Engineering, Amrita School of Computing, Bangalore, for their valuable guidance, We would like to thank express our gratitude all faculty members for their academic support. Finally, we are forever grateful to our parents, who have loved, supported, and encouraged us in all our endeavors.

ABHYUD KRISHNA [BL.EN.U4AIE23103]

BANDI AMITH SREENIVASA REDDY [BL.EN.U4AIE23104]

SHREEHARI SM [BL.EN.U4AIE23128]

CHANDRAKIRAN KOPPARAPPU [BL.EN.U4AIE23141]

# **ABSTRACT**

This project proposes an integrated approach utilizing Kruskal's algorithm for minimum spanning tree generation and Dijkstra's algorithm for shortest path determination to optimize water distribution networks. Kruskal's algorithm is employed to design the most cost-effective network layout by connecting water sources, treatment plants, and distribution points while minimizing construction costs. Subsequently, Dijkstra's algorithm is applied to identify the shortest paths within this network, optimizing water flow efficiency and minimizing operational costs. The integration of these algorithms enables the development of a comprehensive solution for water distribution planning, promoting resource conservation, cost-effectiveness, and improved accessibility to clean water for communities.

|  |  |  |  |
| --- | --- | --- | --- |
| **TABLE OF CONTENTS**  **Chapters Page No.**  **1. Acknowledgement i**  **2. Abstract ii**  **3. Table of contents iii**  **4. List of Figures iv**    **Chapter:-**  **1. Introduction** 1  **2. Literature Survey**  2-3  **3. System Specifications** 4  **4.System Design**  5-6  **5. System Implementation** 7-14  **6. Results**  15-16  **7. Conclusion & Future Scope** 17  **8. References** 18 |  |  |  |

**List of Figures**

|  |  |  |
| --- | --- | --- |
| Figure  No | Title | Page  No |
| 5.1 | Input for vertices and edges | 7 |
| 5.2 | Edges and vertices initialized | 8 |
| 5.3 | Source, destination and weight for first edge | 8 |
| 5.4 | Source, destination and weight for second edge | 9 |
| 5.5 | Source, destination and weight for third edge | 9 |
| 5.6 | Source, destination and weight for fourth edge | 10 |
| 5.7 | Source, destination and weight for fifth edge | 10 |
| 5.8 | Source, destination and weight for sixth edge | 11 |
| 5.9 | Source, destination and weight for seventh edge | 11 |
| 5.10 | Number of water sources | 12 |
| 5.11 | Water Source 1 | 12 |
| 5.12 | Water source 2 | 13 |
| 5.13 | Water Source 3 | 13 |
| 5.14 | Destination Node | 14 |
| 6.1 | Result (Shortest path Found) | 15 |
| 6.2 | Result(Shortest path not found) | 16 |

**1.Introduction**

**Objective:**

The main objective of this project is to develop a comprehensive and efficient solution for water distribution planning by integrating Kruskal's algorithm for minimum spanning tree generation and Dijkstra's algorithm for shortest path determination. The goal is to optimize both the infrastructure layout and operational efficiency of the water distribution network, with aims to minimize construction costs, improve water flow efficiency, and enhance accessibility of water for communities.

**Aim:**

The project aims to optimize water distribution planning by integrating Kruskal's algorithm for minimum spanning tree generation and Dijkstra's algorithm for shortest path determination to minimize construction costs and improve operational efficiency.

**Motivation:**

* Traditional approaches often lack optimization, leading to suboptimal infrastructure layouts.
* Minimize construction costs.
* Improve water flow efficiency.

**Problem Statement:**

Optimize water distribution networks by integrating Kruskal's and Dijkstra's algorithms to minimize construction costs and enhance operational efficiency.

**2. Literature Survey**

[1] This paper explores the application of a pruning-based Minimum Spanning Tree algorithm to optimize water supply networks. By minimizing the total length of the network and ensuring efficient water distribution, the proposed method aims to reduce costs and improve service delivery. The study demonstrates the algorithm's effectiveness through simulation on a real-world dataset, highlighting its potential for practical implementation in urban planning.

In this method The primary focus is on minimizing the total length of the network, which assumes uniform costs or weights associated with the connections and May not efficiently handle dynamic changes in the network, such as fluctuating water demands . While effective for certain scenarios, the iterative pruning process can become complex and less efficient for very large networks.

[2] This paper explores the application of the Minimum Cost Spanning Tree algorithm to optimize urban highway networks. The study demonstrates how MST can minimize the total construction cost by connecting all necessary nodes with the least total length of roads. But this paper Focuses on connecting nodes with the minimum total length.

[3] This paper presents advancements in the Kruskal algorithm, aimed at improving its efficiency and applicability in various network design problems. Their research focuses on optimizing the algorithm’s performance, reducing computational complexity, and enhancing its practical application in real-world scenarios. While enhancing Kruskal’s algorithm for MST construction, the focus remains primarily on creating a minimum spanning tree without addressing shortest path optimization.

[4] This paper discusses the design of a real-time monitoring system for municipal water supply networks, emphasizing the optimization of monitoring points to ensure effective and efficient network management. The study highlights the importance of strategically placing monitoring points to enhance data collection, analysis, and decision-making processes.

[5] The application of the Minimum Spanning Tree algorithm in water supply networks provides a fundamental approach to cost reduction by minimizing the total network length. However, in the context of comprehensive water distribution system optimization, the Kruskal-Dijkstra approach offers significant advantages. By combining the benefits of MST construction with shortest path optimization, the Kruskal-Dijkstra approach addresses limitations related to scalability, redundancy, and path efficiency, ensuring a more robust and efficient water distribution system.

[6] This paper investigates the application of both the Shortest Path Algorithm (SPA) for path optimization and the Minimum Spanning Tree algorithm in the reconfiguration of distribution networks. The study aims to enhance network efficiency by combining the strengths of both algorithms to optimize network structure and operation.

[7] This paper presents a methodology for the optimal operation of water distribution networks using predictive control based on Mixed-Integer Nonlinear Programming (MINLP). The study aims to optimize the operation of the network in real-time by considering dynamic constraints and predictive models.

Predictive control using MINLP can be computationally intensive, especially for large-scale water distribution networks.

[8] This paper focuses on the development of algorithms and software designed to address geometric path planning problems. The study aims to improve the efficiency and accuracy of path planning solutions through advanced algorithmic approaches and robust software implementations. Algorithms may face challenges when scaling to large and complex networks, particularly in real-time applications.

1. **System Specifications**

3.1 Hardware Requirements

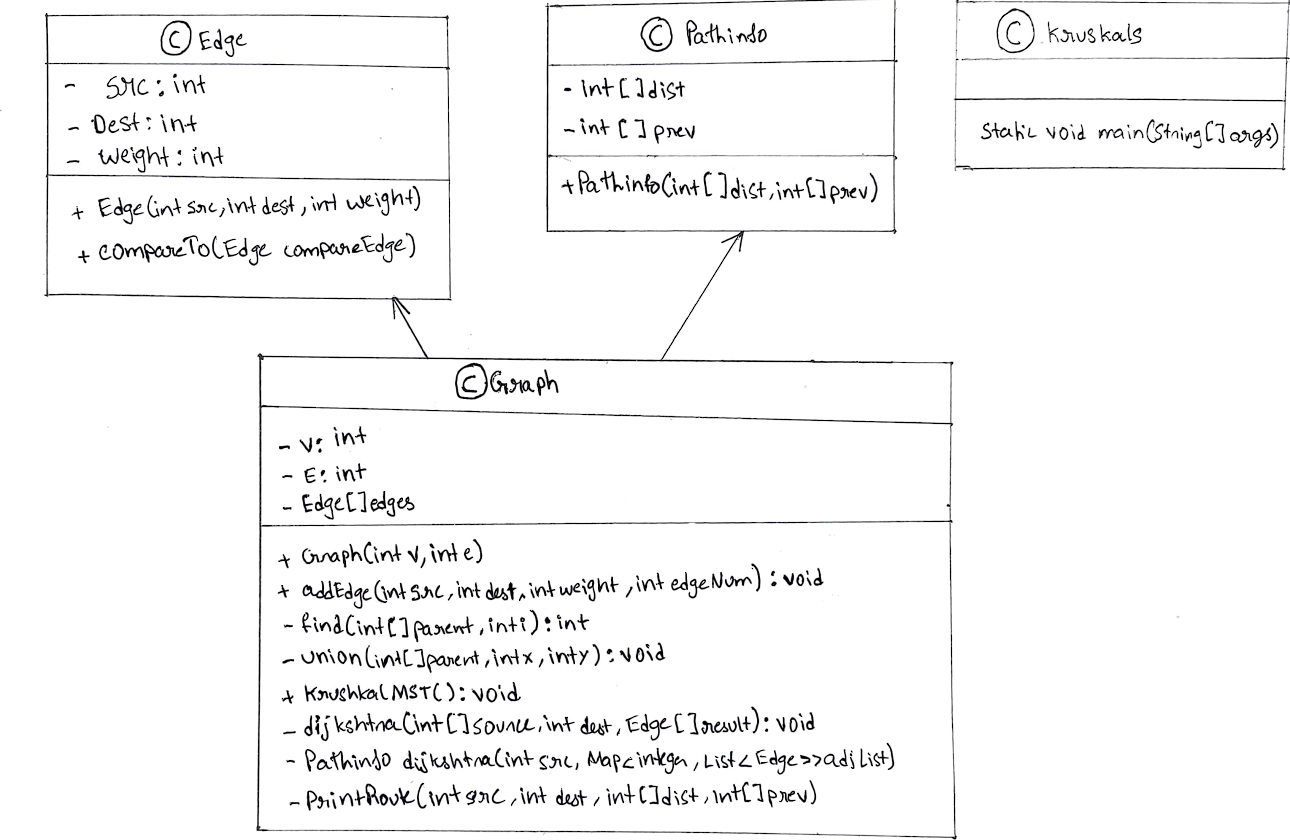
* Windows 10/11
* Intel(R) Core (TM) i5-72001J CPU @ 2.50GHz 2.71 GHz
* Installed RAM:16 GB
* System type:64-bit operating system, x64-based processor

3.2 Software Requirements

* Eclipse IDE
* Java swing

1. **System Design**

**UML DIAGRAM**

****

**Classes**

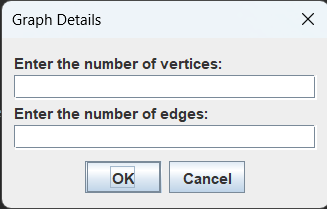
1. Edge
   * Represents an edge in the graph.
   * Attributes:
     + src (int): The source vertex of the edge.
     + dest (int): The destination vertex of the edge.
     + weight (int): The weight of the edge.
   * Methods:
     + Edge(int src, int dest, int weight): Constructor to initialize an edge with source, destination, and weight.
     + int compareTo(Edge compareEdge): Compares edges based on their weight for sorting purposes.
2. Graph
   * Represents the graph and contains methods to perform Kruskal's MST algorithm and Dijkstra's shortest path algorithm.
   * Attributes:
     + V (int): Number of vertices in the graph.
     + E (int): Number of edges in the graph.
     + Edge[] edges: Array of edges in the graph.
   * Methods:
     + Graph(int v, int e): Constructor to initialize the graph with a specified number of vertices and edges.
     + void addEdge(int src, int dest, int weight, int edgeNum): Adds an edge to the graph.
     + int find(int[] parent, int i): Finds the parent of the set containing vertex i.
     + void union(int[] parent, int x, int y): Merges the sets containing vertices x and y.
     + void kruskalMST(): Computes the Minimum Spanning Tree (MST) using Kruskal's algorithm.
     + void dijkstraMST(int[] sources, int dest, Edge[] result): Finds the shortest path from any of the water sources to the destination using Dijkstra's algorithm on the MST.
     + PathInfo dijkstra(int src, Map<Integer, List<Edge>> adjList): Computes the shortest paths from a source vertex using Dijkstra's algorithm.
     + void printRoute(int src, int dest, int[] dist, int[] prev): Prints the shortest route from the source to the destination.
3. PathInfo
   * Helper class to store distance and previous node information for Dijkstra's algorithm.
   * Attributes:
     + int[] dist: Array to store the shortest distance from the source to each vertex.
     + int[] prev: Array to store the previous vertex in the shortest path.
   * Methods:
     + PathInfo(int[] dist, int[] prev): Constructor to initialize the distance and previous vertex arrays.
4. Kruskals
   * Main class that contains the main method to run the program.
   * Methods:

static void main(String[] args): Entry point of the program .

**Relationships**

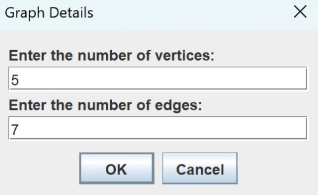
* The Graph class uses the Edge class to represent the edges in the graph.
* The Graph class also uses the PathInfo class to store the results of Dijkstra's algorithm.
* This is shown by a association relationship.

1. **System Implementation**



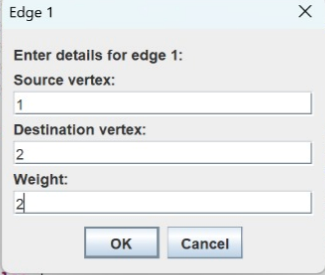
**Fig5.1 Input For Vertices And Edges**

In Fig5.1 The image shows a graphical user interface dialog box titled “Graph Details.” This dialog box is used to gather initial input from the user regarding the graph they want to work with. It contains two labeled text fields and two buttons, “OK” and “Cancel.” The label “Enter the number of Vertices” prompts the user to input the number of vertices (nodes) in the graph. Vertices are fundamental units or points in the graph where edges (connections) meet. This label prompts the user to input the number of edges in the graph. Edges are the connections between the vertices.



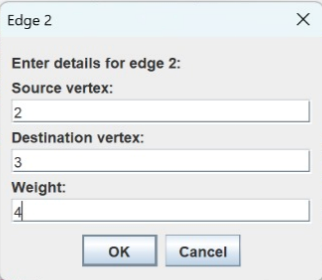
**Fig5.2 Edges and Vertices Initialized**

In Fig5.2 By specifying the number of vertices and edges, the user helps define the structure of the graph, which will be used later to input individual edges and their weights, and eventually perform operations like finding the Minimum Spanning Tree using Kruskal's algorithm and calculating shortest paths using Dijkstra's algorithm.



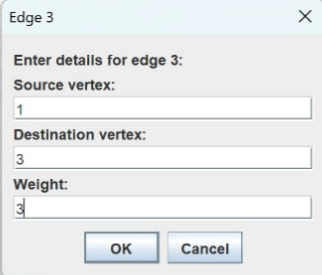
**Fig5.3 Source Destination And Weight For First Edge**

In Fig 5.3 The values i.e the source, destination and weight of the first edge is inputted.By entering these values, the user defines the connections and associated weights between vertices, which are then used by the underlying algorithms (Kruskal's and Dijkstra's) to compute the Minimum Spanning Tree (MST) and the shortest paths.



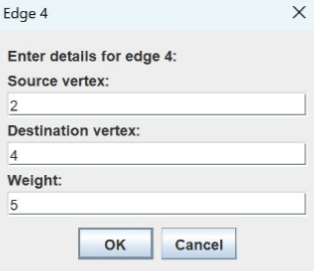
**Fig5.4 Source Destination And Weight For Second Edge**

In Fig 5.4 The values i.e the source, destination and weight of the second edge is inputted.By entering these values, the user defines the connections and associated weights between vertices, which are then used by the underlying algorithms (Kruskal's and Dijkstra's) to compute the Minimum Spanning Tree (MST) and the shortest paths.



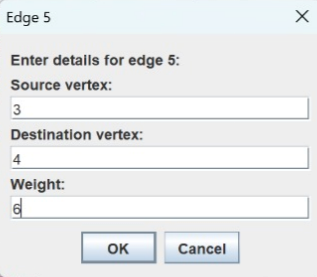
**Fig.5.5 Source Destination And Weight For Third Edge**

In Fig5.5 The values i.e the source, destination and weight of the third edge is inputted.By entering these values, the user defines the connections and associated weights between vertices, which are then used by the underlying algorithms (Kruskal's and Dijkstra's) to compute the Minimum Spanning Tree (MST) and the shortest paths.



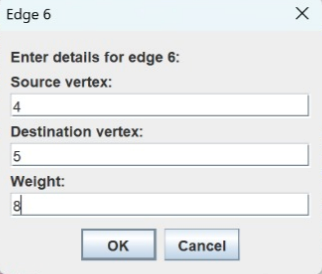
**Fig5.6 Source Destination And Weight For Fourth Edge**

In Fig5.6 The values i.e the source, destination and weight of the fourth edge is inputted.By entering these values, the user defines the connections and associated weights between vertices, which are then used by the underlying algorithms (Kruskal's and Dijkstra's) to compute the Minimum Spanning Tree (MST) and the shortest paths.



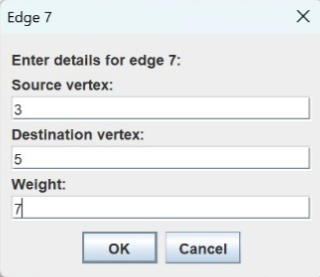
**Fig5.7 Source Destination And Weight For FifthEdge**

In Fig5.7 The values i.e the source, destination and weight of the fifth edge is inputted.By entering these values, the user defines the connections and associated weights between vertices, which are then used by the underlying algorithms (Kruskal's and Dijkstra's) to compute the Minimum Spanning Tree (MST) and the shortest paths.



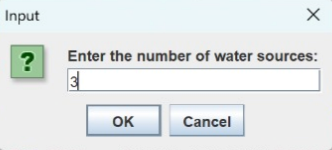
**Fig5.8 Source Destination And Weight For Sixth Edge**

In Fig5.8 The values i.e the source, destination and weight of the sixth edge is inputted.By entering these values, the user defines the connections and associated weights between vertices, which are then used by the underlying algorithms (Kruskal's and Dijkstra's) to compute the Minimum Spanning Tree (MST) and the shortest paths.



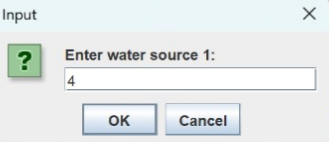
**Fig5.9 Source Destination And Weight For Seventh Edge**

In Fig5.9 The values i.e the source, destination and weight of the seventh edge is inputted.By entering these values, the user defines the connections and associated weights between vertices, which are then used by the underlying algorithms (Kruskal's and Dijkstra's) to compute the Minimum Spanning Tree (MST) and the shortest paths.



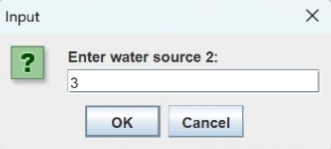
**Fig5.10 Number Of Water Sources**

In Fig 5.10 The primary purpose of this box is to determine how many water sources are present in the graph. This information is necessary to know how many times to prompt the user to input the specific nodes representing these water sources. Based on the number entered, the program will know how many subsequent input boxes to display. The program uses Dijkstra’s algorithm to find the shortest path from each water source to a specified destination node.



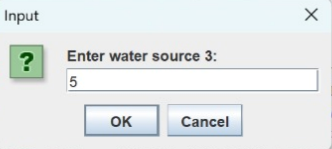
**Fig5.11 Water Source 1**

In Fig5.11 The label "Enter water source 1:" prompts the user to provide the node number for the first water source. The program uses Dijkstra’s algorithm to find the shortest path from each water source to a specified destination node. The program will compare the shortest paths from all specified water sources to the destination and choose the best (shortest) path.



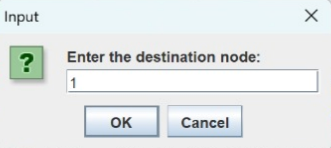
**Fig5.12 Water Source 2**

In Fig5.12 The label "Enter water source 2:" prompts the user to provide the node number for the second water source. The program uses Dijkstra’s algorithm to find the shortest path from each water source to a specified destination node. This helps in determining the most efficient source of water to supply to the destination node.



**Fig5.13 Water Source 3**

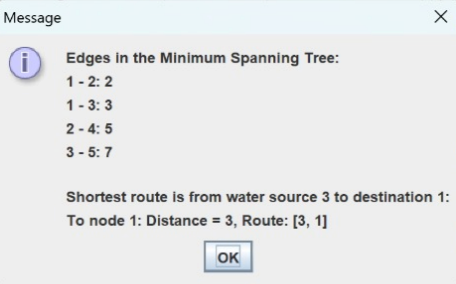
In Fig5.13 The label "Enter water source 3:" prompts the user to provide the node number for the third water source. The program uses Dijkstra’s algorithm to find the shortest path from each water source to a specified destination node. This helps in determining the most efficient source of water to supply to the destination node.



**Fig5.14 Destination Node**

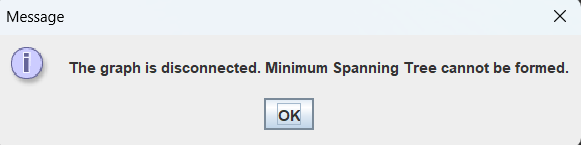
In Fig 5.14 The label "Enter the destination node:" asks the user to provide the node number of the destination within the graph. The text field below this label shows the user-entered value, which in this case is 1. The program uses Dijkstra’s algorithm to find the shortest path from each water source to a specified destination node. This input is crucial for the program’s later stages, specifically for the Dijkstra's algorithm part of the code where the shortest path from these water sources to a specified destination node is calculated. The program collects these water source nodes to determine the best source to supply water to the destination node based on the shortest path in the Minimum Spanning Tree of the graph.

1. **Results**



**Fig6.1 Result (Shortest Path Found)**

In fig 6.1 , The primary purpose of this dialog box is to show the user the edges that are included in the MST of the graph. These edges form a tree that connects all the vertices with the minimum possible total edge weight. By displaying the edges, the user can see which connections (edges) are part of the MST and understand how the vertices are interconnected in the most efficient way. After showing the MST, the program proceeds to collect information about water sources and the destination node, leading to Dijkstra’s algorithm for finding the shortest path from these sources to the destination. Each edge in the MST is represented as a connection between two vertices with a specific weight. For example, an edge "1 - 2: 4" means there is a connection between vertex 1 and vertex 2 with a weight of 4.



**Fig6.2 Result (Shortest Path Not Found)**

In Fig 6.2 , The primary purpose of this dialog box is to inform the user that the graph is disconnected, which means it is impossible to connect all vertices with the given edges. It explains to the user why the MST cannot be formed. An MST requires all vertices to be connected, and a disconnected graph does not meet this criterion. The dialog box prompts the user to acknowledge the situation and understand that no further MST-related operations can proceed with the current graph configuration. When Kruskal's algorithm is executed, it attempts to add edges to form the MST. The algorithm requires V-1 edges to connect all V vertices. During the execution, if Kruskal's algorithm cannot find V-1 edges that connect all vertices, it concludes that the graph is disconnected. This situation arises when there are vertices that cannot be reached from others due to missing connections.

1. **Conclusion and Future Scope**

**Conclusion:**

* By leveraging Kruskal's algorithm for constructing a Minimum Spanning Tree , the system ensures the most efficient connections between various nodes in the network, minimizing the total pipe length and associated costs.
* The implementation of Dijkstra's algorithm allows for the identification of the shortest and most efficient paths from multiple water sources to the destination nodes. This ensures that water distribution is not only cost-effective but also timely, reducing wastage and improving resource allocation.

**Future Scope:**

* Using machine learning algorithms to predict future water demand based on historical data, population growth, and other factors can improve planning and resource allocation.
* Implementing spatial analysis techniques to assess network connectivity, identify vulnerable points, and optimize routing paths based on geographic features.

1. **References**

[1] Mao, Hua, Liwei Xie, and Zhixing Zhang. "Applying pruning-based minimum spanning tree algorithm in water supply networks." *Jisuanji Yingyong yu Ruanjian* 28.2 (2011).

[2] C. Y. Xue, "The Application of Minimum Cost Spanning Tree to Solve the Question of Urban Highway," *Computer Programming Skills & Maintenance*, vol. 16, no. 6, pp. 30-31, 2009.

[3] Li, Haiming, Qiyang Xia, and Yong Wang. "Research and improvement of kruskal algorithm." *Journal of Computer and Communications* 5.12 (2017): 63.

[4] Lin, Y. "Design of real-time monitoring system for municipal water supply network and optimization of monitoring points." *South China University of Technology* (2013).

[5] Cong, Fengxia, and Ying Zhao. "The application of minimum spanning tree algorithm in the water supply network." *2015 International Industrial Informatics and Computer Engineering Conference*. Atlantis Press, 2015.

[6] L. Wang, "The Research of Application in Distribution Network Reconfiguration Based on Shortest Path Algorithm and Minimum Spanning Tree Algorithm," *Master's Degree of Xi'an University of Technology*, 2009.

[7] Biscos, C., et al. "Optimal operation of water distribution networks by predictive control using MINLP." *Water Sa* 29.4 (2003): 393-404.

[8] Chen, Danny Z. "Developing algorithms and software for geometric path planning problems." *ACM Computing Surveys (CSUR)* 28.4es (1996): 18-es.