Logo

Description automatically generated with medium confidence

Fiber Optic Trajectory Optimizing System

(Using Prim’s Algorithm)

Project Report

**Resource Person:**

Professor Dr. Jawad Shamsi

**Course:**

Data Structures

**Group Members:**

|  |  |  |
| --- | --- | --- |
| **Name** | **ID #** | **Contribution** |
| Muhammad Talha | 21K-3349 | Implementation, Testing and Console Manipulation |
| Muhamad Hamza | 21K-4579 | Implementation and Testing/Debugging |
| Muhammad Taha | 21K-3316 | Implementation and Compilation/Documentation |

National University of Computer and Emerging Sciences

Department of Computer Science

Karachi, Pakistan

Tile of the Project: Fiber Optic Trajectory Optimizing System (using Prim’s Algorithm)

Concepts Used: Data Structures (LINKED LIST, MIN-HEAP, GRAPH)

Other/s: File Handling, OOP

# Problem Statement:

A Fiber Optic Trajectory Optimizing System is an application that helps optimizing fiber optics trajectory planning for minimum cost of cabling. This problem is very important since fiber optic is expensive and if not installed optimally, it will cost enormously. Prim's algorithm can optimize by calculating the minimum spanning tree used for fiber optic cable installation. It streamlines and accelerates the transmission of data from source to destination. This optimization is done by preventing graphs from forming cycles. Prim’s algorithm ranks its weight from large to small and make minimum spanning tree. We will implement an efficient Fiber Optic Trajectory Optimizing System by using LINKED LIST, MIN-HEAP and GRAPH along with file handling for effective insertion.

# Functional Details:

1. **Insert Edge:**

This function adds the edge to an adjacency list graph. Time complexity is O (2). It adds edge formed by 2 vertices (say x and y). Since it deals with an undirected graph, therefore it makes ‘link’ from both vertices to each other.

1. **Make Minimum Spanning Tree:**

This function makes minimum spanning tree (MST) of an adjacency list graph inputted by using ‘file handling’. Time complexity of this function is O () when using adjacency matrix, where V is the number of vertices in graph but O ((V + E) log V) when using adjacency list and heaps.

1. **Printing MST:**

This function prints minimum spanning tree in this format:

Edge Weight

e.g.

X-Y W

1. **Calculating Total Weight of MST:**

This function calculates total weight of MST by adding the weight of all edges in MST one-by-one.

# References:

* <https://www.geeksforgeeks.org/prims-mst-for-adjacency-list-representation-greedy-algo-6>
* Books: Problem Solving in Data Structures & Algorithms (Hemanth Jain)

# Project Prototyping:

1. Class ‘Link’ with all function prototypes:

Graphical user interface, text, application, email

Description automatically generated

1. Class ‘Heap’ with all function prototypes:

Graphical user interface, text, application

Description automatically generated

1. Class ‘Graph’ with all function prototypes:

Text

Description automatically generated

1. Execution:

Text

Description automatically generated

A picture containing table

Description automatically generated

Graphical user interface, text, application

Description automatically generated

Text

Description automatically generated with medium confidence

Shape, square

Description automatically generated

Graphical user interface, text, application

Description automatically generated

Text

Description automatically generated with medium confidence