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.

2. **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 (V^2) 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.

3. **Printing MST:**
   This function prints minimum spanning tree in this format:
   Edge Weight
   e.g.
   X-Y W

4. **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:

```
struct listNode //Linked-List Node
{
    int target, weight;
    listNode * next;
    listNode(int t, int w) { target = t, weight = w, next = NULL; }
    ~listNode() { next = NULL; }
};

class List //Linked-List
{
    private:
        listNode *head;
    public:
        List() { head = NULL; }
        void insertAtStart(int, int);  //inserts listNode at start of List for quick insertion
        listNode* getHead() { return head; }
        ~List();
};
```

2. Class 'Heap' with all function prototypes:

```
struct heapItem //Heap Node
{
    int value, key;
    heapItem(int v, int k) { value = v, key = k; }
};

struct minHeap //Minimum Heap
{
    int size, capacity, *position; //var 'position' will be used in 3 functions as a helper
                                   //to get current index of node in minHeap
    heapItem** heapArray;
    minHeap(int);
    void swapHeapItem(heapItem **, heapItem **);
    void minHeapify(int);
    heapItem* extractMin();
    void decreaseKey(int v, int k); //decreases key value 'k' of vertex
    ~minHeap();
};
```

3. Class ‘Graph’ with all function prototypes:

```
class Graph //Graph Data Structure
{
        int numOfVertices;
        List *adjList;
    public:
        Graph(int n) { numOfVertices = n, adjList = new List[numOfVertices]; }
        void insertEdge(int, int, int); //makes 'link' from both vertices
                                        //to each other
        int getNumOfVertices() { return numOfVertices; }
        int * make_MST_via_PrimAlgo();
        int getWeightOfEdge(int x, int y); //returns weight of edge xy
        void printMST(int []); //prints all edges of MST along with their weight
        int getMSTWeight(int []); //returns total weight of all edges in MST
        void printGraph();  //to print graph
        ~Graph() { delete[] adjList; }
};
```

4. Execution:

```
--------------------------------------------------------------------------------------------------------------
                                        |FIBER OPTIC TRAJECTORY OPTIMIZING SYSTEM|
--------------------------------------------------------------------------------------------------------------



                                            Muhammad Anas



1. Login
2.Register
Entire desired option:
```

```
Adding edge '88 - 6' with weight 9040...
Adding edge '42 - 64' with weight 2648...
Adding edge '46 - 5' with weight 5890...
Adding edge '29 - 70' with weight 5350...
Adding edge '6 - 1' with weight 4393...
Adding edge '48 - 29' with weight 2623...
Adding edge '84 - 54' with weight 8756...
Adding edge '40 - 66' with weight 7376...
Adding edge '31 - 8' with weight 6944...
Adding edge '39 - 26' with weight 1323...
Adding edge '37 - 38' with weight 6118...
Adding edge '82 - 29' with weight 6541...
Adding edge '33 - 15' with weight 4639...
Adding edge '58 - 4' with weight 9930...
Adding edge '77 - 6' with weight 1673...
Adding edge '86 - 21' with weight 8745...
Adding edge '24 - 72' with weight 6270...
Adding edge '29 - 77' with weight 5573...
Adding edge '97 - 12' with weight 3986...
Adding edge '90 - 61' with weight 8636...
Adding edge '55 - 67' with weight 3655...
Adding edge '74 - 31' with weight 2052...
Adding edge '41 - 24' with weight 3966...
Adding edge '30 - 7' with weight 8007...
Adding edge '37 - 57' with weight 2287...
Adding edge '53 - 83' with weight 4945...
Adding edge '58 - 21' with weight 8588...
Adding edge '22 - 46' with weight 7506...
Adding edge '30 - 13' with weight 9168...
Adding edge '62 - 55' with weight 7410...
```

```
                                  ----------------
                                 | PROJECT FOTOS |
                                  ----------------
-------------------------------------------------------------------------------------------
|                          FIBER OPTIC TRAJECTORY OPTIMIZING SYSTEM                        |
-------------------------------------------------------------------------------------------
| 0 = EXIT PROGRAM                                                                        |
| 1 = MAKE MST VIA PRIM'S ALGORITHM                                                       |
| 2 = PRINT MST                                                                           |
| 3 = CALCULATE MST WEIGHT                                                                   |
| 4 = PRINT GRAPH
                                     |
-------------------------------------------------------------------------------------------

Enter your choice:
```

```
Node (0)---(1, 8467)

Node (1)---(6, 4393)---(0, 8467)

Node (2)---(85, 4182)---(89, 3195)---(50, 9374)

Node (3)---(10, 6869)---(48, 5200)---(98, 6224)

Node (4)---(58, 9930)---(91, 3902)

Node (5)---(46, 5890)---(45, 3281)

Node (6)---(77, 1673)---(1, 4393)---(88, 9040)

Node (7)---(21, 4310)---(30, 8007)

Node (8)---(44, 2609)---(31, 6944)

Node (9)---(98, 7157)---(16, 8935)

Node (10)---(3, 6869)

Node (11)---(56, 6202)---(22, 7673)

Node (12)---(97, 3986)---(67, 6299)

Node (13)---(17, 9514)---(30, 9168)
```

```
Edge           Weight
0 - 1            8467
85 - 2           4182
48 - 3           5200
91 - 4           3902
46 - 5           5890
1 - 6            4393
21 - 7           4310
44 - 8           2609
98 - 9           7157
3 - 10           6869
22 - 11          7673
67 - 12          6299
30 - 13          9168
0 - 14           0
33 - 15          4639
9 - 16           8935
13 - 17          9514
58 - 18          9796
49 - 19          1556
96 - 20          4021
58 - 21          8588
72 - 22          8538
87 - 23          9314
41 - 24          3966
55 - 25          3434
39 - 26          1323
61 - 27          2995
0 - 28           0
77 - 29          5573
```

```
|                          FIBER OPTIC TRAJECTORY OPTIMIZING SYSTEM                          |
-----------------------------------------------------------------------------------------------
| 0 = EXIT PROGRAM                                                                           |
| 1 = MAKE MST VIA PRIM'S ALGORITHM                                                          |
| 2 = PRINT MST                                                                              |
| 3 = CALCULATE MST WEIGHT                                                                       |
| 4 = PRINT GRAPH
                                          |
-----------------------------------------------------------------------------------------------

Enter your choice: 3

Total weight of MST: 434770

Press any key to continue . . .
```