LABORATORY REPORT

Algorithm Laboratory (CS-39001)

B.Tech Program in ECSc

Submitted By

Name:- SANNIDHI DEB

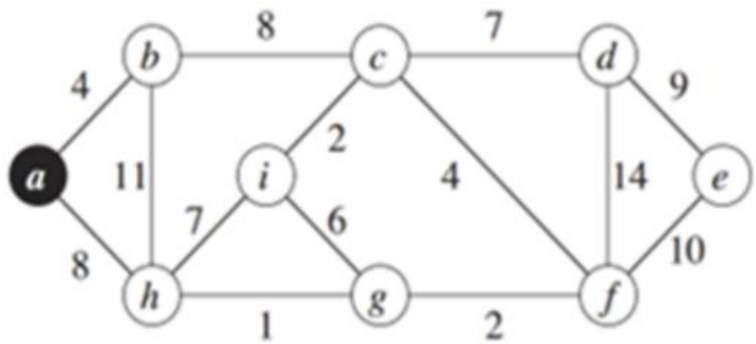
Roll No: 2330044

Kalinga Institute of Industrial Technology

(Deemed to be University) Bhubaneswar, India

Autumn, 2025

Table of Contents

Experiment Number	7.1
Experiment Title	Given an undirected weighted connected graph G(V, E) and starring vertex ‘s’. Maintain a Min-Priority Queue ‘Q’ from the vertex set V and apply Prim’s algorithm to : ● find the minimum spanning tree T(V, E’), and display the cost adjacency matrix of ‘T’, and ● display the total cost of the minimum spanning tree T. Given :  Input: Enter the Number of Vertices: 9 Enter the Starting Vertex: 1 (that is, a)
Date of Experiment	01/10/2025
Date of Submission	10/10/2025

1.Algorithm:-

START

1. Initialize all vertices with infinite key values, except the starting
2. Maintain a Min-priority queue 'Q' containing all vertices with key values.
3. while 'Q' is not empty
   i) extract vertex 'u' with smallest key value.
   ii) for each adjacent vertex 'v' of 'u':
   if 'v' is still in 'Q' and weight (u, v) < key[v], then update key[v] = weight (u, v) and parent [v] = u
4. After vertices are processed, edges (parent[v], v) form MST.
5. Display the adjacency matrix of MST and total cost

END

Sannidhi Deb, 2330044

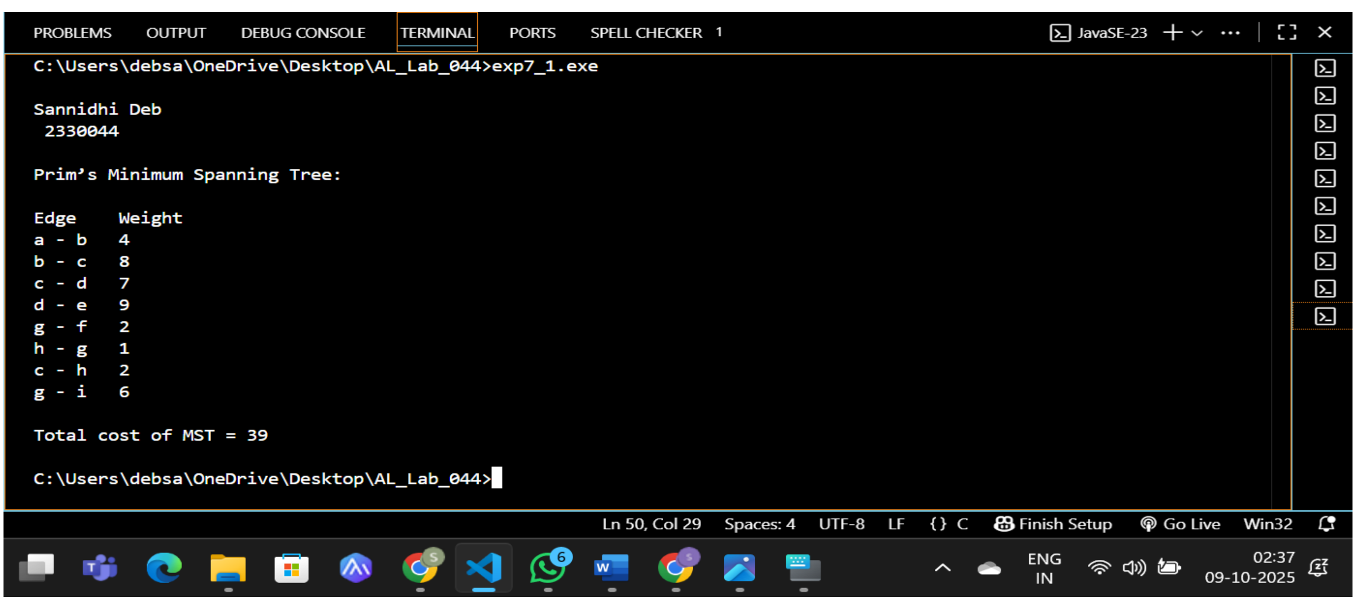
2. Code:-

```
#include <stdio.h>

#include <limits.h>

#define V 9

int minKey(int key[], int mstSet[]) {

   int min = INT_MAX, min_index;

   for (int v = 0; v < V; v++)

      if (mstSet[v] == 0 && key[v] < min)

         min = key[v], min_index = v;

   return min_index;

}

void printMST(int parent[], int graph[V][V]) {

   int totalCost = 0;
```

```
    printf("\nEdge \tWeight\n");
    for (int i = 1; i < V; i++) {
        printf("%c - %c \t%d \n", parent[i] + 'a', i + 'a', graph[i][parent[i]]);
        totalCost += graph[i][parent[i]];
    }
    printf("\nTotal cost of MST = %d\n", totalCost);
}
void primMST(int graph[V][V]) {
    int parent[V];
    int key[V];
    int mstSet[V];
  for (int i = 0; i < V; i++)
        key[i] = INT_MAX, mstSet[i] = 0;
key[0] = 0;
    parent[0] = -1;
for (int count = 0; count < V - 1; count++) {
        int u = minKey(key, mstSet);
        mstSet[u] = 1;
for (int v = 0; v < V; v++)
            if (graph[u][v] && mstSet[v] == 0 && graph[u][v] < key[v])
                parent[v] = u, key[v] = graph[u][v];
    }printMST(parent, graph);
}
int main() {
      int graph[V][V] = {
```

```
        {0,4,0,0,0,0,8,11,0},
        {4,0,8,0,0,0,0,0,0},
        {0,8,0,7,0,4,0,2,0},
        {0,0,7,0,9,14,0,0,0},
        {0,0,0,9,0,10,0,0,0},
        {0,0,4,14,10,0,2,0,0},
        {8,0,0,0,0,2,0,1,6},
        {11,0,2,0,0,0,1,0,7},
        {0,0,0,0,0,0,6,7,0}
    };
    printf("\nSannidhi Deb\n 2330044\n\n");
    printf("Prim’s Minimum Spanning Tree:\n");
    primMST(graph);
    return 0;
}
```

3.Results/Output:- Entire Screen Shot including Date & Time:-



4. Remarks:-

1. What type of algorithm is used?

1. The algorithm used for finding MST is Prim's Algorithm
Sannidhi Deb, 2330044

2. Analyze the complexity of your algorithm.

2. The time complexity using adjacency matrix is $O(V^2)$ and implementing min-heap is $O(E \log V)$ where 'V' is no. of vertex and 'E' is for edge
Sannidhi Deb, 2330044

3. Any other observations?

3. MST connects all vertices with minimum total edge weights and no cycles.
The starting vertex does not change the total cost but the order of chosen edges.
Sannidhi Deb, 2330044

Experiment Number	7.2
Experiment Title	For the same graph, apply Krushkal's algorithm to ● find the minimum spanning tree T(V, E'), and display the selected edges, and ● display the total cost of the minimum spanning tree T.
Date of Experiment	01/10/2025
Date of Submission	10/10/2025

1.Algorithm:-

START

1. Sort all edges of graph in increasing order of their weights.
2. Initialize the MST as an empty set.
3. For each edge (u,v) in sorted list:
   i) if including edge does not form a cycle, add to MST
4. Repeat until MST has (V-1) edges.
5. Display the edges in MST and compute total cost.

END

Sannidhi Deb, 2330044

2. Code:-

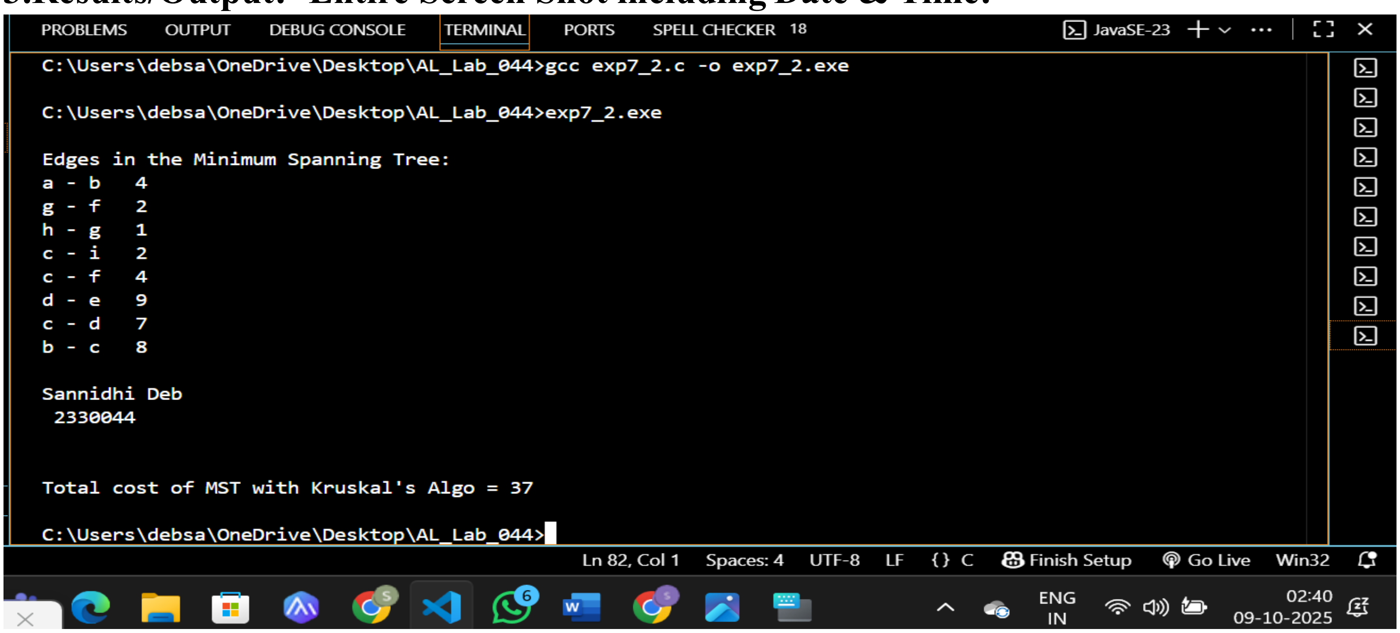
```
#include <stdio.h>

#include <stdlib.h>

#define V 9

#define E 14

struct Edge {
```

```
    int src, dest, weight;
};
struct subset {
    int parent, rank;
};
int find(struct subset subsets[], int i) {
    if (subsets[i].parent != i)
        subsets[i].parent = find(subsets, subsets[i].parent);
    return subsets[i].parent;
}
void Union(struct subset subsets[], int x, int y) {
    int xroot = find(subsets, x);
    int yroot = find(subsets, y);
  if (subsets[xroot].rank < subsets[yroot].rank)
        subsets[xroot].parent = yroot;
    else if (subsets[xroot].rank > subsets[yroot].rank)
        subsets[yroot].parent = xroot;
    else {
        subsets[yroot].parent = xroot;
        subsets[xroot].rank++;
    }
}
int compareEdges(const void* a, const void* b) {
    struct Edge* a1 = (struct Edge*)a;
    struct Edge* b1 = (struct Edge*)b;
```

```
    return a1->weight > b1->weight;
}
void KruskalMST(struct Edge edges[]) {
    struct subset subsets[V];
    struct Edge result[V];
    int e = 0; // index for result[]
    int i = 0; // index for sorted edges
    qsort(edges, E, sizeof(edges[0]), compareEdges);
    for (int v = 0; v < V; v++) {
        subsets[v].parent = v;
        subsets[v].rank = 0;
    }
while (e < V - 1 && i < E) {
        struct Edge next_edge = edges[i++];
         int x = find(subsets, next_edge.src);
        int y = find(subsets, next_edge.dest);
 if (x != y) {
            result[e++] = next_edge;
            Union(subsets, x, y);
        }
    }
printf("\nEdges in the Minimum Spanning Tree:\n");
    int totalCost = 0;
    for (i = 0; i < e; i++) {
        printf("%c - %c \t%d\n", result[i].src + 'a', result[i].dest + 'a', result[i].weight);
```

```
        totalCost += result[i].weight;
    }
    printf("\nSannidhi Deb\n 2330044\n\n");
    printf("\nTotal cost of MST with Kruskal's Algo = %d\n", totalCost);
}
int main() {
    struct Edge edges[E] = {
        {0,1,4}, {0,7,8}, {1,2,8}, {2,3,7}, {3,4,9}, {4,5,10},
        {3,5,14}, {2,5,4}, {2,8,2}, {7,8,7}, {7,6,1}, {6,5,2},
        {6,8,6}, {1,7,11}
    };
  KruskalMST(edges);
    return 0;
}
```

3.Results/Output:- Entire Screen Shot including Date & Time:-



4. Remarks:-

1. What type of algorithm is used?

The algorithm used for finding MST using edge-sorting is Kruskal's Algorithm.

Sannidhi Deb, 2330044

2. Analyze the complexity of your algorithm.

The time complexity is $O(E \log V)$ because of union-find technique where 'E' stands for edge and 'V' for vertices.

Sannidhi Deb, 2330044

3. Any other observations?

Kruskal's algorithm is greedy and works good on sparse graph.
It uses disjoint set (union-find) to detect cycles.
The resulting MST is unique if all edge weights are distinct.

Sannidhi Deb, 2330044

5. Conclusion:-

Both Prim's and Kruskal's algorithms were successfully implemented to find the Minimum Spanning Tree (MST) of a connected weighted graph.The experiments demonstrated how both algorithms use greedy approaches to minimize the total edge cost while connecting all vertices without forming cycles.

Prim's algorithm grows the MST from a starting vertex using a priority queue, Kruskal's algorithm builds it by selecting the smallest edges and applying union–find to avoid cycles. In both cases, the resulting MST had the same total cost, confirming the correctness of the implementations.

Sannidhi Deb
(2330044)

Signature of the FIC

Sannidhi Deb

(Name of the FIC)