Design and Analysis of Algorithm

Assignment 6

Group 9 :

- Aditya Aggarwal - IIT2019210
- Divy Agrawal - IIT2019211
- Aman Rubey - IIT2019212

Problem Statement

To Find the Minimum spanning tree by Prim’s Algorithm

Minimum Spanning tree

Spanning tree is subset of a graph. It has same number of vertices(v) as that of original graph and number of edges are (v-1).

There can more than 1 Minimum spanning trees for a single graph and those MST would have same cost.

Minimum cost spanning tree is a spanning tree (of a weighted graph) whose cost is minimum where cost($vertex_1$,$vertex_2$) is the weight if there exists an edge between $vertex_1$ and $vertex_2$

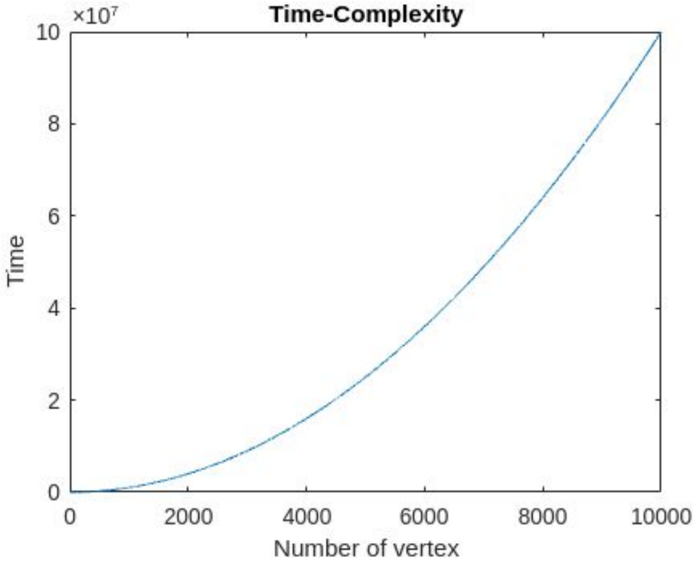
Prim’s Algorithm

1. Create a flag array which has size as number of vertices which will be used to track that which vertex is taken for MST and which not.
2. Create another vector namely, ‘min_weight’ of same size where we will store the weight of the minimum weighted edge its connected to, such that the other endpoint of the edge is already taken in flag array.
3. Start from the first vertex so include it in flag and also min_weight for it will 0.
4. Now, iterate for (vertices-1) times and in every iteration follow next steps:
   a. Get the the minimum weighted index or (u).
   b. A minimum weighted index is that index for which the corresponding vertex is not taken and also it has minimum key in the min_weight vector.
   c. Include u to flag array.
   d. For every v in V update the min_weight , where V is set of all vertices for which graph(u, v)>0.
   e. To update min_weight for ‘v’: if graph(u,v)<min_weight[v] then min_weight[v]=graph(u,v)

Note: graph is a vector which is used to store the original graph in the form of adjacency matrix.

Time Complexity

The time complexity of this problem is O(V^2) where V is number of vertex in the graph.

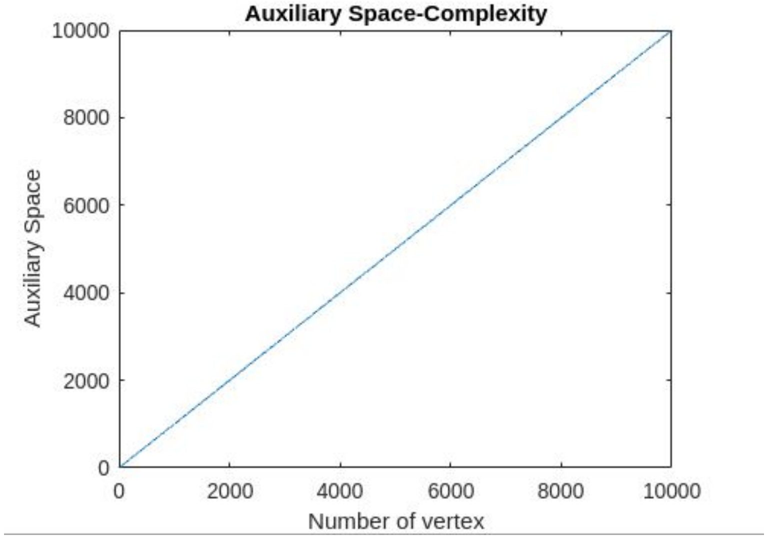


The outer for loop of the Prim_Min_spanning_Tree function is running V times and the getMinWeight function is also running V times. Similarly the Inner for loop in Prim_Min_spanning_Tree function is also running V times. Therefore,

$T(V) = O(V^2)$

Auxiliary Space Complexity

The Auxiliary space complexity is O(V).



In the code we have created only three vectors of size V(number of Vertices) mainly flag, min_weight and min_spanning_tree. Therefore, Auxiliary Space complexity will be,

S(V) = O(V)

TEST CASE

INPUT

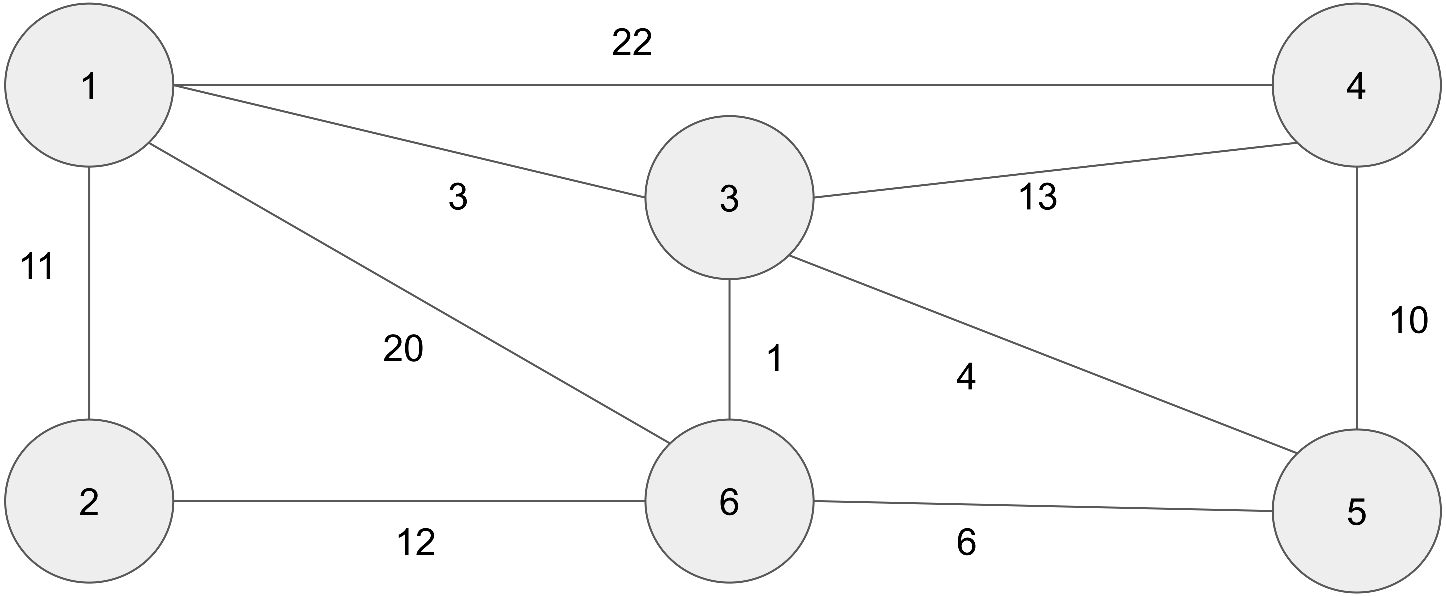
```
Enter Vertex: 6
Enter edges count: 10
1 2 11

1 6 20

1 3 3

1 4 22

2 6 12

3 6 1

3 4 13

3 5 4

4 5 10

5 6 6
```

OUTPUT

```
MST:

1 2 11

1 3 3

5 4 10

3 5 4

3 6 1
```

Example

Consider a Graph shown below, now we have to find the MST using Prim Algorithm



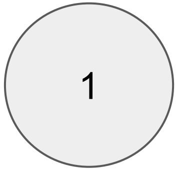
Initially there will be no edge in the MST

Flag Table

1	2	3	4	5	6
F	F	F	F	F	F

min_weight Table

1	2	3	4	5	6
0	∞	∞	∞	∞	∞

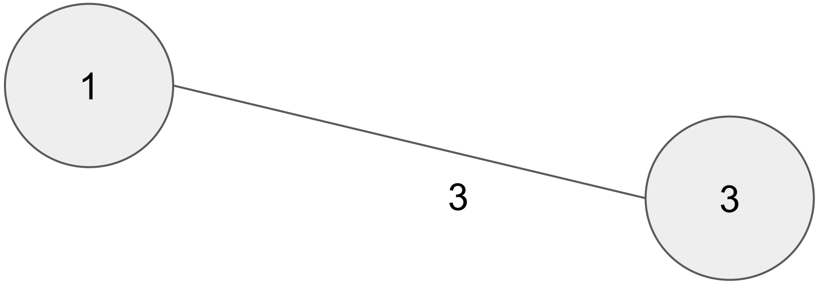


Flag Table

1	2	3	4	5	6
T	F	F	F	F	F

min_weight Table

1	2	3	4	5	6
0	11	3	22	∞	20

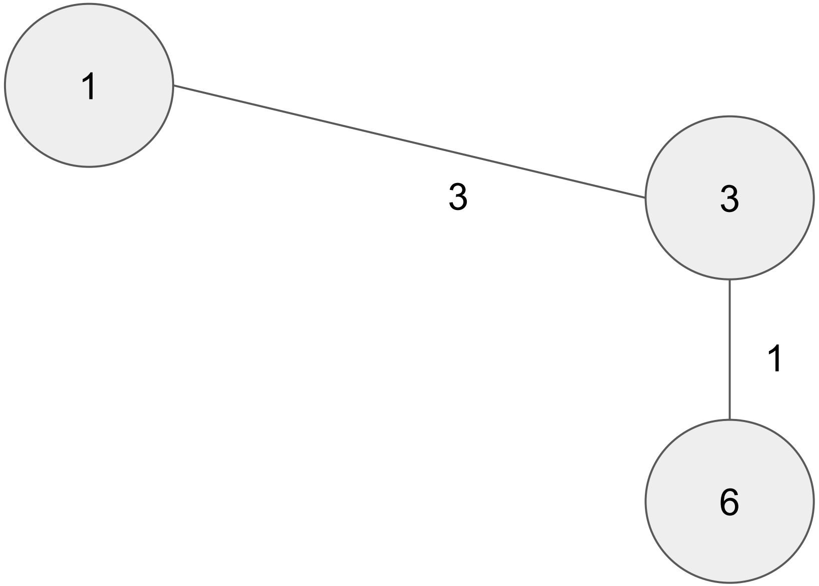


Flag Table

1	2	3	4	5	6
T	F	T	F	F	F

min_weight Table

1	2	3	4	5	6
0	11	3	13	4	1

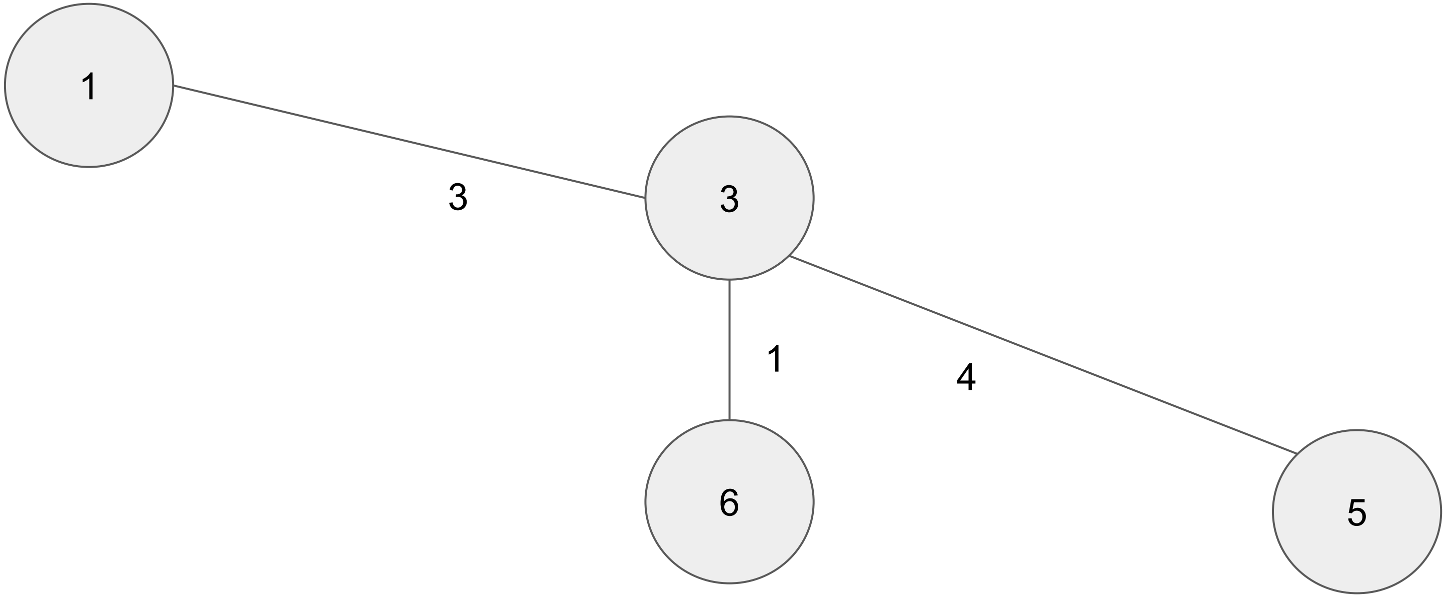


Flag Table

1	2	3	4	5	6
T	F	T	F	F	T

min_weight Table

1	2	3	4	5	6
0	11	3	13	4	1

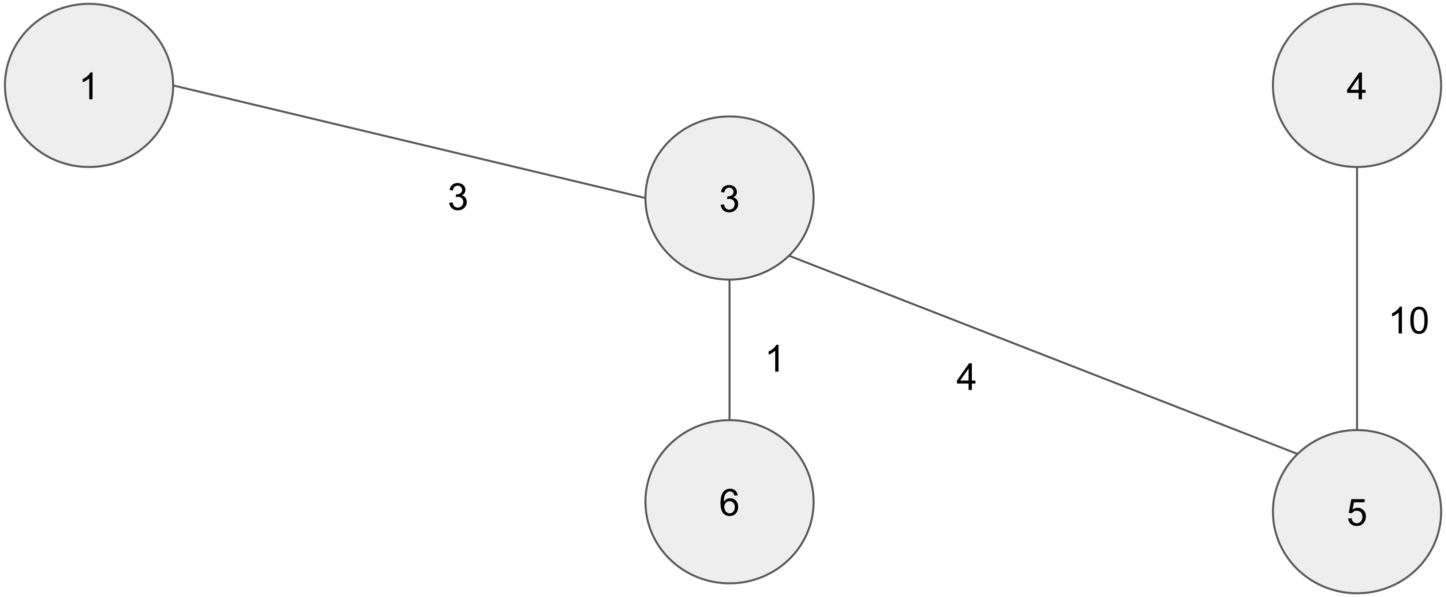


Flag Table

1	2	3	4	5	6
T	F	T	F	T	T

min_weight Table

1	2	3	4	5	6
0	11	3	10	4	1

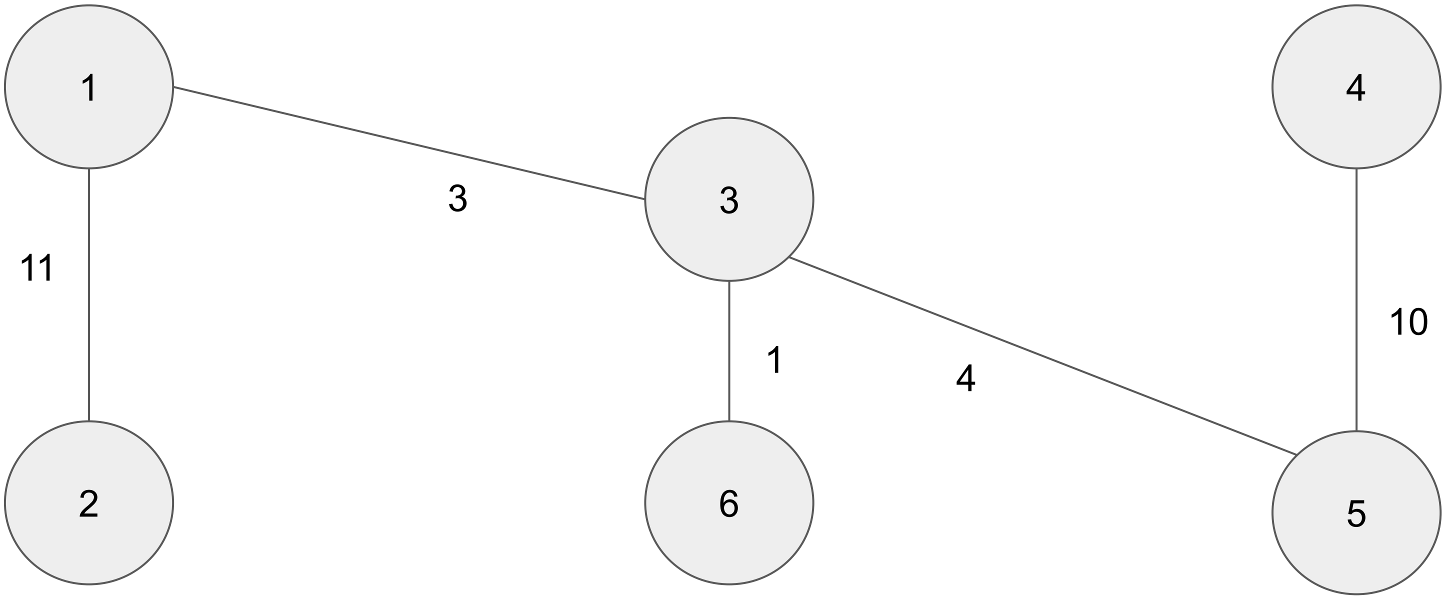


Flag Table

1	2	3	4	5	6
T	F	T	T	T	T

min_weight Table

1	2	3	4	5	6
0	11	3	10	4	1



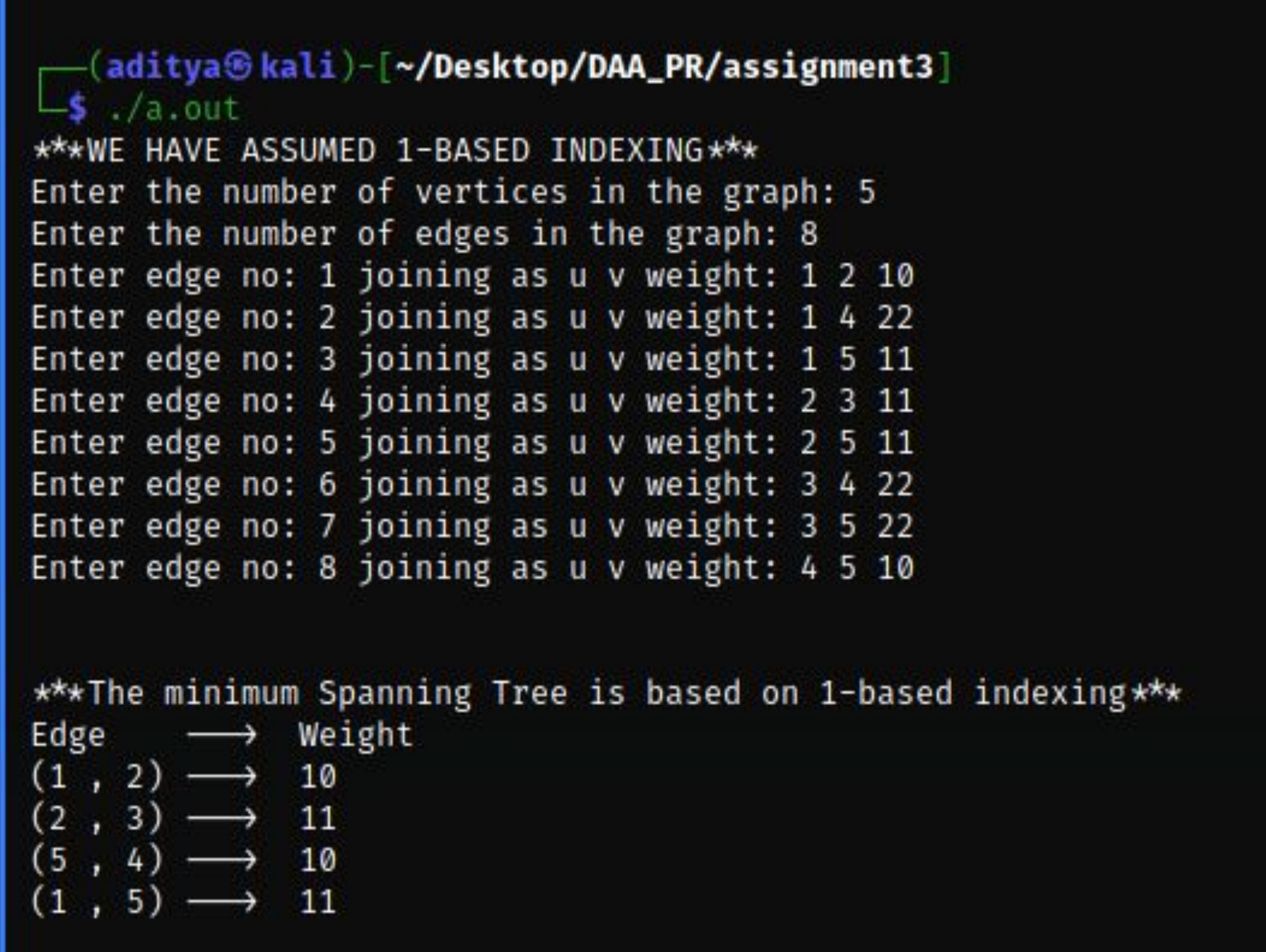
Flag Table

1	2	3	4	5	6
T	T	T	T	T	T

min_weight Table

1	2	3	4	5	6
0	11	3	10	4	1

SCREENSHOT



```
┌──(aditya㉿kali)-[~/Desktop/DAA_PR/assignment3]
└─$ ./a.out
***WE HAVE ASSUMED 1-BASED INDEXING***
Enter the number of vertices in the graph: 5
Enter the number of edges in the graph: 8
Enter edge no: 1 joining as u v weight: 1 2 10
Enter edge no: 2 joining as u v weight: 1 4 22
Enter edge no: 3 joining as u v weight: 1 5 11
Enter edge no: 4 joining as u v weight: 2 3 11
Enter edge no: 5 joining as u v weight: 2 5 11
Enter edge no: 6 joining as u v weight: 3 4 22
Enter edge no: 7 joining as u v weight: 3 5 22
Enter edge no: 8 joining as u v weight: 4 5 10


***The minimum Spanning Tree is based on 1-based indexing***
Edge    ⟶  Weight
(1 , 2) ⟶  10
(2 , 3) ⟶  11
(5 , 4) ⟶  10
(1 , 5) ⟶  11
```

THANKS