Subject: DAA LAB		Software: Ubuntu	
		Hardware: Corce i5	
Branch: CSE	Semester: 4th	Page No. 26	Prog No. 7

PROBLEM STATEMENT: Write a Program to find minimum Cost Spanning Tree of a given connected undirected graph Using Kruskal's Algorithm.

ALGORITHM & CODE: Use Union-Find Algorithm in your Program.

```
#include <stdio.h>
#define MAX_VERTICES 100
#define MAX_EDGES 4950 //maximum edges for 100 vertices
                        (n*(n-1)/2)

// Function to find the Parent of a vertex
int findParent(int Parent[], int vertex)
{ if (Parent[vertex] != vertex)
    Parent[vertex] = findParent(Parent, Parent[vertex]);
  return Parent[vertex];
}
// Function to perform union of two sets
void unionSets(int Parent[], int rank[], int u, int v)
{ int rootU = findParent(Parent, u);
  int rootV = findParent(Parent, v);
  if (rank[rootU] > rank[rootV])
  {Parent[rootV] = rootU;
  }
  elseif (rank[rootU] < rank[rootV])
  {Parent[rootU] = rootV;
  }
```

INPUT GIVEN	
OUTPUT OBTAINED	
REMARKS	
GRADE:	Signature of Faculty Date:
	Signature of Student: Rudranarayan Sahoo Date: 27.03.25

Subject: DAA LAB	Software: Ubuntu
	Hardware: Core i5
Branch: CSE / Semester: 4th	Page No. 27 / Prog No. 07

PROBLEM STATEMENT

ALGORITHM & CODE:

```
else
 {
  Parent[rootV] = rootU;
  rank[rootU]++;
 }
}

// function to impliment kruskal's algorithm
void kruskal(int edges[][3], int vertices, int edgesCount)
{ int Parent[MAX_VERTICES];
  int rank[MAX_VERTICES] = {0};
// initialize Parent arrray
for(int i = 0; i < vertices; i++)
{ parent[i] = i;
}
// using bubble Sort
for(int i = 0; i < edgesCount - 1; i++)
{for(int j=0; j< edgesCount - i - 1; j++)
  {if(edges[j][2] > edges[j+1][2])
   { // Swap edges
     int temp[3];
     temp[0] = edges[j][0];
```

INPUT GIVEN	
OUTPUT OBTAINED	
REMARKS	
GRADE:	Signature of Faculty Date:
	Signature of Student: Rudranarayan Sahoo Date: 27/03/2025

Subject: DAA LAB	Software: Ubuntu
	Hardware: Corei5
Branch: CSE / Semester: Corei5	Page No. 28 / Prog No. 07

PROBLEM STATEMENT

ALGORITHM & CODE:

```
        temp[1] = edges[j][1];
        temp[2] = edges[j][2];
        edges[j][0] = edges[j+1][0];
        edges[j][1] = edges[j+1][1];
        edges[j][2] = edges[j+1][2];

        edges[j+1][0] = temp[0];
        edges[j+1][1] = temp[1];
        edges[j+1][2] = temp[2];
      }
    }
  }
  Printf(" Edges in the minimum Spanning Tree: \n");
  Printf ("Edge \t weight \n");
  int mstEdges = 0;
  int mstWeight = 0;
  for (int i=0; i<edgesCount && mstEdges < vertices-1;
                                    i++)
  { int u = edges[i][0];
    int v = edges[i][1];
    int weight = edges[i][2];
  int root u = findParent (Parent, u);
  int root v = findParent (Parent, v);
```

INPUT GIVEN	
OUTPUT OBTAINED	
REMARKS	
GRADE:	Signature of Faculty Date:
	Signature of Student: Rudranarayan Sahoo Date: 27/03/2025

Subject: DAA LAB

Software: Ubuntu

Hardware: Core i5

Branch: CSE | Semester: 4th | Page No. 29 | Prog No. 07

LEM STATEMENT

ORITHM & CODE:

```
    if (rootU != rootV)
{
    Printf("%d - %d \t %d \n", u, v, weight);
    mstWeight = mstWeight + weight;
    UnionSets (Parent, rank, rootU, rootV);
    mstEdges++;
   }
}
Printf("Total weight of MST : %d \n", mstWeight);
}

int main()
  {
   int vertices, edges;
   int edgeList [MAX_EDGES][3]; // Each edge has u, v, weight
 : Printf("Total weight of MST : %d \n", mstWeight);
 }
int main()
{ int vertices, edges;
   int edgeList [MAX_EDGES][3]; // Each edge has u, v,
                                   weight
   Printf ("Enter number of vertices (max %d):",
                                   MAX_VERTICES);
   Scanf ("%d", &~~edges~~ vertices);
   Printf(" Enter number of edges (max %d):",
                                   MAX_EDGES);
   Scanf ("%d", &edges);
```

INPUT GIVEN

OUTPUT OBTAINED

REMARKS

GRADE:

Signature of Faculty

Date:

Signature of Student: Rudranarayan Sahoo

Date: 27/03/2025

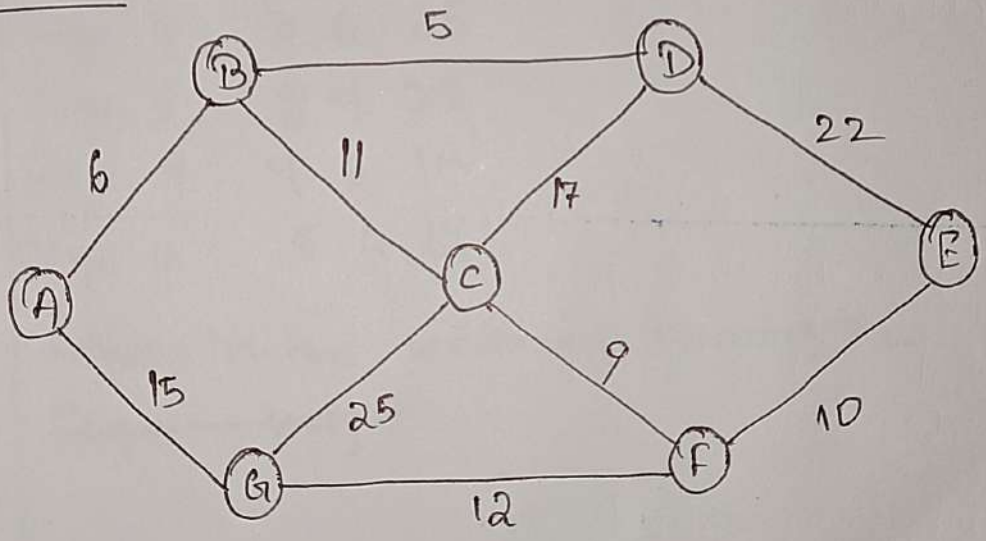
Subject : DAA LAB	Software : Ubuntu
	Hardware : Core i5
Branch : CSE / Semester : 4th	Page No. 30 / Prog No. 07

PROBLEM STATEMENT

ALGORITHM & CODE :

```
printf("Enter edges (u v weight) one by one: \n");
for (int i = 0; i < edges; i++)
{
    printf("Edge %d: ", i+1);
    scanf("%d %d %d", &edgelist[i][0],
&edgelist[i][1], &edgelist[i][2]);
}
kruskal(edgelist, vertices, edges);
return 0;
}
```

INPUT



INPUT GIVEN

OUTPUT OBTAINED

REMARKS

GRADE :

Signature of Faculty
Date :

Signature of Student Rudranarayan Sahoo
Date : 27/03/25

Subject :	Software :
	Hardware :
Branch : / Semester :	Page No. 31 / Prog No. 07

PROBLEM STATEMENT

ALGORITHM & CODE :

OUTPUT

```
Enter number of vertices (max 100): 7
Enter number of edges (max 4950): 10
Enter edges (u v weight) one by one:
Edge 1: 0 1 6
Edge 2: 0 6 15
Edge 3: 1 3 5
Edge 4: 1 2 11
Edge 5: 2 3 17
Edge 6: 2 5 9
Edge 7: 2 6 25
Edge 8: 3 4 22
Edge 9: 4 5 10
Edge 10: 5 6 12

Edges in the minimum Spanning Tree:
Edge  weight
```

INPUT GIVEN	
OUTPUT OBTAINED	
REMARKS	
GRADE :	Signature of Faculty Date : / Signature of Student Date :

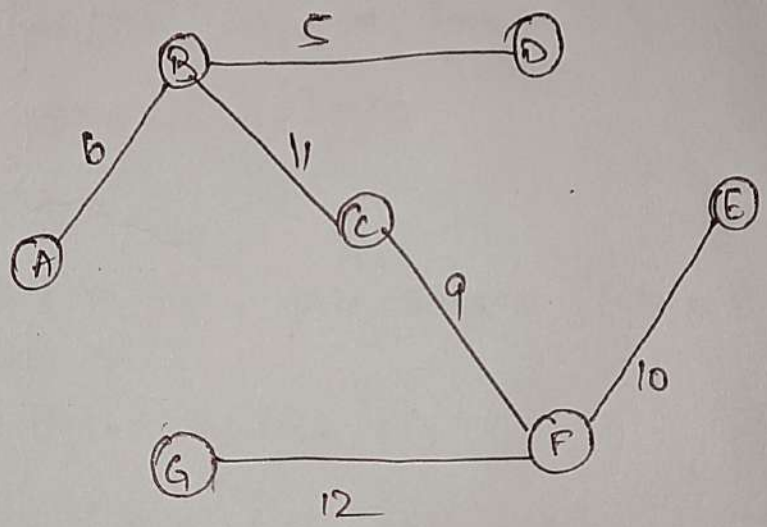
Subject :	Software :
	Hardware :
Branch : / Semester :	Page No. 32 / Prog No. 07

PROBLEM STATEMENT

ALGORITHM & CODE :

Edge	Weight
1-3	5
0-1	6
2-5	9
4-5	10
1-2	11
5-6	12

Total weight of MST : 53



INPUT GIVEN	
OUTPUT OBTAINED	
REMARKS	
GRADE :	Signature of Faculty Date : / Signature of Student Date :