

Kruskal's algorithm

:-1BM21CS232

```
#include <stdio.h>
#include <stdbool.h>
```

```
int n, m,
parent[100]; int
count = 0; int
ET[100][2]; int
cost[100][100];
int sum = 0;
```

```
void unionn(int a, int b)
{   if (a < b)
    parent[b] = a;
    else
    parent[a] = b;
}
```

```
int find(int a)
{   while (parent[a]
    != a)
    {       a =
parent[a];
```

```

    }
return a;
}

void kruskal()
{
    int k = 0;

    for (int i = 1; i <= n; i++)
    {
parent[i] = i;
    }

    while (count != n - 1)
    {
        int min
= 999;        int
u, v;

        for (int i = 1; i <= n; i++)
        {
            for (int j = 1;
j <= n; j++)
            {
                if (cost[i][j] < min &&
cost[i][j] != 0)

```

```

        {
min = cost[i][j];
u = i;          v = j;

        }

    }

}

    int x = find(u);

    int y = find(v);

    if (x !=

y)

    {
        ET[k][0] = u;

ET[k][1] = v;

k++;          count++;

sum += cost[u][v];

unionn(x, y);

    }

    cost[u][v] = cost[v][u] = 999;

    }

}

```

```

int main() {    printf("\n    Kruskal's
algorithm\n");    printf("    -----

```

```

-----");    int u, v, w;

printf("\nEnter the number of
vertices: ");    scanf("%d", &n);

    for (int i = 1; i <= n; i++)
    {        for (int j = 1; j
<= n; j++)
        {            if (i ==
j)                cost[i][j]
= 0;            else
cost[i][j] = 999;
        }
    }

    printf("Enter the number of edges: ");
    scanf("%d", &m);

    printf("Enter the egde with its weight:
\n");    for (int i = 1; i <= m; i++)
    {
        scanf("%d%d%d", &u, &v, &w);
cost[u][v] = cost[v][u] = w;
    }

    kruskal();

```

```
printf("\nMinimum cost = %d\n",
sum);
```

```
printf("Minimum spanning
tree:\n");  for (int i = 1; i < count;
i++)
{
    printf("%d -> %d\n", ET[i][0],
ET[i][1]);
}
return 0;
}
```

OUTPUT:

```
Kruskal's algorithm
-----
Enter the number of vertices: 7
Enter the number of edges: 9
Enter the egde with its weight:
1 2 28
1 6 10
2 7 14
2 3 16
3 4 12
4 7 18
4 5 22
5 7 24
5 6 25

Minimum cost = 99
Minimum spanning tree:
3 -> 4
2 -> 7
2 -> 3
4 -> 5
5 -> 6
```