KRUSKAL ALGORITM

```
#include<bits/stdc++.h>

#define MAXN 100

Using namespace std;

struct edge {

   int u, v, w;

   bool operator<(const edge& p) const

   {

      return w < p.w;

   }

};

int pr[MAXN];

vector<edge> e;

int find(int r)

{

   return (pr[r] == r) ? r : find(pr[r]);

}

int mst(int n)

{

   sort(e.begin(), e.end());

   for (int i = 1; i <= n; i++)

      pr[i] = i;


   int count = 0, s = 0;

   for (int i = 0; i < (int)e.size(); i++) {

      int u = find(e[i].u);
```

```
        int v = find(e[i].v);
        if (u != v) {
            pr[u] = v;
            count++;
            s += e[i].w;
            if (count == n - 1)
                break;
        }
    }
    return s;
}

int main()
{
    // READ("in");
    int n, m;
    cin >> n >> m;
    for (int i = 1; i <= m; i++) {
        int u, v, w;
        cin >> u >> v >> w;
        edge get;
        get.u = u;
        get.v = v;
        get.w = w;
        e.push_back(get);
    }
    cout << mst(n) << endl;
```

```
    return 0;
}
```

INPUT:

```
Nodes=10
Edges=14
0   1  10
0   2  6
2  3   5
 0  3  5
1  3   15
1   4  2
4  6   9
3  6  8
 4  5  24
5  7  6
6  7  10
2  8   4
3  9   6
 8   9  4
 Minimum cost is= 53


Process returned 0 (0x0)   execution time : 7.269 s
```

Press any key to continue.

DIJSKTRA ALGORITM

```
#include <bits/stdc++.h>


using namespace std;
int findMinVertex(int* distance,bool*visited,int n)
{
   int minVertex=-1;
   for(int i=0; i<n; i++)
   {
      if(!visited[i]&&(minVertex ==-1 || distance[i]<distance[minVertex]))
      {
         minVertex=i;


      }
   }
   return minVertex;


}
```

```
void dijskrta(int** edges,int n)
{
    int* distance =new int[n];
    bool* visited=new bool [n];
    for(int i=0; i<n; i++)
    {
        distance[i]=INT_MAX;
        visited[i]=false;
    }
    distance[0]=0;
    for(int i=0; i<n-1; i++)
    {
        int minVertex=findMinVertex(distance,visited,n);
        visited[minVertex]=true;
        for (int j=0; j<n; j++)
        {
            if(edges[minVertex][j]!=0 && !visited [j])
            {
                int dist=distance[minVertex]+edges[minVertex][j];
                if(dist<distance[j])
                {
                    distance[j]=dist;
                }
            }
        }
    }
```

```
    for(int i=0; i<n; i++)
    {
        cout<<i<<" "<<distance[i]<<endl;
    }
    delete[] visited;
    delete [] distance;
}

int main()
{
    int n,e;
    cin>>n>>e;
    int ** edges= new int*[n];
    for(int i=0; i<n; i++)
    {
        edges[i]=new int[n];
        for(int j=0; j<n; j++)
        {
            edges[i][j]=0;
        }

    }
    for(int i=0; i<e; i++)
    {
        int f,s,w;
        cin>>f>>s>>w;
        edges[f][s] =w;
```

```
        edges[s][f]=w;
    }
    cout<<endl;
    dijskrta(edges,n);
    for(int i=0; i<n; i++)
    {
        delete [] edges[i];
    }
    delete [] edges;
    return 0;
}
```

INPUT:

Nodes= 10

Edges= 14

0 1 10

0 2 6

2 3 5

0 3 5

1 3 15

1 4 2

4 6 9

3 6 8

4 5 24

5 7 6

6 7 10

2 8 4

3 9 6

8 9 4

OUTPUT:

vertex weight

0 0

1 10

2 6

3 5

4 12

5 29

6 13

7 23

8 10

9 11

Process returned 0 (0x0) execution time : 59.878 s

Press any key to continue.