```cpp
//日期: 2018/ 时间:
#include <stdio.h>
#include <stdlib.h>
#include <algorithm>
using namespace std;

const int maxe = 1005;
const int INF = 0x7fffffff;
struct edge{
    int u,v;        //边的两个端点
    int cost;       //边权
}E[maxe];       //最多有maxe条边

bool cmp(edge a,edge b){
    return a.cost < b.cost;
}

const int N = 100005;
int father[N];
int findFather(int x){
    int a = x;
    while(x!=father[x]){
        x=father[x];
    }

    while(a!=father[a]){
        int z=a;
        a = father[a];
        father[z] = x;
    }
    return x;
}

//返回最小生成树的边权之和，参数n为顶点个数，m为图的边数
int kruskal(int n,int m){
    //ans为所求边权之和，Num_edge为当前生成树的边数
    int ans = 0,Num_edge = 0;
    for(int i=0;i<n;i++){
        father[i] = i;
    }

    sort(E,E+m,cmp);

    for(int i=0;i<m;i++){
        int faU = findFather(E[i].u);   //查询测试边两个端点所在集合的根节点
        int faV = findFather(E[i].v);

        if(faU != faV){             //如果不在一个集合中
            father[faU] = faV;      //合并集合
            ans += E[i].cost;       //边权之和
            Num_edge++;
            if(Num_edge != n-1) break;  //边数等于顶点数减一时结束算法
        }
    }
    if(Num_edge != n-1) return -1;

```

```
57      return ans;
58  }
59
60  int main(){
61
62
63      return 0;
64  }
65
66
```