MST

● Graded

Student

SRUTHI SUBRAMANIAN

Total Points

60 / 60 pts

Autograder Score

60.0 / 60.0

Passed Tests

Test 1 (10/10)
Test 2 (10/10)
Test 3 (5/5)
Test 5 (5/5)
Test 7 (5/5)
Test 9 (5/5)

Autograder Results

Test 1 (10/10)

Test 2 (10/10)

Test 3 (5/5)

Test 5 (5/5)

Test 7 (5/5)

Test 9 (5/5)

Submitted Files

▾ hw.c　　　　Download

```c
#include <stdio.h>

int main() {
    int flag;
    scanf("%d\n",&flag);
    if (flag==0){
        //DSU Data Structure
        int n,q;
        int query,i,j;
        scanf("%d %d\n",&n,&q);
        int parent[n];
        for (int p=0;p<n;p++){
            parent[p]=-1;
        }
        int Find(int u){
            if (parent[u]==-1){
                return u;
            }
            int x=Find(parent[u]);
            parent[u]=x;
            return x;
        }
        int Merge(int u, int v){
            if (Find(u)==Find(v)){
                return 0;
            }
            int ru,rv;
            ru=Find(u);
            rv=Find(v);
            parent[ru]=rv;
            return 0;
        }
        for (int p=0;p<q;p++){
            scanf("%d %d %d\n",&query,&i,&j);
            i=i-1;
            j=j-1;
            if (query==0){
                //merge query
                Merge(i,j);
            }
            else if (query==1){
                //report query
                if (Find(i)==Find(j)){
                    printf("1 ");
                }
                else{
                    printf("0 ");
                }
            }
        }
    }
    else if (flag==1){
        //MST- Kruskal's algorithm, then sum weights of all the edges in the MST and return that.
        int n;
        scanf("%d\n",&n);
        int parent[n];
        for (int p=0;p<n;p++){
            parent[p]=-1;
        }
        int Find(int u){
            if (parent[u]==-1){
                return u;
            }
            int x=Find(parent[u]);
            parent[u]=x;
            return x;
        }
        int Merge(int u, int v){
            if (Find(u)==Find(v)){
                return 0;
            }
            int ru,rv;
            ru=Find(u);
            rv=Find(v);
            parent[ru]=rv;
            return 0;
        }
        int E[n*n];
        int I[n*n];
        int J[n*n];
        int a,b;
        int c=0;
        for (int i=0;i<n;i++){
            scanf("%d",&a);
            while (a!=-1){
                scanf("%d\n",&b);
                E[c]=b;
                I[c]=i;
                J[c]=a-1;
                scanf("%d",&a);
                c=c+1;
            }
        }
        //finally c is the number of edges
        //now we want to order the edges from smallest to largest(i.e. order E, maintaining the relative ordering of I and J as well)
        int tempE,tempI,tempJ;
        for (int j=1;j<c;j++){
            for (int i=j-1;i--;i>-1){
                if (E[i]>E[i+1]){
                    //swap both
                    tempE=E[i];
                    tempI=I[i];
                    tempJ=J[i];
                    E[i]=E[i+1];
                    I[i]=I[i+1];
                    J[i]=J[i+1];
                    E[i+1]=tempE;
                    I[i+1]=tempI;
                    J[i+1]=tempJ;
                }
            }
        }
        //KRUSKAL'S ALGORITHM
        int j=0;
        int ctr=0;
        int s=0;
        while(ctr!=n-1 || j<c){
            tempI=I[j];
            tempJ=J[j];
            tempE=E[j];
        if (Find(tempI)==Find(tempJ)){
            j++;
        }
        else{
            Merge(tempI,tempJ);
            ctr++;
            j++;
            s+=tempE;
        }
        }
      printf("%d ",s);
      }
      return 0;
    }
```