

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

```
1  #include <stdio.h>
2
3  int main() {
4      int flag;
5      scanf("%d\n",&flag);
6      if (flag==0){
7          //DSU Data Structure
8          int n,q;
9          int query,i,j;
10         scanf("%d %d\n",&n,&q);
11         int parent[n];
12         for (int p=0;p<n;p++){
13             parent[p]=-1;
14         }
15         int Find(int u){
16             if (parent[u]==-1){
17                 return u;
18             }
19             int x=Find(parent[u]);
20             parent[u]=x;
21             return x;
22         }
23         int Merge(int u, int v){
24             if (Find(u)==Find(v)){
25                 return 0;
26             }
27             int ru,rv;
28             ru=Find(u);
29             rv=Find(v);
30             parent[ru]=rv;
31             return 0;
32         }
33         for (int p=0;p<q;p++){
34             scanf("%d %d %d\n",&query,&i,&j);
35             i=i-1;
36             j=j-1;
37             if (query==0){
38                 //merge query
39                 Merge(i,j);
40             }
41             else if (query==1){
42                 //report query
43                 if (Find(i)==Find(j)){
44                     printf("1 ");
45                 }
46                 else{
47                     printf("0 ");
48                 }
49             }
```

```

50     }
51 }
52 else if (flag==1){
53     //MST- Kruskal's algorithm, then sum weights of all the edges in the MST and return that.
54     int n;
55     scanf("%d\n",&n);
56     int parent[n];
57     for (int p=0;p<n;p++){
58         parent[p]=-1;
59     }
60     int Find(int u){
61         if (parent[u]==-1){
62             return u;
63         }
64         int x=Find(parent[u]);
65         parent[u]=x;
66         return x;
67     }
68     int Merge(int u, int v){
69         if (Find(u)==Find(v)){
70             return 0;
71         }
72         int ru,rv;
73         ru=Find(u);
74         rv=Find(v);
75         parent[ru]=rv;
76         return 0;
77     }
78     int E[n*n];
79     int I[n*n];
80     int J[n*n];
81     int a,b;
82     int c=0;
83     for (int i=0;i<n;i++){
84         scanf("%d",&a);
85         while (a!=-1){
86             scanf("%d\n",&b);
87             E[c]=b;
88             I[c]=i;
89             J[c]=a-1;
90             scanf("%d",&a);
91             c=c+1;
92         }
93     }
94     //finally c is the number of edges
95     //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)
96     int tempE,tempI,tempJ;
97     for (int j=1;j<c;j++){
98         for (int i=j-1;i-->-1){
99             if (E[i]>E[i+1]){
100                 //swap both

```

```
101         tempE=E[i];
102         tempI=I[i];
103         tempJ=J[i];
104         E[i]=E[i+1];
105         I[i]=I[i+1];
106         J[i]=J[i+1];
107         E[i+1]=tempE;
108         I[i+1]=tempI;
109         J[i+1]=tempJ;
110     }
111 }
112 }
113 //KRUSKAL'S ALGORITHM
114 int j=0;
115 int ctr=0;
116 int s=0;
117 while(ctr!=n-1 || j<c){
118     tempI=I[j];
119     tempJ=J[j];
120     tempE=E[j];
121     if (Find(tempI)==Find(tempJ)){
122         j++;
123     }
124     else{
125         Merge(tempI,tempJ);
126         ctr++;
127         j++;
128         s+=tempE;
129     }
130 }
131 printf("%d ",s);
132 }
133 return 0;
134 }
```
