MST

● Graded

Student
ANSHUMAN SINGH

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>
#include <stdlib.h>

// Disjoint set union

typedef struct{
    int *parent;
    int *rank;
    int n;
} DJS;

DJS *dsu;

void create(int n){
    dsu =(DJS*)malloc(sizeof(DJS));
    dsu->parent =(int*)malloc((n+1)*sizeof(int));
    dsu->rank =(int*)malloc((n+1)*sizeof(int));
    dsu->n =n;
    for(int i=1;i<=n;i++){
        dsu->parent[i]=i;
        dsu->rank[i]=0;
    }
}

int find(int x){
    while(x!=dsu->parent[x])
        x=dsu->parent[x];
    return x;
}

void merge(int i,int j){
    if(find(i)==find(j))
    return;
    int hi=find(i);
    int hj=find(j);
    if(dsu->rank[hi]>dsu->rank[hj])
        dsu->parent[hj]=hi;
    else{
        dsu->parent[hi]=hj;
        if(dsu->rank[hi]==dsu->rank[hj])
            dsu->rank[hj]++;
    }
}

int report(int i,int j){
    if(find(i)==find(j))
    return 1;
    else
    return 0;
```

```
}

void freeSet(DJS *dsu){
    free(dsu->parent);
    free(dsu->rank);
    free(dsu);
}

// MST

typedef struct{
    int u, v, wt;
}Edge;

void swap(Edge *a,Edge *b){
    Edge temp = *a;
    *a = *b;
    *b = temp;
}

void heapify(Edge *a,int n,int i){
    int k=i;
    if(2*i+1<n&&a[2*i+1].wt>a[k].wt)
        k=2*i+1;
    if(2*i+2<n&&a[2*i+2].wt>a[k].wt)
        k=2*i+2;
    if(k!=i){
        swap(&a[i],&a[k]);
        heapify(a,n,k);
    }
}

void heapSort(Edge *a,int n){
    for(int i =n/2-1;i>=0;i--)
        heapify(a,n,i);
    for (int i =n-1;i>=0;i--){
        swap(&a[0],&a[i]);
        heapify(a, i, 0);
    }
}

int MST(Edge *edges, int tot_edges, int n) {
    create(n);
    int mst_wt=0;
    int ctr=0;

    for(int j=0;j<tot_edges&&ctr<n-1;j++){
        Edge edge=edges[j];
        if(find(edge.u)!=find(edge.v)){
            merge(edge.u,edge.v);
            mst_wt+=edge.wt;
            ctr++;
```

```
        }
    }
    freeSet(dsu);

    return mst_wt;
}



int main(){
    int flag;
    scanf("%d",&flag);
    if(flag==0){
        int n,q;
        scanf("%d %d",&n,&q);
        create(n);
        while(q-->0){
            int type,i,j;
            scanf("%d %d %d",&type,&i,&j);
            if(type==0){
                merge(i,j);
            }
            else{
                int ans=report(i,j);
                printf("%d ",ans);
            }
        }
        freeSet(dsu);
    }
    else{
        int n;
        scanf("%d",&n);
        Edge *edges=(Edge *)malloc(n*n*sizeof(Edge));
        int tot_edges=0;
        for(int i=1;i<=n;i++){
            int j,wt;
            while(1){
                scanf("%d",&j);
                if(j==-1)
                break;
                scanf("%d",&wt);
                int k=tot_edges;
                edges[k].u=i;
                edges[k].v=j;
                edges[k].wt=wt;
                tot_edges++;
            }
        }
        heapSort(edges,tot_edges);
        int mst_wt = MST(edges, tot_edges, n);
        printf("%d ", mst_wt);
    }
```

```
154     return 0;
155 }
```