Kruskal

```cpp
#include <iostream>
#include <algorithm>
#include <vector>

using namespace std;
#define maxn 100005
typedef struct EDGE{
    int from,to,dis;
    EDGE(){}
    EDGE(int u,int v,int w){
        from=u;to=v;dis=w;
    }
}Edge;
vector<Edge> edges;
vector<int> G[maxn];
int tot;
void init(int n){
    for(int i=0;i<=n;i++){
        G[i].clear();
    }
    edges.clear();
    tot = 0;//edges.size()
}

bool cmp(Edge e1,Edge e2){
    return e1.dis<e2.dis;
}

int fa[maxn];
int find(int x){return fa[x]==x?x:fa[x]=find(fa[x]);}

int main(){
    freopen("1.in","r",stdin);
    freopen("1.out","w",stdout);

    // int T;
    // cin>>T;
    int n;
    int m;
    while(cin>>n&&n){
        m = n*(n-1)/2;
        init(n);
        for(int i=0,u,v,w;i<m;i++){
            cin>>u>>v>>w;
            edges.push_back(Edge(u,v,w));
            // edges.push_back(Edge(v,u,w));
            // G[u].push_back(tot++);
            // G[v].push_back(tot++);
        }
        sort(edges.begin(),edges.end(),cmp);
        for(int i=0;i<=n;i++)fa[i]=i;
        int ans = 0;
        for(int i=0;i<m;i++){
            Edge &e = edges[i];
            int tofa = find(e.to);
            int fromfa = find(e.from);
            if(tofa==fromfa)continue;
            fa[tofa]=fromfa;
            ans += e.dis;
        }
        cout<<ans<<endl;
    }
    return 0;
}
```

Segment tree

```cpp
#include <iostream>
#include <algorithm>

using namespace std;
#define maxn 100005
#define lc (root<<1)
#define rc ((root<<1)+1)
typedef struct NODE{
    int l,r,g;
}Node;

Node tree[maxn<<2];
int a[maxn];
int gcd(int a,int b){if(!b)return a;return gcd(b,a%b);}
void build(int l,int r,int root){
    if(l==r){
        tree[root].l = tree[root].r = l;
        tree[root].g = a[l];
        return;
    }
    int mid = (l+r)/2;
    build(l,mid,lc);
    build(mid+1,r,rc);
    tree[root].l = l;
    tree[root].r = r;
    tree[root].g = gcd(tree[lc].g,tree[rc].g);
}

int query(int l,int r,int root){
    if(tree[root].l==l && tree[root].r==r){
        return tree[root].g;
    }
    int mid = (tree[root].l+tree[root].r)/2;
    if(r<=mid){
        return query(l,r,lc);
    }else if(l>mid){
        return query(l,r,rc);
    }
    return gcd(query(l,mid,lc),query(mid+1,r,rc ));
}
int main(){
    int n,m;
    cin>>n>>m;
    for(int i=1;i<=n;i++){
        cin>>a[i];
    }
    build(1,n,1);

    return 0;
}
```

Binary indexed tree

```cpp
#include <iostream>
#include <algorithm>
#include <vector>
#include <cstdio>
#include <cstring>
using namespace std;
#define maxn 100005
int e[maxn],n;
void init(){memset(e,0,sizeof(e));}
int lowbits(int x){return x&-x;}
void update(int x,int v){for(;x<=n;x+=lowbits(x)) e[x]+=v;}
int sum(int x){
    int rs = 0;
    for(;x>0;x-=lowbits(x))rs+=e[x];
    return rs;
}
```

```
int main(){
    freopen("1.in","r",stdin);
    freopen("1.out","w",stdout);
    while(cin>>n){
        init();
        for(int i=1,a;i<=n;i++){
            cin>>a;
            update(i,a);
        }
        for(int i=1;i<=n;i++){
            cout<<e[i]<<" ";
        }
        cout<<endl;
    }
    return 0;
}
```