```cpp
/// Kruskal( 1174-IP-TV):
#include<bits/stdc++.h>
using namespace std;
int par[2005],nn;
map<string,int>mp;
struct edge{ int u,v,w; };
vector<edge>vv;
bool cmp(edge x, edge y){
    return x.w<y.w;
}
int findparent(int x){
    if(par[x]==x) return x;
    return par[x]=findparent(par[x]);
}
void Unionparent(int x,int y){
    par[findparent(y)] = findparent(x);
}
int kruskalMST(int n){
    for(int i=1; i<=n; i++) par[i]=i;
    int sum=0,cnt=0;

    sort(vv.begin(),vv.end(),cmp);

    for(int i=0; i<vv.size(); i++){
        edge top = vv[i];
        int parU = findparent(top.u);
        int parV = findparent(top.v);

        if(parU!=parV){
            Unionparent(parU,parV);
            sum += top.w;
            cnt++;
            if(cnt==n-1)break;
        }
    }
    vv.clear();
    return sum;
}
int main(){
    int tt; scanf("%d",&tt);
    for(int ks=0; ks<tt; ks++){
        if(ks)printf("\n");
        int n; scanf("%d",&n);
        int m; scanf("%d",&m);

        int k=0; mp.clear();
        for(int i=1; i<=m; i++){
            string s1,s2; int w;
            cin >> s1 >> s2 >> w;

            if(mp[s1]==0)mp[s1]=++k;
            if(mp[s2]==0)mp[s2]=++k;

            int u = mp[s1]; int v = mp[s2];
            edge pp; pp.u=u, pp.v=v, pp.w=w;
            vv.push_back(pp);
        }

        int ans = kruskalMST(n);
        printf("%d\n",ans);
    }
    return 0;
}
```

```cpp
/// Kruskal Algorithm(MST):
#include<bits/stdc++.h>
using namespace std;
#define mx 100
int par[mx];
struct edge{
    int u,v,w;
    edge(int a, int b, int c){
        u=a;v=b;w=c;
    }
    bool operator < (const edge& p) const{
        return p.w < w;
    }
};
priority_queue<edge>pq;
queue<edge>Q;

int findparent(int x){
    if(par[x]==x) return x;
    else return par[x]=findparent(par[x]);
}

int kruskalMST(int node){
    int Mincost=0;
    for(int i=1; i<=node; i++) par[i]=i;

    for(int i=1; i<=node-1; i++){
        label:
        edge top = pq.top();
        pq.pop();

        int parU = findparent(top.u);
        int parV = findparent(top.v);

        if(parU==parV){
            goto label;
        }else{
            par[parU]=parV;
            Q.push(top);
            Mincost += top.w;
        }
    }
    return Mincost;
}

int main(){
    int node,edg,u,v,w,mincst,src;
    cin >> node >> edg;

    for(int i=0; i<edg; i++){
        cin >> u >> v >> w;
        pq.push(edge(u,v,w));
    }

    mincst=kruskalMST(node);

    cout <<endl<<"Minimum Spanning Tree:"<<endl<<endl;
    cout << " Edges " << "  Weight" << endl;

    for(int i=1; i<=node-1; i++){
        edge top=Q.front();
        Q.pop();
        printf("%2d  %2d   %3d\n\n",top.u,top.v,top.w);
    }
    cout <<endl<<"Total Minimum Cost: "<<mincst<<endl;

    return 0;
}
```