MST Variation

```
#include<bits/stdc++.h>
using namespace std;
#define mx 100005
long long  par[mx+5];
map<string,int>mp;
struct edge
{
   long long  u,v,w;
   edge(long long  a,long long  b,long long  c)
   {
      u = a,v = b,w = c;
   }
   bool operator <(const edge&p) const
   {
      return w>p.w;
   }
};
vector<edge>vec,vec2;
vector<long long> graph[mx + 5],cost[mx + 5];

long long  findd(long long  val)
{
   if(par[val] == val) return val;
   return par[val]=findd(par[val]);
}
void Union(long long a,long long b)
{
   par[findd(b)] = findd(a);
}
void mst(long long  sz)
{
   long long  s=0,nod = 0;
   sort(vec.begin(),vec.end());
```

```
    reverse(vec.begin(),vec.end());
    for(int i=0; i<=sz; i++)
        par[i] =i;
    for(long long  i=0; i<(long long )vec.size(); i++)
    {
        long long u = findd(vec[i].u);
        long long v =findd(vec[i].v);
        if(u!=v)
        {
            Union(u,v);
            graph[vec[i].u].push_back(vec[i].v);
            graph[vec[i].v].push_back(vec[i].u);
            cost[vec[i].u].push_back(vec[i].w);
            cost[vec[i].v].push_back(vec[i].w);
            nod++;
            s+=vec[i].w;
            if(nod == sz-1)
                break;
        }
    }
    return;
}
long long vis[mx + 5];
long long new_cst[mx + 5];
long long mxx;
long long bfs(long long scc)
{
    long long node_s;
    mxx = 0 ;
    queue<long long >Q;
    vis[scc] = 1;
    Q.push(scc);
     new_cst[scc] = 0;
    while(!Q.empty())
```

```
    {
        long long store = Q.front();
        Q.pop();
        for(int i=0; i<graph[store].size(); i++)
        {
            long long haha = graph[store][i];
            if(vis[haha] == -1)
            {
                vis[haha] = 1;
                new_cst[haha] = new_cst[store] + cost[store][i];
                Q.push(haha);
            //cout<<"cost of node  = "<<new_cst[haha]<<endl;
                if(new_cst[haha]>=mxx)
                {
                    mxx  = max(mxx,new_cst[haha]);
                    node_s = haha;
                }
            }
        }
    }
    return node_s;
}
int main()
{
    long long t,w = 0;
    scanf("%lld",&t);
    while(t--)
    {
        long long  n,m,costt;
        string frm,to;
        scanf("%lld %lld",&n,&m);
        long long cntt = 0;
        long long cnt = 1;
        while(m--)
```

```
{
   cin>>frm>>to;
   if(mp[frm] == 0) mp[frm] = cnt++;
   if(mp[to] == 0) mp[to] = cnt++;
   scanf("%lld",&costt);
   edge k(mp[frm],mp[to],costt);
   vec.push_back(k) }
mst(n);
memset(vis,-1,sizeof(vis));
long long kkk =  bfs(1);
 memset(vis,-1,sizeof(vis));
long long kkkk =  bfs(kkk);
printf("Case %lld: %lld\n",++w,mxx);
vec.clear();
vec2.clear();
mp.clear();
for(int i=0;i<=mx;i++)
{
   graph[i].clear();
   cost[i].clear();
}
```