//DSU

```
#include<bits/stdc++.h>
#define endll "\n"
#define ll long long
using namespace std;
const ll mx=1e5+123;
vector<ll>parent(mx),sz(mx);
ll Find(ll u)
{
   if(parent[u] == u)
   return u;
   return parent[u]=Find(parent[u]);
}
void Union(ll u,ll v)
{
   ll a=Find(u);
   ll b=Find(v);
   if(a!=b)
   {
      if(sz[a]<sz[b])
      swap(a,b);
      parent[b]=a;
      sz[a]+=sz[b];
   }
}
int main()
{
ios_base::sync_with_stdio(false);cin.tie(NULL)
;cout.tie(NULL);
   ll i,j,n,u,m,v,k,maxi=-99999999999;
   cin>>n>>k;
   for(i=1;i<=n;i++)
   {
      parent[i]=i;
      sz[i]=1;
   }
   while(k--) {
      cin>>u>>v;
      if(Find(u)!=Find(v))
      Union(u,v);
   }
   cin>>m;
   while(m--){
      cin>>u>>v;
      if(Find(u)==Find(v))
      {
         sz[Find(u)]=0;
      }
   }
   for(i=1;i<=n;i++){
      if(parent[i]==i)
      maxi=max(maxi,sz[i]);
   }
   cout<<maxi<<endll;
}
```

// DIJKSTRA

```
#include<bits/stdc++.h>
#define endl "\n"
#define ll long long
using namespace std;
const ll mx=1e5+123;
const ll infLL = 9000000000000000000;
```

```
vector<pair<ll,ll>>adj[mx];
ll vis[mx];
ll dist[mx];
void dijkstra(ll s,ll n)
{
    for(ll i=0;i<=n;i++) dist[i]=infLL;
    dist[s]=0;

priority_queue<pair<ll,ll>,vector<pair<ll,ll>
>,greater<pair<ll,ll>>>pq;
    pq.push({0,s});
    while(!pq.empty())
    {
        ll u=pq.top().second;
        ll curr_dist=pq.top().first;
        pq.pop();
        if(dist[u]<curr_dist) continue;
        for(auto it:adj[u])
        {
            if(dist[it.first]>curr_dist+it.second)
            {
                dist[it.first]=curr_dist+it.second;
                pq.push({dist[it.first],it.first});
                vis[it.first]=u;
            }
        }
    }
}
int main()
{
ios_base::sync_with_stdio(false);cin.tie(N
ULL);cout.tie(NULL);
    ll n,m,u,v,w,i,j,k;
    cin>>n>>m;
    while(m--) {
        cin>>u>>v>>w;
        adj[u].push_back({v,w});
        adj[v].push_back({u,w});
    }
    dijkstra(1,n);
    if(dist[n]==9000000000000000000)
    return cout<<-1<<endl,0;
    vector<ll>par={n};
    while(vis[n]!=0){
        par.push_back(vis[n]);
        n=vis[n];
    }
    reverse(par.begin(),par.end());
    for(auto it:par)
    cout<<it<<" ";
    cout<<endl;
}
```

// some logic of bitmask

x^(1<<k) inverts the kth bit of x

x&~(1<<k) sets the kth bit of x to 0

x|(1<<k) sets the kth bit of x to 1

x&(x-1) sets the last one bit of x to 0

x&-x sets all bits except the last one bit equal to 0

x|(x-1) inverts all bit after the last set bit

x>>k means that x is devided by 2^k

x<<k means that x is multiplied by 2^k

~x means –x-1

x is divisible by 2^k if and only if x&(2^(k) - 1)==0

x is a power of two if and only if x&(x-1)==0