

线段树分治

算法简介

一种将修改和询问操作转移到线段树上，最后通过遍历线段树求解的离线算法。

算法例题

[洛谷p5787](#)

题意

初始给定 n 个点，图上没有边。现有 m 条边，每条边出现的时间为 $[l_i, r_i]$ 询问每个时刻图是否为二分图。

题解

考虑对时间序列建线段树，然后将每条边插入线段树。

遍历线段树，同时使用拓展域并查集动态维护此时是否为二分图。如果当前区间已经不是二分图或者遍历到叶子节点即可得到答案并回溯。

由于修改操作需要支持回溯，所有考虑使用按秩合并的可撤销并查集。

修改次数为 $O(m \log n)$ 于是总时间复杂度 $O(m \log^2 n)$

```
const int MAXN=1e5+5;
vector<pair<int,int> >a[MAXN<<2];
int lef[MAXN<<2],rig[MAXN<<2];
void build(int k,int L,int R){
    lef[k]=L,rig[k]=R;
    if(L==R) return;
    int M=L+R>>1;
    build(k<<1,L,M);build(k<<1|1,M+1,R);
}
void add(int k,int L,int R,pair<int,int> edge){
    if(L<=lef[k]&&rig[k]<=R)
        return a[k].push_back(edge),void();
    int mid=lef[k]+rig[k]>>1;
    if(mid>=L)add(k<<1,L,R,edge);
    if(mid<R)add(k<<1|1,L,R,edge);
}
int n,fa[MAXN<<1],dep[MAXN<<1],top;
pair<int,bool> Stack[MAXN<<2];
int Find(int x){return x==fa[x]?x:Find(fa[x]);}
```

```
void Union(int u,int v){
    int x=Find(u),y=Find(v);
    if(x==y) return;
    if(dep[x]>dep[y]) swap(x,y);
    Stack[++top]=make_pair(x,dep[x]==dep[y]);
    fa[x]=y,dep[y]+=dep[x]==dep[y];
}
void solve(int k){
    int cur=top;bool flag=true;
    _for(i,0,a[k].size()){
        pair<int,int> temp=a[k][i];
        if(Find(temp.first)==Find(temp.second)){
            _rep(i,lef[k],rig[k]) puts("No");
            flag=false;
            break;
        }
        Union(temp.first,temp.second+n),Union(temp.first+n,temp.second);
    }
    if(flag){
        if(lef[k]==rig[k]) puts("Yes");
        else solve(k<<1),solve(k<<1|1);
    }
    while(top>cur){
        dep[fa[Stack[top].first]]-=Stack[top].second;
        fa[Stack[top].first]=Stack[top].first;
        top--;
    }
}
int main()
{
    int n=read_int(),m=read_int(),k=read_int();
    ::n=n;
    build(1,1,k);
    _rep(i,1,n<<1) fa[i]=i,dep[i]=1;
    while(m--){
        int u=read_int(),v=read_int(),l=read_int(),r=read_int();
        if(l!=r) add(1,l+1,r,make_pair(u,v));
    }
    solve(1);
    return 0;
}
```

算法练习

[牛客暑期多校\(第八场\) A 题](#)

题意

现有 n 个球员 m 个球迷。每个球员有 k_i 个粉丝。

球迷 i 喜欢球员 j 当且仅当以下两个条件至少有一个满足：

1. 球迷 i 是球员 j 的粉丝
2. 球迷 i 和 i' 都喜欢球员 j 且球迷 i' 喜欢球员 j

接下来 q 次修改操作，每次修改球迷 x 与球员 y 之间粉丝关系。

每次修改后询问至少需要选择多少个球员才能保证每个球迷至少有一个喜欢的球员。如果没有满足条件的方案，输出 -1 。

题解

把球员和球迷当成点，粉丝关系视为边，于是题目答案等价于连通块个数减去孤立球员个数。

特殊情况为如果存在孤立球迷，此时答案为 -1 。

考虑对时间序列建线段树，然后可撤销并查集维护连通块个数、孤立球员个数、孤立球迷个数即可。

时间复杂度 $O\left((q + \sum_{i=1}^n k_i) \log q \log n\right)$

```

const int MAXN=2e5+5,MAXK=7e5+5;
vector<pair<int,int> >a[MAXN<<2];
int lef[MAXN<<2],rig[MAXN<<2];
void build(int k,int L,int R){
    lef[k]=L,rig[k]=R;
    if(L==R)return;
    int M=L+R>>1;
    build(k<<1,L,M);build(k<<1|1,M+1,R);
}
void add(int k,int L,int R,pair<int,int> edge){
    if(L<=lef[k]&&rig[k]<=R)
        return a[k].push_back(edge),void();
    int mid=lef[k]+rig[k]>>1;
    if(mid>=L)add(k<<1,L,R,edge);
    if(mid<R)add(k<<1|1,L,R,edge);
}
int m,fa[MAXN<<1],sz[MAXN<<1],top,Stack[MAXK],block_cnt,Iosn,Iosm;
int Find(int x){return x==fa[x]?x:Find(fa[x]);}
void Union(int u,int v){
    int x=Find(u),y=Find(v);
    if(x==y)return;
    block_cnt--;
    if(sz[x]==1){
        if(x>m)Iosn--;
        else Iosm--;
    }
}

```

```
    if(sz[y]==1){
        if(y>m)Iosn--;
        else Iosm--;
    }
    if(sz[x]>sz[y])swap(x,y);
    Stack[++top]=x;
    fa[x]=y,sz[y]+=sz[x];
}
void Inv_union(int x,int y){
    block_cnt++;
    sz[y]-=sz[x];
    fa[x]=x;
    if(sz[x]==1){
        if(x>m)Iosn++;
        else Iosm++;
    }
    if(sz[y]==1){
        if(y>m)Iosn++;
        else Iosm++;
    }
    top--;
}
void solve(int k){
    int cur=top;
    _for(i,0,a[k].size()){
        pair<int,int> temp=a[k][i];
        Union(temp.first,temp.second);
    }
    if(lef[k]==rig[k]){
        if(Iosm)
            puts("-1");
        else
            enter(block_cnt-Iosn);
    }
    else
        solve(k<<1),solve(k<<1|1);
    while(top>cur)Inv_union(Stack[top],fa[Stack[top]]);
}
map<pair<int,int>,int> st;
int main()
{
    int n=read_int(),m=read_int(),q=read_int(),u,v;
    pair<int,int> temp;
    ::m=m;
    _rep(i,1,n+m)fa[i]=i,sz[i]=1;
    block_cnt=n+m,Iosn=n,Iosm=m;
    build(1,1,q);
    _rep(i,1,n){
        int k=read_int();
        while(k--)
```

```
    st[make_pair(read_int(),i+m)]=1;
}
_rep(i,1,q){
    u=read_int(),v=read_int()+m;
    temp=make_pair(u,v);
    if(st.count(temp)){
        int l=st[temp];
        if(l!=i)add(1,l,i-1,temp);
        st.erase(temp);
    }
    else
        st[temp]=i;
}
for(auto it=st.begin();it!=st.end();++it)
add(1,it->second,q,it->first);
solve(1);
return 0;
}
```

From:
<https://wiki.cvbbacm.com/> - CVBB ACM Team

Permanent link:
https://wiki.cvbbacm.com/doku.php?id=2020-2021:teams:legal_string:jxm2001:%E7%BA%BF%E6%AE%B5%E6%A0%91%E5%88%86%E6%B2%BB&rev=1597721241

Last update: 2020/08/18 11:27