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

https://ac.nowcoder.com/acm/contest/5673/A

题意

题解