比赛链接

补题情况

题目	蒋贤蒙	王赵安	王智彪
B	0	0	0
C	0	0	0
F	2	0	0
G	0	0	0
H	0	0	0
I	0	0	0
J	2	0	0

题解

J. Tree

题意

给定一棵树和两个人的初始位置 s,t，两个人在树上轮流移动，两个人走过的点都不能再走且两个人不能在同一个点。

每个人的分数为自己无法移动时走过的点数，两个人都希望最大化自己和另一个人的分数差，问最终先手的分数减去后手分数的值。

题解

首先以先手为根建树，树形 dp 求出每个点 u 不经过 s,t 链上的点能到达的最远距离 $a(u)$，设链上的点为 $v_1,v_2\cdots v_k$，其中 $v_1=s,v_k=t$。

定义 $\text{solve}(l,r,0/1)$ 表示第一个人位于 v_l 第二个人位于 v_r，$0/1$ 表示轮到先手/后手行动的答案。

对 $\text{solve}(l,r,0)$，先手如果选择进入子树，此时答案为 $a(v_l)+l-1-\max_{l\lt x\le r}(a(v_x)+k-x)$。

如果选择不进入子树，则答案为 $\text{solve}(l+1,r,1)$，所以两种情况取 $\max$ 即可，后手情况类似。

上述转移可以使用 ST 表，时间复杂度 $O(n\log n)$。

```
const int MAXN=5e5+5,MAXM=20;
struct Edge{
    int to,next;
}edge[MAXN<<1];
int head[MAXN],edge_cnt;
void Insert(int u,int v){
    edge[++edge_cnt]=Edge{v,head[u]};
```

```
    head[u]=edge_cnt;
}
int p[MAXN];
multiset<int> dp[MAXN];
void dfs(int u,int fa){
    dp[u].insert(1);
    p[u]=fa;
    for(int i=head[u];i;i=edge[i].next){
        int v=edge[i].to;
        if(v==fa)continue;
        dfs(v,u);
        dp[u].insert(*dp[v].rbegin()+1);
    }
}
struct ST{
    int d[MAXN][MAXM],lg2[MAXN];
    void build(int *a,int n){
        lg2[1]=0;
        _rep(i,2,n)
        lg2[i]=lg2[i>>1]+1;
        _for(i,0,n)
        d[i][0]=a[i];
        for(int j=1;(1<<j)<=n;j++){
            _for(i,0,n+1-(1<<j))
            d[i][j]=max(d[i][j-1],d[i+(1<<(j-1))][j-1]);
        }
    }
    int query(int lef,int rig){
        int len=rig-lef+1;
        return max(d[lef][lg2[len]],d[rig-(1<<lg2[len])+1][lg2[len]]);
    }
}st1,st2;
int mv1[MAXN],mv2[MAXN];
int solve(int l,int r,int pos){
    if(l+1==r)
    return mv1[l]-mv2[r];
    if(pos==0)
    return max(mv1[l]-st2.query(l+1,r),solve(l+1,r,!pos));
    else
    return min(st1.query(l,r-1)-mv2[r],solve(l,r-1,!pos));
}
int main(){
    int n=read_int(),s=read_int(),t=read_int();
    _for(i,1,n){
        int u=read_int(),v=read_int();
        Insert(u,v);
        Insert(v,u);
    }
    dfs(s,0);
    vector<int> vec;
```

```
    int pos=t;
    while(pos){
        vec.push_back(pos);
        pos=p[pos];
    }
    reverse(vec.begin(),vec.end());
    _for(i,0,vec.size()){
        int v=vec[i];
        if(i!=vec.size()-1){
            int v2=vec[i+1];
            dp[v].erase(dp[v].find(*dp[v2].rbegin()+1));
        }
    }
    _for(i,0,vec.size()){
        mv1[i]=*dp[vec[i]].rbegin()+i;
        mv2[i]=*dp[vec[i]].rbegin()+vec.size()-i-1;
    }
    st1.build(mv1,vec.size());
    st2.build(mv2,vec.size());
    enter(solve(0,vec.size()-1,0));
    return 0;
}
```