计算学部寒假十大打卡活动——“龙舞编程新春会”编程打卡

> 2024年2月2日
>
> AcWing 354. 天天爱跑路

一、题目

题目描述

小 C 同学认为跑步非常有趣，于是决定制作一款叫做《天天爱跑步》的游戏。《天天爱跑步》是一个养成类游戏，需要玩家每天按时上线，完成打卡任务。

这个游戏的地图可以看作一棵包含 n 个结点和 $n-1$ 条边的树，每条边连接两个结点，且任意两个结点存在一条路径互相可达。树上结点编号为从 1 到 n 的连续正整数。

现在有 m 个玩家，第 i 个玩家的起点为 s_i，终点为 t_i。每天打卡任务开始时，所有玩家在第 0 秒同时从自己的起点出发，以每秒跑一条边的速度，不间断地沿着最短路径向着自己的终点跑去，跑到终点后该玩家就算完成了打卡任务。（由于地图是一棵树，所以每个人的路径是唯一的）

小 C 想知道游戏的活跃度，所以在每个结点上都放置了一个观察员。在结点 j 的观察员会选择在第 w_j 秒观察玩家，一个玩家能被这个观察员观察到当且仅当该玩家在第秒 w_j也正好到达了结点 j。小 C 想知道每个观察员会观察到多少人?

注意：我们认为一个玩家到达自己的终点后该玩家就会结束游戏，他不能等待一 段时间后再被观察员观察到。 即对于把结点 j 作为终点的玩家：若他在第 w_j 秒前到达终点，则在结点 j 的观察员不能观察到该玩家；若他正好在第 w_j秒到达终点，则在结点 j 的观察员可以观察到这个玩家。

输入格式

第一行有两个整数 n 和 m。其中 n 代表树的结点数量, 同时也是观察员的数量, m 代表玩家的数量。

接下来 $n-1$ 行每行两个整数 u 和 v，表示结点 u 到结点 v 有一条边。

接下来一行 n 个整数，其中第 j 个整数为 w_j , 表示结点 j 出现观察员的时间。

接下来 m 行，每行两个整数s_i和t_i，表示一个玩家的起点和终点。

对于所有的数据，保证 $1 \le s_i, t_i \le n, 0 \le w_j \le n$。

输出格式

输出 1 行 n 个整数，第 j 个整数表示结点 j 的观察员可以观察到多少人。

输入输出样例

输入 #1

```
6 3
2 3
1 2
1 4
4 5
4 6
0 2 5 1 2 3
1 5
1 3
2 6
```

输出 #1

```
2 0 0 1 1 1
```

输入 #2

```
5 3
1 2
2 3
2 4
1 5
0 1 0 3 0
3 1
1 4
5 5
```

输出 #2

```
1 2 1 0 1
```

二、解答

思路：线段树合并

(由于ll今日有事，故选用了本人之前写的题解与代码，大家多多理解呀QwQ)

1.Analysis

某条路线对当前节点x有贡献时只可能是

x在s到t的路线上（即s到lca到t）

那么首先我们可以对从s到t的路线划分为

从s到lca(s,t)，以及从t到lca

a.

from s to lca
要有贡献

易得dep[s]-dep[x]=valx

观察可得寻找s满足dep[s]=dep[x]+val[x]的数量

b.

from lca to t
要有贡献

易得dep[s]+dep[x]-dep[lca]-dep[lca]=val[x] (出现时间)

观察可得寻找s满足dep[s]-dep[lca]-dep[lca]=val[x]-dep[x]的数量

为了计算.我们规定lca被算在a部分中

2.实现

a.

不断从叶子到lca累加
易联想到差分

b.

由于种类不同可以开权值线段树
在递归回fa时合并线段树

```
#include<bits/stdc++.h>
#define re return
#define inc(i,l,r) for(int i=l;i<=r;++i)
using namespace std;
template<typename T>inline void rd(T&x)
{
    char c;bool f=0;
    while((c=getchar())<'0'||c>'9')if(c=='-')f=1;
    x=c^48;
    while((c=getchar())>='0'&&c<='9')x=x*10+(c^48);
    if(f)x=-x;
}

const int maxn=300005,maxz=600000,maxl=300000;
int n,m,k,hd[maxn],ans[maxn],val[maxn];
struct node
{
    int to,nt;
}e[maxn<<1];

inline void add(int x,int y)
{
    e[++k].to=y;e[k].nt=hd[x];hd[x]=k;
    e[++k].to=x;e[k].nt=hd[y];hd[y]=k;
}

struct solu{
    //分为s,t两部分
    int top,tot,cnt;
    int rt[maxn*30],sum[maxn*30][2],ls[maxn*30],rs[maxn*30],first[maxn];
    int rab[maxn*30];
    struct ll{
```

```
    int flag,op,val,nt;
}st[maxn<<2];

inline void insert(int x,int y,int z,int f)
{
    st[++top]=(ll){f,z,y,first[x]};
    first[x]=top;
}


inline int New()
{
    int now;
    if(tot)now=rab[tot--];
    else now=++cnt;
    ls[now]=rs[now]=sum[now][0]=sum[now][1]=0;
    re now;
}
inline void Throw(int x)
{
    rab[++tot]=x;
}


inline int query(int rt,int l,int r,int pos,int f)
{
    if(!rt)re 0;
    if(l==r)
        re sum[rt][f];
    int mid=(l+r)>>1;
    if(pos<=mid)re query(ls[rt],l,mid,pos,f);
    else re query(rs[rt],mid+1,r,pos,f);
}

inline void add(int &rt,int l,int r,int pos,int vvl,int f)
{
    if(!rt) rt=New();
    if(l==r)
    {
        sum[rt][f]+=vvl;
        re ;
    }
    int mid=(l+r)>>1;
    if(pos<=mid)add(ls[rt],l,mid,pos,vvl,f);
    else add(rs[rt],mid+1,r,pos,vvl,f);
}

inline int merge(int x,int y,int l,int r)
{
    if(!x||!y)re x+y;
    if(l==r)
    {

        sum[x][0]+=sum[y][0];
        sum[x][1]+=sum[y][1];
```

```
            Throw(y);
            re x;
        }

        int mid=(l+r)>>1;

        ls[x]=merge(ls[x],ls[y],l,mid);
        rs[x]=merge(rs[x],rs[y],mid+1,r);
        Throw(y);
        re x;
    }
}T;

struct Tree_lca
{
    int top[maxn],size[maxn],son[maxn],dep[maxn],fa[maxn];

    inline void dfs(int x)
    {
        dep[x]=dep[fa[x]]+(size[x]=1);
        for(int i=hd[x];i;i=e[i].nt)
        {
            int v=e[i].to;
            if(v!=fa[x])
            {
                fa[v]=x;
                dfs(v);
                size[x]+=size[v];
                if(size[v]>size[son[x]])son[x]=v;
            }
        }
    }

    inline void dfs2(int x,int topf)
    {
        top[x]=topf;
        if(son[x])
        {
            dfs2(son[x],topf);
            for(int i=hd[x];i;i=e[i].nt)
            {
                int v=e[i].to;
                if(!top[v])
                dfs2(v,v);
            }
        }
    }

    inline int Lca(int x,int y)
    {
        while(top[x]!=top[y])
        {
            if(dep[top[x]]<dep[top[y]])x^=y^=x^=y;
            x=fa[top[x]];
        }
```

```
        re dep[x]<dep[y]?x:y;
    }

}S;

inline void dfs(int x)
{
    for(int i=hd[x];i;i=e[i].nt)
    {
        int v=e[i].to;
        if(v!=S.fa[x])
        {
            dfs(v);
            if(x==1&&v==2)
            x=1;
            T.rt[x]=T.merge(T.rt[x],T.rt[v],1,maxz);
        }
    }

    for(int i=T.first[x];i;i=T.st[i].nt)
        T.add(T.rt[x],1,maxz,T.st[i].val,T.st[i].op,T.st[i].flag);

    ans[x]=T.query(T.rt[x],1,maxz,S.dep[x]+val[x]+maxl,0);
    ans[x]+=T.query(T.rt[x],1,maxz,val[x]-S.dep[x]+maxl,1);

}

int main()
{
//  freopen("in.txt","r",stdin);

    int x,y,z;
    rd(n),rd(m);
    inc(i,2,n)
    {
        rd(x),rd(y);
        add(x,y);
    }

    S.dfs(1);
    S.dfs2(1,n+1);

    inc(i,1,n)
    rd(val[i]);

    inc(i,1,m)
    {
        rd(x),rd(y);
        int lca=S.Lca(x,y),flca=S.fa[lca];
        T.insert(x,S.dep[x]+maxl,1,0);T.insert(flca,S.dep[x]+maxl,-1,0);
        T.insert(y,S.dep[x]-(S.dep[lca]<<1)+maxl,1,1);T.insert(lca,S.dep[x]-
(S.dep[lca]<<1)+maxl,-1,1);
    }
    dfs(1);
```

```
    inc(i,1,n)
    printf("%d ",ans[i]);
    re 0;
}
```

三、总结

1. 容易出现炸空间的问题，直接线段树回收一波即可