树上差分的两种思路

[天天爱跑步](https://www.luogu.org/problemnew/show/P1600)；

[运输计划](https://www.luogu.org/problemnew/show/P2680)；

[疫情控制](https://www.luogu.org/problemnew/show/P1084)；

[松鼠的新家](https://www.luogu.org/problemnew/show/P3258)；

一堆noip的难题由此诞生。

好了，四不四已经跃跃欲试了呢？让我们先看一下

## 差分为何物。

放一个水题：

“给你一个m×n的矩阵，然后使用k块地毯铺地。每片地毯都给出左下角和右上角坐标。问所有地毯铺完之后，还有多少个整点（所谓整点，即横、纵坐标均为整数的点）没有被地毯覆盖。”

想到暴力：1.暴力枚举每张地毯

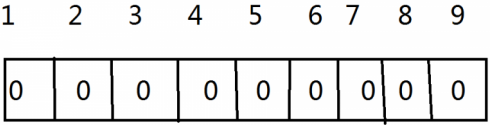
2.将所有被覆盖的点均做上标记

3.最后再枚举所有整点，若未被标记则ans+1；

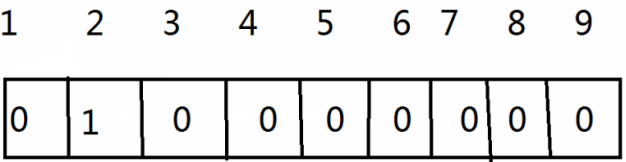
然而时间复杂度是O（mnk）的，直接超时。

竟然有人想用线段树？太强了，考场祝您一路顺风。

考虑差分

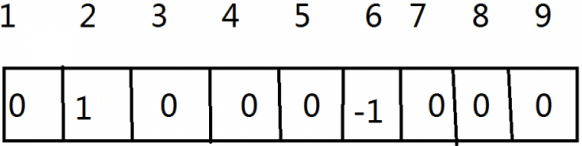


用前缀和的方式进行维护，比如我们覆盖了2到5.

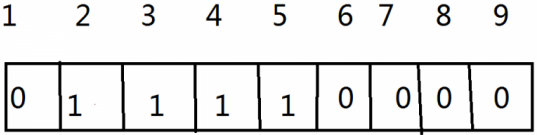


在2那里加上1，那么2之后的前缀和就都是1（表示覆盖）了。

然而我们要找的是2到5，不是2以后，所以在6那里要减1。



所以我们要求的前缀和



完美解决

让我们考虑树上差分

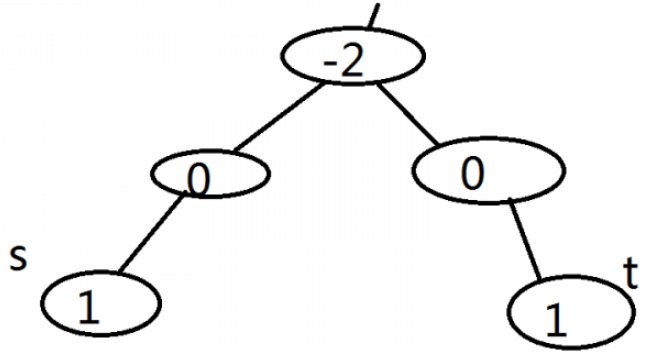
## 一.关于边的差分（如找被所有路径共同覆盖的边）

首先我们除了一般的grand，depth等数组以外，多开两个数组：tmp和prev。

tmp用来记录点的出现次数（具体点说实际上记录的是点到其父亲的边的出现次数），prev记录每个点到其父亲的那条边。对于一条起点s，终点t的路径。我们这样处理：

tmp[s]++,tmp[t]++,tmp[LCA(s,t)]-=2。（记住：最后要从所有叶结点把权值向上累加。）以一次操作为例，我们来看看效果（可以画一张图）。首先tmp[s]++，一直推上去到根，这时候s到root的路径访问次数都+1，tmp[t]++后，t到lca路径加了1，s到lca路径加了1，而lca到根的路径加了2。

这时，我们只需要tmp[LCA(s,t)]-=2，推到根，就能把那些多余的路径减掉，达到想要的目的。而这是一次操作，对于很多次操作的话，我们只需要维护tmp，而不必每次更新到根，维护好tmp最后Dfs一遍即可。这时如果tmp[i]==次数的话，说明i到其父亲的边是被所有路径覆盖的。如图



放一个例题代码：运输计划

#include<iostream>

#include<stdio.h>

#include<cstring>

using namespace std;

struct ss{

int next,to,val;

};ss data[600010];

struct truck{

int s,t,lca;

};truck node[300010];

int n,m,p,flag,cnt,maxn;

int head[300010],deep[300010],f[300010][25],dis[300010],pre[300010],sum[300010];

void change(int &a,int &b)

{

int t=a;a=b;b=t;

}

void add(int a,int b,int c)

{

data[++p].to=b;

data[p].next=head[a];

data[p].val=c;

head[a]=p;

}

void dfs(int a,int fa)

{

deep[a]=deep[fa]+1;

f[a][0]=fa;

for(int i=1;i<=20;i++)

f[a][i]=f[f[a][i-1]][i-1];

for(int i=head[a];i;i=data[i].next)

{

int v=data[i].to;

if(v==fa) continue;

dis[v]=dis[a]+data[i].val;

pre[v]=data[i].val;

dfs(v,a);

}

}

int getlca(int a,int b)

{

if(deep[a]>deep[b]) change(a,b);

for(int i=20;i>=0;i--)

{

if(deep[a]<=deep[f[b][i]])

b=f[b][i];

}

if(a==b) return a;

for(int i=20;i>=0;i--)

if(f[a][i]!=f[b][i])

{

a=f[a][i];

b=f[b][i];

}

return f[a][0];

}

int judge(int a,int fa,int cnt,int maxn)

{

int nsum=sum[a];

for(int i=head[a];i;i=data[i].next)

{

int v=data[i].to;

if(v==fa) continue;

nsum+=judge(v,a,cnt,maxn);

}

if(nsum>=cnt&&pre[a]>=maxn) flag=1;

return nsum;

}

int check(long long limit)

{

memset(sum,0,sizeof(sum));

cnt=0,flag=0,maxn=0;

for(int i=1;i<=m;i++)

if(dis[node[i].s]+dis[node[i].t]-2\*dis[node[i].lca]>limit)

{

sum[node[i].s]++,sum[node[i].t]++,sum[node[i].lca]-=2;

cnt++;

maxn=max(maxn,dis[node[i].s]+dis[node[i].t]-2\*dis[node[i].lca]);

}

if(cnt==0) return 1;

int wsb=judge(1,0,cnt,maxn-limit);

return flag;

}

int main()

{

scanf("%d%d",&n,&m);

for(int i=1;i<=n-1;i++)

{

int a,b,c;

scanf("%d%d%d",&a,&b,&c);

add(a,b,c);

add(b,a,c);

}

dfs(1,0);

for(int i=1;i<=m;i++)

{

scanf("%d%d",&node[i].s,&node[i].t);

node[i].lca=getlca(node[i].s,node[i].t);

}

long long l=0,r=3000000000;

while(l+1<r)

{

int mid=(l+r)/2;

if(check(mid)) r=mid;

else l=mid;

}

if(check(l)) printf("%lld\n",l);

else printf("%lld\n",r);

return 0;

}

## 二.关于点的差分（如将路径上的所有点权值加一，求最后点的权值）

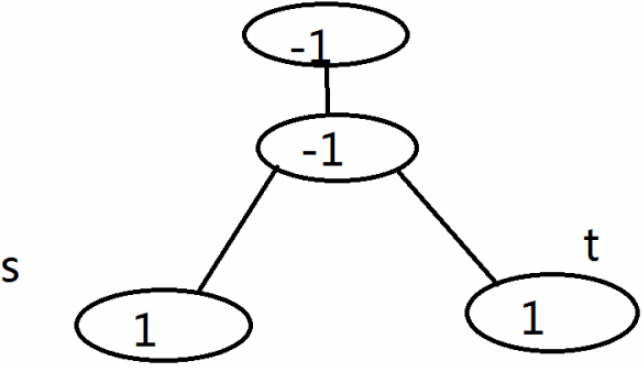
此操作中我们这样维护：每次经过一条边，（如从u到v）我们让tmp[u]++,tmp[v]++,tmp[LCA(u,v)]--,tmp[grand[LCA(u,v)][0]]--。（最后要把tmp推上去）

以一次添加为例想象一下，首先u到根的路径上tmp都+1，此时u到根间结点tmp都为1，之后v到根路径上tmp+1，此时u到LCA前一个，v到LCA前一个点的tmp都+1，而LCA到根的所有点都+2，然后从tmp[LCA]--,更新上去，此时u-v路上所有tmp都+1，已经达到目的。

而多余的是什么部分呢，也就是LCA的上一个结点（grand[LCA][0]）到根的这一段都多加了1，所以tmp[grand[LCA][0]]--,更新上去，也就完成了。

实际操作时也不需要每次更新都推上去，只要把四个tmp维护好，最后Dfs走一边就更新完了。

如图



放一个例题代码：松鼠的新家

#include<iostream>

#include<stdio.h>

using namespace std;

struct ss{

int next,to;

};ss data[600010];

int n,q;

int a[300010],head[600010],deep[300010],p[300010][25],sum[300010];

void change(int &a,int &b)

{

int t=a;a=b;b=t;

}

void add(int a,int b)

{

data[++q].to=b;

data[q].next=head[a];

head[a]=q;

}

void dfs(int a,int fa)

{

deep[a]=deep[fa]+1;

p[a][0]=fa;

for(int i=1;(1<<i)<=deep[a];i++)

p[a][i]=p[p[a][i-1]][i-1];

for(int i=head[a];i;i=data[i].next)

{

int v=data[i].to;

if(v!=fa)

dfs(v,a);

}

}

int lca(int a,int b)

{

if(deep[a]>deep[b]) change(a,b);

for(int i=20;i>=0;i--)

{

if(deep[a]<=deep[b]-(1<<i))

{

//printf("a=%d %d %d\n",a,b,i);

b=p[b][i];

}

}

//printf("%d %d\n",a,b);

if(a==b) return a;

for(int i=20;i>=0;i--)

{

if(p[a][i]!=p[b][i])

a=p[a][i],b=p[b][i];

}

//printf("a=%d\n",a);

return p[a][0];

}

void search(int a)

{

for(int i=head[a];i;i=data[i].next)

{

int v=data[i].to;

if(v==p[a][0]) continue;

search(v);

sum[a]+=sum[v];

}

}

int main()

{

scanf("%d",&n);

for(int i=1;i<=n;i++)

scanf("%d",&a[i]);

for(int i=1;i<=n-1;i++)

{

int u,v;

scanf("%d%d",&u,&v);

add(u,v);add(v,u);

}

dfs(1,0);

for(int i=1;i<=n-1;i++)

{

int x=a[i],y=a[i+1],LCA=lca(x,y);

//printf("lca=%d\n",LCA);

sum[x]++;

sum[y]++;

sum[LCA]--;

sum[p[LCA][0]]--;

}

search(1);

for(int i=2;i<=n;i++)

sum[a[i]]--;

for(int i=1;i<=n;i++)

printf("%d\n",sum[i]);

return 0;

}