

# 树链剖分

## 是什么

## 怎么用

### 普通用法

### trick | keys:LCA

感觉这两道题有一点点共性但又不是显式的，还是放在一起了，无论如何挺值得两连做的。建议思考，仿佛有智力开发的效果

### [BZOJ 3626][LNOI2014]LCA

给出  $n$  个点的有根树  $q$  次询问，每次询问给出  $l, r, z$  求  $\sum_{1 \leq i \leq r} \text{dep}[\text{LCA}(i, z)]$   $1 \leq n, q \leq 50000$

题解：定义询问  $(x, z)$  为  $\sum_{1 \leq i \leq x} \text{dep}[\text{LCA}(i, z)]$  则可以将题中询问拆为询问  $(r, z)$  和  $(l-1, z)$  的差。于是可以离线，从  $1$  到  $n$  逐个加入，加入时将该节点到根节点路径上点的值各加一，线段树维护。而对于每个  $z$  可以在加入  $x$  刚好完成时查询  $z$  到根节点路径上点的值来得到答案。

code:

```
#include<iostream>
#include<cstdio>
#include<cstdlib>
#include<cstring>
#include<algorithm>
#define MAXN 50005
#define Mod 201314
using namespace std;
int n, q, deep[MAXN], head[MAXN], cnt=0, tot=0;
int sz=0, maxv[MAXN], father[MAXN], siz[MAXN], top[MAXN], pos[MAXN], res[MAXN*2];
struct Node1
{
    int next, to;
}Edges[MAXN*2];
struct Node2
{
    int l, r, sum, lazy;
}t[MAXN*4];
struct Node3
{
```

```
int x,z,id;
Node3(int x=0,int z=0,int id=0):x(x),z(z),id(id){}
}Q[MAXN*2];
bool cmp(Node3 a,Node3 b){return a.x<b.x;}
int read()
{
int x=0,f=1;char c=getchar();
while(c<'0' || c>'9'){if(c=='-')f=-1;c=getchar();}
while(c>='0'&&c<='9'){x=x*10+c-'0';c=getchar();}
return x*f;
}
void addedge(int u,int v)
{
Edges[++cnt].next=head[u];
head[u]=cnt;
Edges[cnt].to=v;
}
void dfs1(int u)
{
siz[u]=1;int k=-1;
for(int i=head[u];~i;i=Edges[i].next)
{
int v=Edges[i].to;
deep[v]=deep[u]+1,dfs1(v);
siz[u]+=siz[v];
if(k==-1 || siz[v]>siz[k])k=v;
}
maxv[u]=k;
}
void dfs2(int u,int t)
{
++sz,pos[u]=sz;
top[u]=t;
if(maxv[u]!=-1)dfs2(maxv[u],t);
for(int i=head[u];~i;i=Edges[i].next)
if(maxv[u]!=Edges[i].to)dfs2(Edges[i].to,Edges[i].to);
}
void build(int idx,int l,int r)
{
t[idx].l=l,t[idx].r=r;
if(l==r)return;
int mid=(l+r)>>1;
build(idx<<1,l,mid),build(idx<<1|1,mid+1,r);
}
void pushdown(int idx)
{
if(t[idx].l<t[idx].r&&t[idx].lazy)
{
t[idx<<1].lazy+=t[idx].lazy;
t[idx<<1|1].lazy+=t[idx].lazy;
}
```

```

    t[idx<<1].sum+=t[idx].lazy*(t[idx<<1].r-t[idx<<1].l+1);
    t[idx<<1].sum%=Mod;
    t[idx<<1|1].sum+=t[idx].lazy*(t[idx<<1|1].r-t[idx<<1|1].l+1);
    t[idx<<1|1].sum%=Mod;
    t[idx].lazy=0;
}
}
void add(int idx,int l,int r)
{
    if(l<=t[idx].l&& r>=t[idx].r)
    {
        t[idx].sum+=t[idx].r-t[idx].l+1;
        t[idx].lazy++;return;
    }
    pushdown(idx);
    int mid=(t[idx].l+t[idx].r)>>1;
    if(r<=mid)add(idx<<1,l,r);
    else if(l>mid)add(idx<<1|1,l,r);
    else add(idx<<1,l,r),add(idx<<1|1,l,r);
    t[idx].sum=(t[idx<<1].sum+t[idx<<1|1].sum)%Mod;
}
int query(int idx,int l,int r)
{
    if(l<=t[idx].l&& r>=t[idx].r)return t[idx].sum;
    pushdown(idx);
    int mid=(t[idx].l+t[idx].r)>>1;
    if(r<=mid)return query(idx<<1,l,r);
    else if(l>mid)return query(idx<<1|1,l,r);
    else return (query(idx<<1,l,r)+query(idx<<1|1,l,r))%Mod;
}
void change(int x)
{
    while(top[x])
    {
        add(1,pos[top[x]],pos[x]);
        x=father[top[x]];
    }
    add(1,pos[top[x]],pos[x]);
}
int ask(int x)
{
    int res=0;
    while(top[x])
    {
        res=(res+query(1,pos[top[x]],pos[x]))%Mod;
        x=father[top[x]];
    }
    res=(res+query(1,pos[top[x]],pos[x]))%Mod;
    return res;
}
int main()

```

```
{
    memset(head, -1, sizeof(head));
    n=read(), q=read();
    for(int i=1; i<n; i++){ father[i]=read(); addedge(father[i], i); }
    deep[0]=1, dfs1(0);
    dfs2(0, 0); build(1, 1, sz);
    for(int i=1; i<=q; i++)
    {
        int l=read(), r=read(), z=read();
        ++tot, Q[tot]=Node3(l-1, z, tot);
        ++tot, Q[tot]=Node3(r, z, tot);
    }
    sort(Q+1, Q+1+tot, cmp);
    int now=0;
    for(int i=1; i<=tot; i++)
    {
        while(now<=Q[i].x) change(now), ++now;
        res[Q[i].id]=ask(Q[i].z);
    }
    for(int i=1; i<=tot; i+=2)
    printf("%d\n", (res[i+1] - res[i] + Mod) % Mod);
    return 0;
}
```

## [BZOJ 4012][HNOI2015]开店

给出  $n$  个点的带权树，节点  $i$  的妖怪有年龄  $x_i$ 。  $Q$  个询问，给出  $u, l, r$  询问年龄在  $[l, r]$  的妖怪到  $u$  点的距离之和，强制在线。  $n \leq 150000, Q \leq 200000$ ，年龄上限  $A \leq 10^9$ 。

题解：首先点  $u, v$  间的距离可以转换为  $\text{dis}(u, \text{root}) + \text{dis}(v, \text{root}) - \text{dis}(\text{lca}(u, v), \text{root})$ 。

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

Permanent link: [https://wiki.cvbbacm.com/doku.php?id=2020-2021:teams:wangzai\\_milk:%E6%A0%91%E9%93%BE%E5%89%96%E5%88%86&rev=1590337629](https://wiki.cvbbacm.com/doku.php?id=2020-2021:teams:wangzai_milk:%E6%A0%91%E9%93%BE%E5%89%96%E5%88%86&rev=1590337629)

Last update: 2020/05/25 00:27