长链剖分

算法简介

一种可以在线性时间复杂度维护子树中**仅限深度有关的信息**的算法，主要用于一些特殊的 dp

算法思想

类似重链剖分，将儿子分为重儿子和轻儿子，重儿子组成的链构成长链

但不同的是重链剖分重儿子是子树结点最多的结点，长链剖分长儿子是子树深度最大的结点

长链剖分有两个重要性质

性质一：所有长链长度和为 n

显然所有点均属于且仅属于一条长链，所以性质一成立

性质二：长链剖分的一个重要性质是一个结点 x 的 k 级祖先所在的长链长度一定 $\ge k$

考虑结点 x 和它的 k 级祖先 y，如果 x 和 y 属于同一条长链，该情况下性质二成立

如果 x 和 y 不属于同一条长链，知 y 的重儿子子树的深度一定大于 x 的深度，该情况下性质二也成立

算法应用

树上 k 级祖先

[洛谷p5903]

先一遍 dfs 处理出每个结点的深度、父结点、重儿子，同时用倍增法 $O\left(n\log n\right)$ 时间处理出每个结点的 2^k 级祖先

第二遍 dfs 处理出每个结点所在长链的起点 x

对每个长链的起点 x，暴力处理出 x 上下 $\text{len}(x)$ 个结点，其中 $\text{len}(x)$ 表示 x 所在长链的长度

由于所有长链长度之和为 n，所以该暴力处理的时间复杂度为 $O\left(n\right)$

对每次查询结点 x 的 k 级祖先，记 $h_k=\lfloor \log_2 k\rfloor$，$tk=k-2^{h_k}$

先从 x 向上跳 2^{h_k} 级到 y，根据长链剖分性质，知 y 所在长链长度必然 $\ge 2^{h_k}$

此时还需向上跳 tk 级，$tk\lt 2^{h_k}$，所以 x 的 k 级祖先一定在 y 所在长链起点的上下 $\text{len}(x)$ 级结点范围内

因此在从 y 跳到 y 所在长链起点，最后便可以 $O\left(1\right)$ 访问目标结点

时间复杂度 $O\left(n\log n\right)-O\left(1\right)$

```
#include <iostream>
#include <cstdio>
#include <algorithm>
#include <cstring>
#include <cctype>
#include <vector>
#define _for(i,a,b) for(int i=(a);i<(b);++i)
#define _rep(i,a,b) for(int i=(a);i<=(b);++i)
using namespace std;
typedef long long LL;
inline int read_int(){
    int t=0;bool sign=false;char c=getchar();
    while(!isdigit(c)){sign|=c=='-';c=getchar();}
    while(isdigit(c)){t=(t<<1)+(t<<3)+(c&15);c=getchar();}
    return sign?-t:t;
}
inline void write(LL x){
    register char c[21],len=0;
    if(!x)return putchar('0'),void();
    if(x<0)x=-x,putchar('-');
    while(x)c[++len]=x%10,x/=10;
    while(len)putchar(c[len--]+48);
}
inline void space(LL x){write(x),putchar(' ');}
inline void enter(LL x){write(x),putchar('\n');}
const int MAXN=5e5+5,MAXM=21;
unsigned int s;
inline unsigned int get(unsigned int x){
    x^=x<<13;
    x^=x>>17;
    x^=x<<5;
    return s=x;
}
struct Edge{
    int to,next;
}edge[MAXN<<1];
int head[MAXN],m;
void Insert(int u,int v){
    edge[++m].to=v;
    edge[m].next=head[u];
    head[u]=m;
}
int d[MAXN],fa[MAXN][MAXM],log_2[MAXN];
int h_son[MAXN],mson[MAXN],p[MAXN];
vector<int> Node_up[MAXN],Node_down[MAXN];
```

```
void get_log2(){
    log_2[0]=-1;
    _for(i,1,MAXN)
    log_2[i]=log_2[i>>1]+1;
}
void dfs_1(int u,int depth){
    mson[u]=d[u]=depth;
    _rep(i,1,log_2[d[u]])
    fa[u][i]=fa[fa[u][i-1]][i-1];
    for(int i=head[u];i;i=edge[i].next){
        int v=edge[i].to;
        dfs_1(v,depth+1);
        if(mson[v]>mson[u]){
            h_son[u]=v;
            mson[u]=mson[v];
        }
    }
}
void dfs_2(int u,int top){
    p[u]=top;
    if(u==top){
        for(int i=0,pos=u;i<=mson[u]-d[u];pos=fa[pos][0],i++)
        Node_up[u].push_back(pos);
        for(int i=0,pos=u;i<=mson[u]-d[u];pos=h_son[pos],i++)
        Node_down[u].push_back(pos);
    }
    if(h_son[u])
    dfs_2(h_son[u],top);
    for(int i=head[u];i;i=edge[i].next){
        int v=edge[i].to;
        if(v==h_son[u])
        continue;
        dfs_2(v,v);
    }
}
int query(int u,int k){
    if(!k)return u;
    u=fa[u][log_2[k]],k-=1<<log_2[k];
    k-=d[u]-d[p[u]],u=p[u];
    return k>=0?Node_up[u][k]:Node_down[u][-k];
}
int main()
{
    int n,q,root,x,y;
    cin>>n>>q>>s;
    _rep(i,1,n){
        fa[i][0]=read_int();
        if(fa[i][0])
        Insert(fa[i][0],i);
        else
        root=i;
```

```
    }
    get_log2();
    dfs_1(root,0);
    dfs_2(root,root);
    long long tot=0,last=0;
    int u,k;
    _rep(i,1,q){
        u=(get(s)^last)%n+1;
        k=(get(s)^last)%(d[u]+1);
        last=query(u,k);
        tot^=last*i;
    }
    enter(tot);
    return 0;
}
```

合并信息

洛谷p5904

给定一棵 n 个结点的树，问有多少个无序点对 (i,j,k) 满足 i,j,k 两两间距离相等