// LCT

//该模板基于BZOJ3282

//query操作求的是从u到v的路径的xor和

namespace LCT

{

struct node

{

int ch[2],val,father;

int flip,xsum;

}tree[MAXN\*2];int tot;

inline void Create(int lson,int rson,int val,int father)

{

++tot;

tree[tot].ch[0]=lson;tree[tot].ch[1]=rson;

tree[tot].val=val;tree[tot].father=father;

tree[tot].flip=0;tree[tot].xsum=val;

}

inline void pushup(int cur)

{

tree[cur].xsum=(tree[cur].val^tree[tree[cur].ch[0]].xsum^tree[tree[cur].ch[1]].xsum);

}

inline void pushdown(int cur)

{

if (tree[cur].flip)

{

tree[tree[cur].ch[0]].flip^=1;

tree[tree[cur].ch[1]].flip^=1;

swap(tree[cur].ch[0],tree[cur].ch[1]);

tree[cur].flip=0;

}

}

inline bool isroot(int cur) {return !tree[cur].father || (tree[tree[cur].father].ch[0]!=cur && tree[tree[cur].father].ch[1]!=cur);}

inline void rotate(int x)

{

int y=tree[x].father,z=tree[y].father;

pushdown(y);pushdown(x);

int k=(tree[y].ch[1]==x);

if (!isroot(y)) tree[z].ch[tree[z].ch[1]==y]=x;

tree[y].ch[k]=tree[x].ch[k^1];

tree[x].ch[k^1]=y;

tree[tree[y].ch[k]].father=y;tree[y].father=x;tree[x].father=z;

pushup(y);pushup(x);

}

inline void splay(int x)

{

pushdown(x);

while (!isroot(x))

{

int y=tree[x].father,z=tree[y].father;

if (!isroot(y))

((tree[y].ch[1]==x)^(tree[z].ch[1]==y))?rotate(x):rotate(y);

rotate(x);

}

}

inline void access(int cur)

{

for (int pre=0;cur;pre=cur,cur=tree[cur].father)

{

splay(cur);

tree[cur].ch[1]=pre;

pushup(cur);

}

}

inline void makeroot(int x)

{

access(x);splay(x);

tree[x].flip^=1;

}

inline int find(int x)

{

access(x);splay(x);

while (tree[x].ch[0]) x=tree[x].ch[0];

return x;

}

inline void link(int u,int v) // u is v's son

{

makeroot(u);

tree[u].father=v;

}

inline void cut(int u,int v)

{

makeroot(u);

access(v);splay(v);

if (tree[v].ch[0]==u)

tree[v].ch[0]=0,tree[u].father=0;

}

inline void update(int x,int val)

{

makeroot(x);

tree[x].val=val;

pushup(x);

}

inline int query(int u,int v)

{

makeroot(u);

access(v);splay(v);

return tree[v].xsum;

}

}

// 一份没有换根操作的模板

const int MAXN=1e5;

int n,q;

vector<int> v[MAXN+48];

int depth[MAXN+48],anc[MAXN+48][21];

int val[MAXN+48];

inline void dfs(int cur,int father)

{

int i,j;

for (auto y : v[cur])

if (y!=father)

{

depth[y]=depth[cur]+1;

anc[y][0]=cur;

for (j=1;j<=20;j++)

anc[y][j]=anc[anc[y][j-1]][j-1];

dfs(y,cur);

}

}

inline int jump(int u,int dist)

{

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

if (dist>=(1<<i)) u=anc[u][i],dist-=(1<<i);

return u;

}

namespace LCT

{

struct node

{

int ch[2],val,father;

int flip,xsum;

}tree[MAXN\*2];int tot;

inline void init()

{

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

tree[i].ch[0]=tree[i].ch[1]=tree[i].val=tree[i].father=tree[i].flip=tree[i].xsum=0;

}

inline void pushup(int cur)

{

if (!cur) return;

tree[cur].xsum=(tree[cur].val+tree[tree[cur].ch[0]].xsum+tree[tree[cur].ch[1]].xsum);

}

inline void pushdown(int cur)

{

if (tree[cur].flip)

{

tree[tree[cur].ch[0]].flip^=1;

tree[tree[cur].ch[1]].flip^=1;

swap(tree[cur].ch[0],tree[cur].ch[1]);

tree[cur].flip=0;

}

}

inline bool isroot(int cur) {return !tree[cur].father || (tree[tree[cur].father].ch[0]!=cur && tree[tree[cur].father].ch[1]!=cur);}

inline void rotate(int x)

{

int y=tree[x].father,z=tree[y].father;

pushdown(y);pushdown(x);

int k=(tree[y].ch[1]==x);

if (!isroot(y)) tree[z].ch[tree[z].ch[1]==y]=x;

tree[y].ch[k]=tree[x].ch[k^1];

tree[x].ch[k^1]=y;

tree[tree[y].ch[k]].father=y;tree[y].father=x;tree[x].father=z;

pushup(y);pushup(x);

}

inline void splay(int x)

{

pushdown(x);

while (!isroot(x))

{

int y=tree[x].father,z=tree[y].father;

if (!isroot(y))

((tree[y].ch[1]==x)^(tree[z].ch[1]==y))?rotate(x):rotate(y);

rotate(x);

}

}

inline void access(int cur)

{

for (int pre=0;cur;pre=cur,cur=tree[cur].father)

{

splay(cur);

tree[cur].ch[1]=pre;

pushup(cur);

}

}

inline void makeroot(int x)

{

access(x);splay(x);

tree[x].flip^=1;

}

inline int find(int x)

{

access(x);splay(x);

while (tree[x].ch[0]) x=tree[x].ch[0];

return x;

}

inline void link(int u,int v) // u is v's son

{

//makeroot(u);

access(u);splay(u);

tree[u].father=v;

}

inline void cut(int u)

{

//makeroot(u);

access(u);splay(u);

tree[tree[u].ch[0]].father=0;

tree[u].ch[0]=0;pushup(u);

}

inline void update(int x,int val)

{

makeroot(x);

tree[x].val=val;

pushup(x);

}

}