

[比赛链接](#)

补题情况

题目	蒋贤蒙	王赵安	王智彪
A	0	0	0
B	1	2	2
C	1	0	2
D	0	0	0
E	2	0	2
F	2	0	1
G	2	0	0
H	0	0	0
I	2	0	1
J	2	0	1

题解

G. Yu Ling(Ling YueZheng) and Colorful Tree

题意

给定一棵点权树，初始值各点权值为 \$0\$，接下来两种操作：

1. \$w(u)\gets x\$ 保证 \$w(u)\$ 之前一定为 \$0\$，该类操作的 \$x\$ 都不相同且 \$1 \leq x \leq n\$
2. 给定 \$l, r, x \leq n\$ 查询 \$u\$ 的最近祖先节点 \$v\$ 满足 \$x \mid w(v), w(v) \leq r\$ 输出 \$u, v\$ 之间的距离

题解

考虑离线操作，对每个权值，维护对他有贡献的操作序列。

首先对操作 \$1\$，显然 \$x\$ 会对所有是 \$x\$ 的因子的权值产生贡献，于是将操作 \$1\$ 加入到所有 \$x\$ 的因子的操作序列中。

对操作 \$2\$，直接丢到权值 \$x\$ 的操作序列处理即可。

接下来单独考虑每个权值的操作序列，发现操作序列一定都满足 \$x \mid w(v)\$ 这个约束。

于是只需要考虑 \$v\$ 是 \$u\$ 的最近祖先这个约束。

首先转化一下 \$v\$ 是 \$u\$ 的最近祖先这个约束，不难发现满足该约束的 \$v\$ 就是 \$u\$ 的祖先中序最大的点。

维护一个二维矩阵 \$C[x][y]\$ 表示 \$\text{dfs}\$ 序最大的 \$v\$ 满足 \$w(v)=x\$ 且 \$w(v)\$ 是节点 \$y\$ 的祖先。

于是对操作 \$1\$ 等价于修改

$\$ \$ C[x][\text{\text{\\text\{dfn\}}}(u)] \sim \text{\text{\\text\{dfn\}}}(u) + [\text{\text{\\text\{sz\}}}(u)-1] \text{ gets } \text{\text{\\text\{dfn\}}}(u) \$ \$$

操作 \$2\$ 等价于查询

$\$ \$ \max[\left(C[\mid \sim r][\text{\text{\\text\{dfn\}}}(u)]\right)] \$ \$$

考虑树套树维护 \$C\$ 考虑修改操作共 $O(n \log n)$ 次，查询操作 $O(n)$ 于是总时间复杂度 $O(\log^3 n)$

关于空间复杂度，首先对每个权值，由于他的倍数最多只有 $O(n)$ 个，所以修改操作最大只有 $O(n)$ 次。

树套树第二维是动态开点线段树，于是空间复杂度 $O(\log^2 n)$ 注意处理完每个权值的操作序列后要删除线段树，否则会爆空间。

```
const int MAXN=1.2e5+5,MAXM=150;
struct Edge{
    int to,w,next;
}edge[MAXN<<1];
int head[MAXN],edge_cnt;
void Insert(int u,int v,int w){
    edge[++edge_cnt]=Edge{v,w,head[u]};
    head[u]=edge_cnt;
}
int dfl[MAXN],dfr[MAXN],dfu[MAXN],dfs_t;
LL dis[MAXN];
void dfs(int u,int fa){
    dfl[u]=++dfs_t;
    dfu[dfs_t]=u;
    for(int i=head[u];i;i=edge[i].next){
        int v=edge[i].to;
        if(v==fa) continue;
        dis[v]=dis[u]+edge[i].w;
        dfs(v,u);
    }
    dfr[u]=dfs_t;
}
struct node{
    int u,x,l,r,id;
    node(int u=0,int x=0,int l=0,int r=0,int id=0){
        this->u=u;
        this->x=x;
        this->l=l;
        this->r=r;
        this->id=id;
    }
};
namespace Tree{
    int ch[MAXN*MAXM][2],s[MAXN*MAXM],cnt;
```

```
void clear(){cnt=0;}
void update(int &k,int vl,int vr,int ql,int qr,int v){
    if(!k){
        k=++cnt;
        s[k]=0;
        ch[k][0]=ch[k][1]=0;
    }
    if(ql<=vl&&vr<=qr){
        s[k]=max(s[k],v);
        return;
    }
    int vm=vl+vr>>1;
    if(vm>=ql)
        update(ch[k][0],vl,vm,ql,qr,v);
    if(vm<qr)
        update(ch[k][1],vm+1,vr,ql,qr,v);
}
int query(int k,int vl,int vr,int pos){
    if(!k) return 0;
    if(vl==vr) return s[k];
    int vm=vl+vr>>1;
    if(vm>=pos)
        return max(s[k],query(ch[k][0],vl,vm,pos));
    else
        return max(s[k],query(ch[k][1],vm+1,vr,pos));
}
int rt0[MAXN],rt[MAXN],n;
#define lowbit(x) (x&(-x))
void update(int pos,int u){
    Tree::update(rt0[pos],1,n,dfl[u],dfr[u],dfl[u]);
    while(pos<=n){
        Tree::update(rt[pos],1,n,dfl[u],dfr[u],dfl[u]);
        pos+=lowbit(pos);
    }
}
int query(int l,int r,int u){
    int ans=0;
    while(l<=r){
        ans=max(ans,Tree::query(rt0[r],1,n,dfl[u]));
        for(--r;r-l>=lowbit(r);r-=lowbit(r))
            ans=max(ans,Tree::query(rt[r],1,n,dfl[u]));
    }
    return ans;
}
void del(int pos){
    rt0[pos]=0;
    while(pos<=n){
        rt[pos]=0;
        pos+=lowbit(pos);
    }
}
```

```
}

vector<int> d[MAXN];
vector<node> opt[MAXN];
LL ans[MAXN];
void solve(int v){
    for(node q:opt[v]){
        if(q.x==0){
            int t=query(q.l,q.r,q.u);
            if(t==0)
                ans[q.id]=-1;
            else
                ans[q.id]=dis[q.u]-dis[dfu[t]];
        }
        else
            update(q.x,q.u);
    }
    for(node q:opt[v]){
        if(q.x!=0)
            del(q.x);
    }
}
Tree::clear();
}

int main(){
    n=read_int();
    int q=read_int();
    _for(i,1,n){
        int u=read_int(),v=read_int(),w=read_int();
        Insert(u,v,w);
        Insert(v,u,w);
    }
    dfs(1,0);
    _rep(i,1,n){
        for(int j=i;j<=n;j+=i)
            d[j].push_back(i);
    }
    int qcnt=0;
    while(q--){
        int type=read_int(),u=read_int();
        if(type==0){
            int x=read_int();
            for(int v:d[x])
                opt[v].push_back(node(u,x));
        }
        else{
            int l=read_int(),r=read_int(),x=read_int();
            opt[x].push_back(node(u,0,l,r,qcnt++));
        }
    }
    _rep(i,1,n)
    solve(i);
}
```

```

    _for(i,0,qcnt){
        if(ans[i]==-1)
            puts("Impossible!");
        else
            enter(ans[i]);
    }
    return 0;
}

```

I. Kuriyama Mirai and Exclusive Or

题意

给定一个序列 A 接下来两种操作：

1. $a_i \leftarrow a_i + x$ ($i \in [l, r]$)
2. $a_i \leftarrow a_i + (x + i - l)$ ($i \in [l, r]$)

题解

$a_n = \left(\bigoplus_{i=1}^n b_i \right) \oplus \left(\bigoplus_{i=1}^n \bigoplus_{k=0}^{29} 2^k c_{i,k} \right)$

简单来讲 a_n 是 $b_i, c_{i,k}$ 的前缀和， $c(n,k)$ 是 $d(i,k)$ 的隔 2^k 项前缀和。

然后操作 1 只需要修改 b_i 即可。操作 2 的影响按位考虑影响，将 $[l, r]$ 区间操作等效于 $[l, \inf), [r+1, \inf)$ 两次操作。

每次操作对每个位是按 001111000011110000 的规律周期性变化的。

对该序列进行一次差分，于是得到 001000100010001000 ，这个可以利用 $c(n,k)$ 维护。

再一次差分得到 001000000000000000 发现可以利用 $d(n,k)$ 维护。

最后处理一下最开始非周期的部分即可，时间复杂度 $O((n+q)\log v)$

```

const int MAXN=6e5+5,MAXB=30,mod=1<<30;
int n,a[MAXN],b[MAXN];
bitset<MAXN> c[MAXB];
void update(int pos,int v){
    _for(i,0,MAXB){
        int cyc=1<<(i+1),p1=pos%cyc,p2=(cyc-
v%cyc)%cyc+(1<<i))%cyc,d=pos>>(i+1);
        if(p1>=p2){
            if(p1<p2+(1<<i)){
                b[pos]^=1<<i;
                if(d*cyc+p2+(1<<i)<MAXN)
                    b[d*cyc+p2+(1<<i)]^=1<<i;
            }
        }
    }
}

```

```
        }
        d++;
    }
    else if(p1<p2-(1<<i)){
        b[pos]^=1<<i;
        if(d*cyc+p2-(1<<i)<MAXN)
            b[d*cyc+p2-(1<<i)]^=1<<i;
    }
    if(d*cyc+p2<MAXN)
        c[i].flip(d*cyc+p2);
}
}

int main(){
n=read_int();
int q=read_int();
_for(i,0,n)a[i]=read_int();
while(q--){
    int opt=read_int(),l=read_int()-1,r=read_int()-1,x=read_int();
    if(opt==0){
        b[l]^=x;
        b[r+1]^=x;
    }
    else{
        update(l,(x-l+mod)%mod);
        update(r+1,(x-l+mod)%mod);
    }
}
_for(i,0,MAXB){
    _for(j,0,n)if(j+(1<<i)<MAXN)
        c[i][j+(1<<i)]=c[i][j+(1<<i)]^c[i][j];
    _for(j,1,n)
        c[i][j]=c[i][j]^c[i][j-1];
    _for(j,0,n)
        a[j]^=(c[i][j]<<i);
}
_for(i,1,n)b[i]^=b[i-1];
_for(i,0,n)a[i]^=b[i];
_for(i,0,n)
space(a[i]);
return 0;
}
```

赛后总结

jxm 找操作中的不变量可能是解题的一种思路，比如这场的 \$B\$ 或者博弈论题 \$A\$ 据说是 \$\text{dp}\$ 优化题，但没看懂题解，先咕了。

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

Permanent link:
https://wiki.cvbbacm.com/doku.php?id=2020-2021:teams:legal_string:%E7%BB%84%E9%98%9F%E8%AE%AD%E7%BB%83%E6%AF%94%E8%B5%9B%E8%AE%B0%E5%BD%95:contest6&rev=1627392521

Last update: 2021/07/27 21:28

