线性基

算法简介

一个用长度为 $\log v$ 的数组维护值域为 $[1,v]$ 的序列的异或和的数据结构。

插入一个值和查询第 k 小值的时间复杂度均为 $O(\log v)$

算法思想

线性基本质就是高斯消元，把需要维护的序列的所有数用二进制表示，得到一组 $\log v$ 维的向量组。

把正常的向量加法改成向量异或，进行高斯消元，得到一组极大无关组，将极大无关组的向量成为线性基。

线性基具有如下性质：

性质一：由于极大无关组可以表示向量组的所有向量，所以可以用极大无关组替代原来的向量组来维护异或和。

性质二：由极大无关组的性质知，用 k 个极大无关组中的向量可以组合出 $2^{k}-1$ 个不同的异或和(不包括 0)。

考虑一个问题：求第 k 小的异或和。

先将极大无关组从大到小排好(这里大小是指向量代表的二进制数的大小)，依次决定是否选择该数。

如果能保证当前面的选择相同时，选择该数一定比不选择该数的异或和大。

再利用性质二，可以很容易得出第 k 小异或和。

只要再将极大无关组的矩阵转化为行最简矩阵，便可以满足上述条件。

实现细节稍有不同，具体见代码。

代码模板

洛谷p3812

```
namespace LB{//异或和从小到大排名
    const int MAXL=63;
    LL p1[MAXL],p2[MAXL],p3[MAXL];
    int rk;
    void insert(LL x){
        for(int i=MAXL-1;i>=0;i--){
            if((x>>i)&1){
                if(p1[i])
                x^=p1[i];
                else
```

```
                return p1[i]=x,void();
            }
        }
    }
    LL query(LL x){
        for(int i=MAXL-1;i>=0;i--){
            if((x^p1[i])>x)
            x^=p1[i];
        }
        return x;
    }
    void rebuild(){
        for(int i=MAXL-1;i>=0;i--){
            for(int j=i-1;j>=0;j--)
            if((p1[i]>>j)&1)
            p1[i]^=p1[j];
        }
        _for(i,0,MAXL)
        if(p1[i])
        p2[rk]=p1[i],p3[rk++]=i;
    }
    LL kth(LL k){//第k小
        if(k>=(1LL<<rk)) return -1;
        LL ans=0;
        for(int i=MAXL-1;i>=0;i--){
            if((k>>i)&1)
            ans^=p2[i];
        }
        return ans;
    }
    LL rank(LL x){//查询x的排名,必须保证存在异或和等于x
        LL ans=0;
        _for(i,0,rk)
        if((x>>p3[i])&1)
        ans+=1LL<<i;
        return ans;
    }
};
```

代码练习

练习题一

洛谷p4301

题意

一开始 k 堆火柴，每堆火柴中有若干根火柴，两人轮流取火柴。

每人第一次取火柴时都可以取若干堆火柴，也可以不取，但不能把所有火柴都拿走。

接下来每人每次只能取一堆火柴中的若干根火柴，但不能不取。取走最后一根火柴的人获胜。

问先手一开始至少要取多少根火柴才能保证必胜。

题解

~~没学过 Nim 游戏先学子再来看这题。~~

只需要保证第一回合后手者取过火柴后火柴堆异或和非零即可。

这等价于先手者第一回合取过火柴后剩下的火柴堆数值代表的二进制向量构成线性无关组。

一开始取最少的火柴即要求线性无关组数值和最大。

事实上线性无关组满足交换性质和遗传性质，所以可以直接套拟阵模型。

```
const int MAXN=31,MAXK=105;
int p1[MAXN];
bool Insert(int x){
    for(int i=MAXN-1;i>=0;i--){
        if(x&(1<<i)){
            if(p1[i])
            x^=p1[i];
            else
            return p1[i]=x,true;
        }
    }
    return false;
}
int a[MAXK];
int main()
{
    int n=read_int();
    _for(i,0,n)
    a[i]=read_int();
    sort(a,a+n,greater<int>());
    LL ans=0;
    _for(i,0,n){
        if(!Insert(a[i]))
        ans+=a[i];
    }
    enter(ans);
    return 0;
```

```
}
```

练习题二

洛谷p3292

题意

一棵点权树q 次询问，每次询问从结点 u 到结点 v 的路径上选取若干点得到的最大异或和为多少。

题解

求最大异或和显然是用线性基，考虑合并两条路径信息只需要合并两个线性基，时间复杂度 $O\left(\log^2 v\right)$其中 v 为点权上限。

一种方法为利用倍增求 LCA 的方法暴力合并路径信息，时间复杂度 $O\left((n+q)\log n \log^2 v\right)$

另一种比较好的方法为点分治，时间复杂度$O\left((q+n\log v)\log n + q\log^2 v\right)$

```
const int MAXN=2e4+5,MAXQ=2e5+5,MAXM=63;
struct LinearBase{
    LL p[MAXM];
    void Clear(){mem(p,0);}
    void Insert(LL x){
        for(int i=MAXM-1;i>=0;i--){
            if((x>>i)&1){
                if(p[i])
                x^=p[i];
                else
                return p[i]=x,void();
            }
        }
    }
    LL query(){
        LL ans=0LL;
        for(int i=MAXM-1;i>=0;i--){
            if((ans^p[i])>ans)
            ans^=p[i];
        }
        return ans;
    }
};
LinearBase Merge(const LinearBase &a,const LinearBase &b){
    LinearBase c=a;
    _for(i,0,MAXM)
```

```
    if(b.p[i])
    c.Insert(b.p[i]);
    return c;
}
struct Edge{
    int to,next;
}edge[MAXN<<1];
int head[MAXN],edge_cnt;
void Insert(int u,int v){
    edge[++edge_cnt].next=head[u];
    edge[edge_cnt].to=v;
    head[u]=edge_cnt;
}
int sz[MAXN],mson[MAXN],tot_sz,root,root_sz;
LL a[MAXN],ans[MAXQ];
bool vis[MAXN],mark[MAXN];
LinearBase p[MAXN];
vector<pair<int,int> > q[MAXN];
vector<int> d,sd;
void find_root(int u,int fa){
    sz[u]=1;mson[u]=0;
    for(int i=head[u];i;i=edge[i].next){
        int v=edge[i].to;
        if(vis[v]||v==fa)
        continue;
        find_root(v,u);
        sz[u]+=sz[v];
        mson[u]=max(mson[u],sz[v]);
    }
    mson[u]=max(mson[u],tot_sz-sz[u]);
    if(mson[u]<root_sz){
        root=u;
        root_sz=mson[u];
    }
}
void dfs(int u,int fa){
    d.push_back(u);
    p[u]=p[fa];
    p[u].Insert(a[u]);
    _for(i,0,q[u].size()){
        if(!mark[q[u][i].first])
        continue;
        LinearBase t=Merge(p[u],p[q[u][i].first]);
        t.Insert(a[root]);
        ans[q[u][i].second]=t.query();
    }
    for(int i=head[u];i;i=edge[i].next){
        int v=edge[i].to;
        if(vis[v]||v==fa)
        continue;
        dfs(v,u);
```

```
    }
}
void query(int u){
    sd.clear();
    mark[u]=true;
    p[u].Clear();
    for(int i=head[u];i;i=edge[i].next){
        int v=edge[i].to;
        if(vis[v])
        continue;
        d.clear();
        dfs(v,u);
        _for(i,0,d.size()){
            mark[d[i]]=true;
            sd.push_back(d[i]);
        }
    }
    mark[u]=false;
    _for(i,0,sd.size())
    mark[sd[i]]=false;
}
void solve(int u){
    int cur_sz=tot_sz;
    vis[u]=true;query(u);
    for(int i=head[u];i;i=edge[i].next){
        int v=edge[i].to;
        if(vis[v])
        continue;
        tot_sz=sz[v]>sz[u]?cur_sz-sz[u]:sz[v];root_sz=MAXN;
        find_root(v,u);
        solve(root);
    }
}
int main()
{
    int n=read_int(),t=read_int(),u,v;
    _for(i,0,n)
    a[i]=read_LL();
    _for(i,1,n){
        u=read_int()-1,v=read_int()-1;
        Insert(u,v);
        Insert(v,u);
    }
    _for(i,0,t){
        u=read_int()-1,v=read_int()-1;
        if(u==v)
        ans[i]=a[u];
        else{
            q[u].push_back(make_pair(v,i));
            q[v].push_back(make_pair(u,i));
```

```
        }
    }
    root_sz=MAXN;
    tot_sz=n;
    find_root(0,-1);
    solve(root);
    _for(i,0,t)
    enter(ans[i]);
    return 0;
}
```

练习题三

洛谷p4869

题意

给定一个正整数序列 A，将它的所有子集(包括空集)的异或和结果排序，得到序列 B。

问某个数在序列 B 中第一次出现的下标是多少？(保证该数在序列中一定存在)

题解

给出线性基性质三：设线性基秩为 k，则线性基所能表示的异或和在序列 B 中的出现次数均为 $2^{|\text{A}|-k}$。

证明：不妨设构成线性基的数的集合为 P，数集 $\text{C}\subset \text{A}-\text{P}$。

对任意 $v\in \text{B}$，知 v 和数集 C 全体数的异或和一定能表示用数集 P 的子集的异或和表示。

即 v 可以被数集 C 和数集 P 的某个子集的全体数的异或和表示。

由于数集 C 没有约束条件，故数集 C 的选择有 $2^{|\text{A}|-k}$种。

即对每个 $v\in \text{B}$，至少可以被 $2^{|\text{A}|-k}$个子集可以表示。

又因为 $|\text{B}|\ast 2^{|\text{A}|-k} = 2^{|\text{A}|}$，说明对每个 $v\in \text{B}$，恰好可以被 $2^{|\text{A}|-k}$个子集可以表示。

性质三证毕。

有了性质三，这题就没什么难度了，直接上代码。

```
const int MAXL=63;
LL quick_pow(LL a,LL b,LL mod){
    LL t=1;
    while(b){
```

```
        if(b&1)
        t=t*a%mod;
        a=a*a%mod;
        b>>=1;
    }
    return t%mod;
}
struct LinearBase{
    LL p1[MAXL],p2[MAXL],p3[MAXL],rk;
    void Insert(LL x){
        for(int i=MAXL-1;i>=0;i--){
            if((x>>i)&1){
                if(p1[i])
                x^=p1[i];
                else
                return p1[i]=x,void();
            }
        }
    }
    void rebuild(){
        for(int i=MAXL-1;i>=0;i--){
            for(int j=i-1;j>=0;j--)
            if((p1[i]>>j)&1)
            p1[i]^=p1[j];
        }
        _for(i,0,MAXL)
        if(p1[i])
        p2[rk]=p1[i],p3[rk++]=i;
    }
    LL rank(LL x){
        LL ans=0;
        _for(i,0,rk)
        if((x>>p3[i])&1)
        ans+=1LL<<i;
        return ans;
    }
};
int main()
{
    int n=read_int();
    _for(i,0,n)
    lb.Insert(read_int());
    lb.rebuild();
    int q=read_int();
    enter((lb.rank(q)*quick_pow(2,n-lb.rk,10086)+1)%10086);
    return 0;
}
```

练习题四

洛谷p4151

题意

给定一个连通图，每条路径的权值为该路径所有边的权值的异或和(路径中同一条边重复出现将重复计算贡献)。

求结点 $1\sim n$ 的路径的最大异或和。注意图中可能存在重边和自环。

题解

考虑路径的情况一定是一条 $1\sim n$ 的简单路径(即一条边最多走一次)，再包含一些分支，每个分支经过一个环后返回。

发现每条分支经过两次，于是不计算贡献，所以只包含了环的贡献，而由于图连通所以所有环的贡献都可以被计入。

考虑 dfs 过程中记录当前路径前缀异或和以及访问标记，得到 dfs 树和若干返祖边。

有结论任意简单环可以由若干只含一条返祖边的环通过异或得到。

$O(n+m\log n)$ 得到所有只含一条返祖边的环的贡献，最后线性基即可得到答案。

最后发现一开始随意选择一条 $1\sim n$ 的路径即可，因为如果该路径不是最优，则可以通过与环异或修改为最优路径。

```
namespace LB{
    const int MAXN=63;
    LL p1[MAXN],p2[MAXN],rk;
    void insert(LL x){
        for(int i=MAXN-1;i>=0;i--){
            if(x&(1LL<<i)){
                if(p1[i])
                x^=p1[i];
                else
                return p1[i]=x,void();
            }
        }
    }
    LL query(LL x){
        for(int i=MAXN-1;i>=0;i--){
            if((x^p1[i])>x)
            x^=p1[i];
        }
        return x;
    }
}
```

```
const int MAXN=5e4+5,MAXM=1e5+5;
struct Edge{
    int to,next;
    LL w;
}edge[MAXM<<1];
int head[MAXN],edge_cnt,dfn[MAXN],dfs_t;
void Insert(int u,int v,LL w){
    edge[++edge_cnt]=Edge{v,head[u],w};
    head[u]=edge_cnt;
}
LL dfs_s[MAXN];
void dfs(int u,LL s){
    dfn[u]=++dfs_t;
    dfs_s[u]=s;
    for(int i=head[u];i;i=edge[i].next){
        int v=edge[i].to;
        if(!dfn[v])
        dfs(v,edge[i].w^s);
        else if(dfn[v]<=dfn[u])
        LB::insert(s^edge[i].w^dfs_s[v]);
    }
}
int main()
{
    int n=read_int(),m=read_int(),u,v;
    LL w;
    while(m--){
        u=read_int(),v=read_int(),w=read_LL();
        Insert(u,v,w);
        Insert(v,u,w);
    }
    dfs(1,0LL);
    enter(LB::query(dfs_s[n]));
    return 0;
}
```

练习题五

CF1299D

题意

给定一个边权图，要求删除一些与结点 1 相邻的边，使得不存在路径满足下述条件：

1. 以 1 号节点为起点，1 号节点为终点
2. 此路径经过的所有边的边权异或和为 0
3. 其至少经过了一条边奇数次

输出所有满足条件的方案数模 10^9+7 的结果。

保证图中所有边权 $w\in [0,31]$ 不存在重边和自环且不存在一个长度 $\gt 3$ 的简单环经过了 1 号点。

题解

考虑将点 1 删去，得到若干连通块，由于不存在一个长度 $\gt 3$ 的简单环经过了 1 号点，于是每个连通块仅可能有 $1\sim 2$ 个点与点 1 相邻。

考虑求出每个连通块的所有环的异或和构成的线性基，并且先不妨假设所有连通块都只有一条边与 1 点相连。

如果该线性基在建立过程中插入失败，说明部分环线性相关。则该连通块与 1 相邻后一定会出现非法路径，于是该连通块与 1 的连边必须删去。

对于剩下的连通块，只需要保证所有选择的连通块的线性基在相互合并时不会出现线性相关即可。

考虑对所有线性基的最简形式进行编号，同时预处理任意两个线性基合并得到的新线性基的编号。

设 $\text{dp}(i,j)$ 表示只考虑前 $i-1$ 个连通块时，选择部分连通块的线性基合并最后得到线性基编号为 j 的方案数。

同时记 id_i 表示第 i 个连通块的线性基，于是不难得出状态转移方程

$$\text{dp}(i,\text{Merge}(id_i,k))\gets \text{dp}(i-1,k)$$

如果该连通块与点 1 有 2 条边，则首先可以任意选择保留一条连通，于是可以产生双倍贡献。

另外如果保留两条连边，则记这两条连边的除 1 外的端点为 v_1,v_2 则考虑先选择 $1\to v_1\to v_2\to 1$ 然后再异或上一些环。

于是可以得到新的线性基为原连通块的线性基异或上这三条边，然后考虑新线性基对 dp 的贡献。

最后发现最简线性基共 $k=374$ 种，于是总时间复杂度 $O(kn+k^2+32k\log 32)$

```
const int MAXN=1e5+5,MAXM=400,MAXV=1<<15,MAXL=5,Mod=1e9+7;
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;
}
struct LB{
    int p[MAXL];
    bool insert(int x){
        for(int i=MAXL-1;i>=0;i--){
            if((x>>i)&1){
                if(p[i])
                x^=p[i];
                else{
```

```
                    p[i]=x;
                    _for(j,0,i){
                        if(p[j]&&((p[i]>>j)&1))
                        p[i]^=p[j];
                    }
                    _for(j,i+1,MAXL){
                        if((p[j]>>i)&1)
                        p[j]^=p[i];
                    }
                    return true;
                }
            }
        }
        return false;
    }
    int query(int x){
        for(int i=MAXL-1;i>=0;i--){
            if((x^p[i])>x)
            x^=p[i];
        }
        return x;
    }
    int encode(){
        return p[0]|(p[1]<<1)|(p[2]<<3)|(p[3]<<6)|(p[4]<<10);
    }
    LB(){mem(p,0);}
}lb[MAXM];
int Hash[MAXV],Merge[MAXM][MAXM],lb_cnt;
void dfs(LB u){
    if(!Hash[u.encode()]){
        lb[++lb_cnt]=u;
        Hash[u.encode()]=lb_cnt;
        _for(i,1,32){
            LB temp=u;
            if(temp.insert(i))
            dfs(temp);
        }
    }
}
void Pre_merge(){
    Merge[1][1]=1;
    _rep(i,1,lb_cnt)
    _rep(j,i+1,lb_cnt){
        LB temp=lb[i];
        bool flag=true;
        _for(k,0,MAXL){
            if(lb[j].p[k]&&!temp.insert(lb[j].p[k])){
                flag=false;
                break;
            }
```

```
        }
        if(flag)
        Merge[i][j]=Merge[j][i]=Hash[temp.encode()];
    }
}
int block_root[MAXN],block_id[MAXN],cyc_s[MAXN],block_cnt;
bool fail[MAXN],is_cyc[MAXN];
LB block_lb[MAXN];
int dfn[MAXN],dfs_t,dis[MAXN];
int dp[2][MAXM];
void dfs2(int u,int fa,int s){
    dfn[u]=++dfs_t,dis[u]=s;
    block_id[u]=block_cnt;
    for(int i=head[u];i;i=edge[i].next){
        int v=edge[i].to;
        if(v==1||v==fa)continue;
        if(!dfn[v])
        dfs2(v,u,s^edge[i].w);
        else if(dfn[v]<dfn[u])
        fail[block_cnt]|=!block_lb[block_cnt].insert(s^dis[v]^edge[i].w);
    }
}
int main()
{
    dfs(LB());
    Pre_merge();
    int n=read_int(),m=read_int();
    while(m--){
        int u=read_int(),v=read_int(),w=read_int();
        Insert(u,v,w);
        Insert(v,u,w);
    }
    for(int i=head[1];i;i=edge[i].next){
        int v=edge[i].to;
        if(!block_id[v]){
            block_root[++block_cnt]=v;
            dfs2(v,0,edge[i].w);
        }
        else{
            int idx=block_id[v];
            is_cyc[idx]=true;
            for(int j=head[v];j;j=edge[j].next){
                int vv=edge[j].to;
                if(vv==block_root[idx]){
                    cyc_s[idx]=dis[vv]^edge[j].w^edge[i].w;
                    break;
                }
            }
        }
    }
    int pos=0;
```

```
    dp[pos][1]=1;
    _rep(i,1,block_cnt){
        if(fail[i])continue;
        int h=Hash[block_lb[i].encode()];
        pos=!pos;
        _rep(j,1,lb_cnt)
        dp[pos][j]=dp[!pos][j];
        if(!is_cyc[i]){
            _rep(j,1,lb_cnt){
                if(Merge[j][h])
dp[pos][Merge[j][h]]=(dp[pos][Merge[j][h]]+dp[!pos][j])%Mod;
            }
        }
        else{
            bool flag=block_lb[i].insert(cyc_s[i]);
            int h2=Hash[block_lb[i].encode()];
            _rep(j,1,lb_cnt){
                if(Merge[j][h])
dp[pos][Merge[j][h]]=(dp[pos][Merge[j][h]]+2LL*dp[!pos][j])%Mod;
                if(flag&&Merge[j][h2])
dp[pos][Merge[j][h2]]=(dp[pos][Merge[j][h2]]+dp[!pos][j])%Mod;
            }
        }
    }
    int s=0;
    _rep(i,1,lb_cnt)
    s=(s+dp[pos][i])%Mod;
    enter(s);
    return 0;
}
```

练习题六

HDU6578

题意

给定一个长度为 n 的序列，支持下述操作：

1. 询问区间 $[l,r]$ 的所有子序列的最大异或和
2. 在序列末尾添加一个数

强制在线。

题解

考虑对每个 r 贪心维护区间 $[1,r]$ 的线性基每个位的数值和代表元位置。

具体来说，对每个新插入的数，从高位到低位考虑是否可以将该值插入。

如果该位置没有数则直接插入，否则如果该位置的代表元位置相对于插入数的位置偏左，则将该位置的代表元置换出来。

对区间 $[l,r]$ 考虑 r 维护的线性基当前位的代表元位置是否不小于 l 即可。时空间复杂度 $O((n+q)\log v)$

```
const int MAXN=5e5+5;
namespace LB{
    const int MAXL=31;
    int p1[MAXL],p2[MAXN][MAXL];
    int pos1[MAXL],pos2[MAXN][MAXL];
    void clear(){
        mem(p1,0);mem(p2,0);
        mem(pos1,0);mem(pos2,0);
    }
    void insert(int x,int pos){
        int r=pos;
        for(int i=MAXL-1;i>=0;i--){
            if((x>>i)&1){
                if(!p1[i]){
                    p1[i]=x;
                    pos1[i]=pos;
                    break;
                }
                else if(pos1[i]<pos){
                    swap(p1[i],x);
                    swap(pos1[i],pos);
                }
                x^=p1[i];
            }
        }
        _for(i,0,MAXL){
            p2[r][i]=p1[i];
            pos2[r][i]=pos1[i];
        }
    }
    LL query(int l,int r){
        int ans=0;
        for(int i=MAXL-1;i>=0;i--){
            if(pos2[r][i]>=l&&(ans^p2[r][i])>ans)
            ans^=p2[r][i];
        }
        return ans;
    }
};
int main()
{
```

```
    int T=read_int();
    while(T--){
        int n=read_int(),q=read_int(),lastans=0;
        LB::clear();
        _rep(i,1,n)
        LB::insert(read_int(),i);
        while(q--){
            int opt=read_int();
            if(!opt){
                int l=(read_int()^lastans)%n+1,r=(read_int()^lastans)%n+1;
                if(l>r)swap(l,r);
                enter(lastans=LB::query(l,r));
            }
            else
            LB::insert(read_int()^lastans,++n);
        }
    }
    return 0;
}
```

练习题七

2019牛客暑期多校训练营（第四场□B

题意

给定 n 个集合和 q 个询问，每个集合给定 sz_i 个元素。

每次询问对区间 $[l,r]$ 的每个集合是否都存在子集异或和为 x□

题解

考虑线段树求解，于是问题转化为如何维护两个线性基的交。

线性空间 v_1,v_2 的交 v_3 仍为线性空间，仍然可以用线性基表示，且满足 v_3 的线性基可以用 v_1,v_2 的线性基表示。

不妨记 v_1 的线性基为 a□记 v_2 的线性基为 b□记 v_3 的线性基为 c□

考虑初始空间为 v_1□然后依次将 v_2 的线性基插入，如果插入出现线性相关，则有

$$ a_{x_1}\oplus a_{x_2}\oplus\cdots \oplus a_{x_{t_1}}\oplus b_{y_1}\oplus b_{y_2}\oplus\cdots \oplus b_{y_{t_2}}=b_i $$

移项，得

$$ c=a_{x_1}\oplus a_{x_2}\oplus\cdots \oplus a_{x_{t_1}}=b_{y_1}\oplus b_{y_2}\oplus\cdots

\oplus b_{y_{t_2}}\oplus b_i $$

于是将新得到的 c 插入 v_3 即可。注意在插入 v_2 过程中维护线性基每个位的 v_1 空间的贡献。时间复杂度 $O((n+q)\log n\log v)$

```
const int MAXN=5e5+5,MAXL=32;
struct LB{
    LL p[MAXL];
    void insert(LL x){
        for(int i=MAXL-1;i>=0;i--){
            if((x>>i)&1){
                if(p[i])
                x^=p[i];
                else
                return p[i]=x,void();
            }
        }
    }
    bool query(LL x){
        for(int i=MAXL-1;i>=0;i--){
            if((x>>i)&1){
                if(p[i])
                x^=p[i];
                else
                return false;
            }
        }
        return true;
    }
}a[MAXN],s[MAXN<<2];
int lef[MAXN<<2],rig[MAXN<<2];
void push_up(int k){
    LB t=s[k<<1],v1=s[k<<1],v2=s[k<<1|1];
    _for(i,0,MAXL){
        if(v2.p[i]){
            LL x=v2.p[i],y=0;
            bool flag=true;
            for(int j=MAXL-1;j>=0;j--){
                if((x>>j)&1){
                    if(t.p[j]){
                        x^=t.p[j];
                        y^=v1.p[j];
                    }
                    else{
                        t.p[j]=x;
                        v1.p[j]=y;
                        flag=false;
                        break;
                    }
                }
```

```
            }
            if(flag)
            s[k].insert(y);
        }
    }
}
void build(int k,int L,int R){
    lef[k]=L,rig[k]=R;
    int M=L+R>>1;
    if(L==R)
    return s[k]=a[M],void();
    build(k<<1,L,M);
    build(k<<1|1,M+1,R);
    push_up(k);
}
bool query(int k,int L,int R,LL x){
    if(L>rig[k]||R<lef[k])
    return true;
    if(L<=lef[k]&&rig[k]<=R)
    return s[k].query(x);
    int M=L+R>>1;
    return query(k<<1,L,R,x)&&query(k<<1|1,L,R,x);
}
int main()
{
    int n=read_int(),q=read_int();
    _rep(i,1,n){
        int cnt=read_int();
        while(cnt--)
        a[i].insert(read_LL());
    }
    build(1,1,n);
    while(q--){
        int l=read_int(),r=read_int();
        if(query(1,l,r,read_LL()))
        puts("YES");
        else
        puts("NO");
    }
    return 0;
}
```