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

```
#include <cstdio>
#include <iostream>
#include <vector>
#include <algorithm>
#include <cstring>
#include <cctype>
#include <queue>
#include <cmath>
#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 LL read_LL(){
    LL 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=63;
LL p1[MAXN],p2[MAXN],rk;
void Insert(LL x){//没有用高斯消元，偷了个懒
    for(int i=MAXN-1;i>=0;i--){
        if(x&(1LL<<i)){
            if(p1[i])
            x^=p1[i];
            else
            return p1[i]=x,void();
        }
    }
}
void rebuild(){//转化为行最简矩阵
    for(int i=MAXN-1;i>=0;i--){
        for(int j=i-1;j>=0;j--)
        if(p1[i]&(1LL<<j))
        p1[i]^=p1[j];
    }
    _for(i,0,MAXN)
    if(p1[i])
    p2[rk++]=p1[i];
}
LL kth(int k){//第k小
    if(k>=(1LL<<rk)) return -1;
    LL ans=0;
    for(int i=MAXN-1;i>=0;i--){
        if(k&(1LL<<i))
```

```
        ans^=p2[i];
    }
    return ans;
}
int main()
{
    int n=read_int();
    _for(i,0,n)
    Insert(read_LL());
    rebuild();
    enter(kth((1LL<<rk)-1));
    return 0;
}
```

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 游戏先学了再来看这题。~~

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。

```
#include <cstdio>
#include <iostream>
#include <vector>
#include <algorithm>
#include <cstring>
```

```
#include <cctype>
#include <queue>
#include <cmath>
#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 LL read_LL(){
    LL 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=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;
}
```

练习题一

洛谷p4301

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$O\left((q+n\log v)\log n + q\log^2 v\right)$

```
#include <cstdio>
#include <iostream>
#include <vector>
#include <algorithm>
#include <cstring>
#include <cctype>
#include <queue>
#include <cmath>
#define _for(i,a,b) for(int i=(a);i<(b);++i)
#define _rep(i,a,b) for(int i=(a);i<=(b);++i)
#define mem(a,b) memset((a),(b),sizeof(a))
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 LL read_LL(){
    LL 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=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;
}
```