线性基

定义：

张成：

- $T \subseteq S$所有这样是子集T的异或和组成的集合称为S的**张成**记作$span(S)$

线性相关：

- 存在一个元素S_j可以用其它若干元素异或起来得到
- 或：存在一个子集的异或和为0

线性基：

称B是S的线性基当且仅当：

1. $S \subseteq span(B)$即S是B的张成的子集。
2. B是线性无关的。

线性基的长度：

- 集合B中元素的个数

线性基的基本性质：

1. B是极小的满足线性基性质的集合，它的任何真子集都不可能是线性基；
2. S中的任意元素都可以唯一表示为B中若干个元素异或起来的结果。

构造：

线性基是动态构造的，我们只需要从空的a开始，每次考虑在一个已存在的线性基中插入一个数t即可。

构造流程：

对于每个数x从高位到低位扫描，扫到第i位为1时，若a_i上有数字$x=x \oplus a_i$并向下继续扫，若没有数字$a_i=x$结束扫描。

```
inline void insert(ll x){
    for(int i=50;i>=0;i--)if(x&(1ll<<i)){
        if(a[i]) x^=a[i];
        else{a[i]=x;break ;}
    }
```

```
}
```

大致证明：

未成功插入线性基：

考虑时候不能插入进去呢，显然就是它在尝试插入时异或若干个数之后变成了0 $\Rightarrow$ 新插入的数x可由当前的线性基异或表示 $\Rightarrow$ 所有数都能由当前的线性基异或表示。

成功插入线性基:

假设x成功插入到了a_i显然，它在插入前可能异或若干个数，那么就有：
$x \oplus a_j \oplus a_k \oplus \ldots = a_i$
即：$a_i \oplus a_j \oplus a_k \oplus \ldots = x$
$\Rightarrow$ 新插入的数x可由当前的线性基异或表示 $\Rightarrow$ 所有数都能由当前的线性基异或表示。

性质：

构造出来的线性基满足：

1. 若$a_i=0$
    - **只有**满足$j>i$的a_j的第i位才**可能**为1
2. 若$a_i\ne0$
    - 整个a数组只有a_i的第i位二进制位1
    - a_i的更高位($>i$的位数)**一定**为0
    - a_i的更低位($<i$的位数)**可能**为1

合并：

将一个线性基中的所有元素插入到另一个线性基中。（线性基的个数最多为二进制的位数）

删除：

若x在线性基外：直接删掉 **若x在线性基内：**
没有在线性基中的数，一定是因为线性基中的若干个数可以异或得到他，那么可以记录一下不在线性基中的数都是由线性基中的哪些数异或得到的，那么每一个线性基外的数对应一个集合S这个集合内就是线性基中那些异或起来可以得到他的数。
假如线性基外的某一个数的S中包含x那么就不需要删除x把这个数删除即可。原因是这个数在线性基中是可以代替x的，那么就当这个数代替了x然后x被删除了，然后把线性基中的x当做这个数即可，这样的话线性基不会有变化。（实现起来并不需要维护集合S而是直接维护有哪些数可以代替线性基中的数就好了）
假如x不被线性基外的任何一个数的S包含，那么就另外造一个集合P记录线性基中这个数插入进

来的时候异或过哪些数。然后找到线性基中最小的并且P 包含x的数，让他异或线性基中其他包含x的数即可（包括自己，自己异或完自己后自己这一位就变成了 0），这样就能消除x在线性基中的影响（事实上就等价于用最小的数代替了它）。
总之，由于如果修改了线性基中的某一位会影响到一些比它小的位，所以一般不能修改，要么改最小的并且不会影响到下面的位。

应用：

查询x是否在集合的张成中：

对数字x从高位到地位扫描，若该位为1，$x=x \oplus a_i$
若x最后异或结果为0则x在当前集合的张成中，否则不在

```
inline bool check(ll x){
    for(int i=50;i>=0;i--)
        if(x&(1ll<<i)) x^=a[i];
    return x==0;
}
```

求数集能异或出的最大值：

使用贪心的思想:

```
inline ll getMax(){
    ll ans=0;
    for(int i=50;i>=0;i--)
        ans=max(ans,ans^a[i]);
    return ans;
}
```

证明：
a_i的第i位一定是1，而后面扫描到的$a_{i-1},a_{i-2},\ldots,a_0$第$i$位一定不是1，而如果高位不是1低位再多1也弥补不了，所以当前只需要考虑第i位的情况，若ans的第i位为0异或后为1会变大就进行异或，若为1异或后为0会变小就不进行异或。

求数集能异或出的最小值：

显然是最小的一个a_i最小的%a_i%异或其它数字都会变大。

求能异或出的第k小值：

从一个序列中取任意个元素进行异或，求能异或出的所有数字中第k小的那个。
首先对线性基进行改造，保证只有a_i的第i位是1，其它的第i位都为0。
改造方法是若a_i的第j位为1$(j<i)$ $a_i = a_i \oplus a_j$
求解过程：将k转化为二进制，如果k的第i位为1，ans异或上线性基中的第i个元素。（不是直接异或a_{i-1}，见代码）

```
void init(){//处理线性基
    for(int i=1;i<=50;i++)
        for(int j=0;j<i;j++)
            if(a[i]&(1ll<<j))
                a[i]^=a[j];
}
ll k_th(ll k){
    if(k==1&&tot<n)
        return 0;
    /*特判一下，假如k=1时
  并且原来的序列可以异或出0,
  就要返回0，tot表示线性基中的元素个数,
    n表示序列长度*/
    if(tot<n) k--;
    //类似上面，去掉0的情况，因为线性基中只能异或出不为0的解
    init();
    ll ans=0;
    for(int i=0;i<=50;i++)if(a[i]){
        if(k&1) ans^=a[i];
        k>>=1;
    }
    return ans;
}
```

证明（或者说是理解）：其实处理完之后的线性基其实也还是原序列的一个线性基，回想上面的线性基处理过程，可以发现，处理完之后，线性基中的元素，作用其实都是提供自己最高位上的 1，那么只要使提供出来的 1 可以和 k的二进制的每一位上的 1对应，那么求出的ans一定就是第k小的。

题目：

洛谷4570 [BJWC2011]元素

题意：给定一些矿石编号和矿石价值，问在保证矿石编号的任意子集的异或和不为0的情况下，最大价值是多少。
题解：很显然的贪心，按价值从大到小排序维护线性基。

```
#include<iostream>
#include<cstdio>
#include<cstring>
#include<algorithm>
#include<set>
#include<queue>
#include<vector>
#include<cmath>
#include<cstdlib>
#include<map>
#include<stack>
```

```
using namespace std;
typedef long long ll;
typedef pair<int,ll> pi;
const int N=1005;
ll getint(){
    char c;bool flag=0;ll num=0ll;
    while((c=getchar())<'0'||c>'9')if(c=='-')flag=1;
    while(c>='0'&&c<='9'){num=num*10+c-48;c=getchar();}
    if(flag) num=-num;
    return num;
}
int n,ans;pi x[N];ll a[65];
inline bool insert(ll x){
    for(int i=60;i>=0;i--)if(x&(1ll<<i)){
        if(a[i]) x^=a[i];
        else{a[i]=x;break ;}
    }
    return x;
}
int main(){
    n=getint();
    for(int i=1;i<=n;i++){
        x[i].second=getint();
        x[i].first=-getint();
    }
    sort(x+1,x+n+1);
    for(int i=1;i<=n;i++)
        if(insert(x[i].second))
            ans-=x[i].first;
    cout<<ans<<endl;
}
```

洛谷3292 [SCOI2016]幸运数字

题意：一棵树每个点上有权值，多次查询不同路径x→y的最大幸运值。幸运值：从x→y路径上的点选任意子集的异或和。
题解：倍增lca+线性基

```
#include<iostream>
#include<cstdio>
#include<cstring>
#include<algorithm>
#include<set>
#include<queue>
#include<vector>
#include<cmath>
#include<cstdlib>
```

```
#include<map>
#include<stack>
using namespace std;
typedef long long ll;
typedef pair<int,ll> pi;
const int N=20005;
ll getint(){
    char c;bool flag=0;ll num=0ll;
    while((c=getchar())<'0'||c>'9')if(c=='-')flag=1;
    while(c>='0'&&c<='9'){num=num*10+c-48;c=getchar();}
    if(flag) num=-num;
    return num;
}
int n,q,fa[N],dep[N];
ll luck[N],A[N][17][63],f[N][17],a[63];
int cnt,fir[N],tar[N<<1],nxt[N<<1];
inline void link(int a,int b){
    tar[++cnt]=b,nxt[cnt]=fir[a],fir[a]=cnt;
}
void insert(ll*a,ll x){
    for(int i=60;i>=0;i--)if(x&(1ll<<i)){
        if(a[i]) x^=a[i];
        else{a[i]=x;break ;}
    }
}
void merge(ll*x,ll*y){
    for(int i=0;i<=60;i++)
        if(y[i]) insert(x,y[i]);
}
inline void init(){
    for(int i=2;i<=n;i++){
        insert(A[i][0],luck[i]);
        insert(A[i][0],luck[f[i][0]]);
    }
    for(int i=1;i<=15;i++)
        for(int j=1;j<=n;j++){
            f[j][i]=f[f[j][i-1]][i-1];
            merge(A[j][i],A[j][i-1]);
            merge(A[j][i],A[f[j][i-1]][i-1]);
        }
}
void dfs(int x,int fa){
    for(int i=fir[x];i;i=nxt[i])
        if(tar[i]!=fa){
            f[tar[i]][0]=x;
            dep[tar[i]]=dep[x]+1;
            dfs(tar[i],x);
        }
}
inline ll getMax(){
```

```
    ll ans=0;
    for(int i=60;i>=0;i--)
        ans=max(ans,ans^a[i]);
    return ans;
}
int getk(int x,int k){
    if(!k) return x;
    for(int i=15;i>=0;i--)if(k&(1<<i))
        merge(a,A[x][i]),x=f[x][i];
    return x;
}
int getd(int x,int d){
    return getk(x,dep[x]-d);
}
inline ll query(int x,int y){
    memset(a,0,sizeof a);
    insert(a,luck[x]);
    insert(a,luck[y]);
    if(dep[x]!=dep[y]){
        int md=min(dep[x],dep[y]);
        x=getd(x,md),y=getd(y,md);
    }
    if(x==y) return getMax();
    for(int i=15;i>=0;i--)
        if(f[x][i]!=f[y][i]){
            merge(a,A[x][i]),x=f[x][i];
            merge(a,A[y][i]),y=f[y][i];
        }
    insert(a,luck[f[x][0]]);
    return getMax();
}
int main(){
    n=getint(),q=getint();
    for(int i=1;i<=n;i++)
        luck[i]=getint();
    for(int i=1;i<n;i++){
        int x=getint(),y=getint();
        link(x,y),link(y,x);
    }
    dfs(1,0);
    init();
    while(q--)
        cout<<query(getint(),getint())<<endl;
}
```

洛谷3857 [TJOI2008]彩灯

题意：每个开关可以控制一部分的灯，改变它们的开关状态，计算灯的状态有多少种可能。（对2008取模）
灯的数量和开关的数量都小于50

题解：设线性基的长度为x答案为2^x

```
#include<iostream>
#include<cstdio>
#include<cstring>
#include<algorithm>
#include<set>
#include<queue>
#include<vector>
#include<cmath>
#include<cstdlib>
#include<map>
#include<stack>
using namespace std;
typedef long long ll;
const int N=20005;
int n,m;ll a[55];char str[55];
inline void insert(ll x){
    for(int i=50;i>=0;i--)if(x&(1ll<<i)){
        if(a[i]) x^=a[i];
        else{a[i]=x;break ;}
    }
}
inline int qpow(int x,int p){
    int ans=1;
    while(p){
        if(p&1) ans=ans*x%2008;
        x=x*x%2008,p>>=1;
    }
    return ans;
}
int main(){
    scanf("%d%d",&n,&m);
    for(int i=1;i<=m;i++){
        scanf("%s",str+1);
        ll x=0ll;
        for(int j=1;j<=n;j++){
            x<<=1;
            if(str[j]=='0') x|=1;
        }
        insert(x);
    }
    int cnt=0;
    for(int i=0;i<=50;i++)
        if(a[i]) cnt++;
    printf("%d\n",qpow(2,cnt));
}
```

洛谷4151 [WC2011]最大XOR和路径

题意：给一个无向连通图，求一条从1到n的路径（可以不是简单路径），使经过的边权的异或和最大。
题解：首先注意到一个结论：对于所有的简单环，环上边权的异或和都可以无代价的获取。原因是可以从一号点出发进入该环绕一圈后原路返回。由于一条路径绕两边对答案的贡献为 0，所以这些简单环的异或和都可以无代价取得。那么现在问题就转化成了寻找一条 1到 n的路径，再异或上一些简单环的异或和，最大化答案。
我们考虑应该寻找哪一条路径：事实上任选一条路径即可。原因是选择的路径和答案路径一定可以构成一个环，所以异或上该环的权值就可以得到最优解。