快速幂

```
long long pow(long long a, long long b) {
    long long res = 1;
    while(b) {
        if(b & 1) res = res * a % MOD;
        a = a * a % MOD;
        b >>= 1;
    }
    return res;
}
```

龟速乘

```
ll mul(ll a, ll b, ll mod)
{
```

```
    ll res = 0;
    while(b > 0)
    {
        if(b & 1) res = (res + a) % mod;
        a = (a + a) % mod;
        b >>= 1;
    }
    return res;
}
```

高斯消元

```
double a[105][106];

bool gauss(int n){
    //枚举列
    for(int i = 1; i <= n; ++i){
        int id = i;
        //找到该列中的最大系数
        for(int j = i; j <= n; ++j){
            id = fabs(a[j][i]) > fabs(a[id][i]) ? j : id;
        }
        // 交换
        for(int j = 1; j <= n + 1; ++j){
            swap(a[i][j], a[id][j]);
        }
        // 系数为零说明没有唯一解
        if(!a[i][i]){
            return false;
        }
        // 消去对应列的其他元
        for(int j = 1; j <= n; ++j){
            if(j != i){
                double temp = a[j][i] / a[i][i];
                for(int k = i + 1; k <= n + 1; ++k){
                    a[j][k] -= a[i][k] * temp;
                }
            }
        }
    }

    return true;
}
```

埃氏筛素数

```
bool is_prime[N + 1]; // default true
is_prime[0] = is_prime[1] = false;
for(int i = 2; i <= N; ++i) {
    if(is_prime[i]) {
        for(int j = i << 1; j <= N; j += i) {
            is_prime[j] = false;
        }
    }
}
```

线性筛素数

```
vector<int>p;
vector<bool>prime(N, true);

void Euler_sieve(){
    for(int i = 2; i < N; ++i){
        if(prime[i])p.push_back(i);
        for(int j = 0; j < p.size() && i * p[j] < N; ++j){
            prime[i * p[j]] = false;
            if(i % p[j] == 0)break;
            // i % p[j] == 0
            // 换言之，i 之前被 p[j] 筛过了
            // 由于 pri 里面质数是从小到大的，所以 i 乘上其他的质数的结果一定会被
            // p[j] 的倍数筛掉，就不需要在这里先筛一次，所以这里直接 break
            // 掉就好了
        }
    }
}
```

gcd/exgcd

扩展欧几里得用于解决$ax + by = \gcd(a, b)$的一组可行解

```
int gcd(int a, int b) {
    return b ? gcd(b, a % b) : a;
}

int lcm(int a, int b) {
    return a / gcd(a, b) * b;
}

ll Exgcd(ll a, ll b, ll &x, ll &y){
    if(!b){
        x = 1, y = 0;
```

```
        return a;
    }
    ll d = Exgcd(b, a % b, x, y);
    ll t = x;
    x = y;
    y = t - (a / b);
    return d;
}
```

逆元

费马小定理

要求模数是质数。

☑ = pow(x, MOD - 2);

扩欧

相当于求线性同余方程

$$
\begin{aligned}
\mathrm{inv} &\equiv \frac{1}{x} \pmod{M} \\
\mathrm{inv}x &\equiv 1 \pmod{M} \\
\mathrm{inv}x + kM &\equiv 1 \pmod{M}
\end{aligned}
$$

`std::tie(gcd, inv[x], _) = exgcd(x, MOD);`

若 gcd 不为 1 ，则逆元不存在。

线性求逆元

☑ = (MOD - MOD / x) * inv[MOD % x] % MOD;

口诀：减除乘模

组合数

$$\binom{n}{m} = \binom{n-1}{m-1} + \binom{n-1}{m} = \frac{n!}{m!(n-m)!}$$

卢卡斯定理

要求：模数不大，是质数

$$\binom{n}{m} \bmod p = \binom{\lfloor \frac{n}{p} \rfloor}{\lfloor \frac{m}{p} \rfloor} \cdot \binom{n \bmod p}{m \bmod p} \bmod p$$

```
void init(){
    fac[0] = fac[1] = 1 ;
    for(int i = 2 ; i <= N ; ++i){
        fac[i] = fac[i - 1] * i % p ;
    }


    inv[0] = 1 ;
    //求n!的逆元
    inv[N] = qpow(f[N] , p - 2) ;         //费马小定理
    for(int i = N - 1 ; i > 0 ; --i){
        inv[i] = inv[i + 1] * (i + 1) % p ;
    }

}

long long C(long long n, long long m, long long p) {
    return fac[n] * inv[m] % p * inv[n - m] % p;
}

long long Lucas(long long n, long long m, long long p) {
  return m == 0 ? 1 : (C(n % p, m % p, p) * Lucas(n / p, m / p, p)) % p;
}
```

皮克定理

Pick 定理：给定顶点均为整点的简单多边形，皮克定理说明了其面积 S 和内部格点数目 n、边上格点数目 m 的关系：$S = n + \frac{m}{2} - 1$。

枚举

枚举子集

```
for(int i = 0; i < (1 << n); ++i){
    for(int j = i; j; j = (j - 1) & i){

    }
}
```

分解质因子

```
vector<int>d;   //d[x] = smallest divisor of x
vector<int>factor[N];       //factors[x] = prime factors of x, including
duplicates(重复)


d.assign(N, 1);

for(int i = N - 1; i > 1; --i){
    for(int j = i; j < N; j += i){
        d[j] = i;
    }
}
for(int i = 2; i < N; ++i){
    for(int j = i; j != 1; j /= d[j]){
        factor[i].push_back(d[j]);
    }
}
```

线性基

```
void insert(ull x) {
    for (int i = 63; ~i; --i) {
        if (!(x >> i))  // x 的第 i 位是 0
            continue;
        if (!p[i]) {
            p[i] = x;
            break;
        }
        x ^= p[i];
```

```
    }
}
```

线性基求第k小

```
#include <bits/stdc++.h>
using namespace std;
int n,m,tot = 0;
long long a[1000005];
long long k[61],tt,x,cnt;
bool flag = 1;
void insert(long long x){
    for (int i=60;i>=0;i--){
        if (x&((long long)1LL<<i)) {
            if (k[i]) x=x^k[i];
            else {
                                tot++;
                k[i]=x;
                return ;
            }
        }
    }
}
long long ask(long long x) {
        if (tot < n && x == 1) return 0;
        if (tot < n) x--;
        if (x >= (1LL << (long long)cnt)) return -1;
        long long ans = 0;
        for (int i = 0;i <= 60;i++) {
                if (k[i]) {
                        if (x % 2) ans ^= (long long)k[i];
                        x /= 2;
                }
        }
        return ans;
}
int main(){
        cin>>n;
        for (int i=1;i<=n;i++) {
            cin>>a[i];
                insert(a[i]);
        }
        scanf("%d",&m);
        for (int i = 0;i <= 60;i++) {
                for (int j = 1;j <= i;j++) {
                        if (k[i] & ((long long)1LL << (long long)(j - 1))) k[i] ^=
(long long)k[j - 1];
                }
                if (k[i]) cnt++;
        }
        for (int i = 1;i <= m;i++) {
                long long x;
```

```
            scanf("%lld",&x);
            printf("%lld\n",ask(x));
        }
    return 0;
}
//先求高斯消元法得到的线性基，再枚举k的二进制位数，注意需特判0
```

线性方程组求解

```
#include <iostream>
#include <algorithm>
#include <cmath>

using namespace std;

const int N = 110;
const double eps = 1e-6;

int n;
double a[N][N];

//debug代码的方式
void out()
{
    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < n; j++) printf("%10.2lf", a[i][j]);
    }
}

int gauss()
{
    int c, r;
    for (c = 0, r = 0; c < n; c++)
    {
        int t = r;
        for (int i = r; i < n; i++)          //找到当前这一列绝对值最大的这一行
            if (fabs(a[i][c]) > fabs(a[t][c]))  //fabs是double的，abs是整形的
                t = i;

        if (fabs(a[t][c]) < eps) continue;    //小于eps时，认为是0

        for (int i = c; i < n + 1; i++) swap(a[t][i], a[r][i]);  //把这一行交换到第一
行上去
        for (int i = n; i >= c; i--) a[r][i] /= a[r][c];         //倒着来，最后更新第
一个数，把第一个数变成1

        for (int i = r + 1; i < n; i++)                           //把下面所有行的这列
都消成0
            if (fabs(a[i][c]) > eps)                              //已经是0就不用消了
                for (int j = n; j >= c; j--)                      //从后往前消
                    a[i][j] -= a[r][j] * a[i][c];
```

```
        //out();

        r++;
    }

    if (r < n)   //r不是n，就不是唯一解
    {
        for (int i = r; i < n; i++)
            if (fabs(a[i][n]) > eps)
                return 2;       //无解， 0 = !0)

        return 1;               //无穷多组解   0 = 0
    }

    for (int i = n - 1; i >= 0; i--)                              //倒着把方程消一遍，把
每i行第j列的系数保留1外，都消成0
        for (int j = i + 1; j < n; j++)
            a[i][n] -= a[j][n] * a[i][j];                         //a[j][n]，就是xj的
值。a[i][n]，当前这一行的xi的值
                                                                  //下面一行已经求出xj的
值，只需模拟，当前i行的第n列，-= a[i][j] * xj

    return 0;  //唯一解
}


int main()
{
    cin >> n;
    for (int i= 0; i < n; i++)
        for (int j = 0; j < n + 1; j++)
            cin >> a[i][j];

    int t = gauss();

    if (t == 0)
    {
        for (int i = 0; i < n; i++) printf("%.2f\n", a[i][n]);
    }
    else if (t == 1) puts("0");
    else puts("-1");

    return 0;
}
```

异或方程组求解

```
std::bitset<1010> matrix[2010];  // matrix[1~n]：增广矩阵，0 位置为常数

std::vector<bool> GaussElimination(
    int n, int m)  // n 为未知数个数，m 为方程个数，返回方程组的解
```

```
                    // （多解 / 无解返回一个空的 vector）
{
  for (int i = 1; i <= n; i++) {
    int cur = i;
    while (cur <= m && !matrix[cur].test(i)) cur++;
    if (cur > m) return std::vector<bool>(0);
    if (cur != i) swap(matrix[cur], matrix[i]);
    for (int j = 1; j <= m; j++)
      if (i != j && matrix[j].test(i)) matrix[j] ^= matrix[i];
  }
  std::vector<bool> ans(n + 1);
  for (int i = 1; i <= n; i++) ans[i] = matrix[i].test(0);
  return ans;
}
```