```cpp
bool find(int x)
{
     // 和各个点尝试能否匹配
    for (int i = h[x]; i != -1; i = ne[i])
    {
        int j = e[i];
        if (!st[j])//打标记
        {
            st[j] = true;
            // 当前尝试点没有被匹配或者和当前尝试点匹配的那个点可以换另一个匹配
            if (match[j] == 0 || find(match[j]))
            {
                // 和当前尝试点匹配在一起
                match[j] = x;
                return true;
            }
        }
    }
    return false;
}

int main()
{
    scanf("%d%d%d", &n1, &n2, &m);

    memset(h, -1, sizeof h);
    // 保存图，因为只从一遍找另一边，所以该无向图只需要存储一个方向
    while (m -- )
    {
        int a, b;
        scanf("%d%d", &a, &b);
        add(a, b);
    }

    int res = 0;
    //为各个点找匹配
    for (int i = 1; i <= n1; i ++ )
    {
        memset(st, false, sizeof st);
        //找到匹配
        if (find(i)) res ++ ;
    }

    printf("%d\n", res);

    return 0;
}
```

四、数学知识

> 算法的数学知识定理证明可以在这里查阅：数学部分简介 - OI Wiki (oi-wiki.org)

试除法判定质数

```cpp
bool is_prime(int x)
{
    if (x < 2) return false;
    for (int i = 2; i <= x / i; i ++ )
        if (x % i == 0)
            return false;
    return true;
}
```

试除法分解质因数

```cpp
void divide(int x)
{
    for (int i = 2; i <= x / i; i ++ )
        if (x % i == 0)//i 一定是质数
        {
            int s = 0;
            while (x % i == 0) x /= i, s ++ ;
            cout << i << ' ' << s << endl;
        }
    if (x > 1) cout << x << ' ' << 1 << endl;
    cout << endl;
}
```

埃氏筛法求质数

```cpp
int primes[N], cnt;     // primes[]存储所有素数
bool st[N];         // st[x]存储x是否被筛掉

void get_primes(int n)
{
    for (int i = 2; i <= n; i ++ )
    {
        if (st[i]) continue;
        primes[cnt ++ ] = i;
        for (int j = i + i; j <= n; j += i)
            st[j] = true;
    }
}
```

线性筛法求质数

算法动画讲解：https://www.bilibili.com/video/BV1LR4y1Z7pm

```cpp
int primes[N], cnt;     // primes[]存储所有素数
bool st[N];         // st[x]存储x是否被筛掉

void get_primes(int n)
{
    for (int i = 2; i <= n; i ++ )
```

```
    {
        if (!st[i]) primes[cnt ++ ] = i;
        for (int j = 0; primes[j] <= n / i; j ++ )
        {
            st[primes[j] * i] = true;
            if (i % primes[j] == 0) break;
        }
    }
}
```

试除法求所有约数

```
vector<int> get_divisors(int x)
{
    vector<int> res;
    for (int i = 1; i <= x / i; i ++ )
        if (x % i == 0)
        {
            res.push_back(i);
            if (i != x / i) res.push_back(x / i);
        }
    sort(res.begin(), res.end());
    return res;
}
```

约数个数

约数个数定理和约数和定理公式推导：https://www.bilibili.com/video/BV13R4y1o777

约数个数定理推导：https://www.bilibili.com/video/BV1NY41187GM

$$360 = 2^3 \times 3^2 \times 5^1$$

$$\text{约数个数} = (3+1) \times (2+1) \times (1+1)$$

```
using namespace std;
typedef long long LL;
const int N = 110, mod = 1e9 + 7;
int main()
{
    int n;
    cin >> n;
    unordered_map<int, int> primes;
    while (n -- )
    {
        int x;
        cin >> x;
        for (int i = 2; i <= x / i; i ++ )
            while (x % i == 0)
            {
                x /= i;
```

```
                primes[i] ++ ;
            }
        if (x > 1) primes[x] ++ ;
    }
    LL res = 1;
    for (auto p : primes) res = res * (p.second + 1) % mod;
    cout << res << endl;
    return 0;
}
```

约数之和

约数个数定理和约数和定理公式推导：https://www.bilibili.com/video/BV13R4y1o777

$$\because 360 = 2^3 \times 3^2 \times 5^1$$

$$\therefore \text{约数个数} = (3+1)\times(2+1)\times(1+1)$$

$$\text{约数之和} = (2^0+2^1+2^2+2^3)\times(3^0+3^1+3^2)\times(5^0+5^1)$$

```
using namespace std;
typedef long long LL;
const int N = 110, mod = 1e9 + 7;
int main()
{
    int n;
    cin >> n;
    unordered_map<int, int> primes;
    while (n -- )
    {
        int x;
        cin >> x;
        for (int i = 2; i <= x / i; i ++ )
            while (x % i == 0)
            {
                x /= i;
                primes[i] ++ ;
            }
        if (x > 1) primes[x] ++ ;
    }
    LL res = 1;
    for (auto p : primes)
    {
        LL a = p.first, b = p.second;
        LL t = 1;
        while (b -- ) t = (t * a + 1) % mod;//遍历b次后得到t=p^b+p^(b-1)+...+p+1
        res = res * t % mod;
    }
    cout << res << endl;
    return 0;
}
```

代码第26行解释：

while(b--) t=a*t+1;

当b=3时，经过3次迭代如下

t=1

t=a+1

t=a*(a+1)+1=a^2+a+1

t=a*(a^2+a+1)+1=a^3+a^2+a+1

欧几里得算法(求最大公约数)

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

求欧拉函数

前置知识

> 互质：互质是公约数只有1的两个整数，叫做互质整数。

欧拉函数定义

> $1~N-1$中与N互质的数的个数被称为欧拉函数，记为$\phi(N)$。
>
> 若在算数基本定理中，$N=p_1^{a_1}p_2^{a_2}...p_m^{a_m}$，则：
>
> $\phi(N)=N\cdot\frac{p_1-1}{p_1}\cdot\frac{p_2-1}{p_2}\cdot...\cdot\frac{p_m-1}{p_m}$

欧拉函数推导

> 首先我们要知道$1,2,3...N-1,N$与N互质的个数是$1~N$数列去除N的质因子的倍数。
>
> 例如$N=10$，即$1,2,3,4,5,6,7,8,9,10$去除N的质因子的倍数$，则1,\bcancel{2},3,\bcancel{4},\bcancel{5},\bcancel{6},7,\bcancel{8},9,\bcancel{10}.$
>
> 显然，$1,3,7,9$与10互质。
>
> 由上方结论使用容斥原理进行数学推导如下：
>
> $\because N=p_1^{a_1}p_2^{a_2}...p_m^{a_m}$
>
> ①.从1~n中去掉$p_1,p_2,...,p_k$的所有倍数的个数，即
>
> $n←n-\frac{n}{p_1}-\frac{n}{p_2}-...-\frac{n}{p_k}$

②.由容斥原理，$p_i \cdot p_j$的倍数被①减了两次，所以加上所有$p_i\cdot p_j$的倍数的个数（其中p_i,p_j是$p_1~p_k$的组合），即

$n←n+\frac{n}{p_1\cdot p_2}+\frac{n}{p_1\cdot p_3}+...+\frac{n}{p_{k-1}\cdot p_k}$

③.减去所有$p_i\cdot p_j \cdot p_k$的倍数个数，即

$n←n-\frac{n}{p_1\cdot p_2\cdot p_3}-\frac{n}{p_1\cdot p_2 \cdot p_4}-...-\frac{n}{p{k-2}\cdot p{k-1}\cdot p_k}$

④.同理，加上所有$p_i\cdot p_j \cdot p_k \cdot p_l$的倍数个数，即

$n←n+\frac{n}{p_1\cdot p_2\cdot p_3\cdot p_4}+\frac{n}{p_1\cdot p_2 \cdot p_3\cdot p_5}+...+\frac{n}{p{k-3}\cdot p{k-2}\cdot p{k-1}\cdot p_k}$

$$\vdots$$

因此，

$$\begin{aligned}\phi(n) = n &- \frac{n}{p_1} - \frac{n}{p_2} - \ldots - \frac{n}{p_k} \\ &+ \frac{n}{p_1 \cdot p_2} + \frac{n}{p_1 \cdot p_3} + \ldots + \frac{n}{p_{k-1} \cdot p_k} \\ &- \frac{n}{p_1 \cdot p_2 \cdot p_3} - \frac{n}{p_1 \cdot p_2 \cdot p_4} - \ldots - \frac{n}{p_{k-2} \cdot p_{k-1} \cdot p_k} \\ &+ \frac{n}{p_1 \cdot p_2 \cdot p_3 \cdot p_4} + \frac{n}{p_1 \cdot p_2 \cdot p_3 \cdot p_5} + \ldots + \frac{n}{p_{k-3} \cdot p_{k-2} \cdot p_{k-1} \cdot p_k} \\ &- \ldots\end{aligned}$$

也就是n减去奇数个质因子的倍数个数，加上偶数个质因子的倍数个数，循环往复。

将上式等价变形，得到

$\phi(n)=n\cdot(1-\frac{1}{p_1})\cdot(1-\frac{1}{p_2})...\cdot(1-\frac{1}{p_k})$

证必。

代码模板

```
int phi(int x)
{
    int res = x;
    for (int i = 2; i <= x / i; i ++ )
        if (x % i == 0)
        {
            res = res / i * (i - 1);
            while (x % i == 0) x /= i;
        }
    if (x > 1) res = res / x * (x - 1);

    return res;
}
```

线性筛法求欧拉函数

```
int primes[N], cnt;     // primes[]存储所有素数
int euler[N];           // 存储每个数的欧拉函数
bool st[N];         // st[x]存储x是否被筛掉

void get_eulers(int n)  // 线性筛法求1~n的欧拉函数
```

```
{
    euler[1] = 1;
    for (int i = 2; i <= n; i ++ )
    {
        if (!st[i])
        {
            primes[cnt ++ ] = i;
            euler[i] = i - 1;
        }
        for (int j = 0; primes[j] <= n / i; j ++ )
        {
            int t = primes[j] * i;
            st[t] = true;
            if (i % primes[j] == 0)
            {
                euler[t] = euler[i] * primes[j];
                break;
            }
            euler[t] = euler[i] * (primes[j] - 1);
        }
    }
}
```

快速幂

快速幂公式证明：快速幂 - OI Wiki (oi-wiki.org)

```
// 求 m^k mod p，时间复杂度 O(logk)。
// m为底数，k为幂
int qmi(int m, int k, int p)
{
    int res = 1 % p, t = m;
    while (k)
    {
        if (k&1) res = res * t % p;
        t = t * t % p;
        k >>= 1;
    }
    return res;
}
```

扩展欧几里得算法

扩展欧几里得算法讲解：https://www.bilibili.com/video/BV1KU4y1a7E2/

优秀题解：https://www.acwing.com/solution/content/1393

优秀博客：https://blog.csdn.net/mango114514/article/details/121048335

> x的第一个正解就是(x%k+k)%k
>
> 其中，k=b/gcd(a,b)

```cpp
// 求x, y, 使得ax + by = gcd(a, b)
int exgcd(int a, int b, int &x, int &y)
{
    if (!b)
    {
        x = 1, y = 0;
        return a;
    }
    int d = exgcd(b, a % b, y, x);
    y -= (a/b) * x;
    return d;
}
```

中国剩余定理

中国剩余定理讲解： https://www.bilibili.com/video/BV1AN4y1N7Su/

《孙子算经》
有物不知其数，三三数之剩二，五五数之剩三，七七数之剩二。问物几何？

求解线性同余方程组

$$\begin{cases} x \equiv r_1 \ (mod\ m_1) \\ x \equiv r_2 \ (mod\ m_2) \\ \quad\vdots \\ x \equiv r_n \ (mod\ m_n) \end{cases}$$

其中模数 $m_1, m_2, \cdots, m_n$ 为**两两互质**的整数，求 x 的最小非负整数解。

中国剩余定理 (Chinese Remainder Theorem, CRT)
1. 计算所有模数的积 M
2. 计算第 i 个方程的 $c_i = \frac{M}{m_i}$
3. 计算 c_i 在模 m_i 意义下的逆元 c_i^{-1}
4. $x = \sum_{i=1}^{n} r_i\, c_i c_i^{-1} \ (mod\ M)$

例 $\begin{cases} x \equiv 2 \ (mod\ 3) \\ x \equiv 3 \ (mod\ 4) \\ x \equiv 1 \ (mod\ 5) \end{cases}$

1. $M = 3 \times 4 \times 5 = 60$
2. $c_1 = 20, c_1^{-1} = 2$, 因 $20x \equiv 1 \ (mod\ 3)$
   $c_2 = 15, c_2^{-1} = 3$, 因 $15x \equiv 1 \ (mod\ 4)$
   $c_3 = 12, c_3^{-1} = 3$, 因 $12x \equiv 1 \ (mod\ 5)$
3. $x = (2 \times 20 \times 2 + 3 \times 15 \times 3 + 1 \times 12 \times 3)\%60$
   $= 251\%60 = 11$

证明:
先证明 $\sum_{i=1}^{n} r_i\, c_i c_i^{-1}$ 满足每个 $x \equiv r_i \ (mod\ m_i)$ 。
当 $i \neq j$ 时，$c_j \equiv 0 \ (mod\ m_i)$，则 $c_j c_j^{-1} \equiv 0 \ (mod\ m_i)$
当 $i = j$ 时，$c_i \not\equiv 0 \ (mod\ m_i)$，则 $c_i c_i^{-1} \equiv 1 \ (mod\ m_i)$

$$\begin{aligned} 故\ x &\equiv \sum_{j=1}^{n} r_j\, c_j c_j^{-1} \ (mod\ m_i) \\ &\equiv r_i c_i c_i^{-1} \ (mod\ m_i) \\ &\equiv r_i \ (mod\ m_i) \end{aligned}$$

而 $\sum_{i=1}^{n} r_i\, c_i c_i^{-1} \ (mod\ M)$ 对 m_i 来说，只是减去了 m_i 的若干倍，不影响余数 r_i 。证毕


```cpp
LL exgcd(LL a,LL b,LL &x,LL &y){
    if(b==0){
        x=1,y=0;
        return a;
    }
    LL d=exgcd(b,a%b,y,x);
    y -= (a/b) * x;
    return d;
}
LL CRT(LL m[],LL r[]){
    LL m=1,ans=0;
    for(int i=1;i<=n;i++)M*=m[i];
    for(int i=1;i<=n;i++){
        LL c=M/m[i],x,y;
        exgcd(c,m[i],x,y);
        ans=(ans+r[i]*c*x%M)%M;
    }
    return (ans%M+M)%M;
}
```

扩展中国剩余定理

扩展中国剩余定理讲解： https://www.bilibili.com/video/BV1Ut4y1F7HG/

问题
求解线性同余方程组

$$\begin{cases} x \equiv r_1 \ (mod\ m_1) \\ x \equiv r_2 \ (mod\ m_2) \\ \cdots \\ x \equiv r_n \ (mod\ m_n) \end{cases}$$

其中 $m_1, m_2, \cdots, m_n$ 为**不一定两两互质**的整数，
求 x 的最小非负整数解。

中国剩余定理 (CRT) 已不可行
其构造解 $x = \sum_{i=1}^{n} r_i\, c_i c_i^{-1}\ (mod\ M)$
其中 $c_i x \equiv 1\ (mod\ m_i)$，即 $c_i x + m_i y = 1 = \gcd(c_i, m_i)$
根据裴蜀定理，c_i, m_i 应该互质，$c_i = \frac{m_1 \times \cdots \times m_n}{m_i}$
如果 c_i, m_i 不互质，则 c_i^{-1} 不存在，算法失效

扩展中国剩余定理 (EXCRT)
前两个方程：$x \equiv r_1\ (mod\ m_1), x \equiv r_2\ (mod\ m_2)$
转化为不定方程：$x = m_1 p + r_1 = m_2 q + r_2$
则 $m_1 p - m_2 q = r_2 - r_1$

由裴蜀定理，
当 $\gcd(m_1, m_2) \nmid (r_2 - r_1)$ 时，无解
当 $\gcd(m_1, m_2) \mid (r_2 - r_1)$ 时，有解

由扩欧算法，
得特解 $p = p * \frac{r_2 - r_1}{gcd},\ q = q * \frac{r_2 - r_1}{gcd}$
其通解 $P = p + \frac{m_2}{gcd} * k,\ Q = q - \frac{m_1}{gcd} * k$
所以 $x = m_1 P + r_1 = \frac{m_1 m_2}{gcd} * k + m_1 p + r_1$
前两个方程**等价合并**为一个方程 $x \equiv r\ (mod\ m)$
其中 $r = m_1 p + r_1,\ m = lcm(m_1, m_2)$

所以 n 个同余方程只要合并 $n - 1$ 次，即可求解


```cpp
LL exgcd(LL a,LL b,LL &x,LL &y){
    if(b==0){
        x=1,y=0;
        return a;
    }
    LL d=exgcd(b,a%b,y,x);
    y -= (a/b) * x;
    return d;
}
LL EXCRT(LL m[],LL r[]){
    LL m1,m2,r1,r2,p,q;
    m1=m[1],r1=r[1];
    for(int i=2;i<=n;i++){
        m2=m[i],r2=r[i];
        LL d = exgcd(m1,m2,p,q);
        if((r2-r1)%d){
            return -1;
        }
        p=p*(r2-r1)/d;//特解
        p=(p%(m2/d)+m2/d)%(m2/d);
        r1=m1*p+r1;
        m1=m1*m2/d;
    }
    return (r1%m1+m1)%m1;
}
```

高斯消元法

高斯消元 $O(n^3)$

求解例如下面方程组

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n = b_1 \\ a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n = b_2 \\ \vdots \quad\quad \vdots \quad\quad \vdots \quad\quad \vdots \\ a_{n1}x_1 + a_{n2}x_2 + \ldots + a_{nn}x_n = b_n \end{cases}$$

高斯消元讲解：https://www.bilibili.com/video/BV1Kd4y127vZ/

模板

```cpp
// a[N][N]是增广矩阵
int gauss()
{
    int c, r;
    for (c = 0, r = 0; c < n; c ++ )
    {
        int t = r;
        for (int i = r; i < n; i ++ )   // 找到绝对值最大的行
            if (fabs(a[i][c]) > fabs(a[t][c]))
                t = i;

        if (fabs(a[t][c]) < eps) continue;

        for (int i = c; i <= n; i ++ ) swap(a[t][i], a[r][i]);      // 将绝对值最大的行换到最顶端
        for (int i = n; i >= c; i -- ) a[r][i] /= a[r][c];      // 将当前行的首位变成1
        for (int i = r + 1; i < n; i ++ )       // 用当前行将下面所有的列消成0
            if (fabs(a[i][c]) > eps)
                for (int j = n; j >= c; j -- )
                    a[i][j] -= a[r][j] * a[i][c];

        r ++ ;
    }

    if (r < n)
    {
        for (int i = r; i < n; i ++ )
            if (fabs(a[i][n]) > eps)
                return 2; // 无解
        return 1; // 有无穷多组解
    }

    for (int i = n - 1; i >= 0; i -- )
        for (int j = i + 1; j < n; j ++ )
            a[i][n] -= a[i][j] * a[j][n];

    return 0; // 有唯一解
}
```

应用

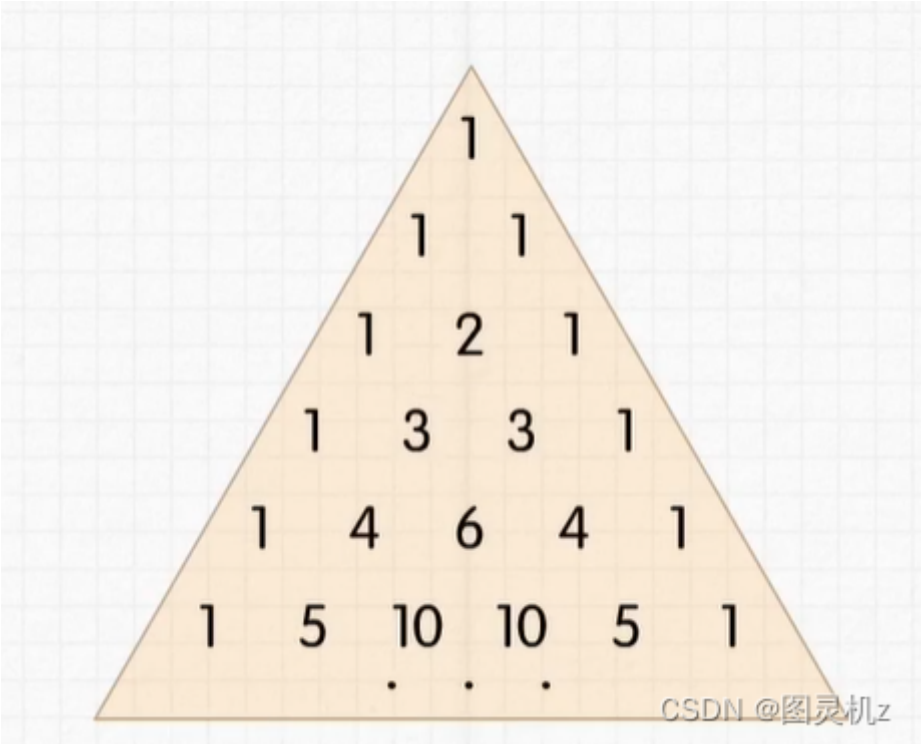
```cpp
using namespace std;

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

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

int gauss()
{
```

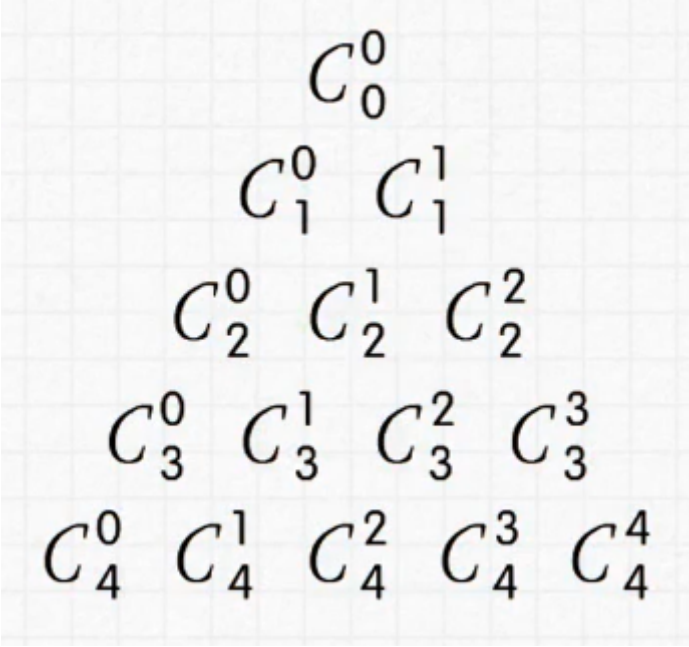
```
    int c, r;// c 代表 列 col ，  r 代表 行 row
    for (c = 0, r = 0; c < n; c ++ )
    {
        int t = r;// 先找到当前这一列，绝对值最大的一个数字所在的行号
        for (int i = r; i < n; i ++ )
            if (fabs(a[i][c]) > fabs(a[t][c]))
                t = i;

        if (fabs(a[t][c]) < eps) continue;// 如果当前这一列的最大数都是 0 ，那么所有数都是 0，就没必要去算了，因为它的约束方程，可能在上面几行

        for (int i = c; i < n + 1; i ++ ) swap(a[t][i], a[r][i]);//// 把当前这一行，换到最上面（不是第一行，是第 r 行）去
        for (int i = n; i >= c; i -- ) a[r][i] /= a[r][c];// 把当前这一行的第一个数，变成 1，  方程两边同时除以 第一个数，必须要到着算，不然第一个数直接变1，系数就被篡改，后面的数字没法算
        for (int i = r + 1; i < n; i ++ )// 把当前列下面的所有数，全部消成 0
            if (fabs(a[i][c]) > eps)// 如果非0 再操作，已经是 0就没必要操作了
                for (int j = n; j >= c; j -- )// 从后往前，当前行的每个数字，都减去对应列 * 行首非0的数字，这样就能保证第一个数字是 a[i][0] -= 1*a[i][0];
                    a[i][j] -= a[r][j] * a[i][c];

        r ++ ;// 这一行的工作做完，换下一行
    }

    if (r < n)// 说明剩下方程的个数是小于 n 的，说明不是唯一解，判断是无解还是无穷多解
    {// 因为已经是阶梯型，所以 r ~ n-1 的值应该都为 0
        for (int i = r; i < n; i ++ )//
            if (fabs(a[i][n]) > eps)// a[i][n] 代表 b_i ,即 左边=0，右边=b_i,0 != b_i, 所以无解。
                return 2;
        return 1;// 否则，  0 = 0，就是r ~ n-1的方程都是多余方程
    }
    // 唯一解 ↓，从下往上回代，得到方程的解
    for (int i = n - 1; i >= 0; i -- )
        for (int j = i + 1; j < n; j ++ )
            a[i][n] -= a[j][n] * a[i][j];//因为只要得到解，所以只用对 b_i 进行操作，中间的值，可以不用操作，因为不用输出

    return 0;
}

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("%.2lf\n", a[i][n]);
    }
    else if (t == 1) puts("Infinite group solutions");
    else puts("No solution");
    return 0;
}
```

求组合数

递推法求组合数

排列组合详细讲解：https://www.bilibili.com/video/BV1e7411J7SC/





1. 左右两侧斜线都是1，$C^0_n=C^n_n=1$
2. 其他数等于其左上角和右上角两数之和$C^{m-1}n+C^m_n=C^m{n+1}$

```
// c[a][b] 表示从a个苹果中选b个的方案数
int c[N][N];
for (int i = 0; i < N; i ++ )
    for (int j = 0; j <= i; j ++ )
        if (!j) c[i][j] = 1;
        else c[i][j] = (c[i - 1][j] + c[i - 1][j - 1]) % mod;
//本质上杨辉三角
```

通过预处理逆元的方式求组合数

模板

```
// 首先预处理出所有阶乘取模的余数fact[N]，以及所有阶乘取模的逆元infact[N]
// 如果取模的数是质数，可以用费马小定理求逆元
int qmi(int a, int k, int p)    // 快速幂模板
{
    int res = 1;
    while (k)
    {
        if (k & 1) res = (LL)res * a % p;
        a = (LL)a * a % p;
        k >>= 1;
    }
    return res;
}

// 预处理阶乘的余数和阶乘逆元的余数
fact[0] = infact[0] = 1;
for (int i = 1; i < N; i ++ )
{
    fact[i] = (LL)fact[i - 1] * i % mod;
    infact[i] = (LL)infact[i - 1] * qmi(i, mod - 2, mod) % mod;
}
```

应用

```
using namespace std;

typedef long long LL;

const int N = 100010,mod=1e9+7;//1e9+7是质数所以与[1,1e9+7)中的数互质

int fact[N],infact[N];

int qmi(int a,int k,int p){
    int res=1;
    while(k){
        if(k&1)res=(LL)res*a%p;
        a=(LL)a*a%p;
        k>>=1;
    }
    return res;
}

int main()
{
    fact[0]=infact[0]=1;
    for (int i = 1; i <= N; i ++ ){
        fact[i]=(LL)fact[i-1]*i%mod;
        infact[i]=(LL)infact[i-1]*qmi(i,mod-2,mod)%mod;
    }

    int n;
    scanf("%d",&n);
    while (n -- ){
        int a,b;
        scanf("%d%d", &a, &b);
```

```
        printf("%d\n",(LL)fact[a]*infact[b]%mod*infact[a-b]%mod);
    }
    return 0;
}
```

Lucas定理求组合数

Lucas定理证明： https://blog.csdn.net/Qiuker_jl/article/details/109528164

模板

```
// 若p是质数，则对于任意整数 1 <= m <= n，有：
// C(n, m) = C(n % p, m % p) * C(n / p, m / p) (mod p)

int qmi(int a, int k, int p)  // 快速幂模板
{
    int res = 1 % p;
    while (k)
    {
        if (k & 1) res = (LL)res * a % p;
        a = (LL)a * a % p;
        k >>= 1;
    }
    return res;
}

int C(int a, int b, int p)  // 通过定理求组合数C(a, b)
{
    if (a < b) return 0;

    LL x = 1, y = 1;  // x是分子，y是分母
    for (int i = a, j = 1; j <= b; i --, j ++ )
    {
        x = (LL)x * i % p;
        y = (LL) y * j % p;
    }

    return x * (LL)qmi(y, p - 2, p) % p;
}

int lucas(LL a, LL b, int p)
{
    if (a < p && b < p) return C(a, b, p);
    return (LL)C(a % p, b % p, p) * lucas(a / p, b / p, p) % p;
}
```

应用

```
using namespace std;

typedef long long LL;

int qmi(int a,int k,int p)
{
    int res = 1;
    while(k)
```

```
    {
        if(k&1)res = (LL)res*a%p;
        a = (LL)a*a%p;
        k>>=1;
    }
    return res;
}

int C(int a,int b,int p)//自变量类型int
{
    if(b>a)return 0;//漏了边界条件
    int res = 1;
    // a!/(b!(a-b)!) = (a-b+1)*...*a / b! 分子有b项
    for(int i=1,j=a;i<=b;i++,j--)//i<=b而不是<
    {
        res = (LL)res*j%p;
        res = (LL)res*qmi(i,p-2,p)%p;
    }
    return res;
}
//对公式敲
int lucas(LL a,LL b,int p)
{
    if(a<p && b<p)return C(a,b,p);//lucas递归终点是C_{bk}^{ak}
    return (LL)C(a%p,b%p,p)*lucas(a/p,b/p,p)%p;//a%p后肯定是<p的,所以可以用C(),但a/p后不一定<p 所以用lucas继续递归
}

int main()
{
    int n;
    cin >> n;
    while(n--)
    {
        LL a,b;
        int p;
        cin >> a >> b >> p;
        cout << lucas(a,b,p) << endl;
    }
    return 0;
}
```

分解质因数法求组合数

模板

```
当我们需要求出组合数的真实值，而非对某个数的余数时，分解质因数的方式比较好用：
    1. 筛法求出范围内的所有质数
    2. 通过 C(a, b) = a! / b! / (a - b)! 这个公式求出每个质因子的次数。 n! 中p的次数是 n / p + n / p^2 + n / p^3 + ...
    3. 用高精度乘法将所有质因子相乘

int primes[N], cnt;     // 存储所有质数
int sum[N];     // 存储每个质数的次数
bool st[N];     // 存储每个数是否已被筛掉


```

```cpp
void get_primes(int n)      // 线性筛法求素数
{
    for (int i = 2; i <= n; i ++ )
    {
        if (!st[i]) primes[cnt ++ ] = i;
        for (int j = 0; primes[j] <= n / i; j ++ )
        {
            st[primes[j] * i] = true;
            if (i % primes[j] == 0) break;
        }
    }
}


int get(int n, int p)       // 求n! 中的次数
{
    int res = 0;
    while (n)
    {
        res += n / p;
        n /= p;
    }
    return res;
}


vector<int> mul(vector<int> a, int b)       // 高精度乘低精度模板
{
    vector<int> c;
    int t = 0;
    for (int i = 0; i < a.size(); i ++ )
    {
        t += a[i] * b;
        c.push_back(t % 10);
        t /= 10;
    }

    while (t)
    {
        c.push_back(t % 10);
        t /= 10;
    }

    return c;
}

get_primes(a);  // 预处理范围内的所有质数

for (int i = 0; i < cnt; i ++ )     // 求每个质因数的次数
{
    int p = primes[i];
    sum[i] = get(a, p) - get(b, p) - get(a - b, p);
}

vector<int> res;
res.push_back(1);

for (int i = 0; i < cnt; i ++ )     // 用高精度乘法将所有质因子相乘
    for (int j = 0; j < sum[i]; j ++ )
        res = mul(res, primes[i]);
```

应用

```cpp
using namespace std;

const int N = 5010;
int primes[N],cnt=0;
// v[i] 记录数字 i 为素数还是合数，v[i]=true时 i 为合数，否则 i 为素数
bool v[N];
// sum[i]=c 表示质数 i 的个数为 c
int sum[N];

// 线性筛法
void get_primes(int n)
{
    for(int i=2;i<=n;++i)
    {
        // i为质数，则存在primes中
        if(!v[i])primes[cnt++]=i;
        // 给当前数i乘上一个质因子pj
        for(int j=0;primes[j]<=n/i;++j)
        {
            v[primes[j]*i]=true;
            if(i%primes[j]==0)break;
        }
    }
}

// 计算 n 里面含有质数 p 的个数，这里的计算是不重不漏的。
// p^k的倍数会被计算k次：第一次算p的倍数时，被加一次；第二次算p^2的倍数时，被加一次；第三次算p^3的倍数时，被加一次...第k次算p^k的倍数时，被加一次。总共被加了k次，是不重不漏的。
int get(int n,int p)
{
    int res=0;
    while(n)
    {
        res+=n/p;
        n/=p;
    }
    return res;
}

// A * b：把 b 看成一个整体，然后与 A 中每一位相乘，A中的数字采用小端存储，即低位数字存储在数组的前面，高位数字存储在数组的后面
vector<int> mul(const vector<int>& A,const int b)
{
    if(b==0)return {0};
    vector<int> res;
    // t 表示乘法进位，这里的进位不限于0 1，可以为任意数字
    for(int i=0,t=0,n=A.size();i<n||t>0;++i)
    {
        // 获得当前位的乘积和
        if(i<n)t+=A[i]*b;
        // 添加个位数字
        res.push_back(t%10);
        // 保留进位
        t/=10;
    }

     // 如 1234 * 0 = 0000，需要删除前导0
    while(res.size()>1&&res.back()==0)res.pop_back();
    return res;
```

```cpp
}

int main()
{
    int a,b;cin>>a>>b;

    // 将 a 分解质因数
    get_primes(a);

    for(int i=0;i<cnt;++i)
    {
        // 当前的质数为 p
        int p=primes[i];
        // 用分子里面 p 的个数减去分母里面 p 的个数。这里的计算组合数的公式为a!/(b!*(a-b)!)，因此用 a 里面 p 的个数减去 b 里面 p 的个数和 (a-b) 里面 p 的个数。
        sum[i]=get(a,p)-get(b,p)-get(a-b,p);
    }

    // 使用高精度乘法把所有质因子乘到一块去就好了
    vector<int> res={1};
    for(int i=0;i<cnt;++i)
        // res*p^k，这里是k个p相乘，不是k*p，所以需要使用一个循环
        for(int j=0;j<sum[i];++j)
            res=mul(res,primes[i]);

    // 倒序打印 res 即可，由于采用小端存储，所以高位在后，从后往前打印即可
    for(int i=res.size()-1;i>=0;i--)printf("%d",res[i]);
    return 0;
}
```

容斥原理应用

经典例题： 890. 能被整除的数 - AcWing题库

AC代码：

```cpp
using namespace std;
typedef long long LL;

const int N = 20;
int p[N], n, m;

int main() {
cin >> n >> m;
for(int i = 0; i < m; i++) cin >> p[i];

int res = 0;
//枚举从1 到 1111...(m个1)的每一个集合状态, (至少选中一个集合)
for(int i = 1; i < 1 << m; i++) {
  int t = 1;              //选中集合对应质数的乘积
  int s = 0;              //选中的集合数量

  //枚举当前状态的每一位
  for(int j = 0; j < m; j++){
      //选中一个集合
      if(i >> j & 1){
          //乘积大于n, 则n/t = 0, 跳出这轮循环
          if((LL)t * p[j] > n){
```

```
            t = -1;
            break;
        }
        s++;                    //有一个1，集合数量+1
        t *= p[j];
    }
  }

  if(t == -1) continue;

  if(s & 1) res += n / t;               //选中奇数个集合，则系数应该是1，n/t为当前这种状态的集合数量
  else res -= n / t;                    //反之则为 -1
}

cout << res << endl;
return 0;
}
```

详细题解：AcWing 890. 能被整除的数 - AcWing

博弈论

NIM游戏

定理1：必胜态的后继状态至少存在一个**必败态**

定理2：必败态的后继状态均为**必胜态**

NIM游戏科普：尼姆游戏（学霸就是这样欺负人的）*哔哩哔哩*bilibili

再看nim游戏*哔哩哔哩*bilibili

经典例题：P2197 【模板】nim 游戏 - 洛谷 | 计算机科学教育新生态 (luogu.com.cn)

AC代码：

```
using namespace std;
int T;

int main() {
    cin >> T;
    while (T--) {
        int n;
        scanf("%d", &n);
        int ans = 0;
        for (int i = 0; i < n; i++) {
            int k;
            scanf("%d", &k);
            ans ^= k;
        }
        if (ans)
            puts("Yes");
        else
            puts("No");
    }
    return 0;
```

```
}
```

结论：

若初态为**必胜态**($a_1\oplus a_2\oplus ...\oplus a_n \neq0$).则先手必胜

若初态为**必败态**($a_1\oplus a_2\oplus ...\oplus a_n =0$).则先手必败

视频讲解：581 尼姆（Nim）游戏【博弈论】*哔哩哔哩*bilibili

台阶型NIM游戏

经典例题：892. 台阶-Nim游戏 - AcWing题库

AC代码：

```cpp
using namespace std;

const int N = 100010;

int main()
{
     int n;
     scanf("%d", &n);
  int res = 0;
     for (int i = 1; i <= n; i ++ )
     {
         int x;
         scanf("%d", &x);
         if (i & 1) res ^= x;
     }
     if (res) puts("Yes");
  else puts("No");

     return 0;
}
```

结论：若奇数台阶上的$a_1\oplus a_3\oplus a_5\oplus ...\neq0$，则先手必胜，反之先手必败。

视频讲解：582 台阶型 Nim游戏【博弈论】*哔哩哔哩*bilibili

集合型NIM游戏

经典例题：893. 集合-Nim游戏 - AcWing题库

AC代码：

```cpp
using namespace std;

const int N=110,M=10010;
int n,m;
int f[M],s[N];//s存储的是可供选择的集合,f存储的是所有可能出现过的情况的sg值

```

```cpp
int sg(int x)
{
     if(f[x]!=-1) return f[x];
     //因为取石子数目的集合是已经确定了的,所以每个数的sg值也都是确定的,如果存储过了,直接返回即可
     unordered_set<int> S;
     //set代表的是有序集合(注:因为在函数内部定义,所以下一次递归中的S不与本次相同)
     for(int i=0;i<m;i++)
     {
         int sum=s[i];
         if(x>=sum) S.insert(sg(x-sum));
         //先延伸到终点的sg值后,再从后往前排查出所有数的sg值
     }
     for(int i=0;;i++)
     //循环完之后可以进行选出最小的没有出现的自然数的操作
      if(!S.count(i))
       return f[x]=i;
}

int main()
{
     cin>>m;
     for(int i=0;i<m;i++)
     cin>>s[i];

     cin>>n;
     memset(f,-1,sizeof(f));//初始化f均为-1,方便在sg函数中查看x是否被记录过

     int res=0;
     for(int i=0;i<n;i++)
     {
         int x;
         cin>>x;
         res^=sg(x);
         //观察异或值的变化,基本原理与Nim游戏相同
     }

     if(res) printf("Yes");
     else printf("No");

     return 0;
}
```

思路：转换成**有向图游戏**

视频讲解：583 有向图游戏 SG函数【博弈论】*哔哩哔哩*bilibili

五、动态规划