数论的一些知识

1. $a|c$、$b|c$，且 $(a,b)=1$ 则 $ab|c$
2. $a|bc$ 且 $(a,b)=1$，则 $a|c$
3. $p|ab$ 则 $p|a$ 或 $p|b$

---

1. 若 $d|a$ 且 $d|b$ 则 d 是 a、b 的「公约数」，最大公约数记为 $d=(a,b)$
2. 若 $a|d$ 且 $b|d$ 则 d 是 a、b 的「公倍数」，最小公倍数记为 $d=[a,b]$
3. $(a,b)*[a,b]=ab$

最大公约数：

```
long long gcd(long long x, long long y) {
    return y == 0 ? x : gcd(y, x % y);
}
```

最小公倍数：

```
long long res = a / gcd(a, b) * b;
```

如果需要求 $a_1, a_2, a_3, \cdots, a_n$ 的最大公约数：

```
d = a1
ls = [a1, a2, a3, ..., an]
for i in ls:
    d = gcd(d, i)
```

这段代码的时间复杂度是：$n + log(a_{max})$

---

1. 如果 a 和 b 都是奇数，那么 $(a,b)=(a-b,b)$
2. 如果 a 是偶数，b 是奇数，那么 $(a,b)=(a/2,b)$
3. 如果 a，b 都是偶数，那么 $(a,b)=2(a/2,b/2)$

关于整数解：

1. 若 $(a,b)|sum$，则必定存在 $ax+by=sum$，且 x、y 都是整数（*存在整数解*）
2. 对于方程 $ax+by=c$，如果 $(a,b)|c$ 则存在整数解，否则一定没有！
3. 因为 $(a,b)|(ax+by)$，所以 $(a,b)|c$
4. 对于方程 $ax+by+cz=d$，则 $(a,b,c)|(ax+by+cz)$，即 $(a,b,c)|d$

扩展欧几里得：

```
long long exgcd(long long a, long long b, long long &x, long long &y) {
    if (b == 0) { x = 1; y = 0; return a; }
    long long d = exgcd(b, a % b, y, x);
    y -= a / b * x;
    return d;
}
```

求方程 $ax+by=c$ 的通解：

先求出 $ax+by=(a,b)$ 的一对解 $\{x,y\}$

则特解：$\{\frac{c}{(a,b)}*x,\frac{c}{(a,b)}*y\}$

通解为：$\{\frac{c}{(a,b)}*x+\frac{b}{(a,b)}*N,\frac{c}{(a,b)}*y-\frac{a}{(a,b)}*N\}$

证明如下：

> $\frac{c}{(a,b)}(ax+by)=c$

> $\frac{c}{(a,b)}\{a(x+k_1), b(y+k_2)=c\}$
>
> $\frac{c}{(a,b)}(ax+by)+\frac{c}{(a,b)}(ak_1+bk_2)=c$，且 $ak_1+bk_2=0$
>
> 所以：$ak_1=-bk_2=-[a,b]*N=-\frac{ab}{(a,b)}*N$
>
> 所以：$k_1=-\frac{b}{(a,b)}*N, k_2=\frac{a}{(a,b)}*N$

求一组 $ax+by=c$ 的正整数解：

如果求出 $ax+by=d$ 的一组解为 x_0, y_0，则原方程的特解为：$\frac{c}{d}*x_0, \frac{c}{d}*y_0$

先让 x_0 变成最小非负整数解：$\frac{c}{d}*x_0+\frac{b}{d}*N$ 转变为问题：$\frac{c}{d}*x_0$ 需要加上或者减去多少个 $\frac{b}{d}$ 才会变成非负数，先求出 $\frac{c}{d}*x_0$ 取模 $\frac{b}{d}$ 的余数，如果余数是负数，则需要再加上一个 $\frac{b}{d}$ 这样就可以编程最小非负整数解！

如果 x_0 已经变成最小非负整数解了，那么 y_0 如果还是负数的话，就需要减去多一个 $-\frac{a}{d}$，那么平行项 x_0 就需要减掉一个 $\frac{b}{d}$ 会变成负数，所以肯定不可能成立！

```cpp
#include <bits/stdc++.h>

long long exgcd(long long a, long long b, long long &x, long long &y) {
    if (b == 0) { x = 1; y = 0; return a; }
    long long d = exgcd(b, a % b, y, x);
    y -= a / b * x;
    return d;
}

long long a, b, c;
long long x, y;

void solve() {
    std::cin >> a >> b >> c;
    long long d = exgcd(a, b, x, y);
    if (c % d) {
        std::cout << -1 << '\n'; return;
    }
    {
        a /= d; b /= d; c /= d;
        __int128 x1 = x; x1 *= c;
        __int128 y1 = y; y1 *= c;
        __int128 x2 = (x1 % b + b) % b;
        __int128 y2 = y1 - (x2 - x1) / b * a;
        if (y2 < 0) {
            std::cout << -1 << '\n'; return;
        }
        std::cout << (long long)x2 << ' ' << (long long)y2 << '\n';
    }
}

int main() {
    std::ios::sync_with_stdio(0);
    std::cin.tie(0); std::cout.tie(0);
    long long t; std::cin >> t; while (t --)
    solve();
    return 0;
}
```

求 $(ax+by) \bmod m$ 的最小值

答案是：0

证明过程如下：

> $(ax+by) \bmod m$
>
> 等价于：$(ax+by+tm) \bmod m$
>
> 而 $(ax+by+tm)=k*\gcd(a,b,m)$
>
> 要让 $k*\gcd(a,b,m)$ 最小，只需要让 $k=0$ 即可

求 $(ax+by+c) \bmod m$ 的最小值

答案：$\min\{c \bmod \gcd(a,b,m), c \bmod \gcd(a,b,m)-\gcd(a,b,m)\}$

证明过程如下：

> $(ax+by+c) \bmod m$
>
> 等价于：$(ax+by+c+tm) \bmod m$
>
> $k*\gcd(a,b,m)+c$ 最小
>
> 所以令：$k=-c/\gcd(a,b,m)$
>
> k 是否是真的可以让取模后的值最小呢？
>
> 假设 k 并不是答案，那么再加上 K 个 $\gcd(a,b,m)$ 后的式子 $k*\gcd(a,b,m)+K*\gcd(a,b,m)+c$ 才是答案
>
> 由于 $\gcd(a,b,m)|m$ 所以无论再继续加多少个 $\gcd(a,b,m)$ 都只会是一个以 $k*\gcd(a,b,m)+c$ 为首项，$\gcd(a,b,m)$ 为公差的循环节
>
> 还可以讨论一下为什么不是 $m-\gcd(a,b,m)+k*\gcd(a,b,m)+c$ 最小

> 因为 $k*\gcd(a,b,m)+c<\gcd(a,b,m)$
>
> 而 $\gcd(a,b,m)|m$，所以 $m=k\gcd(a,b,m),k\geq 1$ 所以：$m-\gcd(a,b,m)+k*\gcd(a,b,m)+c\geq (k-1)*\gcd(a,b,m)$ 对于 $k>1$ 的情况都成立，但是如果 $k=1$ 呢？还是有可能的哦

关于方程 $ax+by+cz=d$ 的特解：

1. 首先 $(a,b,c)|d$ 必定成立
2. 先求出 $ax+by=(a,b)$ 的一对特解记为 x_1,y_1
3. 再求出 $(a,b)t+cz=((a,b),c)$ 的一对特解，记为 t_1,z_1
4. 则 $ax+by+cz=(a,b,c)$ 的特解为：(x_1t_1,y_1t_1,z_1)

同余的一些性质

若 $a\equiv b(\bmod m)$ 且 $a\equiv b(\bmod n)$ 成立，则 $a\equiv b(\bmod [m,n])$

> $m|a-b,n|a-b$ 所以 $[m,n]|a-b$

若 $(k,m)=d$，且 $ka\equiv ka\prime(\bmod m)$ 则 $a\equiv a\prime(\bmod \frac{m}{d})$

> $m|k(a-a\prime)$
>
> $\frac{m}{d}|\frac{k}{d}(a-a\prime)$
>
> 因为 d 是 m 和 k 的最大公约数，所以 $\frac{m}{d}$ 与 $\frac{k}{d}$ 互质
>
> 所以就只可能：$\frac{m}{d}|a-a\prime$

如何求线性同余方程：$ax\equiv b(\bmod m)$

> $ax+my=b$
>
> 用 $exgcd$ 求出一个特解

容斥原理

假设有 n 个集合：S_1，S_2，…，S_n，求：$S_1 \cup S_2 \cup \cdots \cup S_n$

答案：1个集合的组合 - 2个集合的组合 + 3个集合的组合 - 4个集合的组合 … …

题目：给定一个整数 n 和 m 个整数，求 $1 \sim n$ 中能被这 m 个整数的某一个整除的个数有多少个？

```cpp
#include <bits/stdc++.h>
#define long long long

const long N = 1e2;
long n, m;
long a[N];
long res;

inline long lcm(long x, long y) {
    return x / std::__gcd(x, y) * y;
}

std::vector<long> get(long x) {
    long i = 1, j = 0;
    std::vector<long> res;
    while (i <= x) {
        if (x & i) res.push_back(a[j + 1]);
        i <<= 1; j += 1;
    }
    return res;
}

void solve() {
    std::cin >> n >> m;
    for (long i = 1; i <= m; i ++) std::cin >> a[i];
    for (long i = 1; i < (1 << m); i ++) {
        auto vt = get(i);
        long mo = (vt.size() & 1) ? 1 : -1;
        long t = 1;
        for (auto x : vt) {
            t = lcm(t, x);
            if (t > n) break;
        }
        if (t > n) continue;
        res += mo * n / t;
    }
    std::cout << res << '\n';
}

signed main() {
    std::ios::sync_with_stdio(0);
    std::cin.tie(0); std::cout.tie(0);

    solve();
    return 0;
}
```

组合数

排列组合

用二进制表示组合排列：

```
for (long i = 0; i < (1 << n); i ++)
```

用 `dfs` 求组合排列

求组合数

$C(a,b) = \frac{a!}{b!(a-b)!} = C(a-1,b) + C(a-1,b-1)$ 杨辉三角的推导方法推出下面一项

如果需要求的组合数很大，并且需要对一个质数 p 取模，可以用卢卡斯定理求：

> $C(a,b) = C(a \bmod p, b \bmod p) * C(a/p, b/p)$ 递归求下一项

```cpp
#include <bits/stdc++.h>
#define long long long

const long N = 1e5 + 100;
long fac[N]; // 阶乘
long a, b, p;

long exgcd(long a, long b, long& x, long& y) {
    if (b == 0) { x = 1; y = 0; return a; }
    long d = exgcd(b, a % b, y, x);
    y -= a / b * x;
    return d;
}

long C(long a, long b, long mod) {
    long p = fac[a], q = fac[b] * fac[a - b] % mod;
    long d, x, y; d = exgcd(q, mod, x, y);
    return (p * x % mod + mod) % mod;
}

long lucas(long a, long b, long mod) {
    if (b == 0) return 1;
    return C(a % mod, b % mod, mod) * lucas(a / mod, b / mod, mod) % mod;
}

void solve() {
    fac[0] = 1;
    std::cin >> a >> b >> p;
    for (long i = 1; i < N; i ++) fac[i] = fac[i - 1] * i % p;
    std::cout << lucas(a, b, p) << '\n';
}

signed main() {
    std::ios::sync_with_stdio(0);
    std::cin.tie(0); std::cout.tie(0);
    long t; std::cin >> t; while (t --)
    solve();
    return 0;
}
```

欧拉函数

求 $1 \sim n$ 范围内与 n 互质的个数：

$n = p_1^{a_1} p_2^{a_2} \cdots p_k^{a_k}$

则答案为：$n * \frac{p_1 - 1}{p_1} * \frac{p_2 - 1}{p_2} * \frac{p_3 - 1}{p_3} * \cdots * \frac{p_k - 1}{p_k}$

该式子也称为欧拉函数！

```
#include <bits/stdc++.h>
#define long long long

long n;
long res;

void solve() {
    std::cin >> n;
    res = n;
    for (long i = 2; i * i <= n; i ++) {
        if (n % i) continue;
        res = res / i * (i - 1);
        while (n % i == 0) n /= i;
    }
    if (n != 1) res = res / n * (n - 1);
    std::cout << res << '\n';
}

signed main() {
    std::ios::sync_with_stdio(0);
    std::cin.tie(0); std::cout.tie(0);
    long t; std::cin >> t; while (t --)
    solve();
    return 0;
}
```

欧拉定理

令 $1 \sim m$ 范围内与 m 互质的个数为 $f(m)$，若 $\gcd(a, m) \equiv 1$ 则 $a^{f(m)} \equiv 1 (\bmod m)$

特殊情况下，若 m 是质数，则 $f(m) = m - 1$，所以有：$a^{m-1} \equiv 1 (\bmod m)$

逆元

快速幂求逆元：

```
long qpow(long a, long b, long mod) {
    a = (a % mod + mod) % mod;
    long ans = 1;
    while (b) {
        if (b & 1) ans = ans * a % mod;
        a = a * a % mod;
        b >>= 1;
    }
    return ans;
}

bool ny(long a, long b, long& res) {
    if (std::gcd(a, b) != 1) return false;
    res = qpow(a, b - 2, b);
    return true;
}
```

$exgcd$ 求逆元：

> $ax \equiv 1 (\bmod b)$
>
> $ax + by \equiv 1$

```
long exgcd(long a, long b, long& x, long& y) {
    if (b == 0) { x = 1; y = 0; return a; }
    long d = exgcd(b, a % b, y, x);
    y -= a / b * x;
    return d;
}

bool ny(long a, long b, long& res) {
    long d, x, y;
    d = exgcd(a, b, x, y);
    if (d != 1) return false;
    res = (x % b + b) % b;
    return true;
}
```

逆元递推式：

令 $f(i)$ 代表 i 在 $\bmod p$ 的逆元，则：

> $f(i) = (p - p/i) * f(p \bmod i) \bmod p$
>
> 注意 p/i 为取整除

可以快速求 $1 \sim n$ 的逆元，时间复杂度：$O(n)$

```
#include <bits/stdc++.h>
#define long long long

const long N = 1e7 + 10;
long n, m;
long inv[N];

void solve() {
    std::cin >> m >> n;
    inv[1] = 1;
    for (long i = 2; i <= n; i ++) {
        inv[i] = (m - m / i) * inv[m % i] % m;
    }

    long res = 0;
    for (long i = 1; i <= n; i ++) res ^= inv[i];
    std::cout << res << '\n';
}

signed main() {
    std::ios::sync_with_stdio(0);
    std::cin.tie(0); std::cout.tie(0);

    solve();
    return 0;
}
```

借助前缀乘法求逆元

给定一个集合 $a_1, a_2, a_3, \cdots a_n$ 求每一个元素的逆元：

> 令 $s_i = a_1 * a_2 * \cdots a_i$ 前缀乘法

> 求出 $s_1, s_2, s_3, \cdots, s_n$
>
> 先求出 s_n 的逆元为 t_n，则 $t_n = \frac{1}{a_1} * \frac{1}{a_2} * \frac{1}{a_3} * \cdots * \frac{1}{a_n}$
>
> 所以 $\frac{1}{a_n} = t_n * s_{n-1}$，$t_{n-1} = t_n * a_n$

```cpp
#include <bits/stdc++.h>
#define long long long

unsigned int A, B, C;
inline unsigned int rng61() {
    A ^= A << 16;
    A ^= A >> 5;
    A ^= A << 1;
    unsigned int t = A;
    A = B;
    B = C;
    C ^= t ^ A;
    return C;
}

const long N = 1e7 + 10;
long mod, n;
long a[N], s[N], inv[N];

long exgcd(long a, long b, long& x, long& y) {
    if (b == 0) { x = 1; y = 0; return a; }
    long d = exgcd(b, a % b, y, x);
    y -= a / b * x;
    return d;
}

bool ny(long a, long mod, long& res) {
    long d, x, y;
    d = exgcd(a, mod, x, y);
    if (d != 1) return false;
    res = (x % mod + mod) % mod; return true;
}

void solve() {
    s[0] = 1;
    for (long i = 1; i <= n; i ++) {
        s[i] = s[i - 1] * a[i] % mod;
    }
    long t; ny(s[n], mod, t);
    for (int i = n; i; i --) {
        inv[i] = t * s[i - 1] % mod;
        t = t * a[i] % mod;
    }

    long res = 0;
    for (long i = 1; i <= n; i ++) res ^= inv[i];
```

```
    std::cout << res << '\n';
}

int main() {
    std::ios::sync_with_stdio(0);
    std::cin.tie(0); std::cout.tie(0);
    std::cin >> mod >> n >> A >> B >> C;
    long cnt = 0;
    for (long i = 1; i <= n; i ++) {
        long t = rng61() % mod;
        if (t == 0) continue;
        a[++ cnt] = t;
    }
    n = cnt;
    if (n == 0) {
        std::cout << 0 << '\n'; exit(0);
    }
    solve();
    return 0;
}
```

生成随机数

```
unsigned long r64 = time(0);
unsigned long rlong() {
    r64 ^= r64 >> 12;
    r64 ^= r64 << 25;
    r64 ^= r64 >> 27;
    return r64;
}
```

```
unsigned int r32 = time(0);
unsigned int rint() {
    r32 ^= r32 << 13;
    r32 ^= r32 >> 17;
    r32 ^= r32 << 5;
    return r32;
}
```

```c
long rnum(long n) {
    return rlong() % n + 1;
}
```