数论基础

2024 年 7 月

feecle8146

认识求和号

在今明两天的学习中，我们将会多次用到求和符号 $\sum$，它具有以下性质：

1. 可交换。$\sum_i \sum_j f(i,j) = \sum_j \sum_i f(i,j)$。可以理解成枚举一张数表，按行枚举和按列枚举是等价的。
   注意：交换求和号时，不要改变变量的取值范围。例如，
   $\sum_i \sum_{j \geq i} f(i,j) = \sum_j \sum_{i \leq j} f(i,j)$。
2. 加法运算律：$\sum(a+b) = \sum a + \sum b$。
3. 乘法分配律：$\sum_i \sum_j f(i)g(j) = \sum_i f(i) \times \sum_j g(j)$。

在数论问题中，常常需要求出一个很长的带求和号的算式的值。此时，我们的目标一般来说是：

<u>分离变量，使得各个变量间互不影响，从而用上面的2,3分解成子问题</u>

分解质因数

每一个正整数都可以唯一写成

$$n = \prod_i p_i^{k_i}$$

的形式，其中 p_i 是严格递增排列的质数。该形式称为 n 的唯一分解。

可以使用试除法在 $O(\sqrt{n})$ 复杂度内求出 n 的质因数分解。

```
for (int i = 2; i * i <= n; i++) {
    if (n % i) continue;
    p[++cnt] = i;
    while (n % i == 0) n /= i, pk[cnt]++;
}
```

数论函数

定义 $\sigma_0(n)$ 或 $d(n)$ 表示 n 的约数个数。可以用求和号表示为

$$\sigma_0(n) = \sum_{d|n} 1$$

若已知 m 和 n 的唯一分解分别为 $\prod p_i^{l_i}, \prod p_i^{k_i}$，则 $m|n$ 等价于 $\forall i, l_i \le k_i$。

请你推导用 k_i 表示 $\sigma_0(n)$ 的公式。

$$\sigma_0(n) = \prod_i (k_i + 1)$$

数论函数

约数 k 次幂和

定义 $\sigma_k(n)$ 表示 n 的约数的 k 次方和。可以用求和号表示为

$$\sigma_k(n)=\sum_{d|n}d^k$$

请你用等比数列求和公式推导用 p_i,k_i 表示 $\sigma_t(n)$ 的公式。

$$\sigma_1(n)=\prod_i\frac{1-p_i^{k_it+1}}{1-p_i^t}$$

数论函数

gcd 和 lcm

定义 $\gcd(a,b)$ 表示 a,b 的最大公因数，也即 $\max\limits_{u|a,u|b} u$。
定义 $\operatorname{lcm}(a,b)$ 表示 a,b 的最小公倍数，也即 $\min\limits_{a|u,b|u} u$。

gcd 的唯一分解就是 a,b 的唯一分解中，把每个质数的幂次分别取 min。
lcm 的唯一分解就是 a,b 的唯一分解中，把每个质数的幂次分别取 max。

因为 $\min(a,b)+\max(a,b)=a+b$，所以 $\gcd(a,b)\times\operatorname{lcm}(a,b)=ab$。

数论函数

gcd 和 lcm

有结论：$u|a, u|b \rightarrow u|\gcd(a,b)$；$a|u, b|u \rightarrow \operatorname{lcm}(a,b)|u$。试着证明该结论。

同时，我们有 $\gcd(a,b) = \gcd(a, a+kb)\ (k \in \mathbb{Z})$。

因此，可以用辗转相除法求出两个数的 gcd，进而求出 lcm。

对于多个数的 gcd/lcm，只需按任意顺序依次求出。

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

注意到 $a|b \to a \le b$，所以 $\gcd(a,b) \le \min(a,b)$。进一步，若 $a \ne b$，则 $\gcd(a,b) \le |a-b|$。

另一方面，若 $a \ne b$，则 $a|b \to a \le \frac{b}{2}$，所以 $a \ne b \to \gcd(a,b) \le \frac{\min(a,b)}{2}$。

这些简单的放缩，有时可能成为题目的突破口。

若 $\gcd(a,b) = 1$，就说 a,b 互质，有时记作 $a \perp b$。互质等价于唯一分解里没有共同因子。

$a|b$ 等价于 $\gcd(a,b) = a$。

数论函数

符号 $a \equiv b \pmod c$ 表示 $c|a-b$，也就是存在正整数 k 使得 $a = b + ck$。

若 $a \equiv b \pmod c, d|c$，则也有$a \equiv b \pmod d$。换句话说，模数更小的同余式能提供更多的信息。

同余意义下也可以执行加减乘操作。对于整除操作，若 $g|a, g|b$，则

$$\frac{a}{g} \equiv \frac{b}{g} \left(\bmod \frac{c}{\gcd(c,g)} \right)$$

定义 $\varphi(n)$ 表示 $\leq n$ 的正整数中与 n 互质。可以用求和号表示为

$$\varphi(n)=\sum_{i\leq n}[i\perp n]$$

有下列计算 φ 的公式，它的本质是容斥原理：总的 - 钦定一个质因子的 + 钦定两个质因子的 - …。

若 $n=\prod p_i^{k_i}$，则

$$\varphi(n)=n\times\prod_i(1-\frac{1}{p_i})=\prod_i(p_i^{k_i}-p_i^{k_i-1})$$

积性函数

积性函数是指满足如下性质的函数：若 $a \perp b$，则 $f(ab) = f(a)f(b)$。
由于不同质数的幂次互质，所以如果知道唯一分解，积性函数的求值就只需要求出质数幂处的值，也就是

$$f(n) = \prod f(p_i^{k_i})$$

类似地，完全积性函数是指，$\forall a, b, f(ab) = f(a)f(b)$ 的函数。
此时，只需知道质数处的值就能求出任意 $f(n)$：

$$f(n) = \prod f(p_i)^{k_i}$$

前面提到的 σ_k, φ 都是积性函数，但不是完全积性函数。

积性函数

事实上，积性函数远不止上述提到的几个。很多多元函数固定一个变量，对另一个变量也是积性的：例如，固定 X，令 $f(n) = \gcd(X, n)$，则 f 也是积性函数。

对于部分常用的积性函数，有对应的记号表示：

- $id(n) = n, id^k(n) = n^k$
- $1(n) = 1$
- $\iota(n) = [n = 1]$

数论函数

莫比乌斯函数 $\mu(n)$ 的定义较为绕口：

- 若 $\exists k_i \geq 2$，则 $\mu(n) = 0$。
- 否则，若 n 有 u 个质因子，则 $\mu(n) = (-1)^u$。

$\mu(n)$ 有如下性质：

$$\sum_{d|n} \mu(d) = [n = 1]$$

试着证明！
实际上就是二项式定理，或者也可以理解为 -1 和 1 相消。
莫比乌斯函数的重要用途是“莫比乌斯反演”，但我们今天不涉及。

筛法

埃式筛

我们希望求出 $2 \sim n$ 的每一个数是不是质数。

普通筛法的想法是，枚举每个正整数 i 的倍数 $2i, 3i, \ldots$ 并把他们标记为非质数。

思考：这段代码的时间复杂度如何表达？

可以写为 $\sum_i \frac{n}{i} = n \times \sum_{i \le n} \frac{1}{i} = O(n \log n)$ 。

不过，如果注意到只枚举质数作为 i 就够了，可以将复杂度变为 $O(n \log \log n)$，这个复杂度作为结论记住就够了。
此时的筛法被称为埃式筛。

```
for (int i = 2; i <= n; i++) {
    for (int j = i + i; j <= n; j += i) {
        vst[j] = 1;
    }
}
```

```
for (int i = 2; i <= n; i++) {
    if (vst[i]) continue;
    for (int j = i + i; j <= n; j += i) {
        vst[j] = 1;
    }
}
```

即使是 $\log\log$ 复杂度，也不是线性的。能不能在 $O(n)$ 时间内求出 $2\sim n$ 每个数的是否是素数的情况？

埃式筛的症结是，每个非质数被筛了多次，具体地，被筛了质因子个数次。

线性筛中，每个非质数只会被最小质因子筛一次，保证了 $O(n)$ 复杂度。换句话说，我们希望埃式筛中，质数 i 枚举到的 j 总满足 j 的最小质因子是 i。可行吗？

这并不容易，因为“最小质因子为 i 的数”不好枚举。

那就换一种方式：不要求 i 是质数，但要求 $\frac{j}{i}$ 是质数，而且是最小质因子。

筛法

那就换一种方式：不要求 i 是质数，但要求 $\frac{j}{i}$ 是质数，而且是最小质因子。

对于每个 i，枚举 p_j，希望 $i \times p_j$ 的最小质因子为 p_j，此时怎么判断？

从小到大枚举 p_j，到 $p_j | i$ 时停止，就可以了！

```
for (int i = 2; i <= n; i++) {
    if (!vst[i]) {
        p[++cnt] = i;
    }
    for (int j = 1; j <= cnt && i * p[j] <= n; j++) {
        vst[i * p[j]] = 1;
        if (i % p[j] == 0) break;
    }
}
```

线性筛获得的信息

当执行斜体这一句代码时，我们已经获知了 $i \times p_j$ 的最小质因子就是 p_j。

记 $mn[i \times p_j] = p_j$，通过不停除掉 mn，可以在 $O(w(n))$ 时间内求出任何 $\leq n$ 的数的质因子分解。（w 表示质因子个数）

进一步，加上 dfs，还能在 $O(d(n))$ 复杂度内还原出任一数 n 的所有因数。

```
for (int i = 2; i <= n; i++) {
    if (!vst[i]) {
        p[++cnt] = i;
    }
    for (int j = 1; j <= cnt && i * p[j] <= n; j++) {
        vst[i * p[j]] = 1;
        if (i % p[j] == 0) break;
    }
}
```

筛法

线性筛积性函数

进一步，线性筛筛到 n 的过程，其实就是从大到小一个一个添加质因子的过程。

通过记录 mn（最小质因子），mnk（最小质因子的次数），我们就可以在 $O(n)$ 时间内筛出任何积性函数在 $1 \sim n$ 的所有值！

在筛 $i \times p_j$ 时，

1. 如果 p_j 不是 i 的最小质因子（也就是 p_j 不整除 i），$f(ip_j) = f(i)f(p_j)$。
2. 否则，$f(ip_j) = f\left(\frac{i}{p_j^{mnk_i}}\right) \times f(p_j^{mnk_i+1})$。

筛法

线性筛

线性筛积性函数

线性筛约数个数：

```
for (int i = 2; i <= n; i++) {
    if (!vst[i]) {
        p[++cnt] = i, d[i] = 2, mnk[i] = 1;
    }
    for (int j = 1; j <= cnt && i * p[j] <= n; j++) {
        vst[i * p[j]] = 1;
        if (i % p[j] == 0) {
            mnk[i * p[j]] = mnk[i] + 1;
            d[i * p[j]] = d[i] / (mnk[i] + 1) * (mnk[i] + 2);
            break;
        } else {
            mnk[i * p[j]] = 1;
            d[i * p[j]] = d[i] * 2;
        }
    }
}
```

筛法

线性筛积性函数

对于求质数处的值或者质数幂处的值不是 $O(1)$ 而是 $O(f(n))$ 的函数，线性筛的复杂度是 $O(n+nf(n)/\log n)$，因为质数（及其幂）有 $O(n/\log n)$ 个。

例如，在 $O(n)$ 内求 $1^n,2^n,\dots,n^n$ 就可以线性筛。
（节约快速幂的 log！）

再如，给定 x 在 $O(n)$ 内求 $\gcd(i,x)$ $(1\le i\le n)$ 就可以线性筛。
（节约 gcd 的 log！）

筛法

线性筛

线性筛积性函数

线性筛 $\gcd(i, x)$:

（同时也代表了线性筛一般函数）

```
for (int i = 2; i <= n; i++) {
    if (!vst[i]) {
        p[++cnt] = i, pk[i] = i;
    }
    if (pk[i] == i) {
        val[i] = gcd(i, x); // i 为质数幂
    }
    for (int j = 1; j <= cnt && i * p[j] <= n; j++) {
        vst[i * p[j]] = 1;
        if (i % p[j] == 0) {
            pk[i * p[j]] = pk[i] * p[j];
            val[i * p[j]] = val[i] / val[pk[i]] * val[pk[i] *
p[j]];
            break;
        } else {
            pk[i * p[j]] = p[j];
            val[i * p[j]] = val[i] * val[p[j]];
        }
    }
}
```

exgcd

exgcd

exgcd 的目的是解方程 $ax+by=\gcd(a,b)$。我们将构造性证明，该方程一定有解。

当 $b=0$ 时，$(1,0)$ 是解；否则，假设 $bx'+(a \bmod b)y'=\gcd(a,b)$，令 $a \bmod b=a-cb$，左侧就是

$$x'b+y'a-cby'=ay'+b(x'-cy')$$

故只需递归求解 $(b, a \bmod b)$，再令 $x=y', y=x'-cy'$。

可以证明，$|x|,|y| \le \max(|a|,|b|)$。

```
void Exgcd(int a, int b, int &x, int &y) {
    if (!b) return x = 1, y = 0, void();
    int xx, yy;
    Exgcd(b, a % b, xx, yy);
    x = yy, y = xx - (a / b) * yy;
}
```

exgcd

逆元

exgcd 的目的是解方程 $ax + by = \gcd(a, b)$。

将 x 加上 $k \times b/\gcd(a, b)$，y 减去 $k \times a/\gcd(a, b)$，即可得到方程的全部解。由此可求出给定范围内的解数/最小解等，如 P5656。

特别地，当 $a \perp b$ 时，相当于 $ax \equiv 1 \pmod y$。此时，把 a 叫做 x 模 y 的逆元，写作 $a \equiv x^{-1} \pmod y$。

上述推导也说明，exgcd 是求逆元的一种方法。

exgcd

不定方程

exgcd 的目的是解方程 $ax+by=\gcd(a,b)$。

设 $\gcd(a,b)\mid c$，x,y 都乘上 $c/\gcd(a,b)$，就得到 $ax+by=c$ 的一组解；还可以用前述通解性质来缩小 x,y 的绝对值。

简单的例题：P1082，P2613

试看看！

青蛙的约会 P1516

两只青蛙在长为 n 的圆环上跳动，第一只青蛙 t 时刻在位置 $a+bt$，第二只青蛙 t 时刻在 $c+dt$，求第一次相遇的整数时刻。

设时刻为 t，就是要

$$a+bt \equiv c+dt \pmod n$$
$$a-c \equiv (d-b)t \pmod n$$
$$a-c+(d-b)t+kn=0$$

可以将 t,k 视为未知数解方程。

exgcd

合并同余方程

设 $x \equiv a_1 \pmod{p_1}, x \equiv a_2 \pmod{p_2}$ 同时满足，能不能把两个式子合为一个更大的同余式?

$$x = kp_1 + a_1$$
$$x = up_2 + a_2$$

将 k, u 视为未知数，可得 k 的通解：

$$k \equiv k_0 \left(\bmod \frac{p_2}{\gcd(p_1, p_2)}\right)$$

因此

$$x \equiv k_0 p_1 + a_1 \pmod{\operatorname{lcm}(p_1, p_2)}$$

这就是所谓的 excrt。

exgcd

合并同余方程

$$x \equiv k_0 p_1 + a_1 \pmod{\operatorname{lcm}(p_1, p_2)}$$

excrt 可以把若干个同余方程合并为一个模数为所有模数 lcm 的同余方程。

简单的例题：P1495

试看看！

屠龙勇士 P4774

有 n 条龙，第 i 条龙初始生命值为 a_i，恢复力为 p_i。
你有 n 把一次性剑，第 i 把剑攻击力为 b_i。

你会按照编号依次攻击龙，攻击每条龙时，选择当前拥有的 $b_j \le a_i$ 的剑中，b_j 最大的一把。若不存在，则选择 b_j 最小的一把。然后，连续攻击第 i 条龙 X 次，造成它的生命值减少 b_jX；然后，它的生命值每时刻回复 p_i。若某个时刻生命值为 0，则龙死亡。

你需要选定 X，使得所有龙都会死亡。

就是要 $b_jX \ge a_i$ 且 $b_jX \equiv a_i \pmod{p_i}$。

试看看！

屠龙勇士 P4774

有 n 条龙，第 i 条龙初始生命值为 a_i，恢复力为 p_i。
你有 n 把一次性剑，第 i 把剑攻击力为 b_i。

你需要选定 X，使得 $b_jX \geq a_i$ 且 $b_jX \equiv a_i \pmod{p_i}$。

$b_jX \geq a_i$，只需求 $ceil(b_j/a_i)$ 最大值，并在最后解同余方程时要求解大于这个最大值。

$b_jX \equiv a_i \pmod{p_i}$ 就是 $Xb_j + p_i n = a_i$，解出 X 的通解，就化为了 $X \equiv P \pmod{Q}$ 的形式了，然后合并同余方程。

exgcd

逆元的性质

逆元具有唯一性。

逆元是一一对应关系，$(x^{-1})^{-1} = x$。

例子 (威尔逊定理)：$(p-1)! \bmod p$ 等于多少？

$p-1$。
除了 1 和 $p-1$，剩下的和逆元两两配对。

费马小定理和欧拉定理

费马小定理和欧拉定理

定理叙述如下：若底数和模数互质，则

$$x^{p-1} \equiv 1 \pmod p$$
$$x^{\varphi(n)} \equiv 1 \pmod n$$

这也说明在底数是质数时，x 的逆元就是 x^{p-2}。

我们还有所谓的“拓展欧拉定理”：任意 x，当 $b \geq \varphi(n)$ 时

$$x^b \equiv x^{(b \bmod \varphi(n))+\varphi(n)} \pmod n$$

换句话说，$\varphi(n)$ 是幂次取模后的值的循环节长度。（当然，不一定是最小循环节长度）

小结

今天我们学习了：

1. 同余的基本性质。
2. 常见积性函数的定义及求法。
3. exgcd，excrt 算法。
4. 逆元的定义及求法。

到此为止的数论知识都较为简单，需要注意以下几点：

1. 分质因子考虑问题是常见的思维方式，例如摸底测试第二题。
2. 线性筛可以筛任何积性函数，方法就是记录最小质因子的幂次。
3. 许多同余问题都可以转化为不定方程，exgcd 求通解。

求逆元

多个数的逆元

如果要求 $a_1, a_2, \ldots, a_n$ 所有数的逆元，有一种巧妙的方式 $O(n + \log mod)$ 求出。

1. 求 s_n 表示 a 的前缀积。
2. 求 $t_n = s_n^{-1}$（只算 t_n 一项，并用 $t_{n-1} = t_n a_n$ 往前递推）
3. $a_n^{-1} = t_n \times s_{n-1}$。

例题：P5431。这里的 k^i 可以递推，保证整个算法都是线性的。

类似的做法可以用来求 $1 \sim n$ 的阶乘逆元，并用来求组合数 $C(n, m)$。当然，这要求模数是质数（或者所有质因子都很大，阶乘和模数还是互质的）。

求组合数

卢卡斯定理

卢卡斯定理的表述是，$C(n,m) \bmod p$ 的值，等于写出 n,m 在 p 进制下的每一位，把每一位的组合数值分别求出，再相乘。

$$C(n,m) = C\left(\frac{n}{p},\frac{m}{p}\right) C(n \bmod p, m \bmod p)$$

这也说明，必须 n 在 p 进制下每一位都大于 m 的对应位，$C(n,m)$ 才非 0；特别地，若 $p = 2$，就是说 m 被 n 包含。

试看看！

古代猪文 P3518

求 $g^{\sum_{d|n} C(n,d)} \bmod 999911659$。$n \le 10^9$。

指数上的大数怎么办？

欧拉定理，只需求指数 mod 999911658。然后呢？

考察 999911658 的质因数分解：$2 \times 3 \times 4679 \times 35617$，四个数都是质数。
如果我们求出了指数模这四个质数分别的值，就可以唯一确定模 999911658 的值。

分别使用 Lucas 定理即可。

试看看！

密码 P3518

给定 n，$0,1,\ldots,n-1$ 里有些数是密码。

已知：若 x,y 都是密码，则 $(x+y) \bmod n$ 也是密码，
且 $a_1,\ldots,a_k$ 不是密码，a_0 是密码。

问：最少可能有几个密码？

$$k \le 10^6, n \le 10^{15}$$

若 g 是密码，则……

$\gcd(n,g)$ 的所有倍数都是密码。因此，“最少几个”其实就是问最小密码最大是多少。

试看看！

密码 P3518

若 x, y 都是密码，则 $(x+y) \bmod n$ 也是密码，且 $a_1, \dots, a_k$ 不是密码，a_0 是密码。

最少可能有几个密码？

$$k \le 10^6, n \le 10^{15}$$

设 g 为最小的密码，由题知 $g|\gcd(a_0, n)$，但 g 不整除 $a_1 \sim a_k$。

可以先将 a_i 与 n 求 gcd，这样就都是 n 的因数了。

用 map 标记所有不希望的 a_i 的因数，再从大到小枚举 a_0 的因数，找到第一个没被标记的即可。

试看看！

密码 P3518

若 x,y 都是密码，则 $(x+y) \bmod n$ 也是密码，且 $a_1,\ldots,a_k$ 不是密码，a_0 是密码。

最少可能有几个密码？

$$k \le 10^6, n \le 10^{15}$$

思考：时间复杂度？

这样的时间复杂度至少是 $O(d(n)^2 \log d(n))$，$d(n)$ 最大能取到 26880，无法通过。需要优化“标记不可行约数”的过程。

思考：类比普通筛法优化到埃式筛的过程，有哪些其实可以不用标记？

事实上，可以从大到小递推，若 x 被标记了，只额外标记 x/p 就够了。

试看看！

密码 P3518

若 x, y 都是密码，则 $(x+y) \bmod n$ 也是密码，且 $a_1, \dots, a_k$ 不是密码，a_0 是密码。

最少可能有几个密码？

$$k \leq 10^6, n \leq 10^{15}$$

事实上，可以从大到小递推，若 x 被标记了，只额外标记 x/p 就够了。

时间复杂度变为 $O(d(n)w(n)\log d(n))$，可以通过。

试看看！

上帝与集合的正确用法 P4139

设 $a_0 = 1, a_n = 2^{a_{n-1}}$。给定 $p \le 10^9$，可以证明 n 足够大时 a_n 模 p 是定值，求这个定值。

为了求 $a_n \bmod p$，只需求 $a_{n-1} \bmod \varphi(p)$，再加上 $\varphi(p)$ 作为指数即可（这是因为拓展欧拉定理，且 n 足够大）。

依此类推，只需求 $a_{n-2} \bmod \varphi\big(\varphi(p)\big), \ldots$，直到 φ 嵌套足够多层后 p 变成 1，此时 $a_{n-\ldots}$ 的具体值已经不重要了。

思考：如何分析上述算法的时间复杂度？

递归两层，p 至少减小一半

试看看！

前缀 XOR ARC137D

给定 $a_{0,i}$ $(1 \le i \le n)$ 的值。定义 $a_{k,i}$ 是 $a_{k-1,1}, \dots, a_{k-1,i}$ 这些数的异或和。

求 $a_{1,n}, a_{2,n}, \dots, a_{m,n}$。

$$m, n \le 10^6$$

提示：首先需要给问题一个合适的模型。
整个矩阵里的数，有什么性质？

它们都是由 $a_{0,i}$ 异或而来的——换句话说，它们是 $a_{0,i}$ 的一个子集的 XOR 和。
怎么知道每个数具体是哪个子集异或来的，也就是谁对它有贡献？

试看看！

前缀 XOR ARC137D

给定 $a_{0,i}$ $(1 \le i \le n)$ 的值。定义 $a_{k,i}$ 是 $a_{k-1,1}, \dots, a_{k-1,i}$ 这些数的异或和。

求 $a_{1,n}, a_{2,n}, \dots, a_{m,n}$。

$$m, n \le 10^6$$

考虑 $a_{0,i}$ 是否对 $a_{k,n}$ 有贡献，实际上是“格路计数”类似物：每步可以向下走一格再向右走任意格，走到终点的一个方案就意味着一次贡献，而偶数次贡献等价于没有贡献，所以只需求出方案数模 2。

从 $(0, i)$ 走到 (k, n) 的方案数是多少？
$C((k-1)+(n-i), k-1)$，因为除了第一步必须向下，其余是任意走

试看看！

前缀 XOR ARC137D

给定 $a_{0,i}$ $(1 \le i \le n)$ 的值。定义 $a_{k,i}$ 是 $a_{k-1,1}, \dots, a_{k-1,i}$ 这些数的异或和。

求 $a_{1,n}, a_{2,n}, \dots, a_{m,n}$。

$$m, n \le 10^6$$

$$C((k-1)+(n-i), k-1)$$

ans_k 就是所有 $C((k-1)+(n-i), k-1)$ 为奇数的位置 i 的 a_i 异或和，那如何求出？首先需要把条件中的组合数转化得更好看。

等价于 $k-1+n-i$ 包含 $k-1$，（分离变量）也就是 $n-i$ 和 $k-1$ 无交！

试看看！

前缀 XOR ARC137D

给定 $a_{0,i}$ $(1 \le i \le n)$ 的值。定义 $a_{k,i}$ 是 $a_{k-1,1}, \dots, a_{k-1,i}$ 这些数的异或和。

求 $a_{1,n}, a_{2,n}, \dots, a_{m,n}$。

$$m, n \le 10^6$$

ans_k 就是所有 $n-i$ 和 $k-1$ 二进制无交的位置 i 的 a_i 异或和。

提示：转化为 $ans‘_k$ 是 i 包含于 k 的位置的 $\mathrm{rev}(a)_i$ 异或和，再递推。这个 k 是原来的 $\sim(k-1)$。

“包含的异或和”可以按位递推，或者称为“高维前缀和”：初值就是原数组。枚举每一位 2^u，对于包含这位的 t，令 a_t 异或上 a_{t-2^u}。

小结

在解决前面几道例题的过程中，需要注意的要点有：

1. 将对 n 取模的问题用 excrt 转为对 n 的质因子（幂次）取模。
2. 处理“是因数”条件时，每次只除掉一个质因子而非枚举所有因数。“是倍数”也可以这样处理（埃式筛）。
3. 扩展欧拉定理处理幂塔。
4. 分析操作次数，转化为组合数。
5. 利用按位递推求与位运算有关的式子的值，也称作“高维前缀和”。