数论

2024 年 11 月

feecle8146

QQ：3576754855，有问题欢迎课后提问。

分解质因数

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

同余

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

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

数论函数

欧拉函数

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

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

数论函数

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

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

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。

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：

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

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。

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

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

筛法

线性筛

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

线性筛

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;
    }
}
```

筛法

线性筛

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

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，$f(ip_j) = f\left(\frac{i}{p_j^{mn\ i}}\right) \times f(p_j^{mnk_i+1})$。

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

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 $1^n, 2^n, \ldots, n^n$ 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

线性筛积性函数

线性筛 $\gcd(i,x)$：

（同时也代表了线性筛一般函数）

线性筛

```
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

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 $\gcd(a, b) | c$，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

excrt

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：

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

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

excrt

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 $Xb_j + p_in = a_i$，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；进入循环节可能有下界）

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

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 $a_1, \ldots, a_k$ 不是密码，a_0 是密码。

最少可能有几个密码？

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

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？

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

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 $a_1, \ldots, a_k$ 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

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（或者指数上很大的值）。

小结

我们现在已经学习了：

1. 唯一分解定理。
2. 许多常用的数论函数，及求出它们的方法：线性筛。
3. 不定方程的解法：exgcd，以及由此引出的逆元。
4. 合并同余方程：excrt。
5. Lucas 定理处理组合数模小质数。

这些基础知识后面仍然会大量用到。接下来，我们还将学习

1. 简单的“式子处理”，这就涉及到整除分块。
2. 学习模意义下简单的指数相关问题的解法，这就涉及到 BSGS 和原根。当然，前述的（拓展）欧拉定理仍然会发挥很大的作用。

整除分块

整除分块

约数和 P2424

给定 X,Y，求 $\sum_{X\le i\le Y} d(i)$。$Y \le 2\times 10^9$

只需求出前缀和 $\sum_{i\le n} d(i)$，再相减。

尝试化简上式？提示：将 d 拆成一个求和号，再交换。

$$\sum_{i\le n} d(i) = \sum_{i\le n}\sum_{j|i} 1$$

$$= \sum_{j\le n}\sum_{j|i} 1 = \sum_{j\le n} floor\left(\frac{n}{j}\right)$$

整除分块

整除分块的用途就是求 $\sum_{j\le n} floor\left(\frac{n}{j}\right)$ 这样的式子的值。为了简便，我们以后都省略 floor。事实上，它可以求任何 $\sum_{j\le n} g(j)\times f\left(\frac{n}{j}\right)$，其中 f 是好算的函数，而 $g(j)$ 的前缀和好算。

$\frac{n}{j}$ 可以有几种取值？提示：根号分治。

当 $j\le\sqrt{n}$ 时，j 只有 $\sqrt{n}$ 个；
当 $j>\sqrt{n}$ 时，$\frac{n}{j}\le\sqrt{n}$，只有 $\sqrt{n}$ 个。

因此，只有 $\le 2\sqrt{n}$ 种取值；换句话说，我们只需计算 $O(\sqrt{n})$ 种 $f(x)$，然后把每种 $f(x)$ 累加对应的次数。

整除分块

整除分块

有两种实现方法：

1. 直接实现根号分治的代码，讨论 $x \le \sqrt{n}, x > \sqrt{n}$，并分别算出这个 x 对应几个 j。由于较为繁琐，在此不提。
2. 仍然枚举 j，但是注意到 $\frac{n}{j}$ 是分段相等且单调递减的，因此可以一次枚举一段。

我们（以及主流 OI 界）将采用第二种实现。这就需要知道，当我们枚举到 j 时，下一步应当跳到哪里，也就是最大的 k 使得 $\frac{n}{k} = \frac{n}{j}$。

试着自己推出 k 的表达式？

$$\frac{n}{floor\left(\frac{n}{j}\right)}$$

整除分块

整除分块

枚举 j，但是注意到 $\frac{n}{j}$ 是分段相等且单调递减的，因此可以一次枚举一段。

当我们枚举到 j 时，下一步应当跳到哪里，也就是最大的 k 使得 $\frac{n}{k}=\frac{n}{j}$。

$$k=\frac{n}{floor\left(\frac{n}{j}\right)}$$

```
for (int i = 1, j; i <= n; i = j + 1) {
    j = n / (n / i);
    ans += (j - i + 1) * f(n / i);
}
```

试看看！

余数求和 P2261

求 $\sum_{i\le n}(k \bmod i)$，其中 $n,k \le 10^9$。

首先把 $k \bmod i$ 写成 $k-\frac{k}{i}\times i$，然后

$$ans = nk - \sum_{i\le n} i \times \frac{k}{i}$$

细节：

1. 代码里的 j 要和 n 取 min。
2. 当 $\frac{k}{i}=0$ 时，j 的真实值为无穷大，但在这里应该设为 n。

整除分块相关练习题：CF1706D2

试看看！

模积和 P2260

求 $\sum_{i\le n,j\le m,i\neq j}(n \bmod i)(m \bmod j)$，其中 $n,m \le 10^9$。

不妨假设 $n \le m$，则可以写成：所有 i,j，减去 $i=j$ 的。（这样有什么好处？）

拆式子的过程同上题，最终会出现同时含有 $\frac{n}{i},\frac{m}{i}$ 的表达式。如何处理？

提示：分界点的个数乘 2 后还是 $O(\sqrt{n})$ 的。

```
for (int i = 1, j; i <= n && i <= m; i = j + 1)
{
    j = min(n / (n / i), m / (m / i));
    ans += ... * f(n / i, m / i);
}
```

原根　　阶

阶

使得 $x^n \equiv 1 \pmod p$ 的最小正整数 n 称为 x 模 n 的阶，记作 $|x|$。

显然，阶存在的充要条件是 $\gcd(x,p)=1$。

所有 $x^n \equiv 1 \pmod p$ 的 n 都是阶的倍数。这是因为阶也满足“辗转相除法”的性质。

结合拓展欧拉定理，可得 $|x| \mid \varphi(p)$。

阶是 x 的幂模 p 的最小循环节。由此可以推出，$\left|x^k\right| = \frac{|x|}{\gcd(k,|x|)}$。

（可以画图理解）

原根

原根

原根

结合拓展欧拉定理，可得 $|x| \mid \varphi(p)$。

若 $|x| = \varphi(p)$，就说 x 是模 p 的一个原根。此时，$x^0, \ldots, x^{\varphi(p)-1}$ 不重不漏地取遍了所有 $< p$ 的与 p 互质的数。

结论：有原根的数只有 $2, 4, p^k, 2p^k$。且如果存在原根，最小原根不会超过 $n^{1/4}$，所以求原根的方法就是从小到大枚举，并判断阶是否为 $\varphi(p)$。

如何快速判断？

根据阶的性质，只需求出 $\varphi(p)$ 的所有质因子 q，并判断 $x^{\frac{\varphi(p)}{q}}$ 是否是 1。已知 $p, \varphi(p)$ 前提下，单次判断时间复杂度 $O(\log p\, w(p))$。

原根

所有原根

根据 $\left|x^k\right| = \frac{|x|}{\gcd(k,|x|)}$ 以及原根的次幂取遍所有与 p 互质的数，可以推出：

若 g 是某个原根，则所有原根就是 g^k，其中 $k \perp \varphi(p)$。

这说明原根若存在，则有 $\varphi(\varphi(p))$ 个。

练习题：P6091

原根 阶

求阶

求 $|x|$ 的暴力方法就是枚举 $\varphi(p)$ 的因数看每个因数满不满足 $x^k = 1$，利用的性质太少了。

能不能利用 $|x|$ 的倍数都满足 $= 1$ 的性质？

可以使用“试除法”：依次试图除掉每个质因子，若除掉之后 x^k 不再是 1 了，说明不能除。

至多试除 $O(\log p)$ 次，在已知 $p, \varphi(p)$ 的前提下复杂度是 $O(\log^2 p)$。

一个没什么关系的例题：P8993

BSGS

BSGS

求解指数同余方程

上面的“求阶”，其实是求阶指数同余方程的特殊情况。

指数同余方程的一般形式为，求下面方程 x 最小正整数解：

$$a^x \equiv b \pmod{p}$$

我们今天只解决 $\gcd(a,p)=1$ 的情况。

BSGS 算法的思想就是 meet in the middle。取块长 B，设 $x=iB-j$，则 $a^x \equiv b$ 等价于 $(a^B)^i \equiv b \times a^j$。

将 $(a^B)^i$ 放入哈希表内（若有重复只保留 i 最小的），再依次查询 $b \times a^j$。

BSGS

BSGS

求解指数同余方程

将 $(a^B)^i$ 放入哈希表内（若有重复只保留 i 最小的），再依次查询 $b \times a^j$。

由于 $x \leq \varphi(p)$ 而 $\varphi(p) = O(p)$，所以取 $B = \sqrt{n}$ 可做到复杂度 $O(\sqrt{n})$。

特别地，如果固定 a, p 有多组 b，则只需要一次预处理（复杂度 $O\left(\frac{n}{B}\right)$），单次查询复杂度 $O(B)$，平衡复杂度后可做到总共 $O(\sqrt{nq})$。

由于在实数意义下，求 $a^x = b$ 的运算是“对数”，所以指数同余方程问题又称为离散对数问题。

试看看！

随机数生成器 P3306

给定 x_1, a, b, p，保证 p 是质数，$x_i = ax_{i-1} + b$。

求最小的 i 使得 $x_i \equiv b \pmod{p}$。

利用等比数列求和公式化简，再分离变量 a^i，就转化为 $a^i \equiv c \pmod{p}$ 了。

试看看！

前缀离散对数 经典问题

给定 a,p，对于每个 $b=1,2,\ldots,T$，求方程 $a^x \equiv b \pmod p$ 的最小解，保证 a,p 互质。

直接用 BSGS 是 $O(\sqrt{pT})$ 的。能不能利用 b 是前缀的性质，取得再优秀一点的复杂度？

提示：若 $a^{x_1} \equiv b_1, a^{x_2} \equiv b_2$，则 $a^{x_1+x_2} \equiv b_1 \times b_2$。

离散对数满足 $f(ab)=f(a)+f(b)$，可以线性筛！
只需知道质数处的值，就能筛出所有值（不一定最小）。

时间复杂度 $O(\sqrt{pT/\log T})$。当然，筛出的值不一定最小，所以还需要先算出 a 的阶，然后把答案对它取个模。

试看看！

心跳 Codechef CHEFMOD

固定 $P = 10^8 + 7$，给定 $1 \le a < P$，定义无穷序列 $a_i = a^i \bmod p$。求 a_i 的所有前缀最小值之和。

有 T 次询问，$T \le 500$。

先考虑一次询问怎么做。

提示：

1. “取模”和“比大小”两个操作看起来毫无关系。
2. i 第一次在序列中出现的位置就是 i 以 a 为底的离散对数。

可以认为模意义下的序列 a^i 几乎是随机的！

试看看！

心跳 Codechef CHEFMOD

固定 $P = 10^8 + 7$，给定 $1 \le a < P$，定义无穷序列 $a_i = a^i \bmod p$。求 a_i 的所有前缀最小值之和。

有 T 次询问，$T \le 500$。

由随机性（类比随机排列），我们可以认为经过足够多（不太多）次之后，前缀最小值已经很小了。

$O(i)$ 次后最小值大约是 $O\left(\frac{P}{i}\right)$，特别地，$O(\sqrt{P})$ 次后最小值就该 $\le 10^4$ 了。

这很难严谨证明。不过，穷举所有 $1 \le a < P$ 后可以发现是对的。

试看看！

心跳 Codechef CHEFMOD

固定 $P = 10^8 + 7$，给定 $1 \le a < P$，定义无穷序列 $a_i = a^i \bmod p$。求 a_i 的所有前缀最小值之和。

有 T 次询问，$T \le 500$。

暴力计算 $O(\sqrt{n})$ 项，后面只有 10^4 种不同的数。

用离散对数求出每种数 i 的出现位置，就能够回答询问了！这样，单次时间复杂度为 $\sqrt{10^4 P / \log 10^4}$。

如何拓展到所有 a？提示：取原根 g，设 $a = g^k$。

试看看！

心跳 Codechef CHEFMOD

固定 $P = 10^8 + 7$，给定 $1 \le a < P$，定义无穷序列 $a_i = a^i \bmod p$。求 a_i 的所有前缀最小值之和。

有 T 次询问，$T \le 500$。

取原根 g，设 $a = g^k$。

假设现在要求 i 在 $\{a^u\}$ 这一序列里出现的位置。先用 BSGS 预处理 i 在 g^k 序列里第一次出现位置 p，则其出现的所有位置就是 $p + j(P-1)$ $(j \in \mathrm{N})$；又因为 $a^u = g^{ku}$，所以只需求出最小的 j，使得 $k | p + j(P-1)$。

这就是二元不定方程，exgcd 即可，注意这里的 exgcd 可以做到 $O(1)$ 。

小结

小结

“心跳”这道例题是原根处理指数问题的极好范例，我们将其中的思想总结如下：

1. 增长得很快的函数在模意义下的值可以认为是伪随机数。
2. 原根和阶两个概念使我们完全搞清楚了（有原根前提下）任何一个与 mod 互质的数 a 的所有次幂在模 mod 意义下的出现规律。

因此，我们得以将大部分与次幂的值相关的问题转化为关于指数的同余方程问题。

小结

拓展：原根与群的同构

如果你了解过“群”的语言，你应当看出，原根的作用实际上是给出了模 p 缩系下的乘法群和模 $\varphi(p)$ 意义下的加法群的同构——这两个群都是元素个数为 $\varphi(p)$ 的循环群。

乘法群是我们不熟悉的，但循环群（可以想象为圆环上跳动）是我们熟悉的，而且有同余方程等各种工具来处理。例如，加法群的一个生成元是 1，乘法群的一个生成元是 g。换句话说，g 在乘法里的作用约等于 1 在加法里的作用。

知识总结

到此为止，我们的课程已经涵盖了联赛难度的所有数论知识。然而，在这些简单的知识基础上，仍然可以出出千变万化的难题、趣题。

用于求解同余方程的 exgcd 是基础。

当出现多个同余方程时，使用 excrt（本质就是 exgcd）来合并。

指数上的问题，如果还没有到要考虑阶的地步，（拓展）欧拉定理也可以将其转为同余问题。

与整除相关的求和，数论分块来计算。

指数上更难的问题，可以考虑原根，并合理利用阶的性质。

试看看！

幂塔方程（简单版） P8457

解方程 $x^x \equiv a \pmod{p}$，其中 p 是质数。只需求出任意一个 $\leq 10^{18}$ 的解或判断无解。

$$p \leq 10^9$$

提示：
固定底数，指数以 $p-1$ 为周期。
固定指数，底数以 p 为周期。

$(p, p-1) = 1$，所以可以……
同时固定底数和指数！

试看看！

求和 CF1717E

求 $\sum_{a+b+c=n} \operatorname{lcm}(c, \gcd(a,b))$，其中 $n \le 10^5$。

回忆处理求和号的思路：尽量让枚举的变量独立，转化为子问题。

你认为应该先处理哪个变量？

枚举 c，和式变为

$$\sum_{c \le n} \sum_{a+b=n-c} \operatorname{lcm}(c, \gcd(a,b))$$

你认为接下来应该枚举什么变量？

试看看！

求和 CF1717E

求 $\sum_{a+b+c=n} \operatorname{lcm}(c, \gcd(a,b))$，其中 $n \le 10^5$。

回忆处理求和号的思路：尽量将枚举的变量独立，转化为子问题。

$$\sum_{c \le n} \sum_{a+b=n-c} \operatorname{lcm}(c, \gcd(a,b))$$

直接枚举 $\gcd(a,b) = d$，要求 $d | n-c$：

$$\sum_{c \le n} \sum_{d | n-c} \operatorname{lcm}(c,d) f(d, n-c)$$

其中 $f(d)$ 表示：有多少对 $a+b=n$ 满足 $\gcd(a,b)=d$。怎么处理 f？

试看看！

求和 CF1717E

求 $\sum_{a+b+c=n} \operatorname{lcm}(c, \gcd(a,b))$，其中 $n \le 10^5$。

有多少对 $a+b=n$ 满足 $\gcd(a,b)=d$。

注意到 $\gcd(a,b)=\gcd(a,a+b)=\gcd(a,n)$，所以就是有多少个 $a<n$ 满足 $\gcd(a,n)=d$。

$\gcd(a,n)=d$，其实就是 $\gcd\left(\frac{a}{d},\frac{n}{d}\right)=1$！

所以，满足条件的 a 有 $\varphi\left(\frac{n}{d}\right)$ 个。

直接套前述公式计算，复杂度 $O(n\log n)$。

小结

前两题相对简单，但其中都蕴含了普适的思维方式：

1. p 是质数的话，底数和指数都有各自的循环节。如果都要控制（都出现了未知数），无非是合并一下同余方程。
2. 对于求和类问题，想清楚枚举什么变量能使问题最简单。

试看看！

我这个坏东西 AT_tenka1_2017_f

解方程 $a^x \equiv x \pmod{m}$。只需求出任意一个 $\leq 2 \times 10^{18}$ 的解，或判断解不存在。

$a, m \leq 10^9$，≤ 100 组数据。

提示：类比第一题“幂塔方程”。

提示：底数、指数的同余性质我们了如指掌，可惜幂的值我们知之甚少。怎么办？

提示：那就假设我们知道幂的值 $a^x \equiv x \equiv T \pmod{m}$，此时 x 就可以 BSGS 了。当然，BSGS 不是重点（因为其实不知道 T），重点是……

试看看！

我这个坏东西 AT_tenka1_2017_f

解方程 $a^x \equiv x \pmod{m}$。只需求出任意一个 $\leq 2 \times 10^{18}$ 的解，或判断解不存在。

$a, m \leq 10^9$，≤ 100 组数据。

假设我们知道幂的值 $a^x \equiv x \equiv T \pmod{m}$，此时 x 就可以 BSGS 了。当然，BSGS 不是重点（因为其实不知道 T），重点是，它给出了 x 应该满足的（有下界的）同余方程：$x \equiv T' \left(mod\ \varphi(m)\right)$！

（当然，x 可能很小，还没进入循环节，但我们现在只是对问题进行思考，不必纠结于边界情况）

此时你能推出什么新东西？

试看看！

我这个坏东西 AT_tenka1_2017_f

解方程 $a^x \equiv x \pmod{m}$。只需求出任意一个 $\leq 2 \times 10^{18}$ 的解，或判断解不存在。

$a, m \leq 10^9$，≤ 100 组数据。

此时，我们就知道了两个关于 x 的方程同时满足！

$$\begin{cases} x \equiv T \pmod{m} \\ x \equiv T' \pmod{\varphi(m)} \end{cases}$$

这说明 $T \equiv T' \pmod{\gcd(m, \varphi(m))}$！再看看 T' 和 T 是什么？

T 就是 $a^{T'} \bmod m$，而 $\gcd(m, \varphi(m))$ 又是 m 的约数，所以……

试看看！

我这个坏东西 AT_tenka1_2017_f

解方程 $a^x \equiv x \pmod{m}$。只需求出任意一个 $\leq 2 \times 10^{18}$ 的解，或判断解不存在。

$a, m \leq 10^9$，≤ 100 组数据。

T 就是 $a^{T'} \bmod m$，而 $\gcd\big(m, \varphi(m)\big) = g$ 又是 m 的约数，所以

$$a^{T'} \equiv T \equiv T' \pmod{g}$$

换句话说，通过模 m 的解 x，我们能推出此时的 T' 是模 g 的解。
这个过程，能不能反过来？如果已知一个模 g 的解 T'，那……

想想 T' 是怎么定义来的？

试看看！

我这个坏东西 AT_tenka1_2017_f

解方程 $a^x \equiv x \pmod{m}$。只需求出任意一个 $\leq 2 \times 10^{18}$ 的解，或判断解不存在。

已知一个模 g 的解 T'，而 T' 的定义就是 $x \equiv T' \ (mod\ \varphi(m))$，这说明 $T = a^x \bmod m$ 也随之确定了！

而已知 T', T，方程组 $\begin{cases} x \equiv T \pmod{m} \\ x \equiv T' \pmod{\varphi(m)} \end{cases}$ 也确定了，可以直接解出 x！

因此，算法流程就是不停递归 $m \to \gcd(m, \varphi(m))$，并通过上述方式回带出 x 的值。当然，为了保证扩展欧拉定理成立，需要解同余方程时取一个 $\geq 10^9$ 的解。代码：https://atcoder.jp/contests/tenka1-2017/submissions/55884596

小结

本题是笔者非常喜爱的一道题。

其具有较高的难度，但它的难度既不是繁琐的讨论，也不是莫名其妙无厘头的结论，而是有迹可循的，能踏踏实实一步一步还原出思维链条。

1. 希望把问题化为“我们熟悉”的形式，也就是要么限制指数要么限制底数。那就设出幂的值和离散对数的值，强行构造一组同余方程。
2. 构造同余方程后，发现能推出一个更小的模数满足同一形式的式子。
3. 上述都是在“已知指数和底数”前提下从答案逆推；反过来，如果只知道这个同一形式的式子的解，其实也可以还原出来设出的幂值和离散对数值！
4. 做法豁然开朗。

诸君若有兴趣，则可细细品味，或是看看另一个类似的题：P8457。