好题分享

李秉霈

长郡中学

简要题意

题目描述

- 给定整数 m, r, n, k, M。
- 求 $\{1, 2, \cdots, n\}$ 有多少个 k 元子集元素之和模 m 余 r。
- 设答案为 ans，你需要输出 $(ans \cdot m) \bmod M$。

数据范围

- $m \le 10^{11}$， $n \le 10^{18}$
- 对每个 $d \mid m$，保证 $(n \bmod d) \le (k \bmod d)$
- $M \le 2^{62}$， M 的最大质因子 $\le 10^5$
- TL 1.5s

如何刻画原问题

如何刻画原问题

考虑使用二元生成函数，以下的 ans 都是原题面中的 $ans \cdot m$。

$$ans = m[x^k y^{?m+r}] \prod_{i=1}^{n} (1 + xy^i)$$

如何刻画原问题

考虑使用二元生成函数，以下的 ans 都是原题面中的 $ans \cdot m$。

$$ans = m[x^k y^{?m+r}] \prod_{i=1}^{n} (1 + xy^i)$$

单位根反演：

$$ans = \sum_{i=0}^{m-1} \omega^{-ir} [x^k] \prod_{j=1}^{n} (1 + x \cdot \omega^{ij})$$

如何刻画原问题

考虑使用二元生成函数，以下的 ans 都是原题面中的 $ans \cdot m$。

$$ans = m[x^k y^{?m+r}]\prod_{i=1}^{n}(1 + xy^i)$$

单位根反演：

$$ans = \sum_{i=0}^{m-1} \omega^{-ir}[x^k]\prod_{j=1}^{n}(1 + x \cdot \omega^{ij})$$

接下来为了更快的提取 k 次项，我们需要分析这个乘式的性质。

分析乘式性质

分析乘式性质

前置知识：$\prod_{i=0}^{n-1}(x-\omega_n^i)=x^n-1$。

分析乘式性质

前置知识：$\prod_{i=0}^{n-1}(x-\omega_n^i)=x^n-1$。
原乘式相当于在（类似）这个式子的基础上划分为了若干周期。

分析乘式性质

前置知识：$\prod_{i=0}^{n-1}(x-\omega_n^i)=x^n-1$。
原乘式相当于在（类似）这个式子的基础上划分为了若干周期。
具体的，设 $d=\gcd(i,m), m'=m/d$：

$$\prod_{j=1}^{n}(1+x\cdot\omega^{ij})=\prod_{j=1}^{n}(1+x\cdot\omega_{m'}^{(i/d)j})$$

$$=\left(\prod_{j=0}^{m'-1}(1+x\cdot\omega_{m'}^{j})\right)^{n/m'}\prod_{j=1}^{n \bmod m'}(1+x\cdot\omega_{m'}^{(i/d)j})$$

分析乘式性质

分析乘式性质

$$\left(\prod_{j=0}^{m'-1}(1+x\cdot\omega_{m'}^{j})\right)^{n/m'}\prod_{j=1}^{n \bmod m'}(1+x\cdot\omega_{m'}^{(i/d)j})$$

分析乘式性质

$$\left(\prod_{j=0}^{m'-1}(1+x\cdot\omega_{m'}^{j})\right)^{n/m'}\prod_{j=1}^{n \bmod m'}(1+x\cdot\omega_{m'}^{(i/d)j})$$

由前置知识可知，前面部分对 x 指数的贡献只能是 m' 或 0。

分析乘式性质

$$\left(\prod_{j=0}^{m'-1}(1+x\cdot\omega_{m'}^{j})\right)^{n/m'}\prod_{j=1}^{n \bmod m'}(1+x\cdot\omega_{m'}^{(i/d)j})$$

由前置知识可知，前面部分对 x 指数的贡献只能是 m' 或 0。
故当 $(k \bmod m') > (n \bmod m')$ 时，该式的 x^k 项系数一定为 0。

分析乘式性质

$$\left(\prod_{j=0}^{m'-1}(1+x\cdot\omega_{m'}^{j})\right)^{n/m'}\prod_{j=1}^{n \bmod m'}(1+x\cdot\omega_{m'}^{(i/d)j})$$

由前置知识可知，前面部分对 x 指数的贡献只能是 m' 或 0。
故当 $(k \bmod m') > (n \bmod m')$ 时，该式的 x^k 项系数一定为 0。
否则我们有 $(k \bmod m') = (n \bmod m')$，系数为

$$\omega_{m'}^{\binom{k+1}{2}(i/d)}\binom{n/m'}{k/m'}$$

化简原式

化简原式

现在我们已经会更快的提取乘式的 k 次项了。考虑答案式子:

$$ans = \sum_{i=0}^{m-1} \binom{n/m'}{k/m'} \omega_{m'}^{(\binom{k+1}{2}-r)(i/d)} [(k \bmod m') = (n \bmod m')]$$

$$= \sum_{d|m} [(k \bmod d) = (n \bmod d)] \binom{n/d}{k/d} \sum_{i \perp d, 1 \le i \le d} \omega_d^{(\binom{k+1}{2}-r)i}$$

化简原式

现在我们已经会更快的提取乘式的 k 次项了。考虑答案式子:

$$ans=\sum_{i=0}^{m-1}\binom{n/m'}{k/m'}\omega_{m'}^{(\binom{k+1}{2}-r)(i/d)}[(k \bmod m')=(n \bmod m')]$$

$$=\sum_{d|m}[(k \bmod d)=(n \bmod d)]\binom{n/d}{k/d}\sum_{i\perp d,1\le i\le d}\omega_d^{(\binom{k+1}{2}-r)i}$$

那么我们相当于需要计算 d 的所有本原单位根的某次方之和。

化简原式

现在我们已经会更快的提取乘式的 k 次项了。考虑答案式子:

$$ans = \sum_{i=0}^{m-1} \binom{n/m'}{k/m'} \omega_{m'}^{(\binom{k+1}{2}-r)(i/d)} [(k \bmod m') = (n \bmod m')]$$

$$= \sum_{d|m} [(k \bmod d) = (n \bmod d)] \binom{n/d}{k/d} \sum_{i \perp d, 1 \le i \le d} \omega_d^{(\binom{k+1}{2}-r)i}$$

那么我们相当于需要计算 d 的所有本原单位根的某次方之和。接下来我们研究一下这个怎么求。

本原单位根之和

本原单位根之和

让我们先研究一个简单一点的情形：求 n 的本原单位根之和。

本原单位根之和

让我们先研究一个简单一点的情形：求 n 的本原单位根之和。

$$
\begin{aligned}
\sum_{i=1,i\perp n}^{n} \omega_n^i &= \sum_{i=1}^{n} \epsilon(\gcd(i,n))\omega_n^i \\
&= \sum_{i=1}^{n} \sum_{d|i,d|n} \mu(d)\omega_n^i \\
&= \sum_{d|n} \mu(d) \sum_{i=1}^{n/d} \omega_{n/d}^i \\
&= \mu(n)
\end{aligned}
$$

本原单位根之和

让我们先研究一个简单一点的情形：求 n 的本原单位根之和。

$$
\begin{aligned}
\sum_{i=1, i\perp n}^{n} \omega_n^i &= \sum_{i=1}^{n} \epsilon(\gcd(i,n))\omega_n^i \\
&= \sum_{i=1}^{n} \sum_{d|i, d|n} \mu(d)\omega_n^i \\
&= \sum_{d|n} \mu(d) \sum_{i=1}^{n/d} \omega_{n/d}^i \\
&= \mu(n)
\end{aligned}
$$

非常的简洁！

本原单位根 k 次方和

本原单位根 k 次方和

这个时候相当于在上一页的基础上划分为了若干周期。

本原单位根 k 次方和

这个时候相当于在上一页的基础上划分为了若干周期。
令 $d=\gcd(n,k)$，那么有

$$\sum_{i=1,i\perp n}^{n}\omega_n^{ik}=\frac{\mu(n/d)\varphi(n)}{\varphi(n/d)}$$

最终式子

最终式子

令 $k' = \binom{k+1}{2} - r, d' = \gcd(d, k')$，带回原式得到:

$$ans = \sum_{d|m} [(k \bmod d) = (n \bmod d)] \binom{n/d}{k/d} \frac{\mu(d/d')\varphi(d)}{\varphi(d/d')}$$

最终式子

令 $k' = \binom{k+1}{2} - r, d' = \gcd(d, k')$，带回原式得到：

$$ans = \sum_{d|m} [(k \bmod d) = (n \bmod d)] \binom{n/d}{k/d} \frac{\mu(d/d')\varphi(d)}{\varphi(d/d')}$$

这样我们就得到了最终的式子。m 所有因数的 μ 和 φ 可以通过 dfs 质因子 $\mathcal{O}(\sqrt{m})$ 预处理好。

最终式子

令 $k' = \binom{k+1}{2} - r, d' = \gcd(d, k')$，带回原式得到：

$$ans = \sum_{d|m} [(k \bmod d) = (n \bmod d)] \binom{n/d}{k/d} \frac{\mu(d/d')\varphi(d)}{\varphi(d/d')}$$

这样我们就得到了最终的式子。m 所有因数的 μ 和 φ 可以通过 dfs 质因子 $\mathcal{O}(\sqrt{m})$ 预处理好。
所以我们现在需要解决的问题只剩下：求 d(m) 次 10^{18} 级别的组合数，对 M 取模。

最终式子

令 $k' = \binom{k+1}{2} - r, d' = \gcd(d, k')$，带回原式得到：

$$ans = \sum_{d|m} [(k \bmod d) = (n \bmod d)] \binom{n/d}{k/d} \frac{\mu(d/d')\varphi(d)}{\varphi(d/d')}$$

这样我们就得到了最终的式子。m 所有因数的 μ 和 φ 可以通过 dfs 质因子 $\mathcal{O}(\sqrt{m})$ 预处理好。
所以我们现在需要解决的问题只剩下：求 d(m) 次 10^{18} 级别的组合数，对 M 取模。
解决方向是明确的：对 M 质因数分解，对每个 p^e 求出答案，最后 CRT 回去。那么我们需要重点利用的就是 $p \leq 10^5$ 这个性质。

初步转化

初步转化

可以 $\mathcal{O}(\log n)$ 计算出 $n!$ 中 p 的指数 $(n!)_p = \sum_{k\geq 1} \lfloor \frac{n}{p^k} \rfloor$。

初步转化

可以 $\mathcal{O}(\log n)$ 计算出 $n!$ 中 p 的指数 $(n!)_p=\sum_{k\geq 1}\lfloor\frac{n}{p^k}\rfloor$。

$$\binom{n}{m}=\frac{n!}{m!(n-m)!}=\frac{\frac{n!}{p^{(n!)_p}}}{\frac{m!}{p^{(m!)_p}}\frac{(n-m)!}{p^{((n-m)!)_p}}}p^{(n!)_p-(m!)_p-((n-m)!)_p}$$

初步转化

可以 $\mathcal{O}(\log n)$ 计算出 $n!$ 中 p 的指数 $(n!)_p = \sum_{k\geq 1}\lfloor \frac{n}{p^k} \rfloor$。

$$\binom{n}{m} = \frac{n!}{m!(n-m)!} = \frac{\frac{n!}{p^{(n!)_p}}}{\frac{m!}{p^{(m!)_p}} \frac{(n-m)!}{p^{((n-m)!)_p}}} p^{(n!)_p-(m!)_p-((n-m)!)_p}$$

问题落在了计算 $\dfrac{n!}{p^{(n!)_p}}$ 上。

初步转化

初步转化

如果我们能够计算 $F(n)=\prod_{i=1,i\perp p}^{n} i$ 的话？

初步转化

如果我们能够计算 $F(n) = \prod_{i=1, i\perp p}^{n} i$ 的话？

$$\frac{n!}{p^{(n!)_p}} = \prod_{k\geq 0} F(\lfloor \frac{n}{p^k} \rfloor)$$

初步转化

如果我们能够计算 $F(n)=\prod_{i=1,i\perp p}^{n} i$ 的话？

$$\frac{n!}{p^{(n!)_p}}=\prod_{k\geq 0} F(\lfloor\frac{n}{p^k}\rfloor)$$

所以如果我们可以以 $\mathcal{O}(t)$ 单次的复杂度回答 F，那么原问题就可以以 $\mathcal{O}(t\log n)$ 的复杂度被解决。

传统派

传统派

设计多项式 $G_n(x)=\prod_{i=1,i\perp p}^{n}(x+i)$，那么 $F(n)=G(0)$。

传统派

设计多项式 $G_n(x)=\prod_{i=1,i\perp p}^{n}(x+i)$，那么 $F(n)=G(0)$。
考虑从 $G_p(x)$ 开始，倍增得到 $G_{\lfloor \frac{n}{p}\rfloor p}(x)$，再暴力乘 $n \bmod p$ 个 $(x+i)$。

传统派

设计多项式 $G_n(x)=\prod_{i=1,i\perp p}^{n}(x+i)$，那么 $F(n)=G(0)$。
考虑从 $G_p(x)$ 开始，倍增得到 $G_{\lfloor \frac{n}{p} \rfloor p}(x)$，再暴力乘 $n \bmod p$ 个 $(x+i)$。
若要倍增，则需要解决 $G_{ap}\to G_{2ap}$ 和 $G_{ap}\to G_{(a+1)p}$ 的问题。

传统派

设计多项式 $G_n(x) = \prod_{i=1, i\perp p}^{n}(x+i)$，那么 $F(n) = G(0)$。
考虑从 $G_p(x)$ 开始，倍增得到 $G_{\lfloor \frac{n}{p} \rfloor p}(x)$，再暴力乘 $n \bmod p$ 个 $(x+i)$。
若要倍增，则需要解决 $G_{ap} \to G_{2ap}$ 和 $G_{ap} \to G_{(a+1)p}$ 的问题。

- $G_{2ap}(x) = G_{ap}(x) \times G_{ap}(x+ap)$
- $G_{(a+1)p}(x) = G_{ap}(x) \times G_p(x+ap)$

传统派

设计多项式 $G_n(x)=\prod_{i=1,i\perp p}^{n}(x+i)$，那么 $F(n)=G(0)$。
考虑从 $G_p(x)$ 开始，倍增得到 $G_{\lfloor\frac{n}{p}\rfloor p}(x)$，再暴力乘 $n \bmod p$ 个 $(x+i)$。
若要倍增，则需要解决 $G_{ap}\to G_{2ap}$ 和 $G_{ap}\to G_{(a+1)p}$ 的问题。

- $G_{2ap}(x)=G_{ap}(x)\times G_{ap}(x+ap)$
- $G_{(a+1)p}(x)=G_{ap}(x)\times G_p(x+ap)$

注意到我们实际上只需要保留多项式的前 e 项，所以只需要暴力二项式定理展开后面即可。注意需要 $\mathcal{O}(pe)$ 预处理 $G_0\sim G_p$。

传统派

设计多项式 $G_n(x) = \prod_{i=1, i\perp p}^{n}(x+i)$，那么 $F(n) = G(0)$。
考虑从 $G_p(x)$ 开始，倍增得到 $G_{\lfloor \frac{n}{p} \rfloor p}(x)$，再暴力乘 $n \bmod p$ 个 $(x+i)$。
若要倍增，则需要解决 $G_{ap} \to G_{2ap}$ 和 $G_{ap} \to G_{(a+1)p}$ 的问题。

- $G_{2ap}(x) = G_{ap}(x) \times G_{ap}(x+ap)$
- $G_{(a+1)p}(x) = G_{ap}(x) \times G_p(x+ap)$

注意到我们实际上只需要保留多项式的前 e 项，所以只需要暴力二项式定理展开后面即可。注意需要 $\mathcal{O}(pe)$ 预处理 $G_0 \sim G_p$。
计算 F 的时间复杂度为 $\mathcal{O}(e^2 \log n)$，太慢了！

维新派

维新派

本题中询问次数高达 $\mathrm{d}(m)$，因此上面的做法是很不平衡的！

维新派

本题中询问次数高达 d(m)，因此上面的做法是很不平衡的！
我们考虑对 $0 \le i \le 7, 1 \le j < 256$ 预处理 $G_{256^i \cdot j \cdot p}$。

维新派

本题中询问次数高达 $\mathsf{d}(m)$，因此上面的做法是很不平衡的！
我们考虑对 $0 \le i \le 7, 1 \le j < 256$ 预处理 $G_{256^i \cdot j \cdot p}$。
这一部分时间复杂度与 p 无关，大概是 $e^2 \cdot 2048$，因此非常快速。

维新派

本题中询问次数高达 d(m)，因此上面的做法是很不平衡的！
我们考虑对 $0 \le i \le 7, 1 \le j < 256$ 预处理 $G_{256^i \cdot j \cdot p}$。
这一部分时间复杂度与 p 无关，大概是 $e^2 \cdot 2048$，因此非常快速。
那么每次回答 F 的时候就不需要把要用的多项式乘起来了，只需要依次代入需要代入的 x 计算点值，再把点值相乘。

维新派

本题中询问次数高达 $\mathsf{d}(m)$，因此上面的做法是很不平衡的！
我们考虑对 $0 \le i \le 7, 1 \le j < 256$ 预处理 $G_{256^i \cdot j \cdot p}$。
这一部分时间复杂度与 p 无关，大概是 $e^2 \cdot 2048$，因此非常快速。
那么每次回答 F 的时候就不需要把要用的多项式乘起来了，只需要依次代入需要代入的 x 计算点值，再把点值相乘。
回答 F 的时间复杂度优化至 $\mathcal{O}(e \log_{256} n)$。

维新派

本题中询问次数高达 $\mathsf{d}(m)$，因此上面的做法是很不平衡的！
我们考虑对 $0 \le i \le 7, 1 \le j < 256$ 预处理 $G_{256^i \cdot j \cdot p}$。
这一部分时间复杂度与 p 无关，大概是 $e^2 \cdot 2048$，因此非常快速。
那么每次回答 F 的时候就不需要把要用的多项式乘起来了，只需要依次代入需要代入的 x 计算点值，再把点值相乘。
回答 F 的时间复杂度优化至 $\mathcal{O}(e \log_{256} n)$。
最终总时间复杂度为 $\mathcal{O}(p \log M + \mathsf{d}(m) \log M \log n \log_{256} n)$，完全足以通过，其中 $p \le 10^5$ 是 M 的最大质因数。

写在后面：该问题的一些背景

- 计算大组合数模小质数幂是一个经典的问题。目前我已知的时间复杂度最优的做法是 min_25 提出的 $\mathcal{O}(pe)$ 预处理，$\mathcal{O}(e)$ 回答 F。但我实现了这个做法，并在本地对比了其与前文提到的做法的效率差距。以本题为例，最大用时分别为 1.8s 和 0.9s。前文提到的做法由于较小的常数，实际表现是更优秀的。
- 本题的题目来源是 HackerRank Project Euler #242: Odd Triplets。
- 我的博客 https://faqak.github.io/misc/binom-mod-pe/ 中详细描述了 min_25 的算法和前文中阐述的算法。

谢谢大家！

祝大家学业有成！