简单题选讲

whzzt

2019 年 1 月 17 日

安徽师范大学附属中学

Count

给定一个长度不超过 25 的 0/1 串及 $N = \prod_{i=1}^{n} p_i^{e_i}$，求有多少正整数 k 满足 $\gcd(k+i, N) = 1$ 当且仅当 $s_i = 1$。

$|s| \le 25, n \le 10^5$

Count

出门先 CRT，接下来我们考虑 s 全是 1 的情况，相当于在 $\bmod p$ 意义下 ban 掉了一些连续的位置，容易统计。考虑有 0 的情况，只要简单容斥即可。

一个人的数论

给定 n 的质因数分解 $n=\prod_{i=1}^{w}p_i^{\alpha_i}$，求所有小于 n 且与 n 互质的正整数的 m 次方之和模 10^9+7 的值。其中 p_i 为质数且不超过 10^9，$\alpha_i \le 10^9, w \le 1000, m \le 100$。

一个人的数论

给定 n 的质因数分解 $n = \prod_{i=1}^{w} p_i^{\alpha_i}$，求所有小于 n 且与 n 互质的正整数的 m 次方之和模 10^9+7 的值。其中 p_i 为质数且不超过 10^9，$\alpha_i \le 10^9, w \le 1000, m \le 100$。

$$\sum_{i=1}^{n} [\gcd(n,i)=1] i^m$$

$$= \sum_{i=1}^{n} i^m \sum_{d|i,d|n} \mu(d)$$

$$= \sum_{d|n} d^m \mu(d) \sum_{i=1}^{\frac{n}{d}} i^m$$

一个人的数论

由于 $f(n) = \sum_{i=1}^{n} i^m$ 可以表示为一个 $m+1$ 次的多项式，首先 $O(m^2)$ 插值求出。设 $f(n) = \sum_{i=0}^{m+1} f_i n^i$。

一个人的数论

由于 $f(n)=\sum_{i=1}^{n} i^m$ 可以表示为一个 $m+1$ 次的多项式，首先 $O(m^2)$ 插值求出。设 $f(n)=\sum_{i=0}^{m+1} f_i n^i$。

$$
\begin{aligned}
&\sum_{d|n} d^m \mu(d) \sum_{i=1}^{\frac{n}{d}} i^m \\
=&\sum_{d|n} d^m \mu(d) \sum_{i=0}^{m+1} f_i \left(\frac{n}{d}\right)^i \\
=&\sum_{i=0}^{m+1} f_i \sum_{d|n} d^m \mu(d) \left(\frac{n}{d}\right)^i
\end{aligned}
$$

后式是两个积性函数的卷积，也是积性函数，只要对 n 的每个 $p_i^{\alpha_i}$ 因子计算函数值。而 μ 有值的只有两项，时间复杂度 $O(m(m+w))$。

怎样跑的更快

给定 $n \le 10^5; c, d \le 10^9$，现有序列 b 满足 $b_i \equiv \sum_{j=1}^{n} \gcd(i,j)^c \operatorname{lcm}(i,j)^d z_j \pmod{998244353}$，求 z，无解输出 -1。

怎样跑的更快

给定 $n \le 10^5; c, d \le 10^9$，现有序列 b 满足 $b_i \equiv \sum_{j=1}^{n} \gcd(i,j)^c \operatorname{lcm}(i,j)^d z_j \pmod{998244353}$，求 z，无解输出 -1。

首先可以将问题转化为 $y_i \equiv \sum_{j=1}^{n} \gcd(i,j)^w x_j$ 的形式，简单反演即可得到：

怎样跑的更快

给定 $n \le 10^5; c, d \le 10^9$，现有序列 b 满足 $b_i \equiv \sum_{j=1}^{n} \gcd(i,j)^c \operatorname{lcm}(i,j)^d z_j \pmod{998244353}$，求 z，无解输出 -1。

首先可以将问题转化为 $y_i \equiv \sum_{j=1}^{n} \gcd(i,j)^w x_j$ 的形式，简单反演即可得到：

$$y_i \equiv \sum_{T|i} (id^w * \mu)(T) \sum_{i=1}^{\lfloor \frac{n}{T} \rfloor} x_{iT} \pmod{998244353}$$

简单反演即可解决问题。

最小公倍数之和

求 $\sum_{i=1}^{n}\sum_{j=1}^{n}\operatorname{lcm}(i,j)$。$n \le 10^{10}$。

最小公倍数之和

求 $\sum_{i=1}^{n}\sum_{j=1}^{n}\operatorname{lcm}(i,j)$。$n \le 10^{10}$。

考虑一个更加广泛的问题：求 $\sum_{i=1}^{n}\sum_{j=1}^{n}(i \times j)^a \gcd(i,j)^b$，其中 $a, b \le 10^3$。

最小公倍数之和

$$\sum_{i=1}^{n}\sum_{j=1}^{n}(i\times j)^a\gcd(i,j)^b$$

$$=\sum_{T=1}^{n}((id^b*\mu)\circ id^{2a})(T)\left(\sum_{i=1}^{\lfloor\frac{n}{T}\rfloor}i^a\right)^2$$

$$=\sum_{T=1}^{n}(id^{2a+b}*(\mu\circ id^{2a}))(T)\left(\sum_{i=1}^{\lfloor\frac{n}{T}\rfloor}i^a\right)^2$$

最小公倍数之和

$$\sum_{i=1}^{n}\sum_{j=1}^{n}(i\times j)^a\,\mathrm{gcd}(i,j)^b$$

$$=\sum_{T=1}^{n}((id^b*\mu)\circ id^{2a})(T)\left(\sum_{i=1}^{\lfloor\frac{n}{T}\rfloor}i^a\right)^2$$

$$=\sum_{T=1}^{n}(id^{2a+b}*(\mu\circ id^{2a}))(T)\left(\sum_{i=1}^{\lfloor\frac{n}{T}\rfloor}i^a\right)^2$$

困难在于前半部分的前缀和，但容易发现只要求 $\mu\circ id^{2a}$ 的前缀和和 $2a,2a+b$ 次的自然数幂和。杜教筛即可，时间复杂度 $O(\sqrt{na+nb}+n^{2/3})$。

小 C 的本质

小 A 和小 C 在玩一个游戏，每次小 A 从 $1 \sim n$ 中任意选择一个数 i，然后小 C 从 $1 \sim i$ 中任意选择一个数 x，这将会产生 $\gcd(i, x)$ 的价值。众所周知，小 C 是一个复读机，他会重复 k 次这样的操作（但小 A 只选择一个数），游戏的总价值为每次小 C 产生价值的最小公倍数。

现在小 A 想要知道所有情况下游戏总价值的和是多少，由于一些显而易见的原因，你只需要求出在模 $10^9 + 7$ 意义下的和即可。

$n \times k \leq 10^9$

小 C 的本质

容易发现小 A 选一个数的贡献是积性函数，k 小的时候随便求求和，k 大的时候随便暴力就好了。

石像

求

$$\sum_{a_1=1}^{m}\sum_{a_2=1}^{m}\cdots\sum_{a_n=1}^{m}\left(\sigma_0\left(\gcd\left(a_1,a_2,\ldots,a_n\right)^3\right)\right)^3\prod_{i=1}^{k}\left[a_{x_i}\le a_{y_i}\right]$$

在模 2^{32} 意义下的值。

$n \le 20, m \le 10^{10}, k \le n(n-1)$

石像

首先反演一下把 gcd 搞掉，问题变成求积性函数前缀和和求满足后面限制的填充 a 的方案数这两个子问题。前一个问题容易解决，后一个问题可以令 $f_{i,j,S}$ 表示目前用了 i 种值，考虑第 j 个点，已经有值的位置的状态为 S 的方案数。复杂度 $O\left(2^n n^2 + \frac{m^{2/3}}{\log n}\right)$。

经典题

有 A,B,C,D 四种水果，每种都有无限个。现在要求拿出恰好 N 个水果，但要求水果 A 要拿出恰好偶数个，B 的个数是 5 的倍数，C 最多只能拿 4 个，D 最多拿 1 个，问最终的方案数是多少。

$N \leq 10^{18}$

经典题

写出普通生成函数，可以发现答案是 $N+1$。

不动点

现在有 n 个点，每个点有个值 $f(i)$，表示 i 这个点走一步会到 $f(i)$ 这个点。一开始所有的 f 都没有确定。现在给定 k，要求对于每个 i，都满足 i 这个点，走不超过 k 步，能到一个点 c，满足 $f(c) = c$。求满足条件的 f 的个数。

$n \times k \leq 2 \times 10^6, k \leq 3$

不动点

把 $f(i)$ 想象成是 i 指向祖先的边，那么就相当于要统计有多少种森林，满足每个点的深度都不超过 k。

不动点

把 $f(i)$ 想象成是 i 指向祖先的边，那么就相当于要统计有多少种森林，满足每个点的深度都不超过 k。

令 $[x^n]g_k(x)$ 代表深度限制为 k 时，森林点数为 n 的方案数。当 $k=0$ 时，每个点都必须作为根，则有 $g_0(x)=e^x$ ，当 $k=1$ 时，我们考虑，一棵深度不超过 k 的树，相当于一个深度不超过 $k-1$ 的森林和一个根拼在一起，所以一棵深度不超过 k 的树的生成函数就是 $xg_{k-1}(x)$。然后这个新森林又是若干棵树的集合，所以 $g_k(x)=e^{xg_{k-1}(x)}$。

复读机

群里有 k 个不同的复读机。为了庆祝平安夜的到来，在接下来的 n 秒内，它们每秒钟都会选出一位优秀的复读机进行复读。非常滑稽的是，一个复读机只有总共复读了 d 的倍数次才会感到快乐。问有多少种不同的安排方式使得所有的复读机都感到快乐。

$n \le 10^9, d \le 3, k \le 5 \times 10^5$ 且当 $d = 3$ 时 $k \le 1000$。

复读机

我们利用指数级生成函数解决问题。

复读机

我们利用指数级生成函数解决问题。

当 $d = 2$ 时显然答案的生成函数为 $\left(\frac{e^x+e^{-x}}{2}\right)^k$，直接计算即可。

复读机

我们利用指数级生成函数解决问题。

当 $d=2$ 时显然答案的生成函数为 $\left(\frac{e^x+e^{-x}}{2}\right)^k$，直接计算即可。

当 $d=3$ 时显然答案的生成函数为 $\left(\frac{e^x+e^{\omega x}+e^{\omega^2 x}}{3}\right)^k$，其中 ω 是三次单位根。暴力计算即可。

取名字太难了

给定一个大小为 N 的正整数集合 $S = \{1, 2, ..., N\}$，定义 $f(n) = \sum_{T \subset S, |T| = n} \prod_{x \in T} x \pmod p$，对于任意一个 $x \in [0, p-1]$，问 $f(n) = x$ 的 $n \in [0, N]$ 的个数，输出对 998244353 取模。

$N \leq 10^{18}, p \leq 250000$

取名字太难了

即求 $\prod_{i=1}^{N}(x+i)$ 的系数在模 p 下的分布，注意到 $\prod_{i=1}^{p-1}(x+i) \equiv x^{p-1}-1$，问题相当于求 $(x^{p-1}-1)^a$ 和 $\prod_{i=1}^{b}(x+i)$ 在 $\bmod p$ 下的系数分布即可。对于前者我们 Lucas 定理，最后用 fft 合并，总时间复杂度 $O(p\log n)$。

普通的计数题

1. 在序列末端插入一个 0。
2. 在序列中删去一个子序列，并在序列末端插入一个 1。这里对子序列的选取有一定限制，设子序列中包含 x 个 0，y 个 1，则你选取的子序列必须满足：
   - 子序列不可为空，即 $x+y>0$；
   - 当 $y>0$ 时，$x\in A$，这里 A 为给定集合；
   - 当 $y=0$ 时，$x\in B$，这里 B 为给定集合。

你需要对序列执行 n 次操作，求使得最终序列长度为 1 的操作方案有多少种。两种操作方案被视为不同，当且仅当某一次操作的种类不同，或某个第二类操作中选取的子序列的位置不同。

答案对 998244353 取模。

$n\le 1.2\times 10^5$

普通的计数题

将每个数向消去它的数连边，那么 0 一定是叶子，1 就是非叶子，问题变成计数这样的树的贡献之和。考虑给你的 A 和 B 的限制，显然就是 $F(x) = B(x)\int A(x)(e^{F(x)} - 1)\mathrm{d}x$，即 $F'(x) = (B'(x) - A(x)) + A(x)e^{F(x)}$。可以分治 FFT 解决或牛顿迭代解决。

普通的计数题

将每个数向消去它的数连边，那么 0 一定是叶子，1 就是非叶子，问题变成计数这样的树的贡献之和。考虑给你的 A 和 B 的限制，显然就是 $F(x) = B(x) \int A(x)(e^{F(x)} - 1)\mathrm{d}x$，即 $F'(x) = (B'(x) - A(x)) + A(x)e^{F(x)}$。可以分治 FFT 解决或牛顿迭代解决。

其他做法?

普通的计数题

不妨令 $u = \exp f$:

$$\frac{\mathrm{d}\ln u}{\mathrm{d}x} = au + c$$
$$\frac{\mathrm{d}u}{\mathrm{d}x} = au^2 + cu$$
$$\frac{\mathrm{d}u}{u^2\mathrm{d}x} - cu^{-1} = a$$
$$v = u^{-1}$$
$$-\frac{\mathrm{d}v}{\mathrm{d}x} - cv = a$$
$$\frac{\mathrm{d}v}{\mathrm{d}x} + cv = -a$$

普通的计数题

对这样的一阶线性微分方程，有通解：

$$v = \int -ae^{\int c\mathrm{d}x}\mathrm{d}x + C$$

因此有

$$f = \ln \frac{1}{\int -ae^{\int c\mathrm{d}x}\mathrm{d}x + 1}$$

普通的计数题

对这样的一阶线性微分方程，有通解：

$$v = \int -ae^{\int c \mathrm{d}x} \mathrm{d}x + C$$

因此有

$$f = \ln \frac{1}{\int -ae^{\int c \mathrm{d}x} \mathrm{d}x + 1}$$

其实我也不知道上面推的具体式子对不对（

带劲的多项式

多项式因式分解，保证每个不可约多项式因子的重数不同，且均为 $(x+\lambda_i)$ 的形式。

$T = 10, n \leq 2000$

带劲的多项式

考察 $\gcd(f, f')$，容易发现相当于将所有不可约多项式因子的重数消去 1，时间复杂度 $O(Tn^2)$。

EvenPaths

给定一个 n 个点 m 条边的 DAG，有 k 个点可能是障碍点也可能不是障碍点，问 2^k 种情况中有多少种使得 0 号点到 1 号点恰好有偶数条不经过障碍点的路径。

EvenPaths

给定一个 n 个点 m 条边的 DAG，有 k 个点可能是障碍点也可能不是障碍点，问 2^k 种情况中有多少种使得 0 号点到 1 号点恰好有偶数条不经过障碍点的路径。

$n \le 50, m \le 500, k \le 32$

EvenPaths

按照拓扑序将 k 个特殊点分为两个集合 A 和 B。对于一条路径，我们在其访问的 A 中的最后一个点处统计答案。枚举 A 中的障碍点集合，求出 0 号点到 A 中每个点的路径方案数奇偶性；枚举 B 中的障碍点集合，求出从 A 中每个点不经过其他 A 中的点到 1 号点的路径方案数的奇偶性。FWT 合并即可，时间复杂度 $O(2^{\frac{k}{2}}(n+m+k))$。

CRB and String

给定两个字符串 S 和 T，你每次可以在 S 的某个字符 c 后面添加一个字符 d，且要保证 $c \neq d$，问有没有可能把 S 变成 T。

$1 \leq |S| \leq |T| \leq 10^5$

CRB and String

首先 S 必须要是 T 的子序列，且 $S_1 = T_1$。因为字符不同的限制，所以 S 开头连续相同的字符数要不小于 T 开头连续相同的字符数。如果满足以上条件，那么一定可以得到 T。

首先 S 必须要是 T 的子序列，且 $S_1 = T_1$。因为字符不同的限制，所以 S 开头连续相同的字符数要不小于 T 开头连续相同的字符数。如果满足以上条件，那么一定可以得到 T。

时间复杂度 $O(n)$。

String

给定 n 个串，有 m 次询问，每次询问给定串 a, b，问 n 个串中有多少个能表示为 axb，其中 x 为任意字符串。

字符串总长 $\leq 10^6$。

条件等价于前缀为 a，后缀为 b，且长度 $\geq |a| + |b|$ 的个数。

String

条件等价于前缀为 a，后缀为 b，且长度 $\geq |a| + |b|$ 的个数。

不考虑长度的情况下，将正串和反串分别按字典序编号，一个询问对应了编号的一个区间，用字典树或排序 + 二分 +hash 求出区间，然后主席树求答案。

String

条件等价于前缀为 a，后缀为 b，且长度 $\geq |a| + |b|$ 的个数。

不考虑长度的情况下，将正串和反串分别按字典序编号，一个询问对应了编号的一个区间，用字典树或排序 + 二分 +hash 求出区间，然后主席树求答案。

然后考虑减掉长度不足的。直接枚举长度就可以确定出字符串。

条件等价于前缀为 a，后缀为 b，且长度 $\geq |a| + |b|$ 的个数。

不考虑长度的情况下，将正串和反串分别按字典序编号，一个询问对应了编号的一个区间，用字典树或排序 + 二分 +hash 求出区间，然后主席树求答案。

然后考虑减掉长度不足的。直接枚举长度就可以确定出字符串。

时间复杂度 $O(s \log s)$。

迷失的字符串

有一棵 n 个点的树，每条边有一个小写字母。

对于两个不同的点 u, v，将 u 到 v 路径上沿途经过的边上的字符依次写下来，得到一个字符串。

对于一个字符串，如果存在这样一个点对 (u, v)，使得它们路径上的字符串与其完全匹配，那么我们就称这个字符串属于这棵树。

有 m 个字符串，判断每一个字符串是否属于这棵树。

字符串总长 $, n, m \leq 3 \times 10^4$。

迷失的字符串

对单独一个串可以 dp，$f_{w,i}$ 表示走到 w 这条边（边分两个方向），是否能匹配到第 i 个字符。

迷失的字符串

对单独一个串可以 dp，$f_{w,i}$ 表示走到 w 这条边 (边分两个方向)，是否能匹配到第 i 个字符。

将所有串的第二维压到一起，用 bitset 优化，转移时左移一位，与上这条边的字符 c 对应的数组 u_c，再或上 v_c。

$u_{c,i} = 1$ 当且仅当第 i 个位置对应的字符为 c。

$v_{c,i} = 1$ 当且仅当第 i 个位置对应的字符为 c 且它是某个串的第一个字符。

迷失的字符串

对单独一个串可以 dp，$f_{w,i}$ 表示走到 w 这条边（边分两个方向），是否能匹配到第 i 个字符。

将所有串的第二维压到一起，用 bitset 优化，转移时左移一位，与上这条边的字符 c 对应的数组 u_c，再或上 v_c。

$u_{c,i} = 1$ 当且仅当第 i 个位置对应的字符为 c。

$v_{c,i} = 1$ 当且仅当第 i 个位置对应的字符为 c 且它是某个串的第一个字符。

时间复杂度 $O(\frac{ns}{\omega})$。

Little Elephant

给定 n 个字符串。对于每个字符串，询问它有多少个子串是 n 个字符串中至少 k 个的子串。

$1 \leq k \leq n \leq 10^5$，字符串总长 $\leq 10^5$。

首先将所有字符串拼在一起做后缀数组。我们需要求出每个后缀有多少个前缀满足条件 (记为 f_i)。

首先将所有字符串拼在一起做后缀数组。我们需要求出每个后缀有多少个前缀满足条件 (记为 f_i)。

对于每个后缀，预处理出往后扩展到哪里会满足条件。对于这样的一段区间，公共前缀就是 height 的最小值 (记为 x)。

首先将所有字符串拼在一起做后缀数组。我们需要求出每个后缀有多少个前缀满足条件 (记为 f_i)。

对于每个后缀，预处理出往后扩展到哪里会满足条件。对于这样的一段区间，公共前缀就是 height 的最小值 (记为 x)。

一个区间 (l,r,x) 对 f 有以下影响：

- 对于 $l \le i \le r$，$f_i = \max(f_i, x)$。
- 对于 $i > r$，$f_i = \max(f_i, \min(h_l \ldots h_i))$

分别维护即可，时间复杂度 $O(l \log l)$。

给定 n 个字符串 $a_1 \ldots a_n$。有 m 次询问，每次给出 x, y，问 a_x 和 a_y 的满足 [是至少一个字符串的前缀] 的最长公共子串的长度。

Fleet of the Eternal Throne

给定 n 个字符串 $a_1 \dots a_n$。有 m 次询问，每次给出 x, y，问 a_x 和 a_y 的满足 [是至少一个字符串的前缀] 的最长公共子串的长度。

字符串总长 $\leq 10^5, m \leq 100$。

每次询问时对询问的两个串 x 和 y 分别做：

建后缀自动机，求出每个串是这个串的子串的最长前缀。

每次询问时对询问的两个串 x 和 y 分别做:

建后缀自动机，求出每个串是这个串的子串的最长前缀。

每个串对 x 和 y 的答案取 $\min$，所有串的答案取 $\max$ 即可。

每次询问时对询问的两个串 x 和 y 分别做:

建后缀自动机，求出每个串是这个串的子串的最长前缀。

每个串对 x 和 y 的答案取 min，所有串的答案取 max 即可。

时间复杂度 $O(m|s|)$。

Subsequence

定义 a 串是 s 串的次连续子串的条件是：

- a 串是 s 串的子串；
- a 串同时是 s 串的子序列，且这个子序列不能是子串。

现在字符集大小是 k，求有多少个不同的字符串，它的最长次连续子串长度为 w。对 10^9+7 取模。

$1 \le k \le 10^6, 2 \le w \le 10^9$

Subsequence

容易发现最长次连续子串一定是前缀或后缀。考虑最长次连续子串长度 $\leq w$，意味着开头的 $l - w$ 个字符互不相同且末尾的 $l - w$ 个字符互不相同。

Subsequence

容易发现最长次连续子串一定是前缀或后缀。考虑最长次连续子串长度 $\leq w$，意味着开头的 $l-w$ 个字符互不相同且末尾的 $l-w$ 个字符互不相同。

可以发现当 $l \leq 2w$ 时，答案为 $P(k, l-w)^2 * k^{2w-l}$；

当 $l > 2w$ 时，答案为 $P(k, l-2w) * P(k-(l-2w), w)^2$。

容易发现最长次连续子串一定是前缀或后缀。考虑最长次连续子串长度 $\leq w$，意味着开头的 $l-w$ 个字符互不相同且末尾的 $l-w$ 个字符互不相同。

可以发现当 $l \leq 2w$ 时，答案为 $P(k, l-w)^2 * k^{2w-l}$；

当 $l > 2w$ 时，答案为 $P(k, l-2w) * P(k-(l-2w), w)^2$。

于是枚举 l 即可。

字符串识别

给定一个字符串 S 与一个整数 K，定义 S 的子串 $T = S(i,j)$ 是关于第 K 位的识别子串，满足以下两个条件：

- $i \le K \le j$;
- 子串 T 只在 S 中出现过一次。

现在，给定 S，求对于 S 的每一位，最短的识别子串长度是多少。

$n \le 5 \times 10^5$

字符串识别

建后缀自动机，只有 right 集大小为 1 的节点对答案有贡献。

字符串识别

建后缀自动机，只有 right 集大小为 1 的节点对答案有贡献。

若其出现位置右端点为 r，此节点可接受的最短串长为 x，最长串长为 y，则对 $(r-x,r]$ 用 x 更新最小值，对 $r-k(y<k\leq x)$ 则用 k 更新最小值。

字符串识别

建后缀自动机，只有 right 集大小为 1 的节点对答案有贡献。

若其出现位置右端点为 r，此节点可接受的最短串长为 x，最长串长为 y，则对 $(r-x, r]$ 用 x 更新最小值，对 $r-k(y<k\leq x)$ 则用 k 更新最小值。

用两棵线段树维护答案，分别处理以上两种情况。

字符串识别

建后缀自动机，只有 right 集大小为 1 的节点对答案有贡献。

若其出现位置右端点为 r，此节点可接受的最短串长为 x，最长串长为 y，则对 $(r-x,r]$ 用 x 更新最小值，对 $r-k(y<k\leq x)$ 则用 k 更新最小值。

用两棵线段树维护答案，分别处理以上两种情况。

时间复杂度 O(nlogn)

z-function

对于一个长度为 n 的字符串 s，定义函数 $z(i)(2 \le i \le n)$ 等于最大的 t，满足 s 的前 t 位与 s 从第 i 位开始的 t 位对应相等。一个字符串的价值为该字符串最大的 $z()$。求长度为 n，字符集为 k 的所有字符串的价值之和对 $10^9 + 7$ 取模的结果。

$n \le 100$

考虑 $\max z_i$ 的组合意义，相当于每个前缀如果在串中出现过就给答案贡献 1，爆搜 border 写个 DP 容斥就行了。

z-function

考虑 $\max z_i$ 的组合意义，相当于每个前缀如果在串中出现过就给答案贡献 1，爆搜 border 写个 DP 容斥就行了。

基础搜 border 技巧：border 是一条链。

平方串

我们定义一个串为平方串当且仅当这个串非空而且它可以由两个相同串连接而成。例如 naivenaive 和 aaaaaa 为平方串，而 naiveevian 和 aaaaa 不是。

现在 zzq 拿到了一个很长的数字串（每个字符为 $0 \sim 9$），他想要随机地取出一个非空子串，如果这个子串没有前导零且为平方串，那么它的价值就为它的数值，否则它的价值为 0。

zzq 想知道他取出子串价值的期望，$\bmod 666623333$ 输出。

$n \leq 5 \times 10^5$

平方串

分治，每次考虑跨越中间的串。

平方串

分治，每次考虑跨越中间的串。

枚举平方串的长度 $2p$，分别考虑靠左、靠右和正中间的串。

以靠左为例：用后缀数组 +ST 表或二分 +hash 求出中点和中点-p 的最长公共前/后缀，可以用三个参数 (l, r, p) 表示这些平方串：左端点为 $[l, r]$，长度为 p 的所有串。

记 $f(i)$ 表示 $s_1 \dots s_i$ 作为一个数的值，那么如果不要求没有前导零的话，(l, r, p) 的值是 $f(l+p-1) + \dots + f(r+p-1) - 10^p * (f(l-1) + \dots + f(r-1))$。

平方串

分治，每次考虑跨越中间的串。

枚举平方串的长度 $2p$，分别考虑靠左、靠右和正中间的串。

以靠左为例：用后缀数组 +ST 表或二分 +hash 求出中点和中点-p 的最长公共前/后缀，可以用三个参数 (l,r,p) 表示这些平方串：左端点为 $[l,r]$，长度为 p 的所有串。

记 $f(i)$ 表示 $s_1\dots s_i$ 作为一个数的值，那么如果不要求没有前导零的话，(l,r,p) 的值是
$f(l+p-1)+\dots+f(r+p-1)-10^p*(f(l-1)+\dots+f(r-1))$。

考虑前导零的限制，只要将 $s_{i+1}=0$ 的 $f(i)$ 改为 0 即可。

时间复杂度 $O(n\log n)$。

串串划分

给你一个长度为 n 的字符串 S，你需要将其划分成若干个不相交的非空连续子串 $s_1, s_2, \ldots, s_k (k \geq 1)$，满足：

- 每个子串 $s_i (1 \leq i \leq k)$ 是不循环的。
- 相邻的两个子串不同。即 $s_i \neq s_{i+1} (1 \leq i < k)$。
- $s_1, s_2, \ldots, s_k$ 从左到右拼接起来恰好为原串。即 $S = \overline{s_1 s_2 \ldots s_k}$。

一个字符串 S 是循环的，当且仅当存在一个整数 $k (k \geq 2)$ 和一个字符串 s，使得 k 个 s 拼接起来后的字符串与 S 相等。例如字符串 $S = \text{abab}$ 是循环的，因为存在 $k = 2, s = \text{ab}$。一个字符串是不循环的，当且仅当它不是循环的。字符串 ababa 和 abcac 就是不循环的。

你需要计算有多少种不同的划分方法模 998244353 的值。两个划分方法不同是指划分出的子串序列不同。

$|S| \leq 2 \times 10^5$

串串划分

我们定义本原串为非整周期串，本原平方串为本原串重复两次后的结果，那么有结论: 一个长度为 n 的串的前缀本原平方串个数是 $O(\log n)$ 的，证明较为繁琐，这里不加赘述。

串串划分

我们定义本原串为非整周期串，本原平方串为本原串重复两次后的结果，那么有结论: 一个长度为 n 的串的前缀本原平方串个数是 $O(\log n)$ 的，证明较为繁琐，这里不加赘述。

考虑求出所有的本原平方串，可以枚举其长度的一半，这样我们可以求出很多平方串。容易发现对于每一组平方串，我们只要检验一个串即可判定这一组的串是否是本原平方串。因此这一部分的总复杂度为 $O(n \log n)$ 。

串串划分

我们定义本原串为非整周期串，本原平方串为本原串重复两次后的结果，那么有结论：一个长度为 n 的串的前缀本原平方串个数是 $O(\log n)$ 的，证明较为繁琐，这里不加赘述。

考虑求出所有的本原平方串，可以枚举其长度的一半，这样我们可以求出很多平方串。容易发现对于每一组平方串，我们只要检验一个串即可判定这一组的串是否是本原平方串。因此这一部分的总复杂度为 $O(n \log n)$ 。

提取出所有本原平方串后，我们便得到了所有循环串的位置。注意到若循环串次数 ≥ 3 则会在前面同样产生一些本原平方串，因此对每个平方串我们可以 $O(1)$ 进行容斥。总时间复杂度 $O(n \log n)$ 。

串串

fateice 喜欢串串，尤其喜欢双回文串。

如果你不知道啥是双回文串，一个串 s 是双回文串当且仅当存在非空的回文串 a 和 b 满足 s 由 a 和 b 拼接而成。例如，$aabcb$ 为双回文串，而 aba 不是双回文串。

一天 fateice 看到了一个新鲜的字符串 s，他把 s 的所有本质不同的子串列举了出来，并且数出了其中的双回文串个数。两个字符串本质不同当且仅当它们的长度不同或存在一个下标它们对应位置的字符不同。

fateice 当然飞快地报出了答案，但是他想考考你。

$|s| \leq 5 \times 10^5$

串串

有结论: 若 s 有两种以上的双回文拆分，那么 s 一定是个整周期串。

串串

有结论: 若 s 有两种以上的双回文拆分，那么 s 一定是个整周期串。

考虑计数本质不同的双回文串拆分方案数，使用回文树后，问题变成对若干对 (x, y) 的祖先打标记，最后求标记总数。容易利用启发式合并解决该问题，时间复杂度 $O(|s| \log |s|)$ 。

有结论: 若 s 有两种以上的双回文拆分，那么 s 一定是个整周期串。

考虑计数本质不同的双回文串拆分方案数，使用回文树后，问题变成对若干对 (x, y) 的祖先打标记，最后求标记总数。容易利用启发式合并解决该问题，时间复杂度 $O(|s| \log |s|)$ 。

考虑扣除多余部分的贡献，由于所有可能的贡献都一定包含了本原平方串，枚举了本质不同本原平方串后问题变得容易处理。由于本质不同本原平方串个数不超过 $2|s|$，因此这一部分也能 $O(|s| \log |s|)$ 解决。

The end

欢迎大家来听清新小课堂《简单数论算法》!