哈希、字符串

李淳风

长郡中学

2025 年 1 月 13 日

哈希表

当我们需要对一些复杂信息进行统计时，可以用 Hash 函数来把这些复杂信息映射到一个容易维护的值域内。由于值域变简单，范围变小，所以有可能造成两个不同的信息，他们的 Hash 函数值相同的情况。一种常见的处理方式是，对于每个可能的 Hash 函数值建立一个链表，存储所有映射到此的信息。

在 OI 中，我们使用哈希表最常见的情况是对大整数进行统计，一般选择一个大质数 M，定义哈希函数 $H(x) = x \bmod M$。我们对字符串进行统计的情况也不少见，这种情况下我们一般把字符串看作一个 p 进制数，计算这个 p 进制数对 M 取模的结果。一般我们会把每个前缀代表的 p 进制数对 M 取模的值记录下来，这样我们就可以快速计算每个子串的哈希值。

对于字符串哈希，我们可以取 M 为 2^{64}，这样我们只需要对 $unsigned\ long\ long$ 类型的变量进行运算而不需要进行常数较大的取模操作。最后得到的数再对一个大质数取模，用哈希表进行统计。

为了避免冲突，我们可以取若干个（一般两个够用）不同的 P 和 M，只有两种情况下得到的值都相同，我们才认为字符串相同。

KMP

KMP 算法主要在对一个长度为 n 的字符串 s 求一个 $next$ 数组。其中 $next[i]$ 表示：s 以 i 结尾的子串，能与 s 的某个前缀完全匹配的最大长度。即 $s[1\dots k]=s[i-(k-1)\dots i]$，$next[i]$ 就是满足条件的 $k(k<i)$ 的最大值。

假设 $next[1\sim i-1]$ 已经计算完毕，当计算 $next[i]$ 时，根据定义，我们需要找出所有满足 $j<i, s[i-j+1\dots i]=s[1\dots j]$ 的整数 j 并取最大值。这里我们定义满足条件的 j 为 $next[i]$ 的候选项。

由于 $s[i-j+1\dots i]=s[1\dots j]$，所以 $s[i-j+1\dots i-1]=s[1\dots j-1]$，即 $j-1$ 是 $next[i-1]$ 的候选项。而 $next[i-1]$ 的所有候选项从大到小依次是 $next[i-1], next[next[i-1]], \cdots$。因此我们只需要从大到小遍历这些候选项，从中找出满足条件的最大值即可。

由于 $next[i]$ 数组每次最多比 $next[i-1]$ 增大 1，而每一次用 $next$ 往前跳，值至少减小 1，所以总次数是 $O(n)$ 的。因此总复杂度为 $O(n)$。

有了 $next$ 数组之后，我们就可以求出字符串 s 在另一个字符串 A 中每次出现的位置，并求出 s 的循环节长度了。

最小表示法

给定一个长度为 n 的字符串 s，如果我们不断把最后一个字符放到开头，最终会得到 n 个字符串，称这 n 个字符串是循环同构的。这些字符串当中字典序最小的一个称为字符串 s 的最小表示。

我们的朴素做法是，先把 s 复制一遍拼在末尾，对于以 $s[i]$ 和 $s[j]$ 开头的两个字符串，依次往后比较，找到第一个不同的位置。如果 $s[i+k] < s[j+k]$，就说明以 $s[i]$ 开头的字符串字典序更小。比较 $n-1$ 次即可找到最小表示。

考虑优化，当我们发现 $s[i+k] < s[j+k]$ 时，实际上也说明了 $s[i \ldots i+k-1] = s[j \ldots j+k-1]$。又由于 $s[i+k] < s[j+k]$，所以对于以 $s[j], s[j+1], s[j+2], \cdots, s[j+k]$ 开头的字符串都不可能称为最小表示了，因为它们的字典序肯定大于以 $s[i], s[i+1], s[i+2], \cdots, s[i+k]$ 开头的字符串，我们就可以跳过这一段不再进行比较。

所以我们初始令 $i = 1, j = 2$，每次比较后更新 i 或 j 的值，全部比较完成或者发现 $s[i], s[j]$ 开头的字符串完全相等时，算法结束。时间复杂度为 $O(n)$。

Period

对于给定字符串 S 的每个前缀，我们想知道它是否为周期串（周期串定义为由若干最小循环节拼接而成的字符串），若是，输出前缀长度和长度最小的循环节数量。

Period

对于给定字符串 S 的每个前缀，我们想知道它是否为周期串（周期串定义为由若干最小循环节拼接而成的字符串），若是，输出前缀长度和长度最小的循环节数量。

如果一个前缀 $s[1 \ldots i]$ 有循环节 k，那么 $s[1 \ldots i-k] = s[k+1 \ldots i]$，也就意味着 $i-k$ 是 $next[i]$ 的候选项。又由于 $i \bmod k = 0$，所以如果 $i \bmod (i - next[i]) = 0$，那么 $i - next[i]$ 就是最短的循环节。否则如果不整除，则说明 s 不存在除本身以外的循环节。

P8112 [Cnoi2021] 符文破译

Cirno 想要解读一本古老的魔法书。
为了保护魔法书中记载的禁忌的魔法，撰写者将符咒的魔法词缀不加空格地连接在一起，形成一个符文串，记作 S。
而构成符文串的所有魔法词缀都被记载在更远古的先知所编写的魔法辞典中，记作 T。具体地，T 的所有非空前缀均是一个合法的魔法词缀。
简洁是魔法书撰写的第一要务，所以使用的魔法词缀应该尽可能少。
所以在破译魔法书时，将 S 分解成的魔法词缀数越少，破译正确的可能性就越高。
Cirno 想知道，这本魔法书最少的魔法词缀划分段数是多少。
特别地，如果不存在一种合法的划分方案，则表明这本魔法书是假的。
Cirno 将得到一个字符串 Fake。

P8112 [Cnoi2021] 符文破译

Cirno 想要解读一本古老的魔法书。
为了保护魔法书中记载的禁忌的魔法，撰写者将符咒的魔法词缀不加空格地连接在一起，形成一个符文串，记作 S。
而构成符文串的所有魔法词缀都被记载在更远古的先知所编写的魔法辞典中，记作 T。具体地，T 的所有非空前缀均是一个合法的魔法词缀。
简洁是魔法书撰写的第一要务，所以使用的魔法词缀应该尽可能少。
所以在破译魔法书时，将 S 分解成的魔法词缀数越少，破译正确的可能性就越高。
Cirno 想知道，这本魔法书最少的魔法词缀划分段数是多少。
特别地，如果不存在一种合法的划分方案，则表明这本魔法书是假的。
Cirno 将得到一个字符串 Fake。

设 f_i 表示 $S[1\dots i]$ 最少被划分为多少段。不难发现 f_i 是递增的，因为每次在末尾添加一个字符不可能会导致答案变小。
考虑转移方程 $f_i=\min\{f_j\}+1(S[j+1\dots i]=T[1\dots i-j])$，要让 f_j 最小，就是求最长的 S 的后缀和 T 对应长度的前缀匹配。不难发现这就是使用 KMP 算法把 T 在 S 中匹配的过程。

P10992 [蓝桥杯 2023 国 Python A] 最长同类子串

对于两个等长的字符串 A, B，如果对于任意 i, j，都有 $A_i = A_j$ 和 $B_i = B_j$ 同时满足或同时不满足，那么我们称 A, B 是一对同类串。例如，aabab 和 xxkxk 是一对同类串，而 abcde 和 abcdd 则不是。

给定 S, T，找出一个尽可能大的 k 使得 S, T 分别含有一个长度为 k 的子串 S', T'，且 S', T' 是同类串。

$1 \leq |S|, |T| \leq 10^5$，$S, T$ 中仅含英文小写字母。

P10992 [蓝桥杯 2023 国 Python A] 最长同类子串

对于两个等长的字符串 A, B，如果对于任意 i, j，都有 $A_i = A_j$ 和 $B_i = B_j$ 同时满足或同时不满足，那么我们称 A, B 是一对同类串。例如，aabab 和 xxkxk 是一对同类串，而 abcde 和 abcdd 则不是。
给定 S, T，找出一个尽可能大的 k 使得 S, T 分别含有一个长度为 k 的子串 S', T'，且 S', T' 是同类串。
$1 \leq |S|, |T| \leq 10^5$，$S, T$ 中仅含英文小写字母。

首先可以发现，k 是可二分的。现在问题就转化为如何判断两个长度为 k 的子串是同类子串。
可以发现，如果两个字符串是同类字串，那么两个字符串中每个字符上一次出现的位置都是一样的。因此我们二分 k 之后，对于每个长度为 k 的子串求出每个字符上一次的出现位置，然后用字符串哈希判断是否存在两个字符串相等即可。

P7114 [NOIP2020] 字符串匹配

小 C 学习完了字符串匹配的相关内容，现在他正在做一道习题。
对于一个字符串 S，题目要求他找到 S 的所有具有下列形式的拆分方案数：$S = ABC, S = ABABC, S = ABAB \ldots ABC$，其中 A，B，C 均是非空字符串，且 A 中出现奇数次的字符数量不超过 C 中出现奇数次的字符数量。
更具体地，我们可以定义 AB 表示两个字符串 A B 相连接，例如 $A = aab$ $B = ab$，则 $AB = aabab$。
并递归地定义 $A^1 = A, A^n = A^{n-1}A(n \geq 2$且$n$为正整数$)$。例如 $A = abb$，则 $A^3 = abbabbabb$。
则小 C 的习题是求 $S = (AB)^i C$ 的方案数，其中 $F(A) \leq F(C)$，$F(S)$ 表示字符串 S 中出现奇数次的字符的数量。两种方案不同当且仅当拆分出的 A B C 中有至少一个字符串不同。
小 C 并不会做这道题，只好向你求助，请你帮帮他。
本体具有多组数据。$1 \leq T \leq 5, 1 \leq |S| \leq 2^{20}$

P7114 [NOIP2020] 字符串匹配

小 C 学习完了字符串匹配的相关内容，现在他正在做一道习题。
对于一个字符串 S，题目要求他找到 S 的所有具有下列形式的拆分方案数：$S = ABC, S = ABABC, S = ABAB\ldots ABC$，其中 A，B，C 均是非空字符串，且 A 中出现奇数次的字符数量不超过 C 中出现奇数次的字符数量。
更具体地，我们可以定义 AB 表示两个字符串 A B 相连接，例如 $A = aab$ $B = ab$，则 $AB = aabab$。
并递归地定义 $A^1 = A, A^n = A^{n-1}A(n \geq 2$且$n$为正整数$)$。例如 $A = abb$，则 $A^3 = abbabbabb$。
则小 C 的习题是求 $S = (AB)^i C$ 的方案数，其中 $F(A) \leq F(C)$，$F(S)$ 表示字符串 S 中出现奇数次的字符的数量。两种方案不同当且仅当拆分出的 A B C 中有至少一个字符串不同。
小 C 并不会做这道题，只好向你求助，请你帮帮他。
本体具有多组数据。$1 \leq T \leq 5, 1 \leq |S| \leq 2^{20}$

本题最优解应该是使用拓展 KMP 算法的线性做法，我们这里主要讲使用 KMP 或者字符串哈希的 $O(n \log n + 26n)$ 的做法。

P7114 [NOIP2020] 字符串匹配

考虑如果确定了 AB 和 i，那么我们可以判断 $(AB)^i$ 是否是原串的一个前缀。如果是，那么 C 就确定了。既然确定了 C，那么我们就可以考虑 AB 这个串，有多少种划分方法使得 $F(A) \leq F(C)$。这个我们是可以通过预处理来 $O(1)$ 得到对于任意一个后缀 C，有多少个字母出现次数为奇数次，设为 x；现在我们就是要求有多少个前缀满足该前缀中出现奇数次的字母数量不超过 x。这个我们是可以通过长度从小到大枚举 AB，一边枚举一边 $O(26)$ 预处理好答案的，或者使用一个树状数组来统计也可以。

那么我们就考虑枚举 AB 的长度以及 i，总枚举次数是 $O(n \log n)$ 的。然后使用字符串哈希或者 KMP 算法来判断一下 $(AB)^i$ 是否是原串的一个前缀，是的话统计答案即可。