字符串

whzzt

2019 年 7 月 11 日

安徽师范大学附属中学

前言

今天主要稍微讲点字符串的数据结构和一些简单应用。

前言

今天主要稍微讲点字符串的数据结构和一些简单应用。

题目大家可能都做过，求轻喷。

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。

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。也可以分别考虑前缀后缀，用 Trie 树合并来解决这个问题。

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。也可以分别考虑前缀后缀，用 Trie 树合并来解决这个问题。

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。也可以分别考虑前缀后缀，用 Trie 树合并来解决这个问题。

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 $O(s \log s)$。

BZOJ 2061 Country

给定一个只包含小写字母的字符串 S。由于串 S 太长，我们会将一些子串压缩成一个大写字母，其中压缩可能嵌套。

给定一个不含缩写的字符串 T，求 T 在 S 中出现的次数。

$1 \leq |T| \leq 100$，大写字母数不超过 26，每个串长度不超过 100，压缩过程无环。

Solution

考虑 KMP，我们压缩从每个点经过每个压缩串后 KMP 的指针位置和过程中匹配上的次数即可。

Problem

给定一个长度为 n 的字符串 s 和 m 个单词，每个单词有权值。定义一个串的价值为所有单词的价值与其出现次数的积之和。现在有 q 组询问，每次询问 s 的一个区间的价值。强制在线，要求 $\mathrm{poly}(\log)$。

Solution

根号的做法比较容易，应该大家都会。

Solution

根号的做法比较容易，应该大家都会。

考虑对所有串和单词建出 AC 自动机。我们可以注意到一个性质：AC 自动机上的一个节点的 fail 指向的位置之前的位置开始的单词，一定会被当前串完全包括。

Solution

根号的做法比较容易，应该大家都会。

考虑对所有串和单词建出 AC 自动机。我们可以注意到一个性质：AC 自动机上的一个节点的 fail 指向的位置之前的位置开始的单词，一定会被当前串完全包括。

因此直接求和即可。

BZOJ 4231 回忆树

给定一棵 n 个节点的无根树，每条边上有一个小写字符，m 次询问某个串 T 在树上从节点 u 至节点 v 的串中出现的次数。

$n, m \leq 10^5, \sum |T| \leq 3 \times 10^5$

Solution

路径的询问是容易解决的，我们提取出长度为 $2|T|$ 的串，就可以解决出现位置转弯的问题。

Solution

路径的询问是容易解决的，我们提取出长度为 $2|T|$ 的串，就可以解决出现位置转弯的问题。

于是问题转化为到根的询问。我们可以对所有询问串的正反串建立 AC 自动机。我们在原树上 DFS，离线处理询问，即可做到一个 log 。

Problem

给定一棵 n 个点的有根树，树边上有字符。有 q 个询问，每次给出一个字符串 s 和整数 k，求 s 在从根到节点 k 的字符串中出现了多少次。**强制在线。**

$n, q \leq 10^5$，询问串长之和 5×10^5。

Solution

在强制在线的背景下，问题的解法自然有所不同。

Solution

在强制在线的背景下，问题的解法自然有所不同。

如果树的形态是一条链的话，问题容易解决：其中的一种解决办法是求出后缀数组后，利用可持久化线段树处理。

Solution

在强制在线的背景下，问题的解法自然有所不同。

如果树的形态是一条链的话，问题容易解决：其中的一种解决办法是求出后缀数组后，利用可持久化线段树处理。

当树的形态不是一条链时，我们仍然可以沿用该做法。我们可以在树上同样类似地倍增求后缀数组，再用可持久化线段树解决。时间复杂度同样是一个 log。

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 $\leq 10^5$。

Little Elephant

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)。

Little Elephant

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)。

Little Elephant

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 $i > r$，$f_i = \max(f_i, \min(h_l \dots h_i))$

分别维护即可，时间复杂度 $O(l \log l)$。

BZOJ 3145 Str

给定两个串 S 和 T，求允许一次失配的情况下的最长公共子串。

串长 10^5。

Solution

首先算掉一些特殊的情况，考虑枚举失配的位置。不妨假设在第一个串中在 i 位置失配了，而在第二个串中失配的位置是 j，那么答案即为 $\mathrm{LCS}(i-1,j-1)+\mathrm{LCP}(i+1,j+1)+1$。

Solution

首先算掉一些特殊的情况，考虑枚举失配的位置。不妨假设在第一个串中在 i 位置失配了，而在第二个串中失配的位置是 j，那么答案即为 $\mathrm{LCS}(i-1,j-1)+\mathrm{LCP}(i+1,j+1)+1$。

考虑从小到大枚举 LCP，在后缀数组上合并。同时合并时我们用平衡树维护在另一个后缀数组上的位置，启发式合并即可。

Solution

首先算掉一些特殊的情况，考虑枚举失配的位置。不妨假设在第一个串中在 i 位置失配了，而在第二个串中失配的位置是 j，那么答案即为 $\mathrm{LCS}(i-1,j-1)+\mathrm{LCP}(i+1,j+1)+1$。

考虑从小到大枚举 LCP，在后缀数组上合并。同时合并时我们用平衡树维护在另一个后缀数组上的位置，启发式合并即可。

时间复杂度 $O(n\log^2 n)$。

BZOJ 2628 JZPSTR

你要对一个字符串进行以下三种操作：在某个位置插入一个字符串，删去字符串的某个区间，询问某个区间的字符串包含多少次某个给定的子串。

数据范围自行脑补，当然要求尽量优的复杂度。

Solution

在中间插入删除显然可以利用块状链表解决，而求包含某个子串的次数，显然是一个后缀自动机的问题。

后缀自动机

对于一个串的某个子串来说，其在该串中出现了若干次，我们将这些出现位置的右端点的集合称作这个子串的 Right 集合。

后缀自动机

对于一个串的某个子串来说，其在该串中出现了若干次，我们将这些出现位置的右端点的集合称作这个子串的 Right 集合。

容易发现，对于一个串来说，不同的 Right 集合不会超过 $2n$ 种，因为两个不同的 Right 集合要么互不相交，要么互相包含。对于每个 Right 集合来说，它所表示的串的长度一定是某个区间 $[l, r]$，且当串长超过 r 时，Right 集合会变小；当串长不足 l 时，Right 集合会变大。

后缀自动机

对于一个串的某个子串来说，其在该串中出现了若干次，我们将这些出现位置的右端点的集合称作这个子串的 Right 集合。

容易发现，对于一个串来说，不同的 Right 集合不会超过 $2n$ 种，因为两个不同的 Right 集合要么互不相交，要么互相包含。对于每个 Right 集合来说，它所表示的串的长度一定是某个区间 $[l, r]$，且当串长超过 r 时，Right 集合会变小；当串长不足 l 时，Right 集合会变大。

因此，我们可以对一个串的 Right 集合建立一棵树形结构，根节点的 Right 集合是全集，叶子节点的 Right 集合仅包含一个元素。

后缀自动机

我们尝试建立一个能识别给定的串 S 是否是模板串子串的自动机。容易发现，模板串的每一个子串都有一个唯一的 Right 集合，而每个 Right 集合中的串都是模板串的子串。与此同时，对于同一个 Right 集合里的串来说，我们在它们的后面插入同一个字符 c，都会到达相同的 Right 集合。

后缀自动机

我们尝试建立一个能识别给定的串 S 是否是模板串子串的自动机。容易发现，模板串的每一个子串都有一个唯一的 Right 集合，而每个 Right 集合中的串都是模板串的子串。与此同时，对于同一个 Right 集合里的串来说，我们在它们的后面插入同一个字符 c，都会到达相同的 Right 集合。

因此，我们只需要对所有的 Right 集合建立转移边即可。当我们需要判定是否是子串时，可以直接在自动机上行走。而当我们要匹配时，我们可以在向后插入字符 c 不满足子串条件时，每次走向 Right 集合树上的父亲，即保留最长的一个串仍是原串的子串。

后缀自动机

我们考虑递进地构建原串的后缀自动机，即每次在串后增加上一个字符。

后缀自动机

我们考虑递进地构建原串的后缀自动机，即每次在串后增加上一个字符。

当我们在串的末尾增加字符时，实际上就是增加了一个 Right 集合的元素，也就是增加了一个后缀。首先，我们先增加一个表示整串的节点，这个节点显然不会有出边，于是我们只要考虑入边。显然，原本的整串节点是可以转移到这个新状态的。

后缀自动机

我们考虑递进地构建原串的后缀自动机，即每次在串后增加上一个字符。

当我们在串的末尾增加字符时，实际上就是增加了一个 Right 集合的元素，也就是增加了一个后缀。首先，我们先增加一个表示整串的节点，这个节点显然不会有出边，于是我们只要考虑入边。显然，原本的整串节点是可以转移到这个新状态的。

而其他可以转移到当前状态的节点一定都是原来的整串节点在 Right 集合的树上的祖先。有一些这样的祖先节点原本没有当前字符的出边，那么就增加上一条出边到整串节点。

后缀自动机

我们考虑递进地构建原串的后缀自动机，即每次在串后增加上一个字符。

当我们在串的末尾增加字符时，实际上就是增加了一个 Right 集合的元素，也就是增加了一个后缀。首先，我们先增加一个表示整串的节点，这个节点显然不会有出边，于是我们只要考虑入边。显然，原本的整串节点是可以转移到这个新状态的。

而其他可以转移到当前状态的节点一定都是原来的整串节点在 Right 集合的树上的祖先。有一些这样的祖先节点原本没有当前字符的出边，那么就增加上一条出边到整串节点。

如果所有的这些祖先节点都没有过当前字符的出边的话，我们可以发现当前 Right 集合是直接挂在根节点下的，于是直接处理就可以了。

后缀自动机

否则我们考虑最深的一条出边，那么对于这条出边指向的节点来说，我们需要塞一个右端点进 Right 集合。考虑这条出边的起始节点的串长的话，我们有可能新建一个节点：因为原本到达的节点的更长的串并没有新增这个 Right。

后缀自动机

否则我们考虑最深的一条出边，那么对于这条出边指向的节点来说，我们需要塞一个右端点进 Right 集合。考虑这条出边的起始节点的串长的话，我们有可能新建一个节点：因为原本到达的节点的更长的串并没有新增这个 Right。

如果新增了节点的话，需要修改原本指向该节点的出边。这样的一个后缀自动机的构造过程是 $O(n)$ 的。（严格 O(n) 可以写 hash 表，不过效率意外地还不错）。

BZOJ 3926 诸神眷顾的幻想乡

给一棵树，路径上有字符，问有多少种本质不同的路径。

$n \leq 10^5$，叶子节点不超过 20 个

Solution

如果只有一个串，可以直接建出 Right 集合树，并统计所有串的个数。

Solution

如果只有一个串，可以直接建出 Right 集合树，并统计所有串的个数。

有多个串时，我们可以建立“广义后缀自动机”，即对 Trie 建立一个后缀自动机。我们可以通过 BFS 的方式来完成这一点。

Solution

如果只有一个串，可以直接建出 Right 集合树，并统计所有串的个数。

有多个串时，我们可以建立 “广义后缀自动机”，即对 Trie 建立一个后缀自动机。我们可以通过 BFS 的方式来完成这一点。

需要注意的是，若不使用 BFS 而是直接 DFS 建立的话，时间复杂度是和 Trie 的叶子深度和有关的，而不是 Trie 上的节点数。

又一道有趣的字符串题

求一个串的所有子串的后缀树节点数之和。

$n \leq 5 \times 10^5$

Solution

将串翻转，即为所有子串的后缀自动机节点数之和。考虑将后缀自动机的节点分为每次插入字符时的新增节点和我们在连出边时新增的节点。

Solution

将串翻转，即为所有子串的后缀自动机节点数之和。考虑将后缀自动机的节点分为每次插入字符时的新增节点和我们在连出边时新增的节点。

对于一个串来说，第一类节点的数量是固定的，而第二类节点的数量，即为不是该串前缀的 S 的，满足串中至少出现过 aS 和 bS 的 S 的个数。

Solution

首先考虑一个暴力：我们可以枚举原串，枚举 S 来计算答案。但这样的复杂度过高，我们可以考虑在后缀自动机上枚举 S。我们从上一个右端点的基础上接着做，便只要考虑当前的后缀的 S。

Solution

首先考虑一个暴力：我们可以枚举原串，枚举 S 来计算答案。但这样的复杂度过高，我们可以考虑在后缀自动机上枚举 S。我们从上一个右端点的基础上接着做，便只要考虑当前的后缀的 S。

由于同时存在 aS 和 bS，我们只需要考虑后缀自动机每个节点的最长串即可。而对于同一次被某个之前的右端点访问的这些节点来说，其中也一定只有一个 S 是有用的——我们只考虑相邻的 aS 和 bS 就可以处理答案，而这种情况下相邻的 S 的串首字符一定相同。于是，我们可以使用 LCT 维护。

Solution

首先考虑一个暴力：我们可以枚举原串，枚举 S 来计算答案。但这样的复杂度过高，我们可以考虑在后缀自动机上枚举 S。我们从上一个右端点的基础上接着做，便只要考虑当前的后缀的 S。

由于同时存在 aS 和 bS，我们只需要考虑后缀自动机每个节点的最长串即可。而对于同一次被某个之前的右端点访问的这些节点来说，其中也一定只有一个 S 是有用的——我们只考虑相邻的 aS 和 bS 就可以处理答案，而这种情况下相邻的 S 的串首字符一定相同。于是，我们可以使用 LCT 维护。

时间复杂度 $O(n \log n)$。

Manacher

Manacher 算法可以在 $O(n)$ 时间内求出一个串中以所有位置为对称轴的最长回文串长度。

Manacher

Manacher 算法可以在 $O(n)$ 时间内求出一个串中以所有位置为对称轴的最长回文串长度。

该算法的实现也非常简单：记录当前已知的最靠右的回文串右端点即可。

回文树

我们可以类似后缀自动机地来考虑建造回文自动机（回文树），即能够识别一个串的回文子串的自动机。在回文自动机上，一个串的 fail 是其最长回文后缀，转移边 c 代表在串的两端各添加一个字符。

回文树

我们可以类似后缀自动机地来考虑建造回文自动机（回文树），即能够识别一个串的回文子串的自动机。在回文自动机上，一个串的 fail 是其最长回文后缀，转移边 c 代表在串的两端各添加一个字符。

仍然考虑在字符串后增加一个字符 c，显然至多新增一个新的回文串。如果没有新增，我们直接移动到那个节点就可以了，否则我们会新建一个节点。显然这个节点的 fail 指针可以通过匹配求得，而转移边只会新增一条。于是，我们构造出了一个这样的回文自动机。

Problem

给定两个串 S 和 T，翻转 S 中最少的区间使得两串相等。

$n \leq 10^6$

Solution

我们将两个串交替拼在一起，问题变成了将一个串划分成尽量少的长度为偶数的回文子串的问题。这个问题直接暴力是平方的。

Solution

我们将两个串交替拼在一起，问题变成了将一个串划分成尽量少的长度为偶数的回文子串的问题。这个问题直接暴力是平方的。

考虑利用回文自动机解决，由于一个串的后缀回文串的长度构成 $O(\log n)$ 个等差数列，我们可以考虑对一个等差数列一起转移。这样就可以做到 $O(n \log n)$。

z-function

对于一个长度为 n 的字符串 s，定义函数 $z(i)(2 \leq i \leq n)$ 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。

$n \leq 100$

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。

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。

由于一个串的 border 的 border 仍是他的 border，因此我们可以从短到长搜索 border 的长度。

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，mod666623333 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
$f(l+p-1) + \ldots + f(r+p-1) - 10^p * (f(l-1) + \ldots + f(r-1))$。

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
$f(l+p-1) + \ldots + f(r+p-1) - 10^p * (f(l-1) + \ldots + f(r-1))$。

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 $S = \mathrm{abab}$ 是循环的，因为存在 $k = 2, s = \mathrm{ab}$。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。

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 $O(|s| \log |s|)$ 。

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 $O(|s| \log |s|)$ 。

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。

Thanks!