# JZPSTR解题报告

南京师范大学附属中学　顾昱洲

## Contents

**1　题目大意** **2**

**2　基础知识** **2**

2.1　块状链表 . . . . . 2

2.2　字符串 . . . . . 2

2.2.1　多项式hash . . . . . 2

2.2.2　一些字符串数据结构 . . . . . 3

**3　算法分析** **3**

**4　50%的算法** **3**

# 1 题目大意

对一个字符串进行三种操作：

- 在位置$x_i$处插入一个字符串$y_i$；
- 删除位置$[x_i, y_i)$的字符串；
- 查询位置$[x_i, y_i)$的字符串包含多少次给定的子串$z_i$。

(这里是题目大意，忽略输入输出的细节)

$|\Sigma| = 10$

50%的数据，询问个数$\leq 100$；

100%的数据，插入总长度$\leq 2000000$，任何时刻字符串总长度$\leq 1000000$，插入次数$\leq 1001$，删除次数$\leq 1000$，询问的$z_i$的总长度$\leq 10000$。

# 2 基础知识

解决这道题需要一定的数据结构及字符串基础。集训队员应当已经掌握这些知识。为了使得NOI铜牌线的同学能够理解本解题报告，这里简单介绍需要的基础知识。

## 2.1 块状链表

假设有一个长度为$n$的数组，我们要在其上进行一些操作。两个基本的操作是插入和删除。这时使用块状链表是一种不错的方法。

将数组分成$\sqrt{n}$左右个块，每个块大小为$\sqrt{n}$左右。

插入时，将待插入的文本分成大小为$\sqrt{n}$左右的块，然后将原数组中插入位置所在的块在插入位置处断裂成两个块，然后将待插入文本分成的块用链表接进去。

删除时，从删除位置向后数删除长度，将被完全包含的块直接删去，至多有两个块被部分包含且不被完全包含，暴力删除即可。

在插入和删除后，都要保证每个块的大小在$[\sqrt{n}/2, 2\sqrt{n}]$之间，以保证复杂度。

这样插入和删除都是$O(\sqrt{n})$的。

## 2.2 字符串

### 2.2.1 多项式hash

令字符串为$s_1 s_2, \ldots, s_n$，那么设$H_0 = 0$，$H_i = H_{i-1} P + s_i$。

其中$P$为一个奇质数。若使用C++ 的int 或long long 等，可以直接使用以上式子，忽视溢出。

那么该字符串长度为$i$的前缀hash值为$H_i$，该字符串的hash值为$H_n$。

我们预处理$P$的幂，那么可以快速更新在一个字符串右端加上一个字符，或在左端删去一个字符后得到的字符串的hash值。

利用这一点可以在线性的时间内求出一个字符串在另一个字符串中的出现次数，并且常数非常小。

### 2.2.2 一些字符串数据结构

树状数组，其定义及线性构造可以参考 [1]。值得注意的是，线性构造的常数非常大。实践中速度往往不如$O(n \log n)$的倍增算法和$O(n \log^2 n)$的使用hash的暴力sort。

后缀树，其线性构造可以参考 [2]。

后缀自动机，其定义及线性构造可以参考 [3]。

## 3 算法分析

由于字符串长度非常大，我们考虑用块状链表来解决这个问题。

块状链表的插入和删除都属于基础操作，这里我们主要考虑询问。假设询问的串的长度为$m$。

询问的时候，被询问区间肯定覆盖不超过$O(\sqrt{n})$个完整的块，至多非完整覆盖2个块。非完整覆盖的2个块直接处理即可，时间复杂度$O(\sqrt{n})$。下面只考虑完整覆盖的块。

一个询问的串在被询问区间中的出现，只存在两种情况：

- 该串被完全包含在某个块中
- 该串跨越了两个或多个块

跨越了两个或多个块的情况，我们可以知道，这时候的不同的情况至多有$m\sqrt{n}$种，暴力处理即可。这一部分时间复杂度为$O(m\sqrt{n})$。

否则，我们预处理出该块的后缀自动机(或后缀树，但是这里后缀自动机比后缀树的常数和编写复杂度都有优势)。然后进行$O(m)$的询问。这种情况至多出现$O(\sqrt{n})$次。因此这一部分时间复杂度为$O(m\sqrt{n})$。

因此一次询问的总时间复杂度为$O(m\sqrt{n})$。

询问的总的时间复杂度为$O(\sqrt{n}\sum m_i)$，注意这里$\sum m_i \leq 10000$。

但是为了处理询问，现在多出了一个问题。就是插入和删除的时候块内保存的后缀自动机的信息会改变。只要注意到每次插入或删除只会变更常数个数个块的信息。暴力变更即可。

这样插入和删除的时间复杂度为$O(|\Sigma|\, p\sqrt{n})$，其中$p$为插入和删除的总次数。通过优化可以将时间复杂度中的$|\Sigma|$去掉，不过那样常数会极大增加。

总的时间复杂度是$O(|\Sigma|\, p\sqrt{n} + m_s\sqrt{n})$，其中$m_s = \sum m_i$。

空间复杂度是$O(|\Sigma|\, n)$，注意需要一开始就新建所有结点以加快速度。中间过程中使用垃圾回收，以保证当前总的在使用的后缀自动机的结点不超过$2n$个。

这个算法是100%的算法。

## 4 50%的算法

注意到50%的数据中，询问个数非常少，我们可以对于每次询问暴力处理。插入和删除则使用块状链表。由于不需要维护附加信息，因此这个方法的实现非常简单。

时间复杂度$O(p\sqrt{n} + wn)$，其中$w$为询问的总个数。

需要注意的是，有的同学可能会使用C++的crope来简化代码。但是很可惜，在不开O2的时候，crope的速度非常慢(主要体现在询问某位置上的字符非常慢)。因此使用crope是不行的。

## 参考文献


[1] J. Kärkkäinen, P. Sanders, S. Burkhardt, Linear Work Suffix Array Construction, *Journal of ACM*, **53** (6): 918 - 936, 2006

[2] E. Ukkonen, On-line Construction of Suffix Trees, *Algorithmica*, **14** (3): 249 - 260, 1993

[3] M. Mohri, P. Moreno, E. Weinstein, General Suffix Automaton Construction Algorithm and Space Bounds, *Theoretical Computer Science*, **410** (37): 3553 - 3563, 2009