# 全动态模型下最短唯一回文子串问题的数据结构与算法

申治洺

2024 年 5 月 18 日

**摘要**

本文在 Data Structures for Computing Unique Palindromes in Static and Non-Static Strings[1]（以下称原文）的基础上，继续探讨了最短唯一回文子串问题的求解。本文使用较大篇幅概述原文，尤其是 Inoue 算法 [2] 及其改进，这是求解最短唯一回文子串问题的基础。本文调研了回文子串领域的研究成果，主要包括最短唯一回文子串、极小唯一回文子串、极大回文子串这三个方面。最后，本文针对原文仅支持两种半动态模型，不支持全动态模型，适用范围过于狭窄的缺陷，提出了一种全动态模型，并提出了支持在动态串的任意位置插入和删除操作的求解最短唯一回文子串问题的算法。该算法的空间复杂度为 $O(n)$，初始构建数据结构的时间复杂度为 $O(n)$，应对插入或删除操作，更新数据结构并计算最短唯一回文子串的时间复杂度为 $\tilde{O}(m)$，其中 $n$ 为原始串的长度，$m$ 为被操作位到串尾的距离。

**关键词**

- 最短唯一回文子串 (shortest unique palindromic substring, SUPS)
- 极小唯一回文子串 (minimal unique palindromic substring, MUPS)
- 极大回文子串 (maximal palindrome)
- 滑窗模型 (sliding-window model)
- 替换模型 (after-substitution model)

## 1 原文概述

本文在概述原文时，着重介绍数据结构的设计。对于原文中涉及的大量引理 (Lemma)、定理 (Theorem)、命题 (Proposition)、推论 (Corollary)，本文直接引用其中的结论，不再关注其证明过程，因为原文提供了详尽的证明

或证明所在的参考文献。由于原文中直接引用了一些不太常用的数据结构，如后缀树、van Emde Boas 树等，本文会适当补充相关数据结构的定义、实现、性质。对于原文中定义不够明确的概念，本文给出了严格的定义，并描述和论证了相关性质。

## 1.1 问题引入

**MUPS** 给定串 $T$，$1 \le i \le j \le ||T||$，回文子串 $u = T[i..j]$，如果 $u$ 在 $T$ 中是唯一的，$u' = T[i+1..j-1]$ 在 $T$ 中不唯一，则称 $u$ 是 $T$ 的极小唯一回文子串 (minimal unique palindromic substring).

**interval SUPS** 给定串 $T$ 和区间 $[p,q]$，回文子串 $v = T[i..j]$，$v$ 在 $T$ 中唯一且包含区间 $[p,q]$，任何更短的包含区间 $[p,q]$ 的 $T$ 的回文子串都在 $T$ 中不唯一，则称 $v$ 是对区间 $[p,q]$ 的 $T$ 的最短唯一回文子串 (shortest unique palindromic substring)，记作 $v = \mathrm{SUPS}_T([p,q])$.

**point SUPS** 区间 SUPS 问题在 $p = q$ 的情况下退化为点的 SUPS 问题，记作 $v = \mathrm{SUPS}_T(p)$.

**ME** MUPS 向外扩展，即每次起点向前移动一位，终点向后移动一位，直至破坏回文性为止。在此过程中可以得到“极小唯一回文子串的极大扩展”(maximal extension of MUPS).

**MP** 对任意一个半整数，即 1, 1.5, 2, 2.5 .. , n [1]，都能求得以其为中心的极大回文子串 (maximal palindrome).

需要注意的是，MUPS 是极小的，强调的是 MUPS 都无法在自身基础上再缩小一点，因为再缩小会违背 MUPS 的唯一性，MUPS 的定义并不关心每个 MUPS 与其他 MUPS 的长度比较。SUPS 是最短的，SUPS 的定义基于多个包含给定区间或点的回文子串的长度比较，但这并不意味着 SUPS 仅有 1 个。原文中的 Theorem 1 说明 SUPS 至多只有 4 个。

由 MUPS 的定义可知，MUPS 具有唯一性，此外 MUPS 不存在嵌套的情况，可用反证法论证如下：

取 MUPS $M_1 = T[s_1..e_1]$，MUPS $M_2 = [s_2..e_2]$，满足 $s_1 \le s_2 \le e_1 \le e_2$，且 $s_1 \le s_2$ 和 $e_1 \le e_2$ 不能同时取等。找到 $M_1$ 的对称轴 $l$，若 $M_2$ 关于 $l$ 对

[1] 原文中串从 1 开始标号，本文也采用这种规范。

称，则 $M_1$ 显然不满足 MUPS 定义中的极小性；若 $M_2$ 不关于 $l$ 对称，则取串 $M_3$ 关于轴 $l$ 与 $M_2$ 对称，由于 $M_2$ 是回文串，$M_3$ 与 $M_2$ 完全一致，显然违背了 $M_2$ 作为 MUPS 的唯一性。

原文中使用的“极大回文子串”(maximal palindrome) 一词，其实兼指 ME 和 MP。这里明确地区分了这两个概念，并在下文探讨两者的性质和联系。

ME 具有唯一性和非嵌套性，可用反证法论证如下：

假设存在两个完全一致的 ME $m_1$ 和 $m_2$，$m_1$ 对应的 MUPS 为 $M_1$，以 $m_1$ 和 $M_1$ 的相对位置关系在 $m_2$ 中找到回文子串 $M_2$，$M_2$ 一定与 $M_1$ 不重合，于是违背了 $M_1$ 作为 SUPS 的唯一性。ME 的唯一性得证。

取 ME $m_1 = T[s_1..e_1]$，ME $m_2 = [s_2..e_2]$，满足 $s_1 \le s_2 \le e_1 \le e_2$，且 $s_1 \le s_2$ 和 $e_1 \le e_2$ 不能同时取等。找到 $m_1$ 的对称轴 $l$，若 $m_2$ 关于 $l$ 对称，则 $m_2$ 不满足 ME 定义中的极大性；若 $m_2$ 不关于 $l$ 对称，则取串 $m_3$ 关于轴 $l$ 与 $m_2$ 对称，由于 $m_2$ 是回文串，$m_3$ 与 $m_2$ 完全一致，$m_2$ 不满足上文已证的 ME 的唯一性。ME 的非嵌套性得证。

显然，MP 完全包含了 ME。通常情况下，MP 包含了大量的空串和单字符串，并有大量重复情况。例如，串 $T$ = babbbabbababb，以 2 为中心的 MP bab 和以 11 为中心的 MP bab 发生重复，因此两者都不包含在 ME 中。

事实上，唯一且非空的 MP 等同于 ME。证明如下：

对任意非空的 MP，定义“缩小串集”为 MP 缩小[2]过程中产生的所有回文串的总和。易知，“缩小串集”包含了所有回文子串，也就包含了所有 ME。不唯一的 MP 缩小产生的回文串也是不唯一的，因而不能产生 ME。对唯一的 MP 进行缩小，缩小的尽头是空串，必然丧失唯一性。因此，在缩小的过程中一定存在唯一性发生改变的临界，即可找到一个 MUPS，则该唯一的 MP 也就成为一个 ME。综上，唯一且非空的 MP 与 ME 一一对应。

### 1.1.1 最长公共扩展 (LCE) 与后缀树 (suffix tree)

对于长度为 $n$ 的串 $S$，后缀树 (suffix tree) 定义为这样一棵树，满足如下五个条件[3]：

- 有 $n$ 个子节点，分别编号 1 至 $n$

[2]类似于扩展，缩小指每次将回文串的起点向后移动一位，终点向前移动一位。

[3]后缀树的定义和图 1引自 Suffix tree *Wikipedia*, 可参见 https://en.wikipedia.org/wiki/Suffix_tree

- 除根节点外，每个非叶节点都有至少两个子节点
- 每条边都用 $S$ 的一个非空子串来标记
- 起始于同一节点的两条边不能有同一字符开头的标记
- 从根节点到叶节点 $i$,将所有边的标记依次拼接起来,可得到后缀 $S[i..n]$

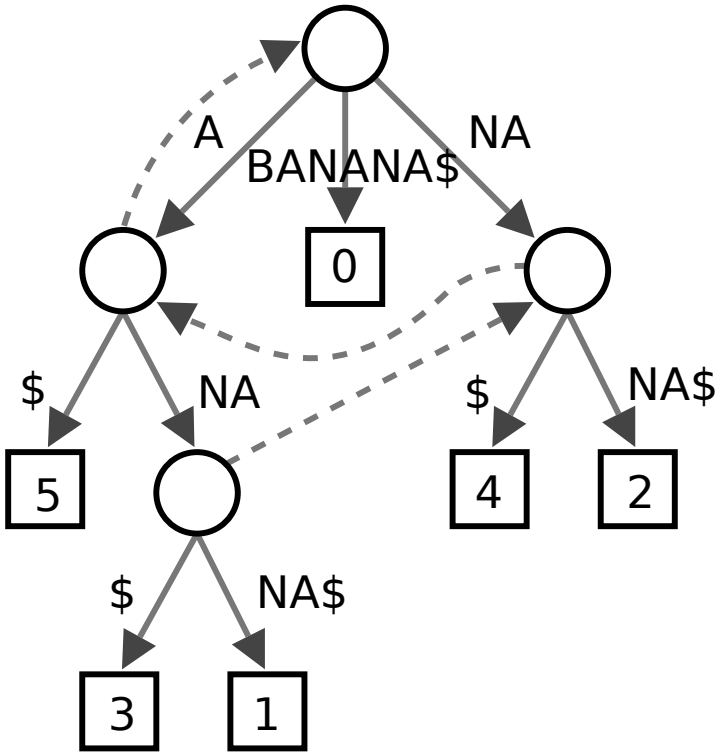


图 1: BANANA$ 的后缀树，虚线为构建过程中的辅助线

最长公共扩展 (longest common extension, LCE) 问题是对一个给定的串 $T$，给定的整数 $i, j$，满足 $1 \leq i, j \leq n$，计算出两个后缀 $T[i..n]$ 和 $T[j..n]$ 的最长公共前缀的长度。

### 1.1.2 区域最小值请求 (RmQ) 及其变体 LogRmQ

区域最小值请求 (range minimum query ,RmQ) 是对一个给定的整型数组 $A$，$A$ 上的两个下标 $i, j$，满足 $i \leq j$，计算出下标 $k$ 使得 $A[k]$ 为 $A[i..j]$

上的最小值。注意 RmQ 数据结构并不要求整型数组有序。

### 1.2 前驱 (predecessor)、后继 (successor) 与 van Emde Boas 树

对一个不减的整型数组 $B$，给定整型数 $x$，定义前驱 (predecessor) 是小于 $x$ 的最大值；同理，定义后继 (successor) 是大于 $x$ 的最小值。

### 1.3 Inoue 算法概述

### 1.4 原文对 Inoue 算法的改进

### 1.5 原文算法的示例

## 2 领域综述

回文串是从前向后和从后向前读取结果相同的序列。由于其独特的性质以及在模式识别、数据压缩和 DNA 序列分析中的广泛应用，在生物信息学、计算机科学等领域，对回文串的研究不断。本节探讨了回文串的主要研究方向，包括但不限于极小唯一回文子串、最短唯一回文子串和极大回文子串。

### 2.1 极小唯一回文子串

极小唯一回文子串 (MUPS) 的定义和性质详见 1.1。求解 MUPS 的算法设计策略总结如下：

- 基于后缀树。对输入的串构建后缀树，从后缀树中获取所有极小唯一的子串，再从中提取回文串 [3]。这种策略需要消耗 $O(n)$ 的时间以完成构建后缀树的预处理 [4]。
- 基于动态规划。使用一个动态规划的表格标记出所有回文子串，再依次检查每个回文子串是否唯一。时间复杂度为 $O(n)$[5]。

### 2.2 极大回文子串

极大回文子串 (MP) 的定义和性质详见 1.1。求解 MP 的算法设计策略总结如下：

- 使用 Manacher 算法 [5]。这是一个求解 MP 的在线 (on-line) 算法[4]，可以在线性时间内获得结果。
- 使用 Eertree (Palindromic Tree)[6]。这种数据结构可以储存所有互异回文子串，以便快速求解 MP。

### 2.3 最短唯一回文子串

最短唯一回文子串 (SUPS) 的定义和性质详见 1.1。求解 SUPS 问题，可以使用 Inoue 算法 [2]，详见 1.3；以及原文 [1] 对 Inoue 算法的改进算法，详见 1.4和 1.5。

### 2.4 回文分解

回文分解 (palindromic factorization) 是将一个串划分为最少数量的回文串的过程，常用于数据压缩和文本挖掘。借助 Eertree 数据结构，k-分解问题可以在 $O(kn)$ 时间内完成 [6]。

## 3 当前算法的不足

原文基于 Inoue 算法提出了区间 SUPS 问题和点的 SUPS 问题的优化算法，只需要 $O(n)$ 空间的数据结构，$O(n)$ 的预处理时间，即可在常数时间内计算出 SUPS 问题的解。如果只使用大 O 记法，不考虑时空复杂度的系数，那么这两类问题的算法已经难以继续优化了。

原文的后半部分提出了两种半动态模型 (semi-dynamic model)，并据此初步探讨了动态情形下 SUPS 问题的求解。

**滑窗模型** 每次支持的操作为，删除最左端的一位，或在最右端添加一位。滑窗模型构建了一个不断向右移动的窗，期间窗的大小[5]可能发生变化。

**替换模型** 每次支持替换串中的一位为字母表[6]中的另一个字符。

原文中为滑窗模型提出了一个相当高效的算法，支持在每次操作后以均摊的 $O(\log\sigma + \log\log W)$ 时间调整数据结构，并在 $O(\log\log W)$ 求

[4]即支持一位一位地按顺序输入串，运行过程中的结果是对于已输入串的问题解。

[5]原文中窗的大小用 $W$ 表示，本文也采取这种规范。

[6]原文中字母表的大小用 $\sigma$ 表示，本文也采取这种规范。

解任何 SUPS 问题。原文中为替换模型提出的算法也相当高效，可以在 $O(\log n \log\log n + k \log\log n)$ 时间[7]内解决任意 $k$ 个 SUPS 问题。这两种算法的空间复杂度和预处理过程的时间复杂度与静态模型一致。

在半动态模型下，被操作位的位置和操作方式都受到较大限制，因而很难满足实际应用的需求。本文基于原文中的两个半动态模型提出全动态模型，支持对原串进行如下三种操作：

- 对原串中任一个间隔，以及开始处和终止处，插入一个新位
- 将原串中任一位删除
- 将原串中任一位替换，即用字母表中的其他字母来取代它

# 4 改进算法的提出

本节为 3提出的全动态模型设计实现算法。

# 参考文献

[1] T. Mieno and M. Funakoshi, “Data structures for computing unique palindromes in static and non-static strings,” *Algorithmica*, vol. 86, no. 3, pp. 852–873, 2024. [Online]. Available: https://doi.org/10.1007/s00453-023-01170-8

[2] H. Inoue, Y. Nakashima, T. Mieno, S. Inenaga, H. Bannai, and M. Takeda, “Algorithms and combinatorial properties on shortest unique palindromic substrings,” *J. Discrete Algorithms*, vol. 52-53, pp. 122–132, 2018. [Online]. Available: https://doi.org/10.1016/j.jda.2018.11.009

[3] D. Gusfield, *Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology*. Cambridge University Press, 1997.

[4] E. Ukkonen, “On-line construction of suffix trees,” *Algorithmica*, vol. 14, no. 3, pp. 249–260, 1995. [Online]. Available: https://doi.org/10.1007/BF01206331

[7] 已包含调整数据结构的时间。

[5] G. K. Manacher, “A new linear-time ”on-line” algorithm for finding the smallest initial palindrome of a string,” *J. ACM*, vol. 22, no. 3, pp. 346–351, 1975. [Online]. Available: https://doi.org/10.1145/321892.321896

[6] M. Rubinchik and A. M. Shur, “EERTREE: an efficient data structure for processing palindromes in strings,” *Eur. J. Comb.*, vol. 68, pp. 249–265, 2018. [Online]. Available: https://doi.org/10.1016/j.ejc.2017.07.021