Details about time complexity

Manacher's Algorithm is an efficient algorithm for finding the longest palindromic substring in a given string. Here are the details about its time complexity:

1.Preprocessing Time Complexity: O(n)

- The preprocessing step involves initializing the LPS (Longest Palindromic Substring) array and setting initial values. This step takes linear time proportional to the length of the input string. Specifically:

  1.Initializing the LPS array takes O(n) time because each position in the array needs to be initialized.

  2. Setting initial values, such as L[0] = 0 and L[1] = 1, takes constant time.

2.Main Algorithm Time Complexity: O(n)

- The core of Manacher's Algorithm is the main loop that traverses the input string to calculate the LPS array. This loop contributes to the linear time complexity of the algorithm.
- Within this loop, there are mainly two types of operations:

  1. Comparisons and updates related to the LPS array, which are typically constant time operations..

  2. Palindrome expansion checks, which involve comparing characters in the string. However, these checks are done efficiently without rechecking palindromic properties unnecessarily.

3.Total Time Complexity: O(n)

- Combining the preprocessing and main algorithm, the total time complexity of Manacher's Algorithm remains linear, denoted as O(n)..
  The algorithm's efficiency comes from its ability to avoid redundant checks. For example:

  1. It utilizes information from previously computed palindromes to avoid rechecking parts of the string unnecessarily.

  2. It leverages symmetry properties of palindromes to efficiently expand and update the LPS array.

In summary, Manacher's Algorithm achieves linear time complexity by smartly managing palindrome expansions and leveraging previously computed information. This makes it highly efficient for finding the longest palindromic substring in a given string.