Documentation

The problem of finding the shortest palindrome involves converting a given string **s** into a palindrome by adding the minimum number of characters to the front of the string. A palindrome is a string that reads the same forwards and backward, and the goal is to achieve this by transforming **s** in the shortest way possible. The key to solving this problem efficiently is identifying the longest palindromic prefix in the string. Once we find this, we can append the remaining non-palindromic characters from the reversed string to the front of **s**.

We approach the solution by reversing the string **s**. This reversed version of the string helps in identifying the palindromic properties of **s**. We then combine the original string and its reversed version using a separator, ensuring no false matches occur while detecting palindromic prefixes. The combined string allows us to apply the Knuth-Morris-Pratt (KMP) algorithm, which is typically used for string matching but, in this case, helps to compute the longest prefix of the string which is also a suffix. This information is stored in the Longest Prefix Suffix (LPS) array, which tells us the length of the longest palindromic prefix in **s**.

The LPS array is built for the combined string, and the value at the last index of the LPS array indicates how much of the original string **s** is already a palindrome. Using this value, we can determine the number of characters that need to be added to the front of the string to make it a palindrome. These characters come from the reversed version of **s**. Once we know which characters to add, we append them to the front of **s**, resulting in the shortest possible palindrome. This method is efficient, with a time complexity of **O(n)** due to the linear time KMP algorithm, and a space complexity of **O(n)** for storing the LPS array. By using this approach, we can quickly and optimally find the shortest palindrome for any given input string.