A palindrome is a string equal to its reversal. Since letters are palindromes, each string can be obtained by concatenating palindromes. The minimum number of factors in such concatenation is the \( \text{palindromic length} \) of a string, denoted by \( \text{PL}(S) \). Thus, if \( S = 01001001 \), then \( \text{PL}(S) = 2 \) because \( S = 0 \cdot 1001001 \) and \( S \) is not a palindrome.

Dynamic programming is a natural approach to the computation of the palindromic length: for a string \( S \) of length \( n \) define an array \( \text{PL}[0..n] \) such that \( \text{PL}[i] \) is the palindromic length of the string \( S[1..i] \) (in particular, \( \text{PL}[n] = \text{PL}(S) \)). Now

\[
\text{PL}[0] = 0, \quad \text{PL}[i] = 1 + \max\{\text{PL}[j] \mid S[j+1..i] \text{ is a palindrome}\}.
\]

In a naive way, the implementation of such an algorithm requires \( \Omega(n^2) \) time and space but this bound can be lowered to \( O(n \log n) \) by processing “series” of palindromic suffixes of the current string rather than individual such suffixes [1]. (Essentially the same algorithm was obtained independently in [2].) There are \( O(\log n) \) series of palindromic suffixes in each position of the word and every series can be processed in constant time. On the other hand, [1] contains an example of a string having \( \Omega(n \log n) \) series of palindromes in total. This example speaks in favor of an idea that \( O(n \log n) \) is the best possible time for finding the palindromic length.

In our IWOCA 2015 paper [3] we exhibit a new data structure called \text{eertree}, which allows one to store all data about palindromes in a string in a compact form. This data structure can be built in linear time (offline; the online bound is only slightly worse). Up to now, we were able to get a new \( O(n \log n) \) algorithm for the palindromic length using eertree. However, the potential of this data structure seems to be big, so we ask the following question:

\[ \text{Is it possible to avoid explicit processing of each single series of palindromic suffixes and improve the } O(n \log n) \text{ time bound for the palindromic length problem?} \]

\[ 1 \text{Probably this citation should be updated later by the official publication.} \]
References

