R-9.11

What is the longest prefix of the string "cgtacgttcgtacg" that is also a suffix of this string?

The longest prefix that is also the suffix of this string is "cgtacg".

C-9.4

Let T be a text of length n, and let P be a pattern of length m. Describe an $O(n+m)$-time method for finding the longest prefix of P that is a substring of T.

Algorithm: we can use KMP pattern matching algorithm in this question and we need to make some modification. We also need to initialize three variables to 0: index_max, length_max and length_current. In the algorithm, we need to add the following function to the KMP algorithm:

if $P(j) = T(i)$, then $\text{length\_current} = \text{length\_current} + 1$.

if $P(j) \neq T(i)$ and $j > 0$,

if $\text{length\_max} < \text{length\_current}$, then $\text{length\_max} = \text{length\_current}$ and $\text{index\_max} = i - j$,

if $\text{length\_max} \geq \text{length\_current}$, then $\text{length\_current} = 0$.

At last, the length_max is the longest length and index_max is the location of this string starting.

Running time: it takes $O(n+m)$-time to run the KMP algorithm. As a result, the total running time is $O(n+m)$.

C-9.5

Say that a pattern P of length m is a circular substring of a text T of length n if there is an index $0 \leq i \leq m$, such that $P = T[n-m+i..n-1] + [0..i-1]$, that is, if P is a substring of T or P is equal to the concatenation of a suffix of T and a prefix of T. Given an $O(n+m)$-time algorithm for determining whether P is a circular substring of T.

Algorithm: first, we can construct a new text T', and $T' = T[n-m..n-1] + [0..m-1]$ whose length is $2 \times m$. Then, we can use the KMP pattern matching algorithm to match the pattern P and text T'.

Running time: it takes $O(m)$-time to construct the new text and $O(n+m)$-time to run the KMP algorithm. As a result, the total running time is $O(n+m)$.

C-9.9

Given an efficient algorithm for deleting a string from a standard trie and analyze its running time.

Algorithm: first, we need to find the external node which is the end of this string. Then, we need to traverse up this trie to the root, if we find a node that is an external node, we should delete it. Finally, we finish and we get the trie without this string.

Running time: we assume that the length of the string we want to delete is n. It takes $O(n)$-time to get to the external node, and it also takes $O(n)$-time to get back to the root. As a result, the total running time is $O(n)$.