

15-351 / 15-650 / 02-613: Exercises 2b

1. Design an efficient in-place algorithm to convert a (4,7)-tree into a (2,3)-tree.

Note: we specify “in-place” to disallow the trivial solution of just using an in-order traversal to get a sorted list and constructing the new tree from scratch, though in practice, that would be a reasonable solution absent memory limitations.

2. A k-mismatch palindrome is a string xy where, $|x| = |y|$ and $\text{reverse}(y)$ and x are the same in all but at most k positions. Give an $O(kn)$ -time algorithm to find all the maximal k -mismatch palindromes in a string S of length n . You may assume constant time access to the function $\text{LCP}(i, j)$, the longest common prefix of two strings at positions i and j .
3. Let x be a string of length n . There are $O(n^2)$ substrings of x . Show how to count the number of distinct substrings of x in $O(n)$ time.
4. Given a set S of k strings, we want to find every string in S that is a substring of some other string in S . Assuming that the total length of all the strings is n , give an $O(n)$ -time algorithm to solve this problem.
5. Suppose you are given a string s of length n . Describe an $O(n)$ -time algorithm to find the longest string t that occurs both forwards and backwards in s . Your solution must use suffix trees or generalized suffix trees.

For example: if $s = \text{yabcxqcbaz}$, your algorithm should return $t = \text{abc}$ or $t = \text{cba}$ because both abc and its reverse cba appear in s and no longer such string exists.

6. Let S be a string of length n . Give an $O(n)$ -time algorithm to find the longest repeated substring of S such that at least two copies of the substring do not overlap in S .
7. **DNA contamination.** When sequencing DNA, often contaminant DNA (e.g. bacteria in the lab equipment, the operator’s DNA, etc.) is accidentally sequenced in addition to the target DNA. Suppose you are given a set of strings $C_1, \dots, C_K$ representing the DNA of known contaminants, where $\sum_i |C_i| = n$.

A string S of length m is sequenced. Give an algorithm with runtime $O(n + m)$ that finds the locations of all the substrings of S that occur in some C_i and that are longer than a given parameter t . (We don’t care which C_i any contaminant comes from, but we do want to know where every contaminant instance occurs in S .)

8. Assume you are using the most straightforward, basic implementation of BWT (the Burrows-Wheeler-Transform), and that anytime you need to sort, you use Merge Sort. Note that the complexity of sorting K integers might be different than the complexity of sorting K strings of length N . Typically, the complexity of a sorting algorithm is written as a function of the number of items to be sorted, and is given in terms of the number of comparisons, but assumes that the length of each item is a constant (i.e. $K \gg N$). This assumption means that each comparison is a constant-time

operation. Is this true for situations when BWT needs to sort? You should see that the aforementioned assumption does not hold (i.e. $K = N$). For a string of length N what is the worst case runtime complexity of a naive function for computing the BWT?