Lecture At the lecture we will talk about randomized algorithms. You should read CLRS section 7 and 9.2, and "Randomized algorithms" by Paul Fischer, Chapter 1 and 2 (minus 2.4 and 2.5). Read also appendix A in the notes.

Exercises

Exercises

In all the exercises it can be assumed, that the alphabet is of constant size and that the suffix tree of a string of length n over an alphabet of size $O(1)$ can be constructed in $O(n)$ time.

Construction of compressed tries and suffix trees In this exercise you don't have to replace the labels by indexes into the string.

- Construct the compressed trie for the words: "trie, tree, try, tire, car, cat, fire, free".
- Construct the suffix tree for "mississippi".

Longest common substring of k strings (Gusfield) In biological strings (DNA, RNA, or protein) the problem of finding substrings common to a large number of strings arises in many contexts. One example is that mutations that occur in DNA after two species diverge changes more rapidly those parts of the DNA or protein that are less functionally important. The parts of the DNA that are critical for the correct functioning of the molecule will be more highly conserved, because mutations in those regions are more likely to be lethal. Therefore, finding DNA or protein substrings that occur commonly in a wide range of species helps to point to regions or sub patterns that might be critical for the function or structure of the biological string. That motivates the following computational problem:

Given k strings $s_1, \ldots, s_k$, of total length n give an algorithm to find the longest substring that is a substring of all k strings. Analyze the time and space consumption of your algorithm.

Smallest k repeat (Gusfield) Given a string S of length n, give and algorithm to find the smallest substring of S occurring *exactly* k times. Analyze the time and space consumption of your algorithm.

DNA contamination (Gusfield) Various laboratory processes used to isolate, purity, clone, copy, maintain, probe, or sequence a DNA string can course unwanted DNA to become inserted into the string of interest or mixed together with a collection of strings. Often, the DNA sequences from many of the possible contaminants are known. This motivates the following computational problem:

Given a string S_1 (the newly isolated and sequenced string of DNA) and a string S_2 (the combined sources of possible contamination), find all substrings of S_2 that occur in S_1 and that are longer than some given length ℓ. These substrings are candidates for unwanted pieces of S_2 that have contaminated the desired DNA string.

Give an algorithm to solve the problem. Analyze the time and space consumption of your algorithm.

Lexicographically smallest shift In chemical databases for circular molecules, each molecule is represented by a circular string of chemical characters. To allow faster lookup and comparisons of molecules, one wants to store each circular string by a canonical linear string. A natural choice for a canonical linear strings the one that is lexicographically smallest. That gives the following computational problem.

Assume we are given a string $T = x_1 \dots x_n$ of length n. A *shift* of T by s, $0 \leq s < n$, is the string $T^s = x_{s+1}x_{s+2} \dots x_n x_1 x_2 \dots x_s$. In this problem we want to find the *lexicographically smallest shift*, i.e. the shift s where T^s is lexicographically smallest among $T^0, \dots, T^{n-1}$. Eg. $T^2 = T^7 =$ `aababaabab` are the lexicographically smallest shifts of the string

$$T = \texttt{abaababaab}$$

Q1 State all s where T^s is a lexicographically smallest shift of the string

$$T = \texttt{bcabaabcabaabcabaa}$$

Q2 Describe an algorithm that given a string T of length n over an alphabet of size $O(1)$ computes all s where T^s is a lexicographically smallest shift of T. State the algorithms running time.

Mandatory assignment (from the exam E15)

Question 1 Draw the suffix tree for the string `pinepie$` (don't replace the labels by indexes into the string, just write the labels in the vertices):

Question 2 You are putting together a set of Christmas songs that will be handed out at the party. The Dean has declared that every song must contain the sentence "Merry␣Christmas␣Dear␣Dean", where "␣" denotes a blank space. E.g. the song:

```
We␣wish␣you␣a␣Merry␣Christmas␣
We␣wish␣you␣a␣Merry␣Christmas␣
We␣wish␣you␣a␣Merry␣Christmas␣
Dear␣Dean␣
Dear␣Dean
```

contains one occurrence of of the sentence "Merry␣Christmas␣Dear␣Dean" (line breaks are disregarded).

Formally, you are given a set S of songs $S_1, \dots, S_k$ and a sentence P. Song S_i contains n_i characters and P contains m characters. Let $n = \sum_{i=1}^{k} n_i$ denote the total number of characters in the songs. All the strings are over an alphabet of size $O(1)$. Describe an algorithm that returns all the songs that contain P. Analyze the asymptotic running time of your algorithm. Remember to argue that your algorithm is correct.