CS240E W24 **Tutorial 10** Mar. 25

String matching

Boyer-Moore. Apply the Boyer-Moore algorithm to the following pattern and text. Show

1. with only the bad-character heuristic,
2. with the good-suffix heuristic.

T :	d	a	y	s	a	y	m	a	y	a	a	a	y	b	a	y	l	a	y	k	a	y	r	a	y	j	a	y
P :	d	a	y	d	a	y	h	a	y	a	y	a	y															
(a)																												
(b)																												

Boyer-Moore.

Boyer-Moore can be modified in many ways. For each of the modifications listed below, state whether the modification is valid, i.e. the modified Boyer-Moore will always successfully find the first occurrence of P in T, if P appears in T, or return `FAIL` if P is not in T.

If the answer is "Yes", provide a brief explanation of why it is still valid. If the answer is "No", demonstrate a counter-example, i.e. trace the algorithm on specific P and T of your choice where the result is incorrect.

(a) Using a first-occurrence function (denoting the index of the first occurrence of the argument character) instead of a last-occurrence function.

(b) When checking a pattern shift, compare characters from the start of the pattern and move forward, instead of scanning backwards from the end of the pattern.

(c) Use the last-occurrence function for $P[0..m-2]$, i.e. P with its last character removed, instead of the last-occurrence function for P.

Most common substring. Let s be a string of length n and let $\mathcal{T}_s$ denote the corresponding suffix tree. For an integer parameter $1 \leq l \leq n$, give a $O(n)$ time algorithm that finds a most commonly occurring substring of length l in s.

Pattern matching.

Consider the pattern $P = 0110101$ and the text T listed in the following table.

0	0	1	1	1	1	0	0	1	1	1

(a) Indicate all the checks that were done by the brute-force method.

(b) Consider the Karp-Rabin fingerprint that simply counts the number of 1s in the bit-string. Is this a rolling hash-function? And using these fingerprints, how many checks were done during Karp-Rabin pattern matching?

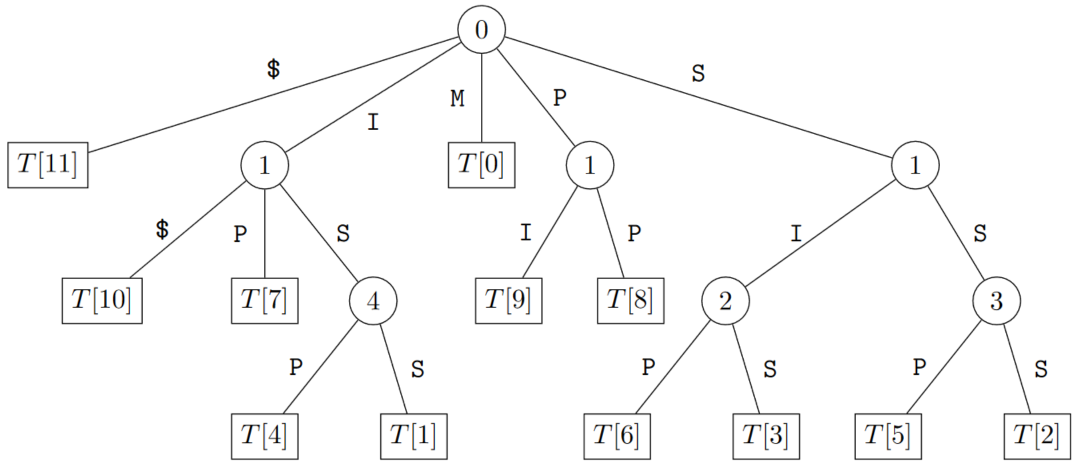
(c) Compute the KMP failure-function for P.

(d) Show the KMP automaton for P.

(e) Consider now the pattern $P =$ `fiddledidi`. Show the Boyer-Moore last-occurrence array.

Suffix trees. Jason discovered a secret message in the form of a suffix tree S, indicating the location of a hidden treasure.

1. Design an algorithm that recovers the original text T from its corresponding suffix tree S. The algorithm should run in $O(n)$ time while using $O(n)$ auxiliary space.
2. Determine the original text for the following suffix tree:



Consecutive strings in a trie.

Given an uncompressed trie T that stores a list of binary strings, design an algorithm *consecutive*(b_1, b_2) that takes two binary strings in T as input, and outputs true if the strings are consecutive in pre-order traversal of the trie, and outputs false otherwise.

Assume that branches are ordered as \$, 0, 1. The runtime should be bounded by $O(|b_1| + |b_2|)$.

For example, suppose T stores $\{000, 01, 0110, 101, 11\}$. Then:

- *consecutive*$(0110, 101)$ returns true
- *consecutive*$(01, 000)$ returns true
- *consecutive*$(11, 000)$ returns false