

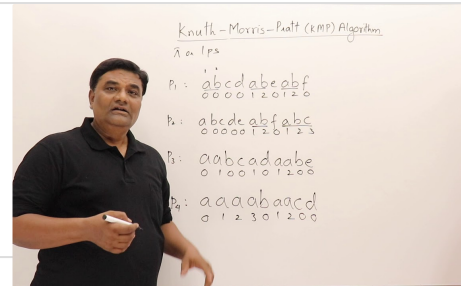
Knuth–Morris–Pratt algorithm

9.1 Knuth-Morris-Pratt KMP String Matching Algorithm

In P3, b is also matching, lps should be 0 1 0 0 1 0 1 2 3 0

Naive Algorithm

https://www.youtube.com/watch?v=V5-7GzOfADQ&ab_channel=AbdulBari



Visualizing String Matching Algorithms

<http://whocouldthat.be/visualizing-string-matching/>

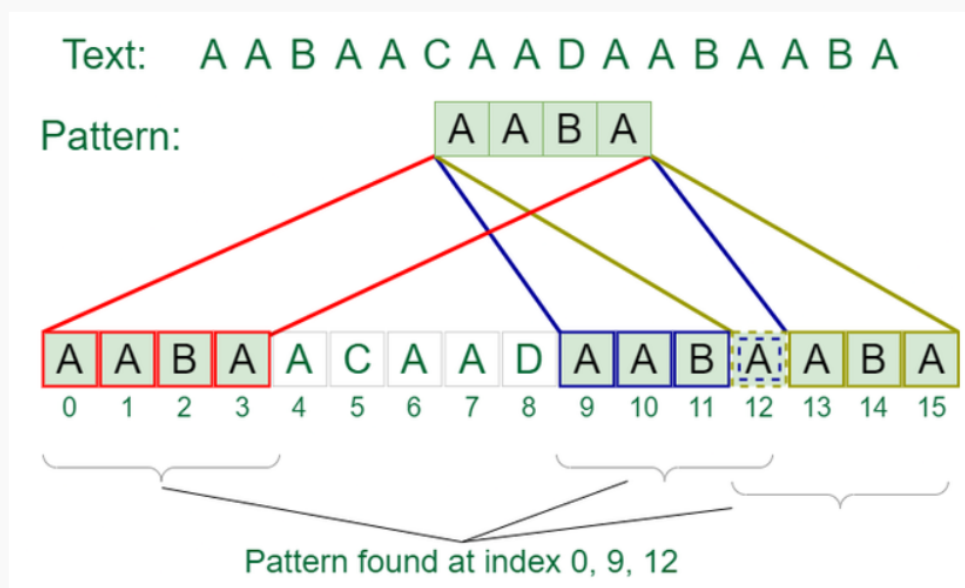
Input: txt[] = "THIS IS A TEST TEXT", pat[] = "TEST"

Output: Pattern found at index 10

Input: txt[] = "AABAACAADAABAABA"

pat[] = "AABA"

Output: Pattern found at index 0, Pattern found at index 9, Pattern found at index 12



Arrivals of pattern in the text

→ KMP Algorithm (used for pattern matching)

ex: (a) pattern: - a b c d a b c
 1 2 3 4 5 6 7

sol: prefix: a, ab, abc, abcd. . . } This is how KMP algorithm solves the problem
suffix: c, bc, abc, dabc. . .

(c) String: a b a b a b d
 0 1 2 3 4 5 6

sol: pattern:

a	b	a	b	d
0	1	2	2	0

→ KMP is a string searching Algorithm, that efficiently finds occurrences of a pattern within a text.

* The key idea behind KMP is to avoid unnecessary comparisons in the text.

* Time Complexity of KMP is $O(n+m)$.

i.e, n = length of the text
 m = length of the pattern.

<http://whocouldthat.be/visualizing-string-matching/>