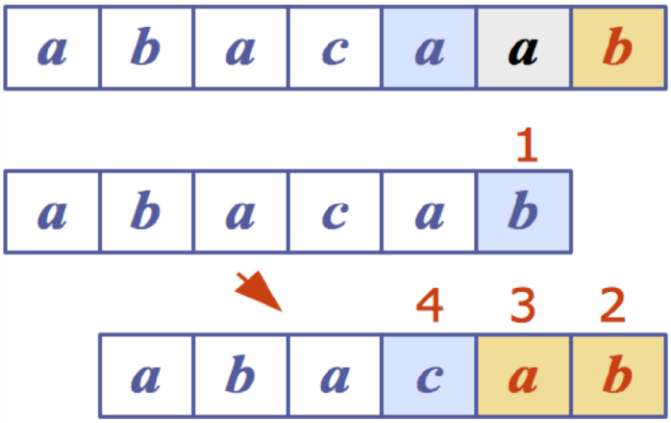
Pattern Matching Algorithms



DOCUMENTATION REPORT

PROJECT 3

ITCS 6114 – Algorithm and Data Structure

DEPARTMENT OF COMPUTER SCIENCE

SUBMITTED TO
Dewan T. Ahmed, Ph.D.

SUBMITTED BY:

Shreeya Gupta **801136226**

Sai Kumar Thallada **801154715**

UNC CHARLOTTE
College of Computing and Informatics

1. Introduction to Pattern Matching Algorithms

The Pattern Searching algorithms are sometimes also referred to as String Searching Algorithms and are considered as a part of the String algorithms. These algorithms are useful in the case of searching a string within another string. Following is the list of pattern matching techniques that we have implemented:

1. Brute Force
2. Boyer-Moore-Horspool Algorithm
3. Knuth-Morris-Pratt Algorithm

I. Brute Force

The Brute Force algorithm compares the pattern to the text, one character at a time, until unmatching characters are found. The algorithm can be designed to stop on either the first occurrence of the pattern, or upon reaching the end of the text.

Data Structure Used:

An array is used to store the return values in Brute Force Algorithm.

Time complexity:

In worst case, Brute Force Algorithm runs in O(nm) time where n is the length of text and m is the length of pattern.

II. Boyer-Moore-Horspool Algorithm

Boyer-Moore-Horspool is an algorithm for finding substrings into strings. This algorithm compares each characters of substring to find a word or the same characters into the string. When characters do not match, the search jumps to the next matching position in the pattern by the value indicated in the Bad Match Table.

The Bad Match Table indicates how many jumps it should move from the current position to the next.

Data Structure Used:

Horspool Algorithm is using an array data structure to store the position and numbers of comparisons used.

Time Complexity:

Preprocessing time of Horspool Algorithm is O(S+m). The worst case runtime of this algorithm is O(nm). The search time and best case runtime of this algorithm is O(n/m) where n is the length of the text and m is the length of the pattern.

III. Knuth-Morris-Pratt Algorithm

It is a clever algorithm that avoids backup. It traverses the pattern from left to right. It never compares the characters that are already been compared. It uses one failure table that can be built by comparing the length of prefix and suffix of the pattern.

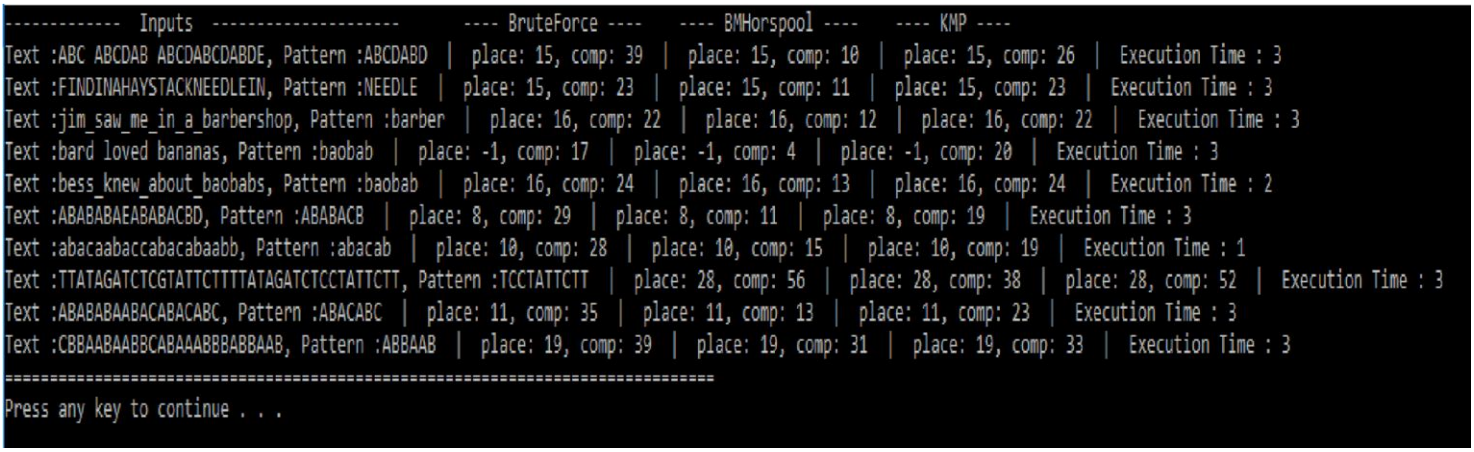
Data Structure Used:

An array data structure is used for storing the failure table information and also the position and comparisons of the pattern.

Time complexity:

The worst case runtime of KMP algorithm is O(n).

Sample Inputs and Outputs:



No of Comparisons:

Text	Pattern	Brute Force	BM Horspool	KMP
ABC ABCDAB ABCDABCDABDE	ABCDABD	39	10	26
FINDINAHAYSTACKNEEDLEIN	NEEDLE	23	11	23
jim_saw_me_in_a_barbershop	barber	22	12	22
bard loved bananas	baobab	17	4	20
bess_knew_about_baobabs	baobab	24	13	24
ABABABAEABABACBD	ABABACB	29	11	19
Abacaabaccabacabaabb	abacab	28	15	19
TTATAGATCTCGTATTCTTTTATAGATCTCCTATTCTT	TCCTATTCTT	56	38	52
ABABABAABACABACABC	ABACABC	35	13	23
CBBAABAABBCABAAABBBABBAAB	ABBAAB	39	31	33