**REPORT**

**Project 3: Pattern Matching Algorithms (1. Brute-force, 2. BM Horspool and 3. KMP)**

**Submitted By**

Jawad Chowdhury, (ID# 801135477)

**Overview of Algorithms (short description)**

Here, I basically used 3 different algorithms for the pattern matching task. The algorithms are as:

1. Brute-force algorithm.
2. Boyer Moore Horspool algorithm and
3. Knuth Morris Pratt (KMP) algorithm.

**Algorithm 1 (Brute-force):**

This algorithm basically takes a text **T** and a patter **P** and for each element of **T**, it starts an execution of matching with pattern **P**. If it matches then the next item of **T** is being tried to match with the next item of **P** and so on. If does not get matched, it then starts the whole matching of patter **P**, starting from the next item of text **T**.

This algorithm does not use any pre-processing to the text **T** or pattern **P**.

**Algorithm 2 (Boyer Moore Horspool):**

This algorithm use searching by some pre-processing of the pattern.

This algorithm need to form a **Shift Table** to determine how much the pattern needs to shifted if a mismatch occurs.

The algorithm executes the searching of the pattern **P** in the text **T**, by following an order from **Right to Left**.

**Algorithm 3 (KMP – Knuth Morris Pratt):**

This algorithm use searching by also some pre-processing of the pattern.

This algorithm need to form a **Failure Function** to determine how much the pattern needs to shifted if a mismatch occurs and also what will be the index of the pattern to match on next.

The algorithm executes the searching of the pattern **P** in the text **T**, by following an order from **Left to Right**.

**Data Structure Used**

For implementing these 3 algorithms, I used the basic data structures such as list and dictionary of the python basic data types.

1. **Brute-force**: This algorithm does not require any pre-processing, so there was no pre-processed structure to build. I just implemented the algorithm for the execution.
2. **Boyer Moore Horspool**: This algorithm does some pre-processing. So I need to form the **Shift Table** here. The basic idea of my implementation was to form a **dictionary** that contains the required shifting value for all the **unique** character in the pattern and a **default** value for all other characters. Now, I have used a function to return the value from the dictionary if there is a key with that character that I am searching for otherwise it would return the **default** value which basically the **length of the pattern.**
3. **KMP:** This algorithm also requires some pre-processing. I have implemented a **Failure Function** here which basically returns python basic list data type, with containing all the **index value of the pattern** that should be used on the **next iteration**, and each of these values are in the index of the returned list where the **index** is related to the **number of matched character before the mismatch**.

**Runtime of Code**

Here, I am using –

**Brute-force**

Runtime (in worst case):

**Boyer Moore Horspool:**

Runtime for Shift Table: **;**

Runtime of Algorithm in worst case:

But in best case it is:

**KMP:**

Runtime of Failure Function:

Runtime of Algorithm:

**Sample Input & Output**

I have used 10 sample input and output for experiments and they are given below.

**EXPERIMENT NO ===> 1**

PATTERN ===> barber

TEXT ===> jim saw me in a barbershop

Brute-force (matched at) index: 16.

BM Horspool (matched at) index: 16.

KMP (matched at) index: 16.

**EXPERIMENT NO ===> 2**

PATTERN ===> fox

TEXT ===> the quick fox jumps over the lazy dog

Brute-force (matched at) index: 10.

BM Horspool (matched at) index: 10.

KMP (matched at) index: 10.

**EXPERIMENT NO ===> 3**

PATTERN ===> abcdabd

TEXT ===> abc abcdab abcdabcdabde

Brute-force (matched at) index: 15.

BM Horspool (matched at) index: 15.

KMP (matched at) index: 15.

**EXPERIMENT NO ===> 4**

PATTERN ===> needle

TEXT ===> findinahaystackneedleina

Brute-force (matched at) index: 15.

BM Horspool (matched at) index: 15.

KMP (matched at) index: 15.

**EXPERIMENT NO ===> 5**

PATTERN ===> baobab

TEXT ===> bird loved bananas

Brute-force (matched at) index: -1.

BM Horspool (matched at) index: -1.

KMP (matched at) index: -1.

**EXPERIMENT NO ===> 6**

PATTERN ===> baobab

TEXT ===> bess knew about baobabs

Brute-force (matched at) index: 16.

BM Horspool (matched at) index: 16.

KMP (matched at) index: 16.

**EXPERIMENT NO ===> 7**

PATTERN ===> country

TEXT ===> i am in a new country

Brute-force (matched at) index: 14.

BM Horspool (matched at) index: 14.

KMP (matched at) index: 14.

**EXPERIMENT NO ===> 8**

PATTERN ===> student

TEXT ===> this is algorithm class

Brute-force (matched at) index: -1.

BM Horspool (matched at) index: -1.

KMP (matched at) index: -1.

**EXPERIMENT NO ===> 9**

PATTERN ===> present

TEXT ===> this pattern is present here

Brute-force (matched at) index: 16.

BM Horspool (matched at) index: 16.

KMP (matched at) index: 16.

**EXPERIMENT NO ===> 10**

PATTERN ===> absent

TEXT ===> this pattern is not present here

Brute-force (matched at) index: -1.

BM Horspool (matched at) index: -1.

KMP (matched at) index: -1.

**Table for Number of Comparison**

The table with the number of comparison need values is given below for each experiment and each of these algorithm.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Alg /Exp No | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| Brute-force | 22 | 13 | 39 | 23 | 16 | 24 | 21 | 19 | 24 | 28 |
| BM Horspool | 12 | 7 | 10 | 11 | 4 | 13 | 10 | 3 | 10 | 11 |
| KMP | 22 | 13 | 26 | 23 | 20 | 24 | 21 | 26 | 24 | 33 |

**Instruction to Run Program**

The running procedure of my implementation is pretty much simple.

I have only one python script named as **project\_3.py.**

In this script, I basically kept a list of tuple with as (pattern, text). So to try with the different pattern and text combination, just this list needs to be updated.

After updating this list with the pattern and text and then running the script will do the work.