

CMP3005-Analysis of Algorithms

Fall 2017

Project

Due Date: December 18, 2017

Aim: Comparing the Brute-Force and Horspool's string matching algorithms.

Submission Instructions: Submit the project through itslearning!

Only itslearning will be accepted. Other ways of submissions such as e-mail, hard-copy or pigeon will NOT be graded. For more details, see the submission instructions below.

1) String-matching Algorithms

a) **Brute-Force string matching** algorithm is given below:

ALGORITHM *BruteForceStringMatch*($T[0..n - 1]$, $P[0..m - 1]$)

//Implements brute-force string matching

//Input: An array $T[0..n - 1]$ of n characters representing a text and

// an array $P[0..m - 1]$ of m characters representing a pattern

//Output: The index of the first character in the text that starts a

// matching substring or -1 if the search is unsuccessful

for $i \leftarrow 0$ **to** $n - m$ **do**

$j \leftarrow 0$

while $j < m$ **and** $P[j] = T[i + j]$ **do**

$j \leftarrow j + 1$

if $j = m$ **return** i

return -1

b) **Horspool's string matching algorithm** is given below:

ALGORITHM *ShiftTable*($P[0..m-1]$)

```
//Fills the shift table used by Horspool's and Boyer-Moore algorithms
//Input: Pattern  $P[0..m-1]$  and an alphabet of possible characters
//Output:  $Table[0..size-1]$  indexed by the alphabet's characters and
//       filled with shift sizes computed by formula (7.1)
for  $i \leftarrow 0$  to  $size-1$  do  $Table[i] \leftarrow m$ 
for  $j \leftarrow 0$  to  $m-2$  do  $Table[P[j]] \leftarrow m-1-j$ 
return  $Table$ 
```

ALGORITHM *HorspoolMatching*($P[0..m-1]$, $T[0..n-1]$)

```
//Implements Horspool's algorithm for string matching
//Input: Pattern  $P[0..m-1]$  and text  $T[0..n-1]$ 
//Output: The index of the left end of the first matching substring
//       or -1 if there are no matches
 $ShiftTable(P[0..m-1])$  //generate  $Table$  of shifts
 $i \leftarrow m-1$  //position of the pattern's right end
while  $i \leq n-1$  do
     $k \leftarrow 0$  //number of matched characters
    while  $k \leq m-1$  and  $P[m-1-k] = T[i-k]$  do
         $k \leftarrow k+1$ 
    if  $k = m$ 
        return  $i-m+1$ 
    else  $i \leftarrow i + Table[T[i]]$ 
return -1
```

2) Implementation Instructions

a) Input

Text: Script of “The Truman Show” which is attached to this message (the_truman_show_script.txt). This input is referred to as ‘T’ in the above-given algorithms.

Pattern: A comparably shorter string entered by the user. This input is referred to as ‘P’ in the above-given algorithms.

b) Execution

When the program is executed, the following menu will be displayed:

i)



```
C:\Windows\system32\cmd.exe
(1) Brute-Force String Matching
(2) Horspool's String Matching
(3) Compare brute-force and Horspool's string matching algorithms
(0) Exit
```

ii) When option 1 (Brute-Force String Matching) is chosen by the user:

- a. the program will ask the user to enter a “Pattern”, P.
- b. The pattern will be searched in the input “Text”, T, with the Brute-force algorithm. The input text is the script of “The Truman Show” kept at “the_truman_show_script.txt”.
- c. The program will finally display how many times the basic operation of the algorithm has been executed and total number of seconds passed to find the pattern.

iii) When option 2 (Horspool’s String Matching) is chosen by the user:

- a. The program will ask the user to enter a “Pattern”, P.
- b. The pattern will be searched in the input “Text”, T, with the Horspool’s algorithm. The input text is the script of “The Truman Show” kept at “the_truman_show_script.txt”.
- c. The program will finally display how many times the basic operation of the algorithm has been executed and total number of seconds passed to find the pattern.

iv) When option 3 (Compare brute-force and Horspool's string matching algorithms) is chosen by the user:

- a. The program will ask the user to enter a “Pattern”, P.
- b. The pattern will be searched in the input “Text”, T, with both Brute-Force and Horspool’s algorithms. The input text is the script of “The Truman Show” kept at “the_truman_show_script.txt”.
- c. The program will finally display how many times the basic operation of the algorithm has been executed and total number of seconds passed to find the pattern for both of the algorithms.

v) When option 0 (Exit) is chosen by the user, the program will be immediately terminated

3) Submission Instructions

- Submit the project through itslearning in a zip/tar file. The file name should be STUDENT_NUMBER.zip.
- Implement the program in C++.
- The project can be done individually or as a group of 2 students.
- The group projects will be submitted for once, i.e. not by both of the students, and the file name in this case should be STUDENT_NUMBER_and_STUDENT_NUMBER.zip.
- Write a header comment in the source code file, which includes your main function. You are supposed to list the following information in the header comment.

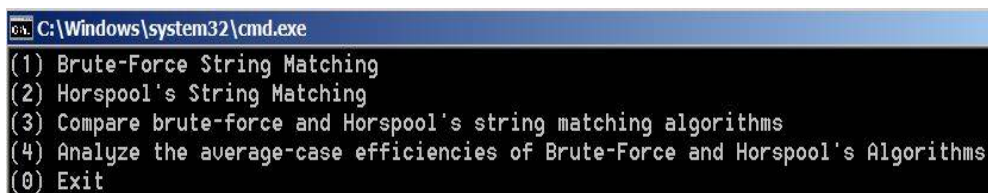
```
/*  
Student Name: Jim Carrey and Jeff Daniels  
Student ID: 1010101010  
Operating System: {Windows[Version] / Linux[Version] / MacOSX}  
Compile Status: {Compiling/Not Compiling} (Is your program successfully compiled?)  
Execution Status: {Working/Not Working} (Is it working?)  
Bonus Part: {Implemented/Not implemented} (Did you implement the bonus part?)  
Comments: Please write your additional comments, if any, about your source code.  
*/
```

BONUS PART

This part will be evaluated over an extra 40 points.

Implementation Instructions for the Bonus Part

- 1) The menu will include an additional option (4) as shown below:



```
C:\Windows\system32\cmd.exe  
(1) Brute-Force String Matching  
(2) Horspool's String Matching  
(3) Compare brute-force and Horspool's string matching algorithms  
(4) Analyze the average-case efficiencies of Brute-Force and Horspool's Algorithms  
(0) Exit
```

- 2) When option 4 (Analyze the average-case efficiencies of Brute-Force and Horspool's Algorithms) is chosen by the user, the below steps will be executed:
 - a. The program will generate 100 random patterns consisting of small-case or upper-case letters, i.e. a-z and A-Z, digits (0-9), exclamation mark ('!'), question mark ('?'), dot ('.'), or comma (','). The length of each pattern should be between 3 and 7.
 - b. For each of the 100 random patterns generated in the above-case, execute both Brute-force and Horspool's algorithms.

- c. Display how many times the basic operations of the algorithms have been performed in average which will give the experimental average-case efficiency of these algorithms.

Cheating Policy

Cheating is strictly prohibited. It must be your own work. Do not use each other's source code. In such case all the cheaters, including the original project, **will be penalized heavily**.

Important: Itslearning has a cheating program which detects very similar source codes automatically.