Program 3 on Horspools String Matching

CSC 2400 Design of Algorithms

# Problem Description

The **String Matching Problem** is the problem of finding an occurrence or occurences of a pattern string within another string or body of text. There are many different algorithms for this type of searching. This problem is also called exact string matching, string searching, or text searching.

In the problem, you have a string that you will be searching in, called the **text**. The number of characters in the text is represented by the letter ‘**n**’. Then, the string that you are finding in the text is called the **pattern**. The number of characters in the pattern is represented by the letter ‘**m**’. The pattern must be smaller than, or equal size to the text (m <= n).

# hORSPOOLS String Matching Algorithm

**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 Bad Match Table of shifts**

i **<-** m**-**1 **//position of the pattern’s right end**

**while** i **<=** n**-**1 **do**

k <- 0 **//number of matched characters**

**while** k **<=** m-1 **and** P[m**-**1**-**k] **=** T[i**-**k] **do**

k **<-** k**+**1

**end while**

**if** k **=** m **//found the pattern in the text**

**return** i**-**m**+**1

**else**

i **<-** i **+** Table[T[i]]

**end while**

**return** -1

# Shift Table Algorithm

**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 (size is 26)**

**----------------------------------------------------------------------------------------**

**for** i<-0 **to** size-1 **do //initialize each element of shift table to pattern length**

Table[i] **<-** m

**end for**

**//for each character in pattern, set shift table character that matches to pattern length – index value in pattern - 1**

**for** j<-0 **to** m-2 **do**

Table[P[j]] **<-** m-1-j

**end for**

**return** Table

# Program Specifications

Write a program in C++ that

* asks the user for the text and the pattern (changing their string to uppercase letters),
* validates that both are of adequate size for the problem scope, and
* implements the Horspool’s string matching algorithm.

## Output

1. Print the bad match table.
2. Print every time the pattern shifts and by how much it shifted.
3. Print whether the pattern was found in the text, or not.
4. If the pattern was found, print the index value in the text where it was found.
5. Print the total number of comparisons completed.

Refer to the sample output provided below to answer questions you may have about how your input and output must look. Your output text does not have to be identical, but must be similar to the output shown in this document.

## Helpful Hints

* For your shift table, you should create an array of 26 characters where each element represents each uppercase letter of the alphabet.
* Each character is represented by a number and is stored as a number in memory. For example, the letter ‘A’ is 65, ‘B’ is 66, ‘C’ is 67, and so on. Refer to the ASCII character chart to see the numbers: <https://www.asciitable.com/>. When you are setting up the shift table, you will do the calculation for m-index-1 for each character in the pattern. So to determine where in the array to put this calculation, you take the letter in the pattern and then subtract 65 and that is the index value of the letter in the array you create.

# SAMPLE OUTPUT

## Sample One

User input is highlighted in **yellow**.

**Enter a string of characters, called the text.**

**TEXT: AAAAAAA**

**Enter a search string to find in the text, called the pattern.**

**Your pattern must be less than 7 characters long.**

**PATTERN: AAAAB**

**BAD MATCH TABLE:**

**A B C D E F G H I J K L M N O P Q R S T U V W X Y Z**

**1 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5**

**Shifting pattern by 1**

**Shifting pattern by 1**

**Shifting pattern by 1**

**Sorry, the pattern was not found in the text.**

**TOTAL COMPARISONS: 3**

## SAMPLE TWO

User input is highlighted in **yellow**.

**Enter a string of characters, called the text.**

**TEXT: APRIL HAS FUJI APPLE WATER NOT APPLE JUICE**

**Enter a search string to find in the text, called the pattern.**

**Your pattern must be less than 42 characters long.**

**PATTERN: APPLE**

**BAD MATCH TABLE:**

**A B C D E F G H I J K L M N O P Q R S T U V W X Y Z**

**4 5 5 5 5 5 5 5 5 5 5 1 5 5 5 2 5 5 5 5 5 5 5 5 5 5 5**

**Shifting pattern by 1**

**Shifting pattern by 5**

**Shifting pattern by 5**

**Shifting pattern by 4**

**LETTER MATCHED-E, NUMBER CHARS MATCHED-1**

**LETTER MATCHED-L, NUMBER CHARS MATCHED-2**

**LETTER MATCHED-P, NUMBER CHARS MATCHED-3**

**LETTER MATCHED-P, NUMBER CHARS MATCHED-4**

**LETTER MATCHED-A, NUMBER CHARS MATCHED-5**

**Pattern was found at index 15!**

**TOTAL COMPARISONS: 9**

## SAMPLE Three

User input is highlighted in **yellow**.

**Enter a string of characters, called the text.**

**TEXT: INTHETIKITIKQTIKITIKIROOM**

**Enter a search string to find in the text, called the pattern.**

**Your pattern must be less than 25 characters long.**

**PATTERN: TIKITIKI**

**BAD MATCH TABLE:**

**A B C D E F G H I J K L M N O P Q R S T U V W X Y Z**

**8 8 8 8 8 8 8 8 2 8 1 8 8 8 8 8 8 8 8 3 8 8 8 8 8 8 8**

**Shifting pattern by 1**

**LETTER MATCHED-I, NUMBER CHARS MATCHED-1**

**LETTER MATCHED-K, NUMBER CHARS MATCHED-2**

**LETTER MATCHED-I, NUMBER CHARS MATCHED-3**

**LETTER MATCHED-T, NUMBER CHARS MATCHED-4**

**Shifting pattern by 2**

**LETTER MATCHED-I, NUMBER CHARS MATCHED-1**

**Shifting pattern by 2**

**Shifting pattern by 8**

**LETTER MATCHED-I, NUMBER CHARS MATCHED-1**

**LETTER MATCHED-K, NUMBER CHARS MATCHED-2**

**LETTER MATCHED-I, NUMBER CHARS MATCHED-3**

**LETTER MATCHED-T, NUMBER CHARS MATCHED-4**

**LETTER MATCHED-I, NUMBER CHARS MATCHED-5**

**LETTER MATCHED-K, NUMBER CHARS MATCHED-6**

**LETTER MATCHED-I, NUMBER CHARS MATCHED-7**

**LETTER MATCHED-T, NUMBER CHARS MATCHED-8**

**Pattern was found at index 13!**

**TOTAL COMPARISONS: 17**

# Submission

**Zip** your source file in one zip/compressed folder named your **username\_prog2** (e.g. acrockett\_prog1).

You will upload your submission to ilearn in an assignment folder named **Program 2**.

# Grading

Look at grading Rubric in Ilearn!