Program 2 on Brute Force String Matching

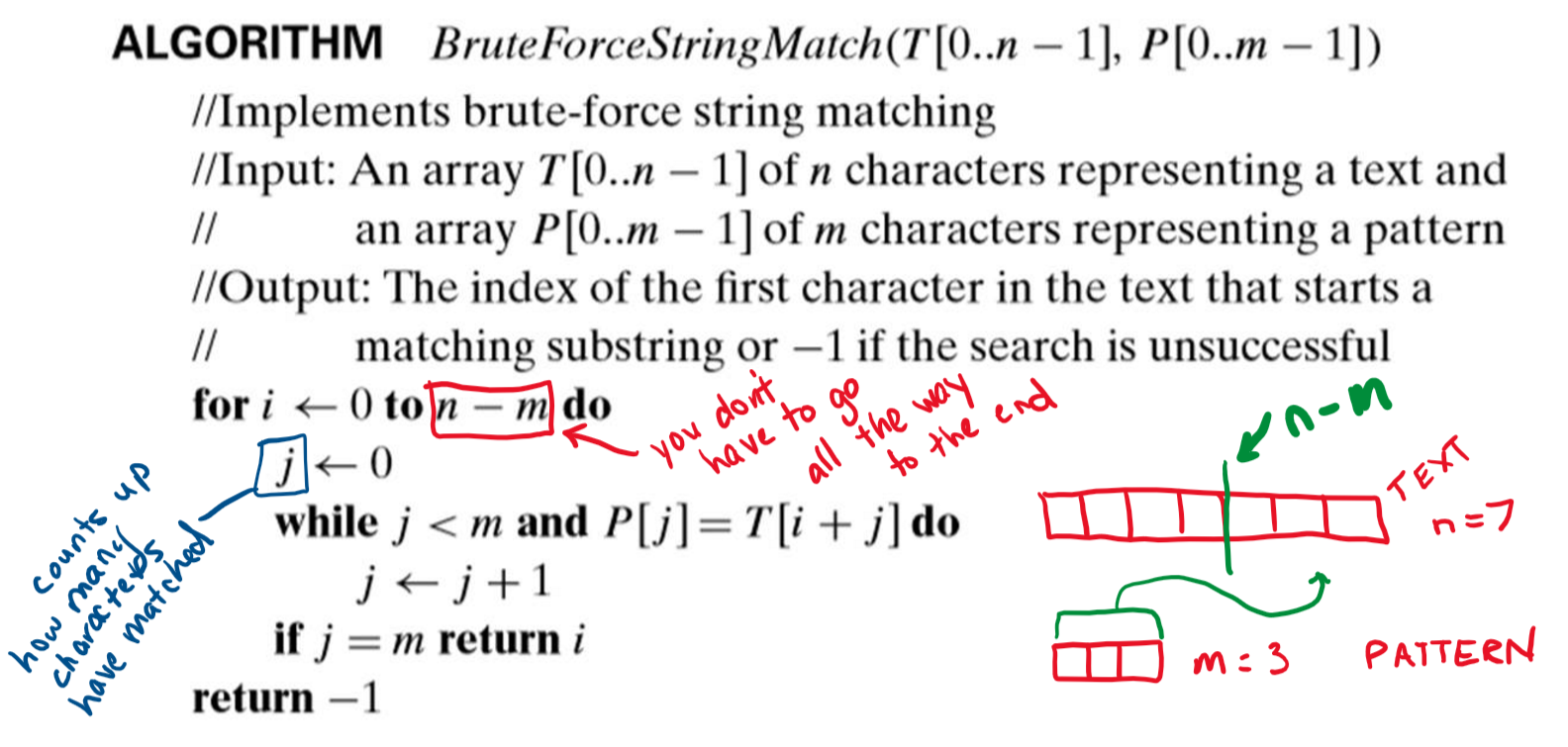
CSC 2400 Design of Algorithms

# 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).

# Brute Force String Matching Algorithm



# Program Specifications

Write a program in C++ that asks the user for the text and the pattern, validates that both are of adequate size for the problem scope, and then implements the brute force string matching algorithm. You should print whether the pattern was found in the text, or not. If found, you should print the index value in the text where it was found. Your program must also 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.

## 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**

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

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

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

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

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

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

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

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

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

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

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

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

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

**TOTAL COMPARISONS: 15**

### 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**

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

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

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

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

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

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

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

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

**Pattern was found at index 15!**

**TOTAL COMPARISONS: 23**

### SAMPLE Three

User input is highlighted in **yellow**.

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

**TEXT: SEVEN**

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

**Your pattern must be less than 5 characters long.**

**PATTERN: SEVEN IS MY FAVORITE**

**Oops! Your pattern must be less than 5 characters long.**

**PATTERN: E**

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

**Pattern was found at index 1!**

**TOTAL COMPARISONS: 2**

### SAMPLE FOUR

User input is highlighted in **yellow**.

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

**TEXT:**

**Oops! You must enter in a text greater than zero characters.**

**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**

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

**Pattern was found at index 13!**

**TOTAL COMPARISONS: 32**

# 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!