STRING MATCHING ASSIGNMENT

* The naive String-matching algorithm is a very basic and straightforward method by considering the other string pattern matching algorithm.
* It is checked, if a string is given and a pattern is given then the problem is to find out whether this pattern is existing inside the string or not.

**Principle of the Naïve Algorithm**

Size of the string = n

Size of the pattern = m

**Step 01** => start on the first letter of the string and compare the first letter of the string with the first letter of the pattern.

**Step 02** => if the first letter, does not matching then shift the pattern to the right side(->) by one step.

**Step 03** => if the first letter does match with the pattern, then compare the 2nd letter of the text with the 2nd letter of the pattern.

**Step 04** => So likewise this process keeps going until (n-m) times.

So, the time complexity of the naïve pattern is: **O(nm).**

**Algorithm of the naïve patterns matching**

Begin

Create function naiveAlgo(text,pattern)

SET Size of the string  = n

SET Size of the pattern = m

SET patternMachingTime = 0

For i in range n-m:

    For j in range m:

        If pattern[j] = text[j] *for* both capital and simple letter then

            i++

            patternMachingTime++

        end If

        Else If pattern[j] != text[j] *for* both capital and simple letter then

            patternMachingTime = 0

*break*;

        end Else If

    end For

    If patternMachingTime = m then

*break*;

    end If

End For

End function

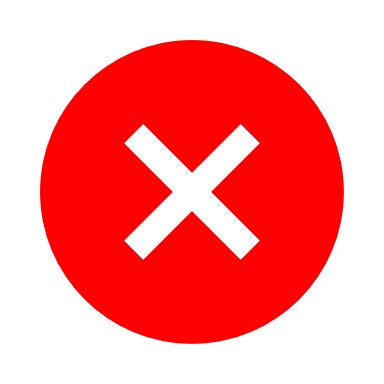
End

**Example:**

String = a b c d e f g h

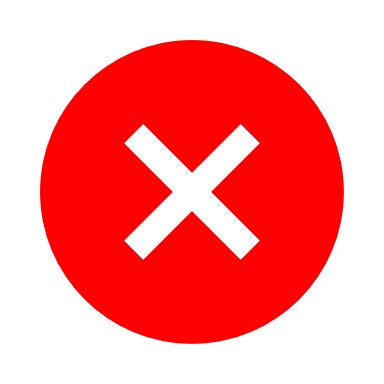
Pattern = e f g

**01. e != a, so shift pattern by one step**

a b c d e f g

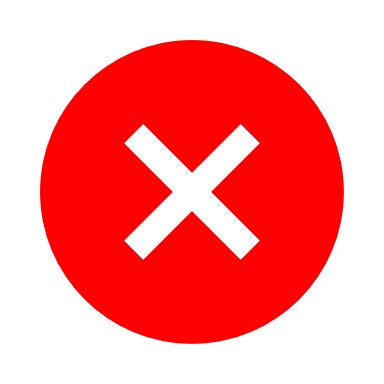
e f g

**02. e != b, so shift pattern by one step.**

a b c d e f g

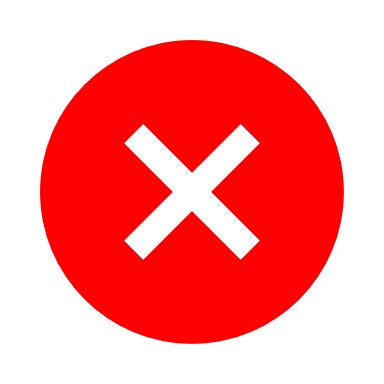
e f g

**03. e != c, so shift pattern by one step.**

a b c d e f g

e f g

**04. e != d, so shift pattern by one step.**

a b c d e f g

e f g

**05. e == e, so then check the next character of both string and pattern.**

a b c d e f g

e f g

**05. f == f, so then also matching.**

**then check the next character of both string and pattern.**

a b c d e f g

e f g

**06. g == g, so then also matching.**

**then check the next character of both string and pattern.**

a b c d e f g

e f g

* So, the pattern is found inside the text at the index of the 4th to 6th.

**\*\*\*\*\***