GitHub

STRING MATCHING ASSIGNMENT

Justification

The naive String-matching algorithm is a very basic and straightforward method by considering the other string pattern matching algorithm. So using this algorithm we can easily implement the program because we haven't more checking conditions like another algorithm when implementing. Because we always check the character one by one. Due to that reason, mostly reduce the mistakes when pattern matching with the string(compare to other algorithms).

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.

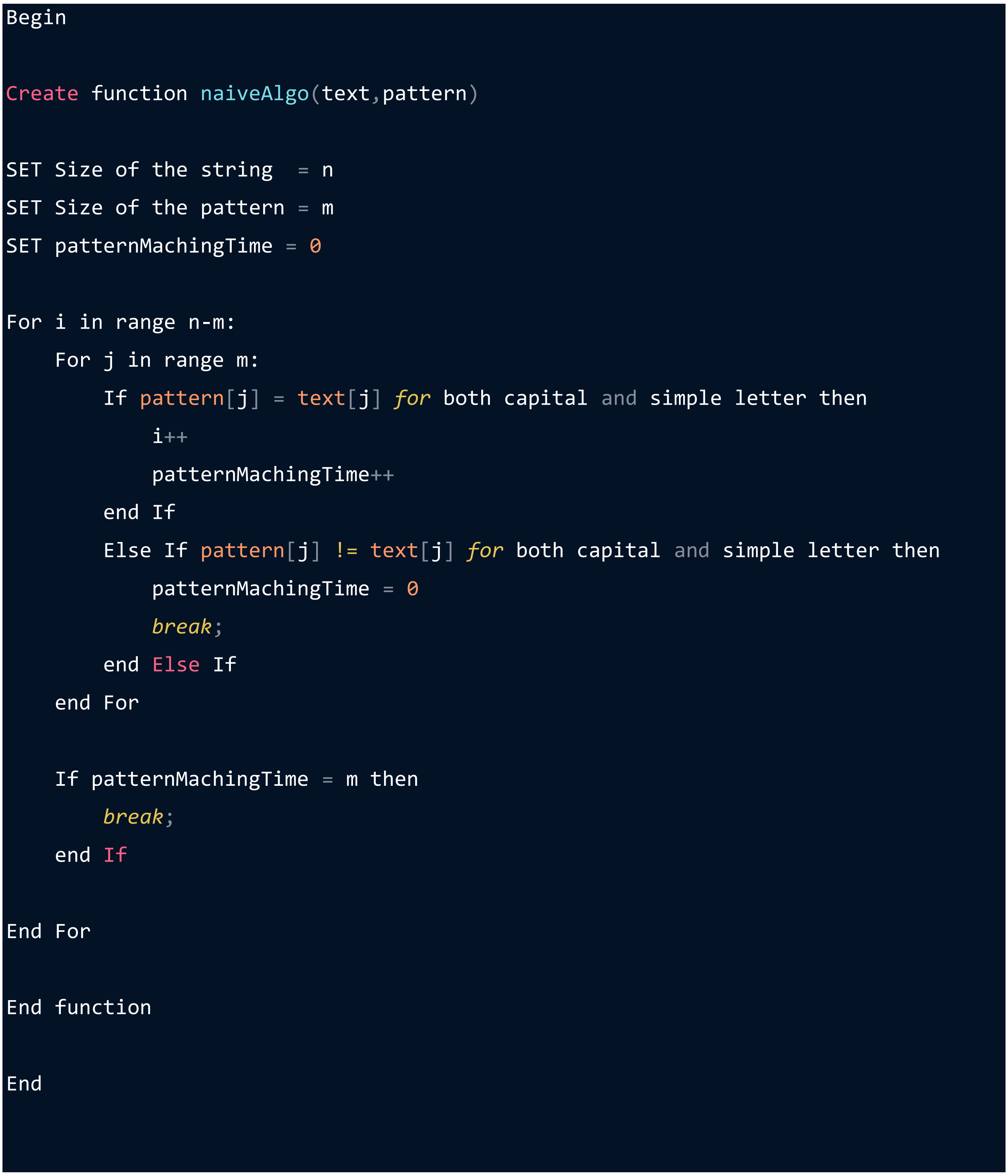
Step 03 => if the first letter does match with the pattern, then compare the 2nd letter of the text with the 2nd letter of the pattern.

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

So, the time complexity of the naïve pattern is:

Worst Case :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.

Test Cases

01.

```
"c:\Users\ACER\Desktop\GitHub\String-Matching-DSA\"20000111-StirngMatching


Enter a search string: Microprocessor
EEE2007 Computer Systems and Microprocessors
EEE2206 Computer Systems and Microprocessors
EEE8022 Microprocessor Systems
Number of matches: 3
```

02.

```
"c:\Users\ACER\Desktop\GitHub\String-Matching-DSA\"20000111-StirngMatching


Enter a search string: circuits
Number of matches: 0
```

03.

```
"c:\Users\ACER\Desktop\GitHub\String-Matching-DSA\"20000111-StirngMatching


Enter a search string: function
MAS8753 Functional Analysis
Number of matches: 1
```

Code Explains

01.

```
void moduleRead()
```

by using moduleRead function,

1.The modules.txt file read one by one line

2.then push that read file in to dynamic array

02.

```
char swapCaseofChar(char character)
{
    if(islower(character))
        character = char(toupper(character));
    else
        character = char(tolower(character));

    return character;
}
```

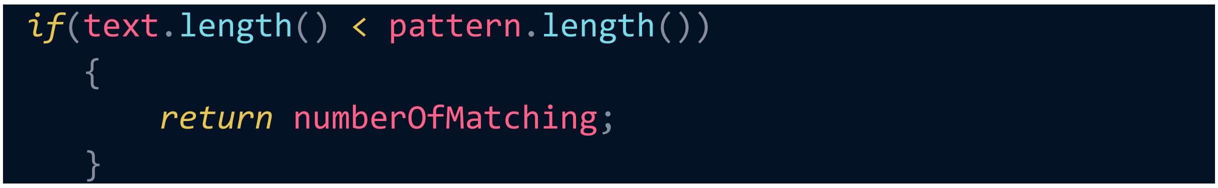
using swapCaseofChar function

01. sawp character case of the given character

02. if given character is 'A' then swapCaseofChar function return 'a'

03. if given character is 'a' then swapCaseofChar function return 'A'

03.



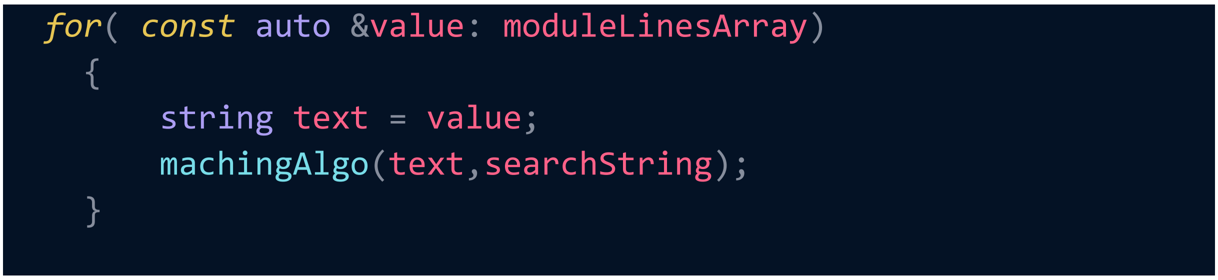
```
 if(text.length() < pattern.length())
    {
        return numberOfMatching;
    }
```

text length mush be greater than pattern length

so if pattern length is greater then text length,

then exit from the matchingAlgo function

04.



```
  for( const auto &value: moduleLinesArray)
    {
        string text = value;
        machingAlgo(text,searchString);
    }
```

using this for loop,

1. pop the string line one by one from the "moduleLinesArray" (dynaminc array)

2. then pass the poped string line and searchString into matchingAlgo function

*****