

Lab_02_AA_Assignment

Aim : Implementation of naïve string matching algo and hoorspool string matching algorithm.

1. Naïve String matching Algorithm:

```
# -*- coding: utf-8 -*-  
"""
```

Created on Fri Jul 17 15:43:05 2020

```
@author: DHRUV  
"""
```

```
def search (pattern, string):  
    len_pattern, len_string = len(pattern), len(string)  
    for ele in range(len_string - len_pattern + 1):  
        i = 0  
        while (i < len_pattern):  
            if (string[ele + i] != pattern[i]):  
                break  
            i += 1  
        if (i == len_pattern):  
            return ele  
    else:  
        return -1
```

```
if __name__ == "__main__":  
    string = input("Enter the text : ")  
    pattern = input("Enter the pattern : ")  
    index_ = search(pattern, string)  
    if (index_ != -1):  
        print(f"pattern found at index {index_}")  
    else:  
        print("pattern not found")
```

output :

```
Enter the text : Welcome to World of Artificial Intelligence.  
Enter the pattern : World  
Pattern found at index 11
```

```
Python 3.7.6 (default, Jan 8 2020, 20:23:39) [MSC v.1916 64 bit (AMD64)]
Type "copyright", "credits" or "license()" for more information.

IPython 7.12.0 -- An enhanced Interactive Python.

In [1]: runcell(0, 'D:/cLg2021/AA/LAB2/BMH_CE054.py')

Enter the string : Hello and enjoy the world of Machine learning

Enter the pattern : Machine
29

In [2]: runcell(0, 'D:/cLg2021/AA/LAB2/naive_CE054.py')

Enter the text : Welcome to World of Artificial Intelligence

Enter the pattern : World
pattern found at index 11

In [3]: |
```

2. BoyerMooreHorspool string matching algorithm :

-*- coding: utf-8 -*-

"""

Created on Sun Jul 19 18:32:26 2020

@author: DHRUV

"""

```
def BoyerMooreHorspool(pattern, string):
    pattern_l = len(pattern)
    skipText_l = len(string)
    if pattern_l > skipText_l:
        return -1
    skip = []
    for i in range(256):
        skip.append(pattern_l)
    for i in range(pattern_l - 1):
        skip[ord(pattern[i])] = pattern_l - i - 1
    skip = tuple(skip)
    i = pattern_l - 1
    while i < skipText_l:
        j = pattern_l - 1
        k = i
        while j >= 0 and string[k] == pattern[j]:
            j -= 1
            k -= 1
        if j == -1:
            return k + 1
```

```

        i += skip[ord(string[i])]
    return -1

if __name__ == '__main__':
    string = input("Enter the string : ")
    pattern = input("Enter the pattern : ")
    pos = BoyerMooreHorspool(pattern, string)
    if pos > -1:
        print (pos)
    else:
        print ("Not Found")

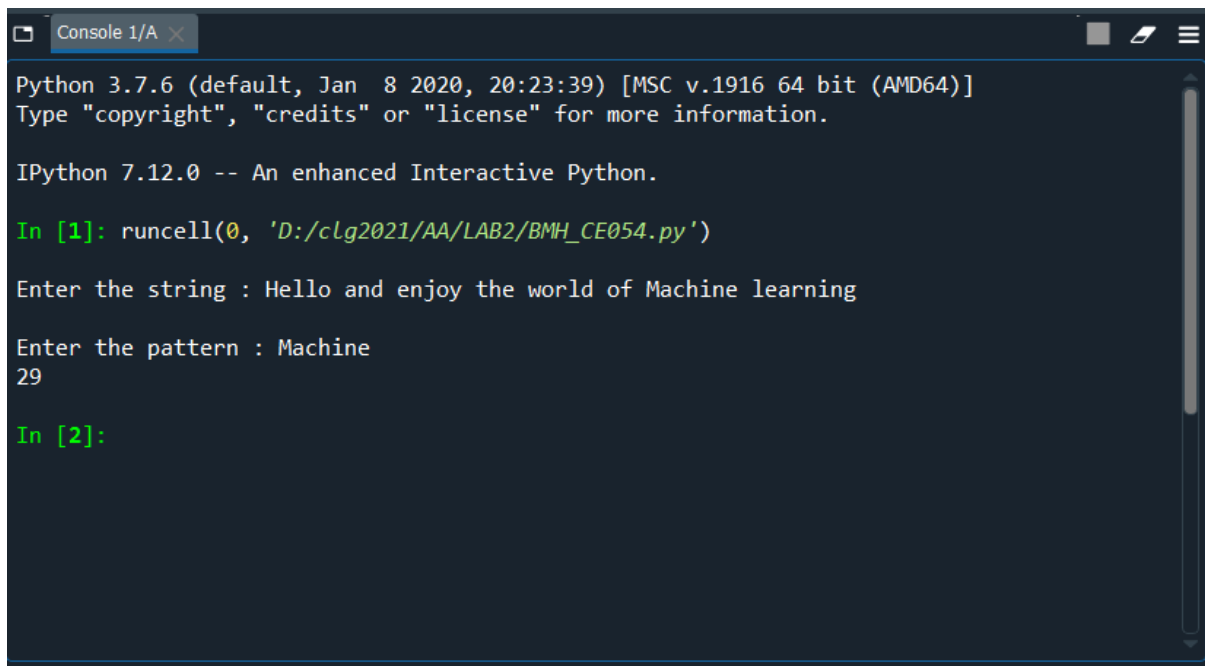
```

output :

```

Enter the string : Hello and enjoy the world of Machine learning
Enter the pattern : Machine
29

```



```

Console 1/A x
Python 3.7.6 (default, Jan 8 2020, 20:23:39) [MSC v.1916 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 7.12.0 -- An enhanced Interactive Python.

In [1]: runcell(0, 'D:/cLg2021/AA/LAB2/BMH_CE054.py')

Enter the string : Hello and enjoy the world of Machine learning

Enter the pattern : Machine
29

In [2]:

```

- Comparison between Naïve string matching and Horspool string matching Algo!

Naïve :

Naïve pattern searching is the simplest method among other pattern searching algorithms. It checks for all character of the main string to the pattern. This algorithm is helpful for smaller texts. It does not need any pre-processing phases. We can find substring by checking once for the string. It also does not occupy extra space to perform the operation.

The time complexity of Naïve Pattern Search method is $O(m*n)$. The m is the size of pattern and n is the size of the main string.

Horspool :

The bad-character shift used in the Boyer-Moore algorithm is not very efficient for small alphabets, but when the alphabet is large compared with the length of the pattern, as it is often the case with the ASCII table and ordinary searches made under a text editor, it becomes very useful.

Using it alone produces a very efficient algorithm in practice. Horspool proposed to use only the bad-character shift of the rightmost character of the window to compute the shifts in the Boyer-Moore algorithm.

The algorithm trades space for time in order to obtain an average-case complexity of $O(n)$ on random text, although it has $O(nm)$ in the worst case, where the length of the pattern is m and the length of the search string is n .

So, i.e for long text we can say that Horspool string matching algorithm is worthy.