String Matching Assignment

20000014

2020/CS/001

**H. D. Sanduni Aaloka**

University of Colombo School of Computing

**Contents**

[Problem 2](#_Toc109059180)

[Why Boyer Moore is used? 3](#_Toc109059181)

[Code Explanation 4](#_Toc109059182)

[Preprocessing function 4](#_Toc109059183)

[boyerMoore function 5](#_Toc109059184)

[Test Cases 7](#_Toc109059185)

[Case 1 7](#_Toc109059186)

[Case 2 8](#_Toc109059187)

[Case 3 8](#_Toc109059188)

# Problem

Write a program that performs the following tasks using a familiar programming language

(Java/C/C++/Python):

1) Prompts the user to enter a pattern P to search for, for example “Architecture”.

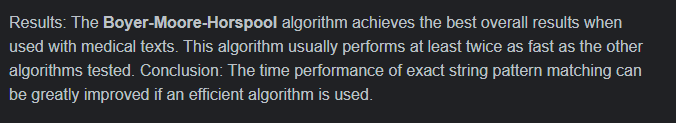
2) It then proceeds by reading the module catalogue stored in the file ‘modules.txt’, where each line contains the title of one module.

3) For every line in the file, the program checks whether it contains the search pattern P. If yes, it prints out this line. For example, the line corresponding to the ARC1015 module contains the string “Architecture” and should be printed. Note: string matching should be case-insensitive, i.e., “Architecture” contains the search string “architecture”.

4) When all lines have been processed, the program prints the number of found matches.

Note: you do not have to follow exactly the same input and output format as shown in the examples. You are also encouraged to experiment by adding other useful features for manipulating the information provided in the module catalogue.

# Why Boyer Moore is used?

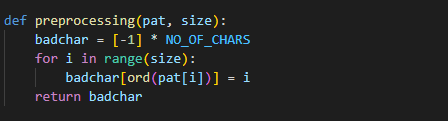


The difference between Boyer Moore and other algorithms is mainly because Boyer Moore algorithm search the string backwards. And if the comparison of the last character doesn’t match the pattern can skip the full length. The worst-case running time of Boyer Moore is O(mn). The best-case time complexity is O(m/n). Practically different string-matching algorithms are suitable at different situations irrespective of its time complexities. If for example KMP algorithm and Boyer Moore algorithm are compared, KMP algorithm works well in situations where the alphabet is not very large and Boyer Moore works well with somewhat longer patterns. Also, Boyer Moore can be sublinear.

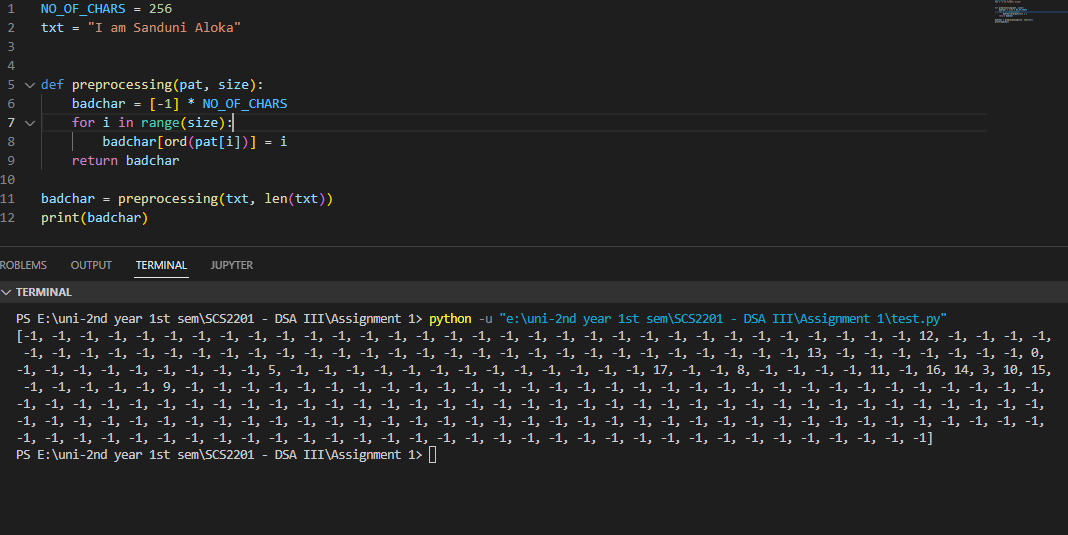
# Code Explanation

The code is written in python language. The Boyer Moore Horspool algorithm is used by using two functions as boyerMoore() and preprocessing().

## Preprocessing function

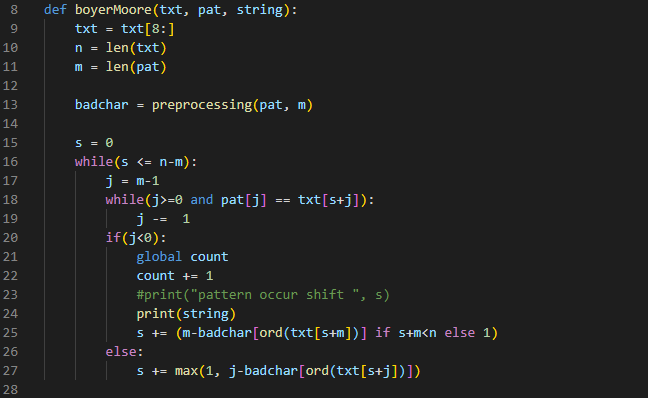


The preprocessing function in Boyer Moore algorithm should result the HpBc table. In this for each character in the alphabet should have a value which is equal to pattern-1 from its rightmost character. The last symbol is not taken into account if this calculation. If only the preprocessing function is used the below array will result.

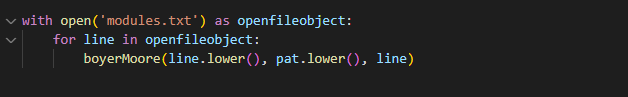


NO\_OF\_CHARS is a variable which is equal to the number 256. Initially the array is initialized with -1. Then for each character in the pattern is the position of it in the pattern is inserted at its index. The index of each character is taken from its Unicode value as ord() function in python returns the Unicode from a given character. At the end of the loop each character in the alphabet will be initialized with its rightmost position in the pattern.

## boyerMoore function



The text and pattern are inserted as arguments into the boyerMoore function. In the module.txt a single line will be for example APL1001 Alternative Practice Histories where “APL1001” is the code of the module and “Alternative Practice Histories” is the name of the module.



As txt lower case of each line is included and as line each line of txt as it is included. The argument string is used because we need to print the line of the module in boyerMoore function if the search is successful. Both txt(text) and pat(pattern) is included as lowercase is because the search should be case sensitive.

Ex: “Architecture” contains the search string “architecture”.

The variable s denotes the shift. Then txt is truncated to get only the name of the module. Thus, the pattern searched will only be searched through the name of the module. j variable is the indicator for the pattern and j starts from the end of the pattern. That’s because in Boyer Moore searching is done backwards. If the character at position j is matching with the corresponding character of the text and if it is not the first character of the pattern the variable j is decremented by 1. Then that character is compared and so on.

If the first character of the pattern is also compared then the index j will become -1.Thus, if j==-1 then there is a match. The count variable which stores the number of times the pattern is found is incremented by 1 and the text is printed. The argument string is used to printing as txt will be all lower case and also it is trunacated.

Then the equation,

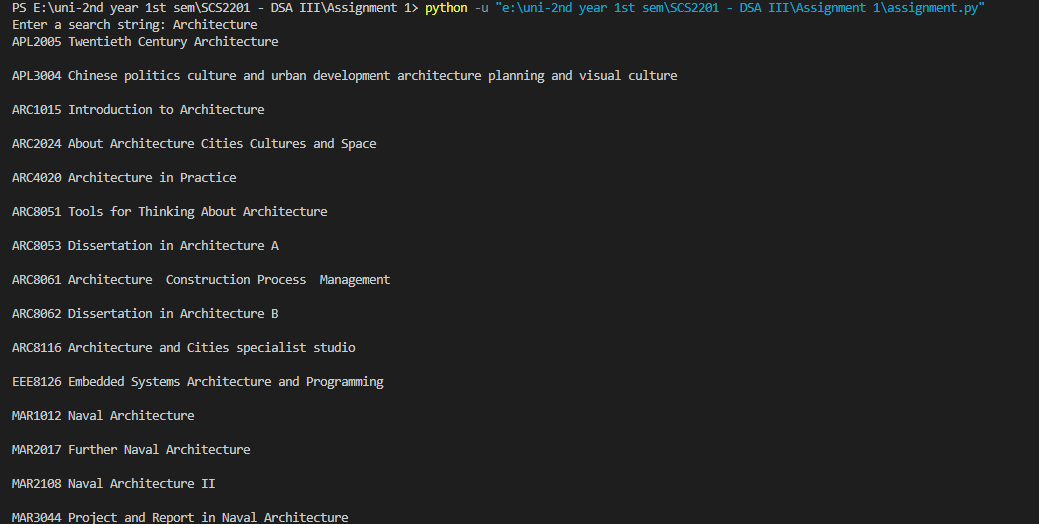


is used to get the next shift.

As s is the shift and m is the length of the pattern s+m gives the last index of the text where the pattern is matched. Badchar[ord(txt[s+m])] gets the value of that particular character in the text from the HpBc table. The value that comes from the badchar array is the index of the final encounter of that particular character in the pattern. Thus substarcting it by m(length of the pattern) gives us the shift where the badcharacter is matched to the pattern.

## Test Cases

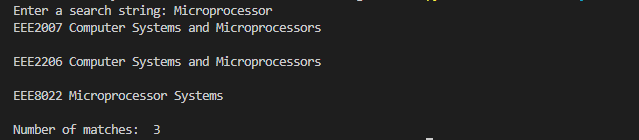
### Case 1



**………..**

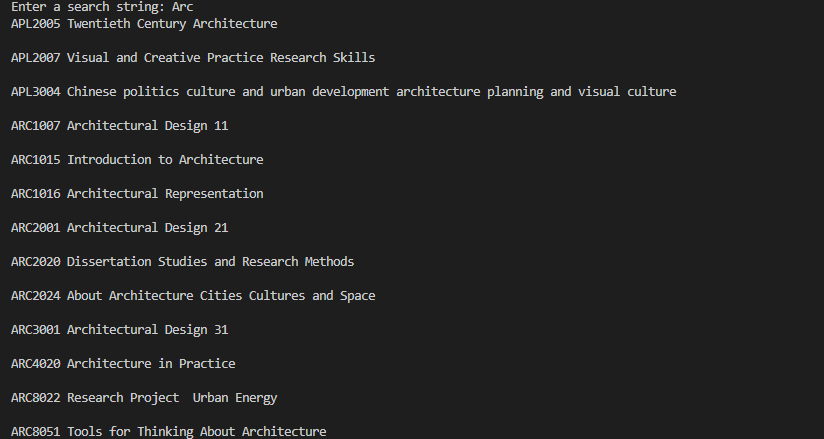


### Case 2



### Case 3

To prove the program is case sensitive





……….

