String Matching Algorithms

Assignment Report

Ramindu Walgama

20201959 - 2020/CS/195

2020cs195@stu.ucsc.cmb.ac.lk

Jul 19, 2022

Data Structures & Algorithms III – SCS 2201

Contents

1 Introduction **4**

2 Comparison of Algorithms **5**

3 Performance **6**

4 Approach **9**

4.1 Pseudo code 9

4.2 Language choose 10

4.3 Output 10

5 Conclusion **11**

List of Figures

1 Search Time (in secs) and Number of Files Carved for Image 11-carve-fat.dd Liao (2015) 6

2 Search Time (in secs) for Image “12-carve-fat.dd” Liao (2015) 6

3 Search Time Comparison for Image ”11-carve-fat.dd” Liao (2015) 7

4 Search Time Comparison for Image ”12-carve-fat.dd” Liao (2015) 8

5 Script output for ”Informatics” in modules.txt dataset 10

List of Tables

1 The time complexity of String Matching Algorithms; Liao (2015) 5

1 Introduction

String matching algorithms are used to find the occurrence of a given pattern in a given text. Below mentioned algorithms are considered algorithms for this assignment.

1. The Naïve Algorithm(Brute Force Algorithm)
2. The Knuth-Morris-Pratt Algorithm(KMP Algorithm)
3. The Rabin-Karp Algorithm
4. The Boyer-Moore Algorithm
5. The Boyer-Moore-Horspool Algorithm

Compared the above-listed algorithms in detail, the second chapter. Furthermore performance wise is analyzed in the third chapter, considering the theoretical as well as practical aspects. In the fourth chapter, the approach to the assignment is discussed briefly, including psuedo code, language selection, and small discription about the output format.

2 Comparison of Algorithms

The Naïve Algorithm is a brute force algorithm that is not suitable for this task when considering the performance. The other four algorithms perform a pre-process before the execution of the process as shown in the below table 1.

Algorithm	**Founder & the year**	**Pre-processing**	**Searching phase**
The Naïve Algorithm(Brute Force Algorithm)	-	-	O(m x n)
The Knuth-Morris-Pratt Algorithm(KMP Algorithm)	Boyer & Moore, 1977	O(m)	O(m x n)
The Rabin-Karp Algorithm	Rabin & Karp, 1987	O(m)	O(m x n)
The Boyer-Moore Algorithm	Knuth et al., 1977	O(m + σ)	O(m x n)
The Boyer-Moore-Horspool Algorithm	Horpool, 1980	O(m + σ)	O(m x n)

Table 1: The time complexity of String Matching Algorithms; Liao (2015)

$\sigma = alphabet\ size\ of\ the\ pattern\ and\ text;\ m = size\ of\ the\ pattern;\ n = size\ of\ the\ text$

KMP algorithm is more suitable when the pattern contains a fewer alphabet and the pattern contains substrings that are repeated in the pattern so that when part of the pattern matches with the text and the rest does not match, it won't start to search from the beginning and avoid another rematch (Bansal, 2020). Here, can't guarantee that the entered text and the pattern are similar to the above-mentioned points since the pattern is user input (Vassar, 2018). The KMP algorithm is more reliable than the Rabin-Karp algorithm because there can be a collision during the hash table lookup and the hash function. The Boyer-Moore algorithm and the KMP algorithm are used widely (Liao, 2015). The Boyer-Moore algorithm is an efficient algorithm in both theoretical as well as practical aspects. The Boyer-Moore-Horspool Algorithm is a simplified algorithm of the Boyer-Moore algorithm. This performs well when the given alphabet is large and has proven that the Horspool algorithm achieves the best overall results in medical texts (Lovis and Baud, 2000) which is much similar to the given dataset.

3 Performance

In practical terms, algorithms, as mentioned earlier, have a big difference even though they have O(m x n) time complexity as mentioned in the above table where m is the length of the pattern and n is the length of the text which is clearly visible on the below-quoted table 1 and table 2 which has performed for different types of image datasets. Here the time measured in seconds. (Liao, 2015).

Algorithm	**doc1**	**doc2**	**gif**	**jpg1**	**jpg2**	**mov1**	**mov2**	**mov3**	**pdf1**	**pdf2**	**wav**	**wmv**
brute force	0.68 3	0.54 3	0.62 1	0.62 2	0.79 1	1.27 1	1.26 2	1.25 1	0.62 1	0.62 2	0.54 1	0.53 2
Boyer-Moore[1]	0.87 3	0.43 3	0.47 1	0.61 2	0.55 1	0.44 1	0.45 2	0.45 1	0.46 1	0.46 2	0.45 1	0.45 2
KMP	0.84 3	0.69 3	0.75 1	0.80 2	1.02 1	0.78 1	0.78 2	0.78 1	0.79 1	0.79 2	0.69 1	0.69 2
Karp-Rabin	0.20 3	0.17 3	0.18 1	0.19 2	0.24 1	0.17 0	0.17 0	0.17 0	0.19 1	0.19 2	0.17 0	0.17 2
Horspool	0.88 3	0.43 3	0.48 1	0.66 2	0.59 1	0.42 1	0.42 0	0.42 1	0.47 1	0.48 2	0.42 0	0.47 2

Figure 1: Search Time (in secs) and Number of Files Carved for Image 11-carve-fat.dd Liao (2015)

Algorithm	**doc1**	**doc2**	**gif**	**jpg1**	**jpg2**	**mov1**	**mov2**	**mov3**	**pdf1**	**pdf2**	**wav**	**wmv**
brute force	1.13	1.06	1.14	1.11	1.07	2.46	2.44	2.47	1.11	1.11	1.07	1.07
Boyer-Moore[1]	1.86	0.98	1.02	1.34	1.02	1.03	1.02	1.02	1.03	1.03	1.02	1.06
KMP	1.47	1.36	1.43	1.42	1.37	1.56	1.56	1.63	1.45	1.42	1.37	1.37
Karp-Rabin	0.35	0.33	0.34	0.34	0.33	0.33	0.33	0.33	0.34	0.34	0.33	0.33
Horspool	1.86	1.00	1.04	1.45	1.04	0.96	0.96	0.96	1.05	1.05	0.94	1.09

Figure 2: Search Time (in secs) for Image "12-carve-fat.dd" Liao (2015)

A better comparison can be done using comparison chart 3 with respect to the table 2 above and chart 4 with respect to the table 2 above. The data is drawn for other algorithms as well, which haven't been discussed in this report. According to the discussed algorithms, it is clear that the Boyer-Moore algorithm and the Boyer-Moore-Horspool algorithm perform better than the other discussed algorithms.

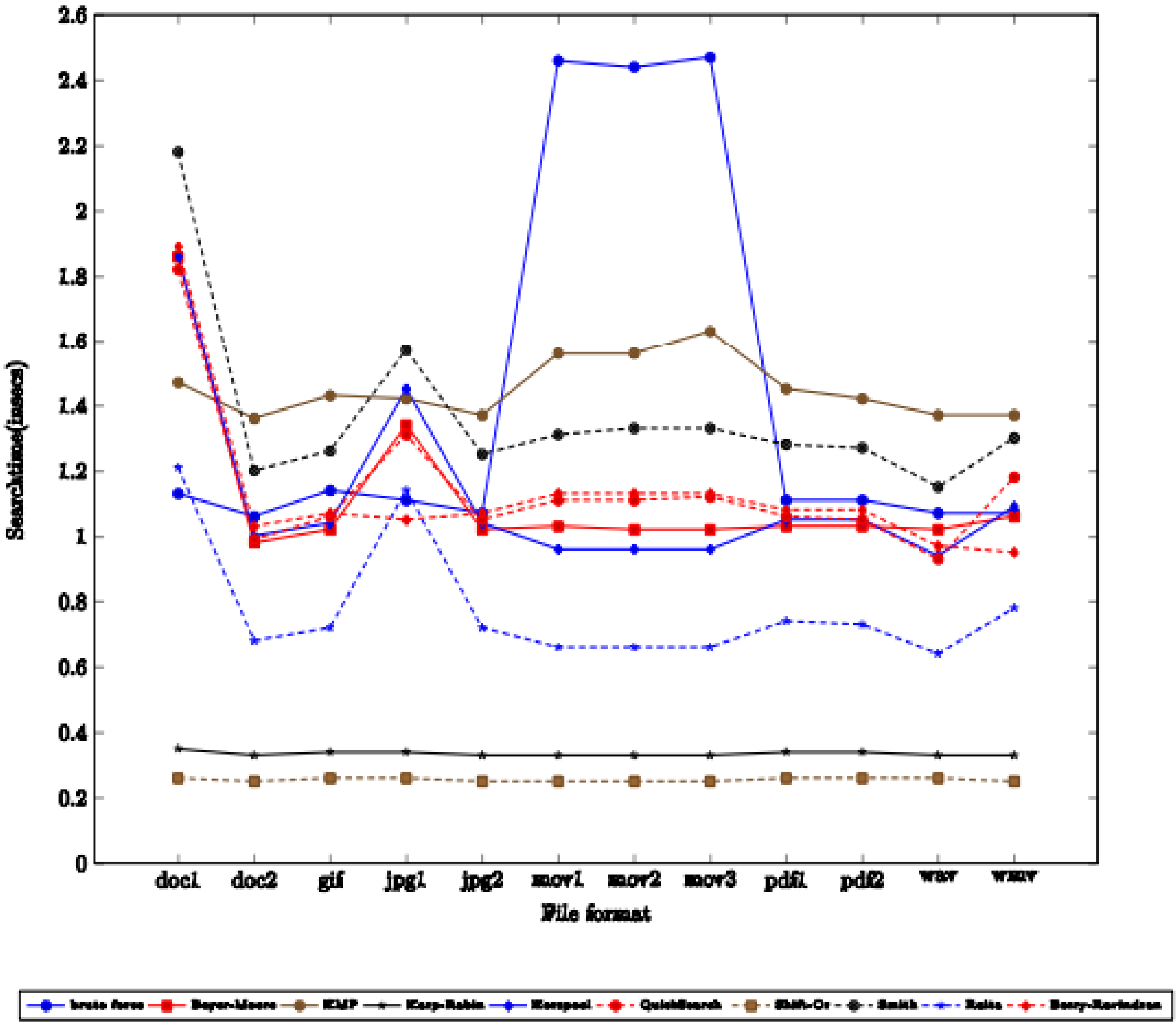

Figure 3: Search Time Comparison for Image "11-carve-fat.dd" Liao (2015)

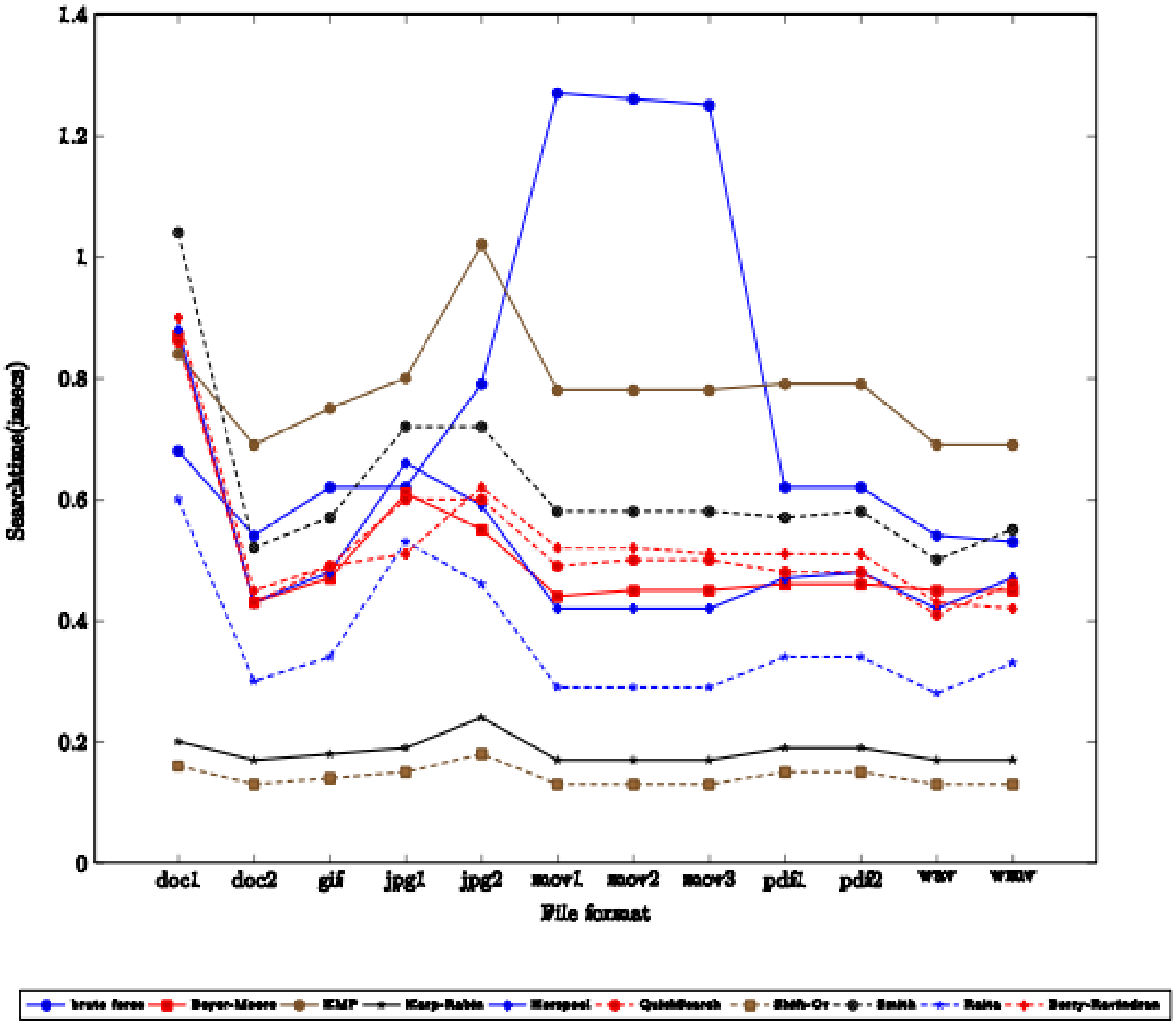

Figure 4: Search Time Comparison for Image "12-carve-fat.dd" Liao (2015)

4 Approach

The Boyer-Moore-Horspool algorithm is used to complete this assignment by referring to the lecture slides, and other online material. The algorithm approach is very simple.

4.1 Pseudo code

Referred lecture notes and external sources such as *lectures of University of Helsinki*, *lectures of Université Gustave Eiffel* and YouTube tutorials to understand and write the algorithm.

Algorithm 1 The Boyer-Moore-Horspool algorithm

function HORSPOOL(txt, pattern)
    $n \leftarrow$ length of the text
    $m \leftarrow$ length of the pattern
    for $i = 0, 1, \ldots, \Sigma$ **do**
        shift[i] $\leftarrow$m
    end for
    for $i = 0, 1, \ldots,$m-1 **do**
        shift[pattern[i]] $\leftarrow$m-$i - 1$
    end for
    $position_{txt} \leftarrow 0$
    while $position_{txt} + m <= n$ **do**
        if $txt[position_{txt} + m - 1] = pattern[m - 1]$ **then**
            $position_{pattern} \leftarrow m - 2$
            while $position_{pattern} >= 0$ and $txt[position_{txt} + position_{pattern}] = pattern[position_{pattern}]$ **do**
                $position_{pattern} \leftarrow position_{pattern} - 1$
                if $position_{txt} < 0$ **then**
                    pattern found
                end if
            end while
        end if
        $position_{txt} \leftarrow position_{txt} + shift[txt[position_{txt} + m - 1]]$
    end while
end function

4.2 Language choose

Used Python to implement the Boyer-Moore-Horspool algorithm. There are several reasons to choose the Python language. Python is widely used for data analytics, big data, and data science. Since here dealing with a large dataset, it is easy to visualise the output in many forms, such as different types of graphs and tables.

4.3 Output

The script required a single input from the user; a string to be searched in the "modules" dataset. After the execution, it will display the results in a table containing the line number of the matching string, module code, module name, number of matches in the same line, and the indices of the matches for that particular line. At the end, it will give a summary with the number of total occurrences and the number of total lines matched. The output for the input "Informatics" is shown below.

Line	Module Code	Module Name	Matches	Indices
689	CEG1713	Informatics 1	1	[9]
711	CEG2722	Informatics 2	1	[9]
735	CEG3715	Environmental Informatics	1	[23]
736	CEG3716	Geospatial Informatics	1	[20]

Total Lines Found	4
Total Word Matches	4

Figure 5: Script output for "Informatics" in modules.txt dataset

For to get the output as shown above a third party library, tabulate being used. Haven't any other libraries and this is used only for the ouput formatting.

5 Conclusion

This report is mainly focused on the algorithms that are being taught within the 2201-DSA module but learned extra learnings by completing this assignment such as new algorithms which are not being discussed during the lectures and the comparisons of these algorithms to more extent.

References

Bansal A. (2020). *What is the best algorithm for pattern searching?* Accessed: 2022-07-19. URL: https://www.quora.com/What-is-the-best-algorithm-for-pattern-searching.

Liao Y.-C. (2015). "String Matching Algorithm". *A Survey of Software-based String Matching Algorithms for Forensic Analysis*. Association for Computing Machinery.

Lovis C. and Baud R. H. (2000). "Fast exact string pattern-matching algorithms adapted to the characteristics of the medical language". *Journal of the American Medical Informatics Association* 7.4, pp. 378–391. DOI: 10.1136/jamia.2000.0070378.

Vassar S. (2018). *KMP algorithm vs Rabin-Karp algorithm*. Accessed: 2022-07-19. URL: https://www.quora.com/Which-is-better-and-efficient-between-the-KMP-algorithm-and-the-Rabin-Karp-algorithm.