American University of Sharjah

College of Engineering

Design and Analysis of Algorithms – CMP340

Fall 2020

Project: String Matching Methods

Brute-Force vs Horspool

Joel D'Souza b00079296

Natasha Madhu g00076735

Sabbir Alam b00079438

Professor: Dr. Salam Dhou

Submission Date: November 30, 2020

Introduction

The simplest method to perform string searching for a pattern in a text file is the brute-force method. The implementation of this method is straight forward, however, it is rarely the most efficient method around. This method does not require a lot of space but sacrifices time spent in comparisons. This algorithm works by comparing every character in the text to the pattern and tries to find a match.

The Horspool string-search algorithm, published by Dr. R. Nigel Horspool in 1980, is a much more efficient approach to search for a pattern in a text file compared to brute force and can be applied to real-life searches. This method gains its efficiency by sacrificing space in order to reduce the time taken to search. The way it does this is by creating a shift table that allows the pattern to align itself in order to match its characters with the text.

Figure 1: Dr. R. Nigel Horspool

Theoretical Analysis

Theoretical comparison of the best, average, and worst-case scenarios for the Brute-Force and Horspool string search algorithms are shown below:

Analysis of Brute-Force String Matching

Basic Operation = Comparison

n = length of the document

m = length of the pattern

Best Case Complexity:

When String is found at the beginning of the document

$$C(n) = m+1$$

The best-case complexity belongs to $\Omega(m)$

Worst Case Complexity:

We arrive at the worst case complexity when the text is the same character repeated multiple times and the pattern is that same character, but with the last letter being different. (Eg. Text file = “BBB...”, Pattern = “ABBB..”)

$$C(n) = \sum (0 \text{ to } n-1) * \sum (0 \text{ to } m-1) * 1$$

$$= (n-1-0+1) * (m-1+1)$$

$$= (n) * (m)$$

$$= nm$$

The worst-case complexity belongs to $O(nm)$

Average Case Complexity:

According to research done on Brute Force string search algorithms, the average-case complexity belongs to Θ (n+m).

Analysis of Horspool String Matching

Basic Operation = Comparison

n = length of the document [0 to n-1]

m = length of the pattern [0 to m-1]

Best Case Complexity:

When String is found at the beginning of the document)

$$C(n) = m+1$$

The best-case complexity belongs to Ω(m)

Worst Case Complexity:

We arrive at the worst case complexity when the text is the same character repeated multiple times and the pattern is that same character, but with the last letter being different. (Eg. Text file = “BBB...”, Pattern = “ABBB..”)

$$C(n) = \sum (m-1 \text{ to } n-1) * \sum (0 \text{ to } m-1) * 1$$

$$= ((n-1)-(m-1)+1)*(m-1-0+1)$$

$$= (n-1-m+1+1)*(m)$$

$$= (n-m+1)*(m)$$

$$= nm-m^2+m$$

The worst-case complexity belongs to O(nm)

Average Case Complexity:

According to research done on Horspool's string search algorithm (Baeza-Yates & Regnier, 1992), the average number of comparisons for binary alphabet is:

$$C(n) = (26 \ln 2 – 8 \ln 3 – 17/2)*n = 1.2782*n$$

Therefore, the average case complexity is $\Theta(n)$

<u>Comparison of the String Matching Algorithms</u>

For the best case, both the brute force and Horspool algorithm have a complexity of $\Omega(m)$.

For the average case, the Horspool algorithm performs better with a complexity of $O(n)$ while the brute force algorithm has a complexity of $O(n+m)$.

For the worst case, both the brute force and Horspool algorithm have a complexity of $O(nm)$.

Experimental Results

To experiment on the difference in complexity between Brute Force string matching and using the Horspool algorithm for string matching, we used 2 books publicly available in the Project Gutenberg library, namely, the entire Sherlock series by Sir Arthur Conan Doyle and War and Peace by Leo Tolstoy. Our experimental results are as shown below:

n = 6,360,210, Text:- SHERLOCK (n→ Number of Characters)

Sentence Number and Description	**Size of pattern (m)**	**Bruteforce** # of comparisons	**Horspool** # of comparisons	**Which performed better**
1 - Plagiarized sentence in the 1st Quarter of the text file	166	2,010,863	86,819	Horspool
2 - Plagiarized sentence in the 2nd Quarter of the text file	104	3,446,902	145,095	Horspool
3 - Plagiarized sentence in the 3rd Quarter of the text file	124	5,668,688	239,219	Horspool
4 -Plagiarized Small Sentence in the middle of the text file	44	3,734,595	206,513	Horspool
5 - Plagiarized Big Sentence in the middle of the text file	344	3,539,066	104,853	Horspool
6 - Not Plagiarized Small Sentence (Does not exist in the file)	109	6,735,613	279,078	Horspool
7 - Not Plagiarized Big sentence (Does not exist in the	530	6,841,636	168,081	Horspool

file)				
8 - Not plagiarized one-word sentence (Does not exist in the file)	9	6,801,780	900,561	Horspool
9 - Plagiarized first sentence in the text file	64	64	64	Equal
10 - Plagiarized last sentence in the text file	25	6,483,795	442,356	Horspool
TOTAL	**-**	45,263,002	2,572,639	**-**

n = 195,216, Text:- WarAndPeace (n → Number of Characters)

Sentence Number and Description	**Size of pattern (m)**	**Bruteforce** # of comparisons	**Horspool** # of comparisons	**Which performed better**
1 - Plagiarized sentence in the 1st Quarter of the text file	156	67,071	2,380	Horspool
2 - Plagiarized sentence in the 2nd Quarter of the text file	120	119,810	4,662	Horspool
3 - Plagiarized sentence in the 3rd Quarter of the text file	112	170,023	7,730	Horspool
4 -Plagiarized Small Sentence in the middle of the text file	28	109,370	7,880	Horspool
5 - Plagiarized Big Sentence in the middle	288	116,640	4,208	Horspool

of the text file				
6 - Not Plagiarized Small Sentence (Does not exist in the file)	105	205,790	8,668	Horspool
7 - Not Plagiarized Big sentence (Does not exist in the file)	530	208,166	4,767	Horspool
8 - Not plagiarized one-word sentence (Does not exist in the file)	9	208,873	27,645	Horspool
9 - Plagiarized first sentence in the text file	60	60	60	Equal
10 - Plagiarized last sentence in the text file	102	211,730	8,722	Horspool
TOTAL	-	1,417,533	76,722	-

Experimental Analysis

Based on the above table, we observe that Sentences 1, 2, 3, and 10 appear within the 1st, 2nd, 3rd Quarter, and end of the text file respectively and these four rows indicate that the number of comparisons necessary to find a pattern increases, for both Brute Force and Horspool, as the pattern appears deeper within the text.

Next, from Sentences 4 and 5, we observe that Horspool performs better when the pattern to be matched is big (around 50% better), however, no such significant trend can be observed with the Brute Force technique. We believe that the reason for this is because a larger pattern implies that we shift by a greater number of characters if we encounter a mismatch that is not present within the shift table.

Furthermore, from Sentences 6, 7, and 8 we observe that if the pattern is not found within the text, Horspool performs worse for smaller sizes of the pattern to be searched, whereas, there

appears to be no such significant difference based on pattern size for the Brute Force algorithm. We again conclude this is due to the smaller maximum number of shifts in the Horspool shift table, due to the smaller pattern size and hence leads to smaller shifts upon mismatch.

Finally, in Sentence 9, both Horspool and Brute Force string matching perform exactly the same when the pattern to be matched appears at the beginning of the text.

Based on the above results, we arrive at the following experimental best, worst and average cases for Brute Force and Horspool string matching.

N = 6,360,210, Text:- SHERLOCK (n → Number of Characters)

	Brute force	**Horspool**
Best	64 (Beginning of the text)	64 (Beginning of the text)
Average	4,526,300.2 comparisons	257,263.9 comparisons
Worst	6,841,636 (Not plagiarized, big sentence)	900,561 (Not plagiarized, one-word sentence)

N = 195,216, Text:- WarAndPeace (n → Number of Characters)

	Brute force	**Horspool**
Best	60 (Beginning of the text)	60 (Beginning of the text)
Average	141,753.3 comparisons	7,672.2 comparisons
Worst	208,873 (Not plagiarized, big sentence)	27,645 (Not plagiarized, one-word sentence)

We can see that both the Brute Force and Horspool string matching algorithms have the best case when the pattern to be matched appears in the first line of the text. This agrees with our theoretical analysis done earlier.

We also see that Brute Force performs the worst on big sentences that cannot be found within the text however, Horspool performs the worst on very small sentences (1 word) that are not present within the text. Yet we were still unable to reach the theoretical worst case for both these

algorithms since that requires specific texts to be searched and patterns to be matched that do not apply when it comes to creating a plagiarism checker.

Lastly, we also found that on average the Horspool string matching algorithm is approximately 18 times more efficient than the Brute Force algorithm, based on the number of comparisons made (17.6x better for Sherlock and 18.5x better for War and Peace). Again, we were unable to match the theoretical average case, since we were unable to attain the absolute worst case, leading to a greater bias towards the best-case scenario.

Conclusions

Through both our experimental and theoretical analysis we can conclude that:

- The Horspool algorithm is always better than or just as good as the Brute Force method of string matching, based on the number of comparisons made.
- The further into the text a given pattern is present, the more the number of comparisons both the algorithms take to search for the pattern.
- Both Horspool and Brute Force have the same best-case scenario.
- Bigger patterns lead to better performance of the Horspool algorithm and vice-versa.
- In a plagiarism detector, if the pattern is not plagiarized and the text being searched is not a bunch of repeated characters, the Horspool algorithm will always have the number of comparisons to be less than the size of the text. Whereas, for the Brute Force method, the number of comparisons will always be greater than or equal to the size of the text.
- Furthermore, given the above conditions for the plagiarism detector, the theoretical worst-case scenario will never be reached.

References

Ricardo A. Baeza-Yates, Mireille Régnier, Average running time of the Boyer-Moore-Horspool algorithm, Theoretical Computer Science, Volume 92, Issue 1, 1992, Pages 19-31, ISSN 0304-3975, https://doi.org/10.1016/0304-3975(92)90133-Z. (http://www.sciencedirect.com/science/article/pii/030439759290133Z)