Computer Science

Extended Essay

Research Question:

To what extent the variation in search pattern will affect the efficiency of Rabin Karp algorithm and Boyer Moore algorithm in the terms of time complexity?

Word count: 3,767

Contents

Introduction :

Every time you type a search query in a search engine how does Google display you precise and needed results in milliseconds even though it owns eight data centres [1] around the world? Or when you type down notes, how spell checker finds errors and suggest suitable spelling suggestions for all the misspelled words even though there are a total of 171476 English words[2]. What happens behind the screen? Many daily processes like these use fundamental algorithms called the string matching algorithms to find one pattern from; the enormous datasets. The ideology of how major tasks rely on these simplistic and basic called strings inspired me to do this research on the efficiency of string matching algorithms. String matching algorithms are fundamental algorithms used for finding a particular pattern from a dataset.

As we grow up, the time has become a crucial factor in our hectic lifestyles and as technology proceeds to develop, the amount of data stored are expanding simultaneously so it is essential to determine the most suitable string searching algorithm with a minimal amount to average runtime for processing the data. Purpose of this investigation is to examine how the entered pattern may influence the time complexity of the two algorithms. For this investigation, I will be taking two popular string matching algorithms the Rabin Karp algorithm and BoyerMoore algorithm to find how the entered pattern will affect the time complexity of both the algorithms.

---

[1] "Google Data Center FAQ & Locations | Data Center Knowledge." 17 Mar. 2017, https://www.datacenterknowledge.com/archives/2017/03/16/google-data-center-faq. Accessed 9 May. 2019.

[2] "How many words are there in the English ... - Lexico.com." https://www.lexico.com/en/explore/how-many-words-are-there-in-the-english-language. Accessed 9 May. 2019.

Background information :

String matching algorithms and their applications:

Words have performed quite a significant role in our day to day lives. Most of us do not know the immense importance of these tiny fundamental blocks of language. Humans have been using words as a communication tool for so long that no one knows when words were invented. Only imagining a world without words may sound highly disastrous. We use words everywhere and in technical terms, words can be known as a form of strings. Strings are known as a combination of characters. Characters are the alphabet, numbers, punctuation, space or symbols[3]. String matching algorithm also called string searching algorithm finds the occurrence of the pattern from the text and returns the position of the pattern as output.

Ever since humankind started storing the data in string format, the problem of finding the string among collective data sets became a significant issue and this led to the discovery of various string matching algorithms. The naive string search algorithm is the basic string search algorithm. We perform this algorithm in our day to day life without recognizing the name of it. If we want to find a word we usually take the pattern which we need to find and infer it through the text by comparing each string of the pattern with the text, checking if the characters match with each other. If it matches we stop inferring and if it does do match, we continue inferring through the text to find where the pattern exists. This algorithm is simple yet inefficient .so this issue further inspired inventors like J Strother Moore, Robert Stephen Boyer, Michael O. Rabin ,and Richard M. Karp to

[3] "Character Definition - TechTerms." https://techterms.com/definition/character. Accessed 29 May. 2019.

discover various string matching algorithms. String searching algorithms are becoming an essential part of our lives as we use it in day to day applications. String matching algorithms are used in spell checkers, in search engines, in plagiarism detection programs, in increasing field of bioinformatics for finding DNA sequences, in Digital Forensics, in information retrieval systems for text mining, and spam filters. All the string matching algorithms are divided into four types according to the way they approach the given data. They are classical algorithms, bit parallelism algorithms, suffix automata algorithms ,and hashing algorithms. For this investigation, two popular algorithms with different approaches will be chosen. from classic algorithms, Booyer Moore algorithm will be chosen as it is said to one of the oldest benchmark string searching algorithms as many variations have been developed lately using this as a base and from the hashing algorithm, Rabin Karp algorithm was chosen as it uses the powerful hash function to process.

Rabin karp algorithm :

Rabin karp was discovered by [4]Richard M. Karp and Michael O. Rabin during the year 1987. Since it uses a hashing approach to the process , a hashing function is used for calculating hash values for all the characters in the string. Each character of the text is provided with a hash value .the hash values for each character present in the text and pattern is generated with the help of a hash function. A substring is a segment of the text which is taken for comparison from the existing text. The entered pattern's hash value is compared with the hash value of the substring and if the hash values match,

[4] "Rabin-Karp Pattern Searching Algorithm - OpenGenus IQ." https://iq.opengenus.org/rabin-karp-string-pattern-searching-algorithm/. Accessed 29 May. 2019.

the hash value of each individual character of the pattern and the hash value of each individual character of the substring is compared. If the hash value of the individual characters does not match, then the algorithm will slide over the text and choose a new substring which is nearby to compare. After a new substring is chosen from the text, again the whole process gets repeated. When the hash value of the individual characters of the pattern matches with the individual characters of the substring which is present in the text, then the pattern is considered to be found and the index values are returned. The use of hashing is believed to speed up the time taken for finding the match required. The more complex the hash functions, the more accurate matches will be found.

This is the hash function which will be used in this investigation:

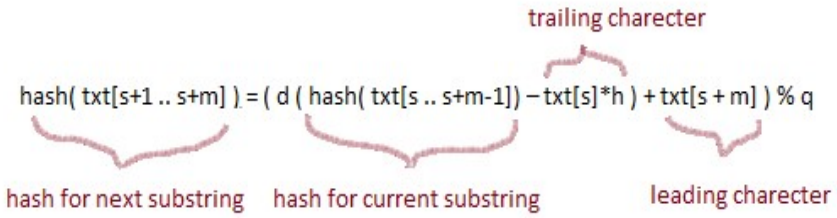


Figure 2: the hash function formula

d: represents the total number of characters present in the ASCII code.

q: represents a prime number

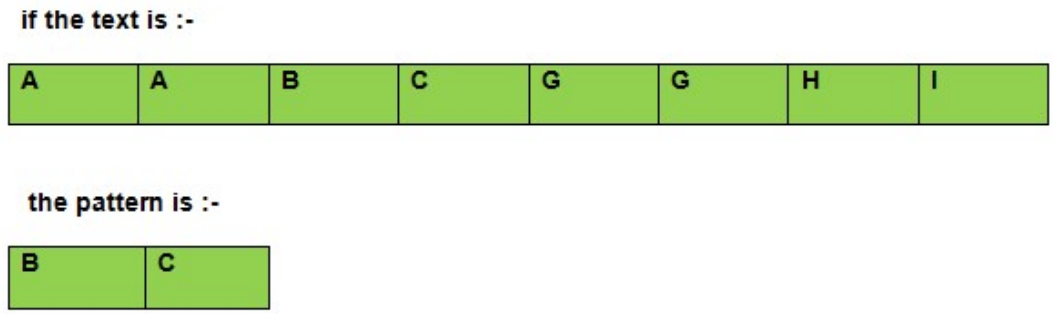
h: represents $d^{(m-1)}$

m: represents the pattern length

n: represents the text length

This formula also includes rehashing was the hash values of next substring will be generated with the help of hash value of current substring and the next character in the text.

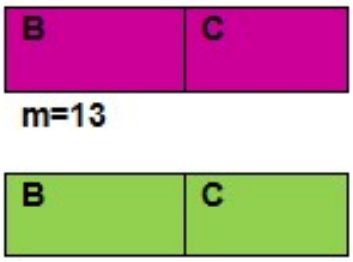
For example:



Assume the hash value of pattern is n and in each character comparison, the hash values of substring and pattern is compared until a match is found. Substring is denoted with the colour magenta.

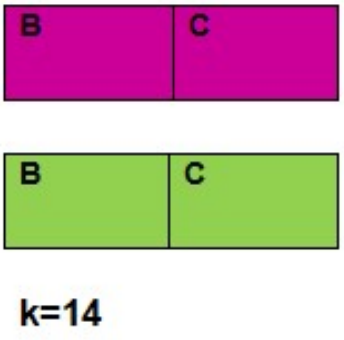
During the first comparison, the hash value of substring the does not match with n so it's a mismatch. Since the mismatch takes place, next substring is taken for comparison from the sequence of text. The hash values of next substring = n, so that particular substring is taken for individual character comparison. Assume hash value of B = 13 and hash value of C = 14 and hash values of letter B in the pattern = m and hash values of letter C in the pattern=k.

First comparison:



Now m = 13 so it is proven during the first comparison, the first character of the substring is same as first character of the pattern.

Second comparison:



During the second comparison, the k matches with hash value of C (k=14).so the match is said to be found.

Boyer Moore algorithm :

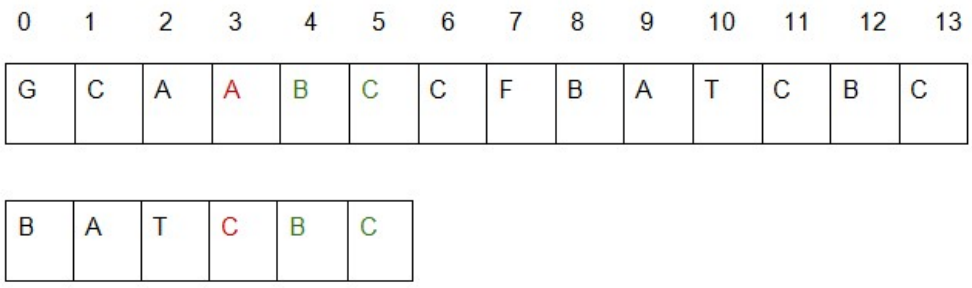
Boyer Moore algorithm was discovered by [5]Robert S. Boyer and J Strother Moore in the year 1977 and it is said to perform fast as the pattern length starts increasing .this algorithm uses classic approach to find the required pattern. There are two ways to approach in Boyer Moore algorithm, which are good suffix rule and bad character heuristics. In this investigation, bad character heuristics will be used.

If a character of the text mismatches with a character of the pattern, that character is called as bad character. So the algorithm compares the pattern with the text from rightmost character in the pattern ,and whenever a mismatch (bad character) is spotted, the algorithm skips alignment until either the pattern matches with the text or until the entered pattern has passed over the mismatched string in the given text.

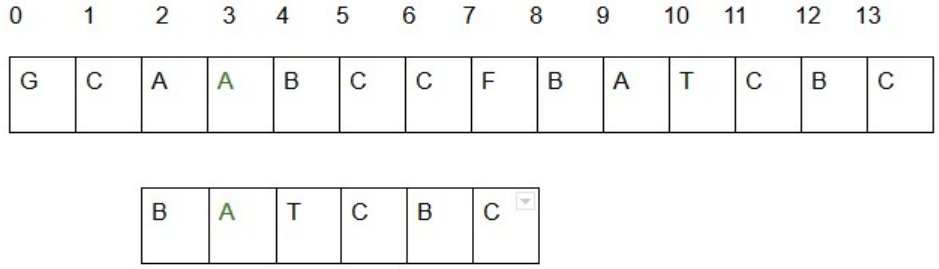
For example:

A text of size 14 and a pattern of pattern length 6 were taken. Whenever a mismatch occurs, the mismatch is marked with font colour of red. All the matches are marked with green.

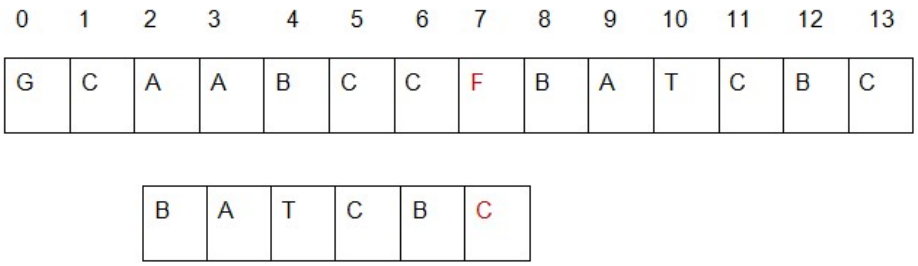
Since pattern was compared with the text from the rightmost character.

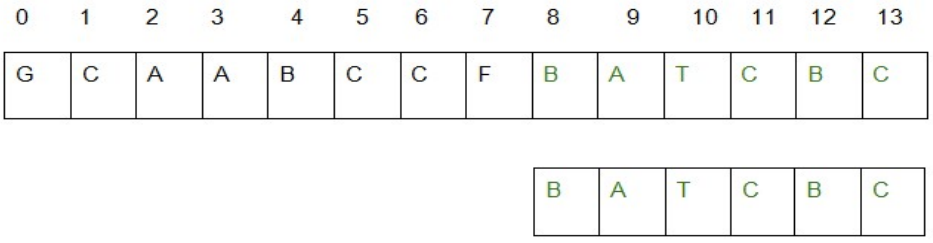


[5] "DAA Boyer-Moore Algorithm - javatpoint." https://www.javatpoint.com/daa-boyer-moore-algorithm. Accessed 29 May. 2019.



First ,there is a mismatch occurring at position 3 and the bad character will be A. now the last occurrence of the bad character (A) will be searched in the pattern and it can be found at position 1.now the pattern will be shifted twice so that the mismatch will become a match.





Now the pattern gets compared from the rightmost character of the pattern. The first character of the pattern mismatches with the first character of the text. There is a mismatch at position 7.the mismatch character “F” does not occur in the pattern before the position seven so now the pattern shall be shifted past the position seven. After the position of the pattern is shifted, again comparison takes place from the rightmost

character if the pattern. All the characters of the pattern match with the characters of the text so the pattern is said to be found and the index value of the position found will be returned as the output.

Time complexity of the algorithms :

In this investigation, time complexity is considered as a measure of efficiency. To find the most efficient algorithm among Rabin Karp algorithm and Boyer Moore algorithm, we will be comparing the running time of the algorithms when different variations of patterns are entered. Time complexity is the amount of time a code or an algorithm takes to run. The run time also called as execution time will be found for both the algorithms when different patterns are entered and this will be measured in nanoseconds. The best case of an algorithm occurs when a minimal amount of processing is needed due to the input being favourable to the optimal conditions of the algorithm. The worst case of an algorithm occurs when the entered input is not favourable to the optimal conditions of the algorithm and when the maximum number of processing is required.

In the following equations represents the length of the pattern and n represents the length of the text.

Time complexity of Boyer Moore algorithm[6]-

Best case: O(m/n) Worst case: O(mn)

Time complexity of Rabin Karp algorithm-

Best case: O(m+n) Worst case: O((n-m)m)

The worst case in Rabin karp algorithm occurs when all the individual characters of the entered pattern and all the individual characters of the text are the same, as the hash values of the pattern will be the same as the hash value of all the substrings.

Hypothesis :

As the number of times the pattern occurs in the text increases, the number of times the algorithm needs to iterate will increase so this might cause the runtime to increase as the number of occurrence of pattern in the text increases for both the algorithms. As the position of the pattern where it can be found in the text increases, the runtime might increase because the algorithm might have to compare more number of times to find the pattern if the pattern needed is placed in the long distance from the first character and the runtime might decrease if the pattern needed is placed in short distance from the first character.

Whenever a mismatch occurs, Boyer Moore has the benefit of skipping many characters of the input pattern. So as the input pattern length increases, the length of mismatch detected will also increase. This increase in the length of mismatch pattern found will cause advantage of an increase in the number of characters that can be skipped. Which means there is only fewer numbers of strings left to be compared when compared to the earlier text? In this case, the time taken to process the data will be reduced.

Whereas Rabin Karp algorithm does not have the ability to skip strings, instead it scans every character of the given string with the text. In this case the time taken to pre-

process the algorithm may consume comparatively more time. So this would extend the runtime of Rabin Karp algorithm. For the second component of the experiment, due to these reasons I hypothesize Boyer Moore algorithm might outrun Rabin Karp algorithm and as the pattern length increases, the run time might increase for Rabin Karp algorithm but Boyer Moore will take less time to process the same pattern.

Investigation :

In pursuance of this investigation, the experimentation will be divided into two components. As the base code for both the algorithms are readily available online (mentioned in appendix), the base code will be taken for both the algorithm and it will be modified to calculate the runtime. The dataset used for both the experiments was a passage which consisted of information taken from a computer science resource website and it had 8,247 characters including space in it and has 1344 words. The text used will be put into array so the position of the pattern will be mentioned as index numbers in the upcoming explanations. To ensure the obtained runtime is highly precise, three trials will be taken for both the algorithms throughout the experimentation.

The runtime was calculated by the following steps:

The start time is declared once the entered pattern is constructed into an array and before the search of pattern begins. The end time is declared after the search process is done and when the occurrences of the patterns are found in the text. Then the total time is obtained by subtracting the end time with the start time.

```
    }
}

/* Driver program to test above function */
public static void main(String[] args)
{
    String txt = "The world wide web started around 1990/91 as a system of servers connect
    String pat;
    pat = "to";
    long startTime = System.nanoTime();
    int q = 101; // A prime number
    search(pat, txt, q);
     long endTime   = System.nanoTime();
long totalTime = endTime - startTime;
System.out.println ( totalTime);
}


}
```

Figure 1.1 declaration of start time and end time in Rabin karp

```
84                      make sure that we get a positive shift.
85                      We may get a negative shift if the last
86                      occurrence  of bad character in pattern
87                      is on the right side of the current
88                      character. */
89                  s += max(1, j - badchar[txt[s+j]]);
90              }
91          }
92
93          /* Driver program to test above function */
94      public static void main(String []args) {
95
96          char txt[] = "The world wide web started around 1990/91 as a system of servers conne
97          char pat[] = "and".toCharArray();
98           long startTime = System.nanoTime();
99          search(txt, pat);
100         long endTime   = System.nanoTime();
101     long totalTime = endTime - startTime;
102     System.out.println ( totalTime);
103     }
104 }
105
```

Figure 1.2: declaration of start time and end time in Boyer Moore algorithm

Variable used in the experiment :

Dependent variable-The runtime will be the dependent variable used throughout the experiment for both the components. It will be calculated by measuring the difference of the start time and the end time of the program. It will be measured in the unit of nano seconds.

Independent variable:

Independent variable is the variable which will be changed in the experimentation. Since this investigation focuses on the pattern, three factors of the pattern will be changed throughout the experimentation. They are the number of times the pattern occurs in the text, the position of the pattern in the text and the length of the pattern.

In the first component of the experimentation, the number of times a pattern occurs and the position of the pattern will be the independent variables. Since I couldn't find enough resources about whether the occurrence of a pattern multiple times in a text will affect the runtime and whether the position of the pattern will affect the runtime of the algorithms, these variables were chosen as the independent variable for the first component of the experiment. In the second component of the experiment, pattern length would be the independent variable.

Controlled variables:

Since the running environment affects the run time, all the trials must be conducted on the same computer and multitasking must be avoided.

Variable	**Description**	**Specifications**
Computer and the operating system	I used Acer aspire Es 15 laptop and windows 10 OS was used.	Processor: Intel(R) Pentium(R) CPU N3710 @ 1.60GHz RAM memory: 2.00 GB where 1.83 GB was usable. System type : 64-bit operating system
The number of times the pattern occurred in the text	This was only used as a controlled variable in the second component of the experimentation.	Since the number of times the pattern occurred in the text affected the runtime taken, all the patterns chosen for the second component was chosen in such a way that it occurred only once throughout the

		entire text.
Same algorithm	Same algorithm (mentioned in the appendix) used for both the components of the experimentation.	
Same dataset	Same data set (text)were used for both the components of the experiment	
Integrated development environment used	Throughout the experimentation, all the programs will be run on same IDE.	Java: 1.8.0_171; Java HotSpot(TM) Client VM 25.171-b11 Runtime: Java(TM) SE Runtime Environment 1.8.0_171-b11 System: Windows 10 version 10.0 running on x86; Cp1252; en_IN (nb)

Experiment part 1 :

The first component of the experiment will focus on how the multiple occurrence of the pattern and position of the pattern in the text might affect the runtime. First experiment will be conducted using only single words. Words were taken with the pattern of length of 2-14, which were placed in different positions in the text. Few pairs of single words were taken which had same pattern length but belonged to different positions to check whether the change in position of string will affect the running time of the algorithms.

Test results of the experiment part 1 :

Pattern	Pattern length	Number of times the pattern was repeated in the text	Average time/nano seconds	
			Rabin Karp	Boyer Moore
the	3	103	7714357.33	45246319.7
as	2	45	5271476.33	4407513
and	3	44	4859098.67	3465778
be	2	33	4160444.67	3092209
web	3	29	4119374.67	2739551
that	4	25	4052700.7	2430848.33
use	3	23	3916265	2525992.67
are	3	18	3702704	2106357.67

information	11	11	3596840.67	1533135
from	4	10	3356547.33	1824150
database	8	9	3257663	1456752
allows	6	8	3061290.33	1439322.33
which	5	7	3030063	1589826.33
standards	9	6	3037917.67	511249
application	11	5	3085006.33	1236289.33
might	5	4	2946821.67	1368333.33
smarter	7	3	2857020.67	1243982.67
international	13	2	2743683	765270
authentication	14	1	2654117.67	1011932.67

Table 1: processed data of experimentation part-1, describes the relationship between times of occurrence of the pattern in the text and the average runtime consumed by both the algorithms.

From the first part of the experiment, it was evident that the position where the pattern is placed in the text does not affect the time taken for the algorithm to find the pattern but the number of times the pattern repeats in the text did affect the time taken. As the number of times the pattern occurred in the text increased, the run time also increased. The patterns which were placed in two different positions in the text, which had the same length, had similar runtimes. In the below table, the same font colour is used to denote the set of words which had the same times of occurrence and same pattern length. Only words which had the same times of occurrence and same pattern length

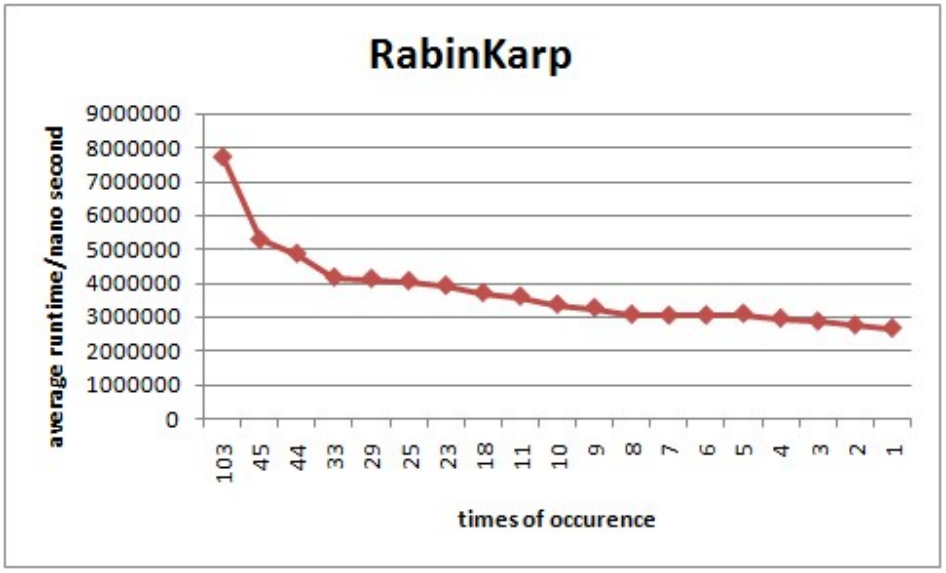
are extracted from the table because this way it was easy to find and show the relationship and compare.

number of occurrences of the pattern	Pattern	index number	Rabin Karp	Boyer Moore	pattern length
1	looked	1035	2601703.67	1144365.67	5
1	meaningful	1916	2668151	1029807	10
1	technology	2924	2622117	1048120	10
1	full time	3117	2631625	1076363.33	8
1	reliability	3298	2617330	1017280	11
1	manipulation	4222	2610970	1034974.33	12
1	credentials	4395	2611016.33	1043081.67	11
1	respective	4754	2689798	1050123	10
1	function	5038	2637637.33	1071606.33	8
1	navigational	5561	2646362.33	1015560	12
1	normal	7092	2635343	1166822.33	5

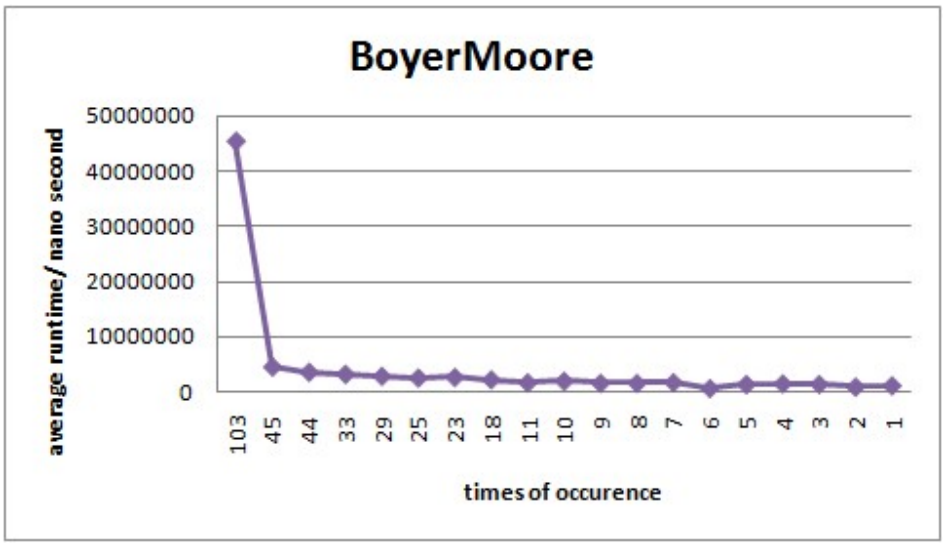
Table 2: processed data of experimentation part-1, describes the relationship between position of the pattern in the text and the average runtime consumed by both the algorithms.

In table For example if we take set of words like manipulation which has index number 4222-navigational which has index number 5561, And credentials which has index number 4395 and reliability which has index number 3295 . Both the set of words have the same pattern length but are placed in different positions in the text but both the set of words took a similar amount of runtime.

Index length is the position of the pattern in the text and it usually starts from 0.the words in table 1 was chosen based on their pattern length and the number of times they occur because these two factors will affect the running time of the algorithms.



Graph 1 - the graph showing the relationship between times of occurrence and average runtime for Rabin Karp algorithm.



Graph 2 - the graph showing the relationship between times of occurrence and average runtime for BoyerMoore algorithm.

The graph 1 and graph 2 clearly shows that as the times of occurrence of the pattern increased, the run time also increased for both the algorithms. In graph 2, there is a drastic increase from 45 to 103 because the interval between 45 and 103 is huge. When Boyer Moore is used, even small intervals have a more noticeable change in runtime as times of occurrence increase comparing to Rabin karp algorithm. In graph 1, the change from 45 to 103 is comparatively less because even though there is change in runtime of the change in not as vast as the change in Boyer Moore.

Experiment part 2 :

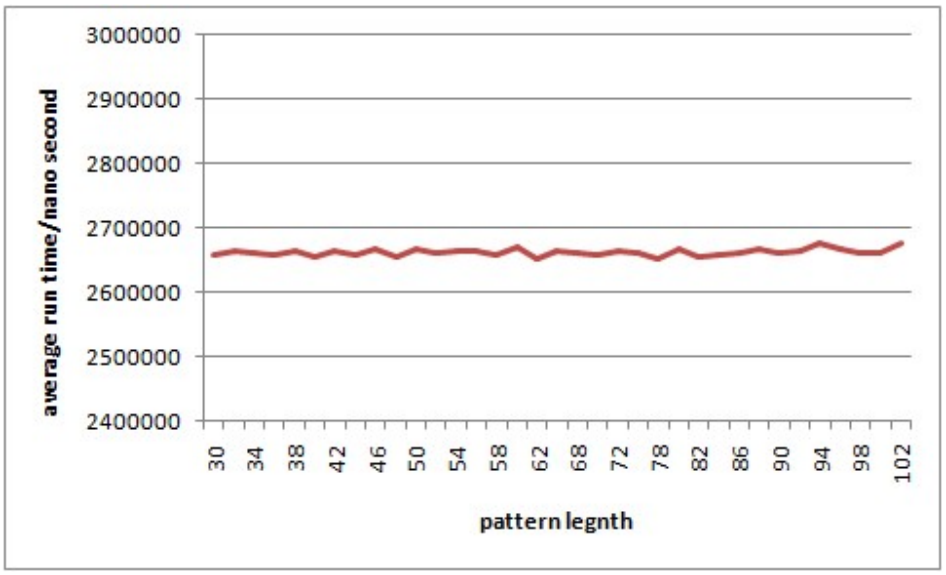
Second experiment will be done with a collection of words. After considering the results of the previous experiment, few changes were done to the variables. The position of the pattern was ignored since it did not cause major changes in the runtime. Set of words were taken for this experiment in the increasing pattern length of 30 to 102.

Test results of experiment part 2 :

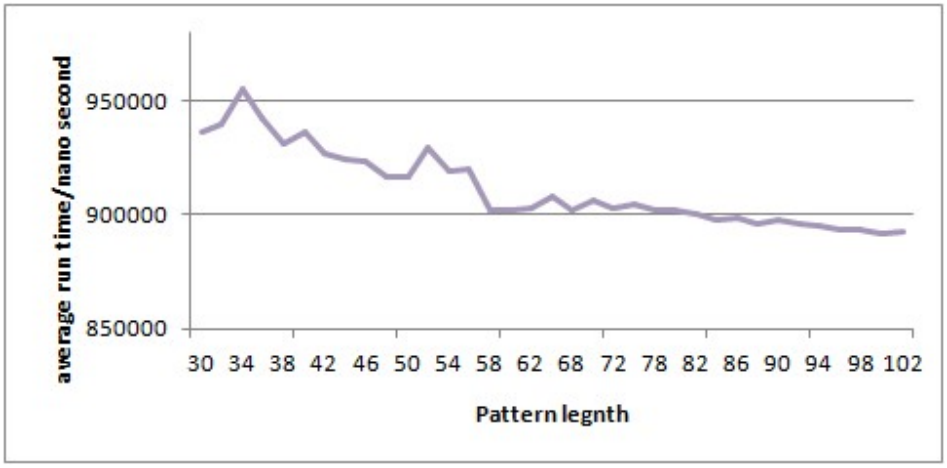
Pattern length	Rabin Karp	Boyer Moore
30	2658804.667	936716.6667
32	2665272.333	940072
34	2660616.333	955339.3333
36	2658961.333	942644.3333
38	2662373.333	931062.6667
40	2654472	936641.3333
42	2664516.333	927201.3333
44	2657593.667	924798.3333
46	2668402.333	923526
48	2654195.667	916860.6667
50	2667556	916292.6667
52	2661121	929743.6667
54	2663283.333	919067.6667
56	2664845	920325.6667
58	2656349.667	901712.6667
60	2669121	902219
62	2652258	902640.3333
64	2663879.333	907973.6667
68	2661470	902340

70	2657725	906037.6667
72	2662532	902774
74	2661623.333	904874
78	2651833	902163.6667
80	2667393	902481.3333
82	2653533	900329
84	2659185	898027
86	2660549.667	898555.3333
88	2666174.667	896339
90	2659447.667	898105.6667
92	2664940	896398.6667
94	2677279.333	894998.3333
96	2668155.333	893554.6667
98	2661439	893745.6667
100	2662282.333	892130.3333
102	2676289	892408

Table 3: processed data of experimentation part 2, describes the relationship between average runtime of both the algorithms and pattern length.



Graph 3 - Average runtime by pattern length graph for RabinKarp algorithm



Graph 4 - Average runtime by pattern length graph for Boyer Moore algorithm

From the results obtained from the second part of the experiment, it is understandable that as the pattern length increased, there was not much change in the runtime for RabinKarp algorithm. The minor fluctuation can be caused by processors and these fluctuations always exist even though the running environment was maintained constant for every trial , so the fluctuations seems like a change in trend in graph 3 but in real life

these fluctuations are very minute and they would not cause much change in the trend . The average running time continued to fluctuate around 2600000.

The change in pattern length did affect the time taken to process when Boyer Moore algorithm was used. The time taken for execution of the program decreased as the pattern length increased as shown in the graph 4.

Conclusion :

Returning to the research question ‘‘*to what extent the variation in the search pattern may affect the efficiency of Rabin Karp algorithm and Boyer Moore algorithm in terms of time complexity*’’, it was evident from the overall results that the Boyer Moore algorithm outperformed Rabin Karp algorithm in all the situations. The run time of Boyer Moore algorithm was much faster than the Rabin Karp algorithm throughout the experimentation. Half of my hypothesis was correct as pattern length increased, the runtime taken decreased for Boyer Moore algorithm. But in Rabin Karp what I hypothesized was wrong as there was no change in the trend when pattern length increased. I was clearly wrong about run time increasing as the position of the pattern in the text increases as the position of the pattern did not affect the run time for both the algorithms . Since it was well evident from my obtained results, I was right about my hypothesis of runtime increasing as the number of occurrence of pattern. Still Rabin karp is used in various plagiarism checking programs because it is said to be more suitable for the application when it comes to handling multiple patterns and also it is uses the unique hashing approach which is not used by other major algorithms. Boyer

Moore algorithm can be used when it comes to handling long patterns since it takes less runtime to find the pattern as the pattern length increases.

Further scope of the investigation :

As only two string searching algorithms of different approaches(classical approach and hashing approach) were taken in this investigation , for further investigation I want to take string searching algorithms from other two approaches (which are Suffix automata approach and Bit parallelism approach) and compare them to find the most efficient string searching algorithm with less average runtime. I also want to check whether the trend might change for different data types like binary alphabets and DNA alphabets and find the most suitable string algorithm for the different data types. Since this time only small data set was used for the text, I want to change the data set sizes and see how it would affect the runtime of the different string matching algorithms.

Limitations :

The investigation was carefully planned so that minimal amount of error will be produced so there weren't much limitation as far as I know. As different people might use different processors and different hard wares, the runtime might be different for different computers as the processor speed might differ but I believe this would not affect the trend of relationship found between the variables.

Bibliography :

[1]"Google Data Center FAQ & Locations | Data Center Knowledge." 17 Mar. 2017, https://www.datacenterknowledge.com/archives/2017/03/16/google-data-center-faq. Accessed 9 May. 2019.

"How many words are there in the English ... - Lexico.com." https://www.lexico.com/en/explore/how-many-words-are-there-in-the-english-language. Accessed 9 May. 2019.

"Character Definition - TechTerms." https://techterms.com/definition/character. Accessed 29 May. 2019.

"Rabin-Karp Pattern Searching Algorithm - OpenGenus IQ." https://iq.opengenus.org/rabin-karp-string-pattern-searching-algorithm/. Accessed 29 May. 2019.

"DAA Boyer-Moore Algorithm - javatpoint." https://www.javatpoint.com/daa-boyer-moore-algorithm. Accessed 29 May. 2019.

"How to calculate the running time of my program? - Stack Overflow." 6 Mar. 2011, https://stackoverflow.com/questions/5204051/how-to-calculate-the-running-time-of-my-program. Accessed 9 May. 2019.

"Rabin-Karp Algorithm for Pattern Searching - GeeksforGeeks." https://www.geeksforgeeks.org/rabin-karp-algorithm-for-pattern-searching/. Accessed 12 May. 2019.

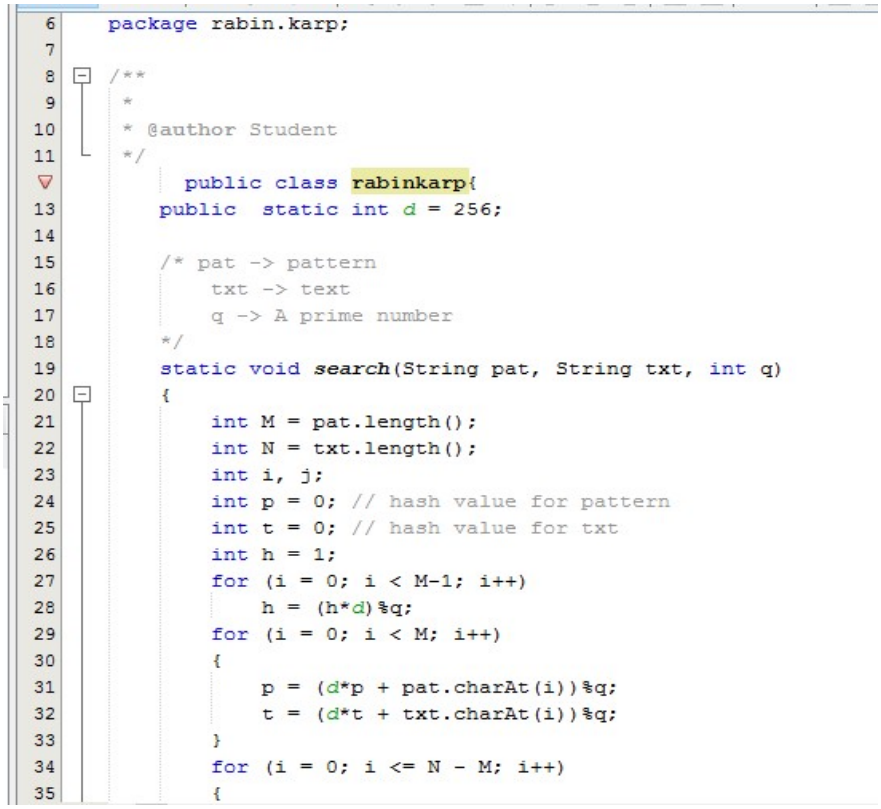
"Boyer Moore Algorithm for Pattern Searching - GeeksforGeeks." https://www.geeksforgeeks.org/boyer-moore-algorithm-for-pattern-searching/. Accessed 29 May. 2019.

"Option C - Web Science - cs-ib." https://www.cs-ib.net/topic/C-web-science.html. Accessed 12 May. 2019.

Appendix :

The code for both the algorithms were taken from a website called geeksforgeeks, and the idea and code to calculate the run time[7] was taken from a web forum called stackoverflow.com where many people across the world share their ideas regarding queries regarding programming.

Coding for Rabin Karp algorithm in java :[8]



```
6   package rabin.karp;
7
8   /**
9    *
10   * @author Student
11   */
       public class rabinkarp{
13     public  static int d = 256;
14
15     /* pat -> pattern
16         txt -> text
17         q -> A prime number
18     */
19     static void search(String pat, String txt, int q)
20     {
21         int M = pat.length();
22         int N = txt.length();
23         int i, j;
24         int p = 0; // hash value for pattern
25         int t = 0; // hash value for txt
26         int h = 1;
27         for (i = 0; i < M-1; i++)
28             h = (h*d)%q;
29         for (i = 0; i < M; i++)
30         {
31             p = (d*p + pat.charAt(i))%q;
32             t = (d*t + txt.charAt(i))%q;
33         }
34         for (i = 0; i <= N - M; i++)
35         {
```

[7] "How to calculate the running time of my program? - Stack Overflow." 6 Mar. 2011, https://stackoverflow.com/questions/5204051/how-to-calculate-the-running-time-of-my-program. Accessed 9 May. 2019.

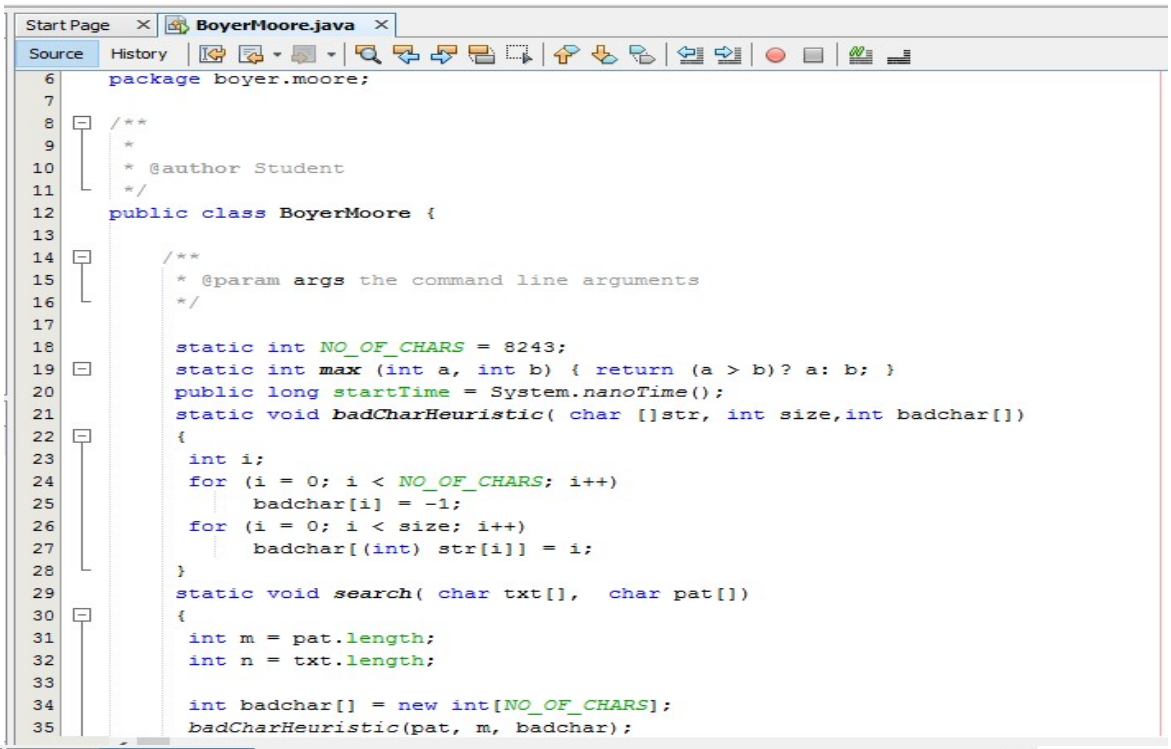
[8] "Rabin-Karp Algorithm for Pattern Searching - GeeksforGeeks." https://www.geeksforgeeks.org/rabin-karp-algorithm-for-pattern-searching/. Accessed 12 May. 2019.

Start Page | rabinkarp.java

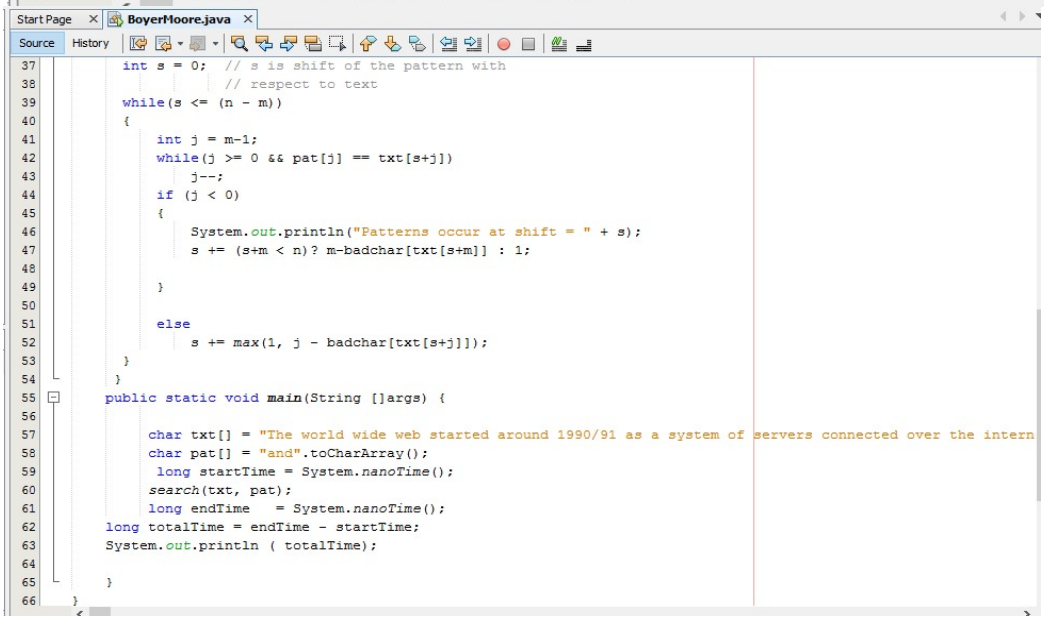
Source | History

```
36              if ( p == t )
37              {
38                  for (j = 0; j < M; j++)
39                  {
40                      if (txt.charAt(i+j) != pat.charAt(j))
41                          break;
42                  }
43                  if (j == M)
44                      System.out.println("Pattern found at index " + i);
45              }
46              if ( i < N-M )
47              {
48                  t = (d*(t - txt.charAt(i)*h) + txt.charAt(i+M))%q;
49                  if (t < 0)
50                  t = (t + q);
51              }
52          }
53      }
54      public static void main(String[] args)
55      {
56          String txt = "The world wide web started around 1990/91 as a system of servers connected over the interne
57          String pat;
58          pat = "to";
59          long startTime = System.nanoTime();
60          int q = 101; // A prime number
61          search(pat, txt, q);
62           long endTime   = System.nanoTime();
63      long totalTime = endTime - startTime;
64      System.out.println ( totalTime);
65      }
```

Coding for Boyer Moore algorithm in java:[9]



```
Start Page | BoyerMoore.java
Source | History
 6  package boyer.moore;
 7
 8  /**
 9   *
10   * @author Student
11   */
12  public class BoyerMoore {
13
14      /**
15       * @param args the command line arguments
16       */
17
18      static int NO_OF_CHARS = 8243;
19      static int max (int a, int b) { return (a > b)? a: b; }
20      public long startTime = System.nanoTime();
21      static void badCharHeuristic( char []str, int size,int badchar[])
22      {
23       int i;
24       for (i = 0; i < NO_OF_CHARS; i++)
25            badchar[i] = -1;
26       for (i = 0; i < size; i++)
27            badchar[(int) str[i]] = i;
28      }
29      static void search( char txt[],  char pat[])
30      {
31       int m = pat.length;
32       int n = txt.length;
33
34       int badchar[] = new int[NO_OF_CHARS];
35       badCharHeuristic(pat, m, badchar);
```



```
Start Page | BoyerMoore.java
Source | History
37       int s = 0;  // s is shift of the pattern with
38                   // respect to text
39       while(s <= (n - m))
40       {
41           int j = m-1;
42           while(j >= 0 && pat[j] == txt[s+j])
43               j--;
44           if (j < 0)
45           {
46               System.out.println("Patterns occur at shift = " + s);
47               s += (s+m < n)? m-badchar[txt[s+m]] : 1;
48
49           }
50
51           else
52               s += max(1, j - badchar[txt[s+j]]);
53       }
54      }
55      public static void main(String []args) {
56
57          char txt[] = "The world wide web started around 1990/91 as a system of servers connected over the intern
58          char pat[] = "and".toCharArray();
59           long startTime = System.nanoTime();
60          search(txt, pat);
61          long endTime   = System.nanoTime();
62      long totalTime = endTime - startTime;
63      System.out.println ( totalTime);
64
65      }
66  }
```

[9] "Boyer Moore Algorithm for Pattern Searching - GeeksforGeeks." https://www.geeksforgeeks.org/boyer-moore-algorithm-for-pattern-searching/. Accessed 29 May. 2019.

Data set used: [10]

The data set used was taken from the cited website.

The world wide web started around 1990/91 as a system of servers connected over the internet that deliver static documents, which are formatted as hypertext mark-up language (HTML) files, which support links to other documents, but also multimedia as graphics, video or audio. In the beginnings of the web, these documents consisted mainly of static information and text, where multimedia was added later. Some experts describe this as a read-only web, because users mostly searched and read information, while there was little user interaction or content contribution. However, the web started to evolve into the delivery of more dynamic documents, enabling user interaction or even allowing content contribution. The appearance of blogging platforms as Blogger in 1999 gives a time mark for the birth of the Web 2.0. Continuing the model from before, this would be the evolution to a read-write web. This opened new possibilities and lead to new concept as blogs, social networks or video-streaming platforms. Web 2.0 might also be looked at from the perspective of the websites themselves evolving in more dynamic and feature-rich. For instance, improved design, JavaScript and dynamic content loading could be considered Web 2.0 features. The internet and thus the World Wide Web is constantly developing and evolving into new directions and while the changes described for the Web 2.0 are clear to us today, the definition for the Web 3.0 is not definitive yet. Continuing the read to read-write description form earlier, it might be argued that the Web 3.0 would be the read-write-execute web. One interpretation of this

[10] "Option C - Web Science - cs-ib." https://www.cs-ib.net/topic/C-web-science.html. Accessed 12 May. 2019.

is that the web enables software agents to work with documents by using semantic mark-up. This allows for smarter searches and the presentation of relevant data fitting into context. This is why Web 3.0 is sometimes called the semantic executive web. It is about user input becoming more meaningful, more semantic, by users giving tags or other kinds of data to their document, that allow software agents to work with the input, e.g. to make it more searchable. The idea is to be able to better connect information that is semantically connected. However, it might also be argued that the Web 3.0 is what some people call the Internet of Things, which is basically connecting every day devices to the internet to make them smarter. In some way, this also fits the read-write-execute model, as it allows the user to control a real life action on a device over the internet. Either way, the web keeps evolving and the following image provides a good overview and an idea where the web is heading to. However, it might also be argued that the Web 3.0 is what some people call the Internet of Things, which is basically connecting every day devices to the internet to make them smarter. It has been founded in 1946 and since then has published over 21000 international standards regarding aspects of technology and manufacturing. The members are from 163 countries including 3 368 technical bodies that help standards to be developed. In addition, the organization has over 135 people working fulltime at the central in Geneva. Experts of the same field work together to develop standards and these are settled on through a consensus process. These standards ensure safety, reliability and quality for products and services, while also providing a common denominator for different processes to communicate, e.g. for technologies. Sites that include server-side programming as well, usually to retrieve content dynamically from a database. This allows for data processing

on the server and allows for much more complex applications. In some way, this also fits the read-write-execute model, as it allows the user to control a real life action on a device over the internet. Either way, the web keeps evolving and the following image provides a good overview and an idea where the web is heading to. ISO is the International Organization of Standardization, an independent, non-governmental organization that develops and publishes international standards. Website logic that runs on the server. Common tasks include the processing of search queries, data retrieval from a database and various data manipulation tasks. Good examples are online-shops, where items are displayed based on a search query. Once the user decides to buy an item, server-side scripts check user credentials and make sure that the shop receives the order. Cookies are small files stored on a user computer. They hold data specific to a website or client and can be accessed by either the web server or the client computer. Cookies contain data values such as first-name and last-name. Once the server or client computers have read the cookie through their respective codes, the data in the cookie can be retrieved and used for a website page. Cookies are created usually when a new web page is loaded. Disabling cookies on your computer will abort the writing operation that creates cookies. However, some sites require cookies in order to function. Cookies are used to transport information from one session on a website to another. They eliminate the use of server machines with huge amounts of data storage, since cookies are more efficient and smaller. A database is an organized collection of data, which allows retrieving specific data easily based on queries. Data are usually organized in a way that allows the application to find data easily. There are different logic models of how to organize data in a database, e.g.

relational models, object models, navigational models and more. A database is access (in order to retrieve data, update them, administration) through a database management system (DBMS), such as for example MySQL, PostgreSQL, MongoDB, etc. . . . These systems usually differ in the database model that they use. XML is a flexible way to structure data and can therefore be used to store data in files or to transport data. It allows data to be easily manipulates, exported, or imported. This way, websites can also be designed independent from the data content. Example uses of XML are RSS feeds, where it is used to store data about a feed. This is a standard protocol for web servers to execute console programs (applications that run from the command line) in order to generate dynamic websites. It implements an interface for the web server (as in the software) to pass on user information, e.g. a query, to the application, which can then process it. This passing of information between the web server and the console application is called the CGI. Thanks to CGI, a variety of programming languages such as Perl, Java, C or C++ can be used, which allow for fast server-side scripting. The surface web is the part of the web that can be reached by a search engine. For this, pages need to be static and fixed, so that they can be reached through links from other sites on the surface web. They also need to be accessible without special configuration. Examples include Google, Face book, YouTube, etc. The deep web is the part of the web that is not searchable by normal search engines. Reasons for this include proprietary content that requires authentication or VPN access, e.g. private social media, emails; commercial content that is protected by pay walls, e.g. online news papers, academic research databases; personal information that is protected, e.g. bank information, health records; dynamic content. Dynamic content is usually a result of

some query, where data are fetched from a database Interoperability can be defined as the ability of two or more systems or components to exchange information and to use the information that has been exchanged. In order for systems to be able to communicate they need to agree on how to proceed and for this reason standards are necessary. Lossy compression or irreversible compression is the class of data encoding methods that uses inexact approximations and partial data discarding to represent the content. These techniques are used to reduce data size for storage, handling, and transmitting content. Lossless data compression algorithms usually exploit statistical redundancy to represent data without losing any information, so that the process is reversible.

Raw data collected during the experimentation :

In the following tables , first row of each pattern will be the runtime taken by Rabin Karp algorithm and second row of each pattern will be the runtime taken by Boyer Moore.

experimentation part 1:

pattern	pattern length	number of times the pattern was repeated in the text	Trial 1	Trial 2	Trial 3	Average time/nano seconds

the	3	103	7821889	7723315	7597868	7714357.333
			111197841	11285445	13255673	45246319.67
as	2	45	5341719	5319317	5153393	5271476.333
			4709006	4760364	3753169	4407513
and	3	44	4843789	4889872	4843635	4859098.667
			3421454	3498898	3476982	3465778
be	2	33	4162151	4174952	4144231	4160444.667
			3118894	3113134	3044599	3092209
web	3	29	4163434	4156703	4037987	4119374.667
			2721254	2762703	2734696	2739551
that	4	25	3937810	4285682	3934610	4052700.667
			2400288	2486848	2405409	2430848.333
use	3	23	4295920	4052509	3400366	3916265
			2573407	2503003	2501568	2525992.667
are	3	18	3733793	3742754	3631565	3702704
			2026993	2014832	2277248	2106357.667
user	4	11	3599693	3541782	3540657	3560710.667
			1839412	1809332	1830325	1826356.333
even	4	1	2699827	2597906	2634371	2644034.667
			1331097	1344539	1380380	1352005.333
static	5	3	2769436	3040635	3241915	3017328.667

			1290294	1269014	1266054	1275120.667
mostly	6	1	2689585	2609427	2579345	2626119
			1135728	1159071	1155873	1150224
from	4	10	3358092	3375373	3336177	3356547.333
			1838041	1834011	1800398	1824150
blogs	5	1	2675505	2573585	2639508	2629532.667
			1235578	1250298	1215395	1233757
might	5	4	2941915	2936154	2962396	2946821.667
			1399427	1378946	1326627	1368333.333
looked	6	1	2503667	2601445	2699999	2601703.667
			1168034	1122931	1142132	1144365.667
design	6	2	2687540	2801791	2709900	2733077
			1235573	1240058	1269413	1248348
argued	6	3	2872791	2884950	2899030	2885590.333
			1284706	1262305	1209182	1252064.333
smarter	7	3	2854085	2819855	2897122	2857020.667
			1244836	1244196	1242916	1243982.667
meaningful	10	1	2675504	2640644	2688305	2668151
			1025801	1035899	1027721	1029807
connect	7	5	2896328	2873287	2800808	2856807.667
			1326758	1254331	1398088	1326392.333

internet	8	6	3052652	3060332	3099207	3070730.333
			1328777	1331738	1334937	1331817.333
following	9	2	2757428	2743347	2751668	2750814.333
			1002345	1102344	1002316	1035668.333
connecting	10	2	2768866	2735606	2769507.999	2757993.333
			1109005	1140367	1109645	1119672.333
technology	10	1	2686384	2594223	2585744	2622117
			1019400	1068120	1056840	1048120
fulltime	8	1	2672884	2634387	2587604	2631625
			1083404	1071243	1074443	1076363.333
reliability	11	1	2580934	2580624	2690432	2617330
			1015600	1019400	1016840	1017280
application	11	5	3091196	3088635	3075188	3085006.333
			1239573	1238292	1231003	1236289.333
applications	12	2	2636947	2645345	2630392	2637561.333
			1075724	1071243	1072763	1073243.333
international	13	2	2716951	2761754	2752344	2743683
			764203	764843	766764	765270
manipulation	12	1	2681264	2577423	2574223	2610970
			1036681	1034121	1034121	1034974.333
credentials	11	1	2610706	2611109	2611234	2611016.333

			1041802	1044361	1043082	1043081.667
respective	10	1	2571664	2601105	2896625	2689798
			1052043	1051403	1046923	1050123
function	8	1	2581264	2575504	2576144	2577637.333
			1094526	1060247	1060046	1071606.333
session	7	1	2571663	2672944	2585104	2609903.667
			1032046	1028850	1030126	1030340.667
allows	6	8	3036969	3052330	3094572	3061290.333
			1443590	1455750	1418627	1439322.333
logic	5	2	2735666	2751027	2743987	2743560
			1264533	126373	1261334	884080
flexible	8	1	2576143	2571663	2774223	2640676.333
			1066123	1068683	1067683	1067496.333
manipulate	10	1	2592144	2790225	2573583	2651984
			1043081	10462817	1043052	4182983.333
console	7	2	2820637	2729907	2635027	2761857
			1074928	1073009	1073008	1073648.333
which	5	7	3026883	3028163	3035143	3030063
			1594946	1584066	1590467	1589826.333
normal	6	1	2570383	2568463	2767183	2635343
			1064689	1067249	1068529	1066822.333

authentication	14	1	2789585	2589585	2583183	2654117.667
			1014280	1010439	1011079	1011932.667
navigational	12	1	2674637	2676146	2588304	2646362.333
			1019400	1011080	1016200	1015560
database	8	9	3283051	3255530	3234408	3257663
			1466145	1423245	1480866	1456752
system	6	5	3069280	2973275	2974555	3005703.333
			1366299	1367580	1377180	1370353
standards	9	6	3024477	3060319	3028957	3037917.667
			120135	122695	1290917	511249
communicate	10	2	2785589	2634387	2732466	2717480.667
			1095031	1098806	1096991	1096942.667
information	11	11	3554500	3664491	3571531	3596840.667
			1521828	1525668	1551909	1533135
reversible	10	2	2695508	2732588	2730546	2719547.333
			1080843	71089163	1084043	24418016.33

Experimentation part 2 :

Pattern taken	Pattern length	Trial 1	Trial 2	Trial 3	Average runtime
non-governmental	30	2653481	2683032	2639901	2658804.667

organization					
		934590	936670	938890	936716.6667
he data in the cookie can be retrieved	32	2616393	2683032	2696392	2665272.333
		938897	940211	941108	940072
every day devices to the interne	34	2638891	2644632	2698326	2660616.333
		934697	949212	982109	955339.3333
where items are displayed based on	36	2659807	2657990	2659087	2658961.333
		942338	940235	945360	942644.3333
through a database management system	38	2659801	2660443	2666876	2662373.333
		931869	930211	931108	931062.6667
variety of programming languages such as	40	2662552	2687672	2613192	2654472
		939974	930615	939335	936641.3333
are usually organized in a way that allows	42	2651333	2674325	2667891	2664516.333
		928695	926774	926135	927201.3333
since cookies are more	44	2684477	2626392	2661912	2657593.667

efficient and smaller					
		928055	923174	923166	924798.3333
be used to store data in files or to transport	46	2635661	2673333	2696213	2668402.333
		922689	923433	924456	923526
where items are displayed based on a search query	48	2592522	2688153	2681912	2654195.667
		916534	916234	917814	916860.6667
information and to use the information that has be	50	2682131	2674544	2645993	2667556
		916534	916344	916000	916292.6667
and thus the world wide web is constantly developing	52	2608889	2681118	2693356	2661121
		916375	936745	936111	929743.6667
perspective of the websites themselves evolving in	54	2682341	2647270	2660239	2663283.333
		902427	924188	930588	919067.6667
ndards regarding aspects of technology and manufacturing	56	2699909	2618318	2676308	2664845
		930042	930588	900347	920325.6667
software agents to work with	58	2694639	2668887	2605523	2656349.667

documents by using semantic m					
		901043	902203	901892	901712.6667
by users giving tags or other kinds of data to their documen	60	2691279	2622418	2693666	2669121
		901099	902677	902881	902219
hile there was little user interaction or content contribution	62	2620456	2689919	2646399	2652258
		902427	901787	903707	902640.3333
between the webserver and the console application is called the	64	2653439	2663439	2674760	2663879.333
		903707	917787	902427	907973.6667
of servers connected over the internet that deliver static documents	68	2681124	2621123	2682163	2661470
		902543	901256	903221	902340
eliminate the use of server machines with huge amounts of data storage	70	2683445	2655489	2634241	2657725
		915534	902345	900234	906037.6667

at from the perspective of the websites themselves evolving in more dynamic and feature-rich	72	2590798	2688345	2708453	2662532
		904967	903221	900134	902774
are used to transport information from one session on website to another	74	2684389	2633040	2667441	2661623.333
		900657	910023	903942	904874
cookies on your computer will abort the writing operation that creates cookies	78	2607513	2687032	2660954	2651833
		902533	902613	901345	902163.6667
while also providing a common denominator for different processes to communicate	80	2633680	2679932	2688567	2667393
		902113	903220	902111	902481.3333
xed, so that they can be reached through links from other sites on the surface web	82	2696399	2623880	2640320	2653533
		900231	900432	900324	900329
structure data and can	84	2633040	2699398	2645117	2659185

therefore be used to store data in files or to transport data					
		898587	890067	905427	898027
allows for smarter searches and the presentation of relevant data fitting into context	86	2638721	2678289	2664639	2660549.667
		899088	898234	898344	898555.3333
ess data compression algorithms usually exploit statistical redundancy to represent data	88	2677654	2644326	2676544	2666174.667
		897958	896592	894467	896339
his allows for data processing on the server and allows for much more complex applications	90	2633680	2788901	2555762	2659447.667
		897889	898773	897655	898105.6667
communicate they need to agree on how to proceed and for this reason standards are necessary	92	2694352	2601038	2699430	2664940

		896778	896890	895528	896398.6667
irreversible compression is the class of data encoding methods that uses inexact approximation	94	2632114	2603821	2795903	2677279.333
		895661	894333	895001	894998.3333
include server-side programming as well, usually to retrieve content dynamically from a database	96	2698238	2615575	2690653	2668155.333
		893999	892330	894335	893554.6667
these documents consisted mainly of static information and text, where multimedia were added later	98	2643729	2689594	2650994	2661439
		893465	893440	894332	893745.6667
of the same field work together to develop standards and these are settled on through a consensus pro	100	2690032	2600075	2696740	2662282.333
		892243	890224	893924	892130.3333
the same field work together to	102	2730075	2693361	2605431	2676289

develop standards and these are settled on through a consensus process					
		893541	892111	891572	892408