ISTANBUL TECHNICAL UNIVERSITY

COMPUTER ENGINEERING DEPARTMENT

BLG 374E PROJECT REPORT

GROUP MEMBERS:

150200059	:	Mehmet Ali Balıkçı
150210061	:	Metin Ertekin Küçük
150200097	:	Mustafa Can Çalışkan
150200016	:	Yusuf Şahin
150200066	:	Yusuf Emir Sezgin

SPRING 2024

Contents

1 INTRODUCTION

A fundamental topic in computer science, pattern searching has many applications, such as bioinformatics, data mining, and text processing. In these jobs, efficient algorithms are essential because they have a big impact on the scalability and performance of programs. The Knuth-Morris-Pratt (KMP) method and the Rabin-Karp algorithm are two popular algorithms for pattern recognition. By comprehending and contrasting different algorithms, one can gain important insights into their advantages and disadvantages and ascertain which strategy is best for a certain pattern-searching activity.

2 METHODOLOGY

The foundation of the KMP method is the idea of a “failure function,” which is employed to ascertain the longest possible suitable suffix of the pattern that corresponds to a pattern prefix. This information makes the algorithm extremely efficient by preventing pointless comparisons when looking for the pattern in the text. To create the failure function, the algorithm preliminary procedures the pattern. This function is then employed in the search phase.

In contrast, the Rabin-Karp algorithm compares the pattern with text substrings using hashing. It calculates the pattern’s hash value and contrasts it with the text’s substrings’ hash values. The algorithm runs a thorough comparison to verify the match if the hash values match. If not, it advances to the following substring. Because the Rabin-Karp algorithm uses hashing, it can compare patterns with substrings quickly and is therefore appropriate in situations where hashing can be implemented effectively.

In this study, the theoretical principles, implementation specifics, and performance characteristics of the KMP and Rabin-Karp algorithms will be compared. To fully compare these two pattern-searching algorithms, we will examine their time complexity, space complexity, and practical performance in a range of contexts.

3 AIM OF THE PROJECT

This report’s goal is to compare and contrast the Brute-Force, KMP, and Rabin-Karp algorithms, emphasizing each algorithm’s advantages and disadvantages under various conditions. The paper will go over the theoretical underpinnings of both algorithms, give a synopsis of how they were implemented, and evaluate how well they performed using a range of measures, including time and space complexity.

4 ANALYSIS

4.1 THEORETICAL ANALYSIS

Test results have shown that the KMP Algorithm, boasting a time complexity of O(m + n) and a space complexity of O(m), performs efficiently when handling large texts and small patterns. Conversely, the Rabin-Karp Algorithm, with a time complexity of O(m * n) and constant space complexity O(1), is better suited for scenarios involving small texts and large patterns. The time complexity of brute force is O((n - m + 1) * m) and the space complexity is O(1).

4.2 PRACTICAL ANALYSIS

Figure 1 and 2 illustrate the varying number of collusions with different hash functions and different moduli for the Rabin-Karp algorithm. As can be seen from the figures, as the modulus increases, all three hash types show approximately the same performance on average.

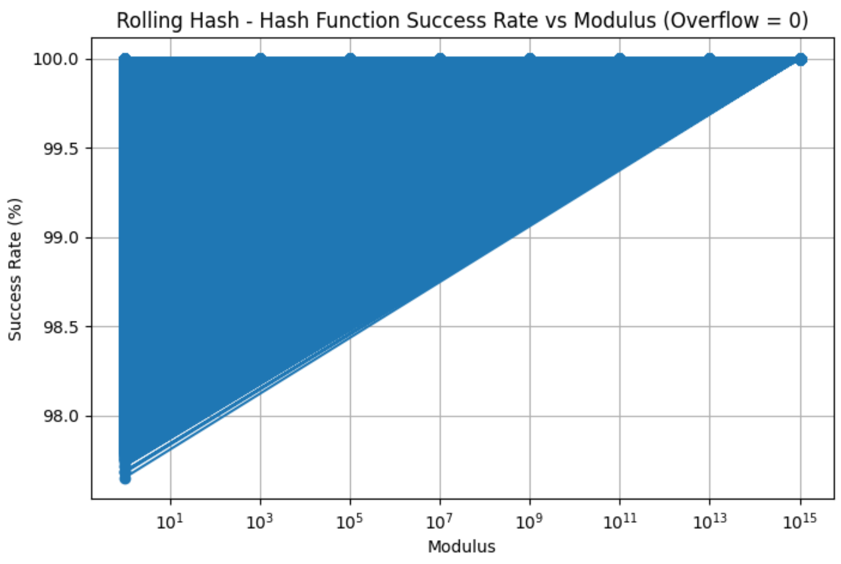

Figure 1: Rolling Hash Function Success Rate vs Modulus

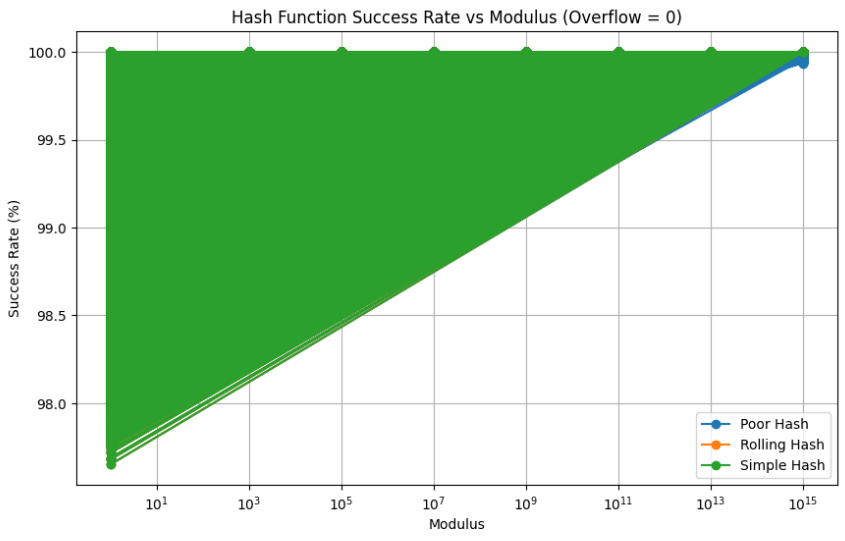

Figure 2: Hash Function Success Rate vs Modulus (Overflow = 0)

As can also be seen from Figure 3, as the length of the text and pattern increases, the execution time of the brute force approach significantly increases compared to the KMP and Rabin-Karp algorithms.

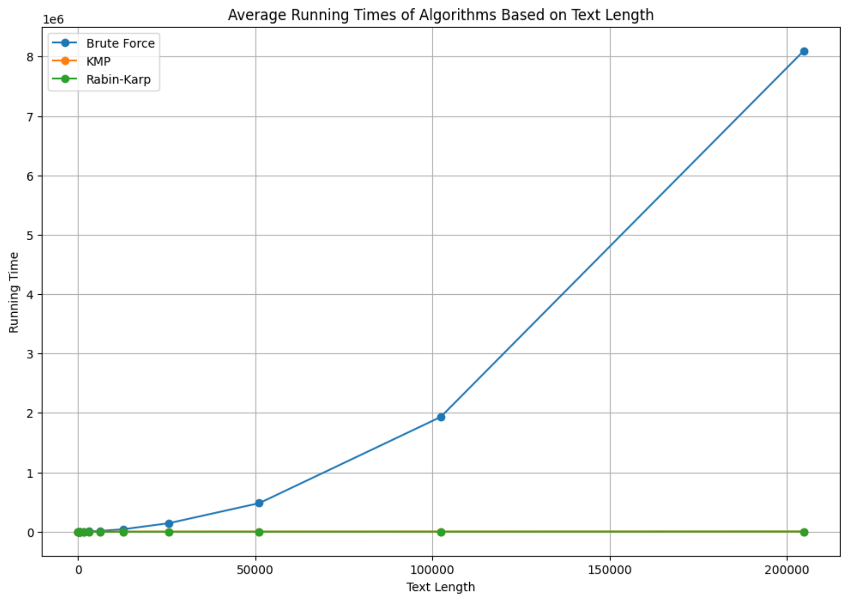

Figure 3: Average Running Times of Algorithms Based on Text Length

When the brute force approach is removed and re-scaled, it can be observed in Figure 4 that the Rabin-Karp algorithm slightly outperforms KMP.

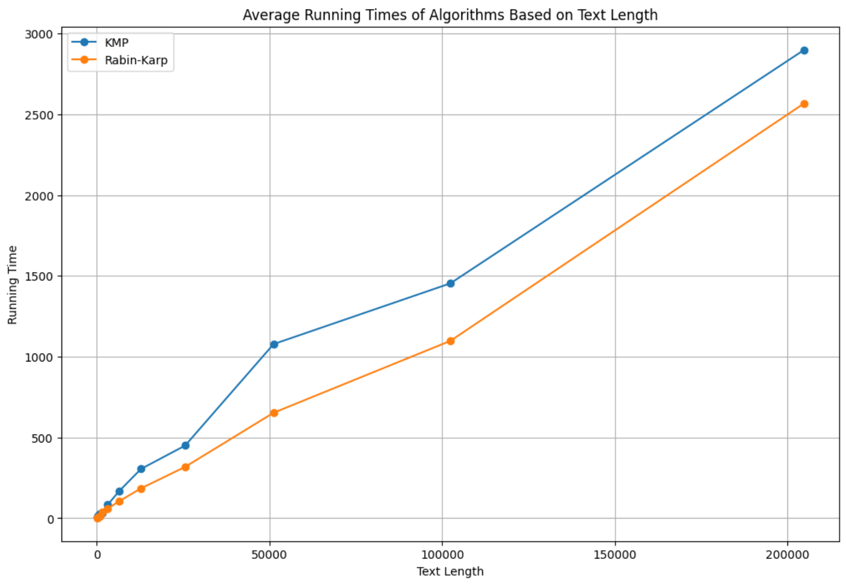

Figure 4: Average Running Times of Algorithms Based on Text Length

- Average Execution Time:

  Brute Force: The average execution time of the Brute Force algorithm for pattern searching was measured at 892463.08μs.

  KMP Algorithm: The average execution time of the KMP algorithm for pattern searching was measured at 544.42μs.

  Rabin-Karp Algorithm: The average execution time of the Rabin-Karp algorithm for pattern searching was measured at 421.25μs.

Based on particular criteria, these results offer insightful information about the performance characteristics of the KMP and Rabin-Karp algorithms and can be used to choose the best method for pattern-searching jobs.

5 CONCLUSION

In conclusion, the comparison between the Knuth-Morris-Pratt (KMP) algorithm and the Rabin-Karp algorithm for pattern searching reveals distinct performance characteristics and suitability for different scenarios.

The KMP algorithm, leveraging its time complexity of O(m + n) and space complexity of O(m + n), proves highly efficient when dealing with large texts and small patterns. On the other hand, the Rabin-Karp algorithm, with its average time complexity of O(m + n), worst time complexity of O(m * n), and constant space complexity O(1), excels in scenarios involving small texts and large patterns, thanks to its hashing-based approach.

Practical analysis, including benchmark tests and performance evaluations, further confirms these theoretical findings. Figures demonstrating the execution times of both algorithms illustrate that while the brute force approach lags significantly behind, the Rabin-Karp algorithm slightly outperforms the KMP algorithm, particularly as the size of the text and pattern increases.

Moreover, the analysis of average execution times highlights the efficiency of both algorithms, with the Rabin-Karp algorithm generally exhibiting faster performance compared to the KMP algorithm.

These findings provide valuable insights for practitioners in selecting the most appropriate algorithm based on specific criteria such as text and pattern sizes, emphasizing the importance of understanding the underlying principles and performance characteristics of each algorithm in pattern-searching tasks.

REFERENCES