String Matching Algorithms: Analysis Report and Our Journey

1. Introduction

This report explains the implementation and analysis of two string matching algorithms: Boyer-Moore and a custom algorithm called GoCrazy. The aim of this study is to create efficient string matching algorithms and develop a system to select the best algorithm for different situations. The GoCrazy algorithm uses Character Density-Oriented Search (CDGS) to improve the search process by focusing on character density in text and patterns. The Boyer-Moore algorithm is particularly efficient for large texts and alphabets. It uses two main rules to speed up the string matching process: the Bad Character Rule and the Good Suffix Rule. On the other hand, the GoCrazy algorithm is a custom solution designed to process specific patterns and texts more efficiently using the CDGS technique. selection process.

2. Implementation of the Boyer-Moore Algorithm

The Boyer-Moore algorithm is among the most efficient string matching algorithms, especially in cases of large alphabets and long texts. It takes advantage of two rules in a very systematic way: the Bad Character Rule and the Good Suffix Rule. The Bad Character Rule states that in the case of a mismatch, the shift of the pattern should be such that the most recently matched character of the pattern aligns with the last occurrence of the same character. In contrast, the Good Suffix Rule shifts the pattern such that the matched suffix of the pattern aligns with the suffix of the matched portion of the text in case of a mismatch. The above task is achieved by preprocessing the pattern to create two tables. The Bad Character Table stores the last occurrence of each character in the pattern so that on occurrence of a mismatch, the algorithm can make an effective shift. The Good Suffix Table helps in determining the best shift by considering the matched portion of the pattern. This enables the algorithm to bypass certain comparisons. Additionally, it also covers edge cases for an empty pattern and patterns longer than text.

3. GoCrazy Algorithm Implementation-Character Density Guided Search (CDGS)

GoCrazy is a simplification of the string matching algorithm to find a certain pattern in a text. Unlike other algorithms, such as Knuth-Morris-Pratt or Boyer-Moore, which guarantee correctness, this one focuses on ease of use and average efficiency. It borrows ideas from right-to-left comparison techniques and introduces additional rules that restrict superfluous character checks during the search process.The algorithm picks up the very last character of the pattern as a reference point and starts aligning it with positions in the text. This method helps the algorithm get rid of many non-matching positions quickly since in most cases mismatches at the end of the pattern mean an unsuccessful match. In the case of a matching reference character, it further investigates the character right before it in pursuit of reducing the likelihood of a full comparison when a match is unlikely.

The algorithm then checks the entire pattern from right to left only when both the reference character and the preceding character in the pattern match. When all characters align, it records the starting index as one of the successful matchings. In either case, a match is found or not; the algorithm traverses the text by using a set skip determined by half the length of the pattern. This makes the search quite fast but might also skip matched text.

4. Implementation of the StudentPreAnalysis Class

The StudentPreAnalysis class is designed to implement a strategy that will help in the selection of the most efficient algorithm based on the characteristics of the text and the pattern. It will make a decision based on many factors that will determine the best performance an algorithm might have in a given situation. Patterns of very short length, ≤ 3, would directly use the Naive algorithm since it's effective in small searches. On the other hand, the KMP algorithm would be preferred if the pattern has repeated prefixes or has low entropy, because it handles such patterns quite efficiently. The RabinKarp would be preferred in the case of a long pattern on large texts since it is really good in handling such a scenario, especially for big data. In general cases, the algorithm would default to BoyerMoore because of its all-around efficiency: it is versatile enough to handle any kind of situation between text and pattern. The idea of this pre-analysis logic is to dynamically select the best option by considering the properties of the text and pattern, which would help in achieving the best performance of the string matching process.

5. Research and Resources

Boyer-Moore Algorithm:

To understand the fundamental principles of the Boyer-Moore algorithm and how it works, we watched Ben Lengead's YouTube videos. Additionally,we examined optimization strategies for Boyer-Moore by looking at GeeksforGeeks, a site that offered detailed information on both the Bad Character Rule and the Good Suffix Rule. Thanks to these resources, we understood the basic structure of the algorithm. During implementation, ChatGPT was effective in further clarifying these concepts and solving the problems we encountered while correctly applying the rules in our code.

GoCrazy:

For the GoCrazy algorithm, we adopted the principles of Character Density Guided Search (CDGS) from various research articles on string matching. This approach focuses on efficiently skipping over sections of text based on specific conditions like character repetition, which significantly improves search efficiency. We also explored the possibility of blending multiple algorithms for hybrid optimization, drawing insights from Medium articles on hybrid algorithm selection and optimization techniques. These articles helped us understand how combining different search algorithms could improve performance, especially in complex or non-standard search scenarios. Throughout the process, ChatGPT played a key role in helping us integrate the CDGS principles into the GoCrazy algorithm. It provided valuable guidance on character density-based search strategies and helped us overcome challenges, particularly in managing repeated characters, ensuring the algorithm’s efficiency and correctness.

PreAnalysis:

We developed our pre-analysis strategy based on performance comparisons of different string matching algorithms. We reviewed GeeksforGeeks and ByteQuest, as well as the Computer Concepts and Abdulbari YouTube channels, which helped us understand how algorithms perform in different situations. With this

information, we created a strategy that selects the most efficient algorithm by considering the pattern, text, and algorithm efficiency. We also improved our strategy by reviewing hybrid algorithm approaches in Medium articles. ChatGPT helped us improve our pre-analysis method, contributing to a more accurate and optimized algorithm selection process.

6. Conclusion

The Boyer-Moore and GoCrazy algorithms, together with the StudentPreAnalysis system, create an efficient approach for string matching. The Boyer-Moore algorithm speeds up the search by using the Bad Character and Good Suffix rules, making it effective for large texts and alphabets.The GoCrazy algorithm serves as an example of how optimizations such as pivot-based checking, early rejection, and fixed skipping enable better average performance for string matching tasks. However, due to its dependence on a fixed skip approach without any formal correctness proof, it cannot be guaranteed to detect every possible occurrence of a given pattern. In this regard, it is best regarded as an experimental or educational tool rather than as a substitute for reliable, theoretically grounded methods of string matching. The StudentPreAnalysis system improves performance by choosing the best algorithm based on factors like text and pattern length, character repetition, and alphabet size. This helps ensure the most efficient algorithm is used, reducing unnecessary work.

7.OUR JOURNEY

This assignment definitely taught us a lot because algorithms are not an easy lesson. In the beginning, we had a tough time grasping the concept of the Boyer-Moore algorithm, particularly the Bad Character rule and Good Suffix rule. But after reading some articles on GeeksforGeeks, as well as YouTube tutorials, we managed to learn the basic principles of the Boyer-Moore algorithm. It is very difficult to execute the algorithm, especially making the Bad Character table and Good Suffix table, with minimum errors, in the implementation of the Boyer-Moore algorithm, but we corrected them with the aid of ChatGPT. The GoCrazy part was quite challenging for us. Understanding the Character-Density Guided Search algorithm was difficult. The hardest part was determining when to skip text. Checking the last character we used as a pivot and the character before it, and managing to skip at the right times was difficult. Initially, we skipped a lot of valid matches, so we frequently changed our skipping strategy. Getting the early rejection mechanism working correctly also took time. Comparing the last character we used as a pivot and the character before it, and being able to skip at the right times was also quite challenging. Initially, we skipped many valid matches, so we frequently changed our skipping strategy. Getting the early rejection part working properly also took time.The StudentPreAnalysis section was easier for us compared to the other sections. It already specified which algorithms should be used in which situations. Only writing the patternEntropy function among the helper functions was a bit difficult. We had to figure out how to calculate the diversity of characters in a pattern, which was a bit confusing. ,

8. References:

1. **Boyer-Moore Algorithm** - Wikipedia, https://en.wikipedia.org/wiki/Boyer%E2%80%93Moore_string-search_algorithm

2. **Boyer-Moore Search Algorithm** - Ben Langemead Youtube Channel : https://www.youtube.com/watch?v=4Xyhb72LCX4&t=200sGeeksforGeeks, https://www.geeksforgeeks.org/boyer-moor-algorithm-for-pattern-searching/

3. **String Matching Algorithms** - GeeksforGeeks, https://www.geeksforgeeks.org/algorithms-gq/pattern-searching/

4. **Google Scholar** - Search for papers on Character Density Guided Search and string matching optimizations.

5. **ChatGPT** was used for generating additional insights into hybrid strategies and understanding their implications on performance in different scenarios.

6. **Stack Overflow** - A community of developers discussing string matching algorithms, optimizations, and troubleshooting (https://stackoverflow.com/).

7. **Abdulbari** YouTube Channel for String Matching Algorithms : https://www.youtube.com/channel/UCY6cZ3Xr-DlGGX64Ym88krg

8. **BroCode** YouTube Channel for String Matching Algorithms : https://www.youtube.com/c/BroCode/