A Comparison between Brute-force, Sunday, KMP, FSM, Rabin-Karp, Gusfield Z

Ceren Ucak Olcay Duzgun

June 5, 2024

1 Introduction

In this paper, we compare single pattern matching algorithms regarding their efficiency. Single pattern matching algorithms are techniques used to find the occurrence of a specific pattern (substring) within a larger text (string). These algorithms scan the text to identify all locations where the pattern appears. Several well-known single pattern matching algorithms are included in this paper: Brute-Force, Sunday, Rabin-Karp, KMP, FSM, and Gusfield Z.

2 Algorithm Descriptions

2.1 Brute-force

Brute-force algorithms involve finding the occurrence(s) of a pattern within a larger text by exhaustively checking each possible starting position in the text. The brute-force algorithm works by sliding the pattern over the text and comparing the characters of the pattern with the corresponding characters in the text. This is repeated for every possible starting position in the text.

Efficiency: Brute-force pattern matching has a time complexity of $O(nm)$, which may be inefficient for large texts or long patterns.

2.2 Sunday

The Sunday algorithm is a string-searching algorithm for single pattern matching that provides an efficient way to find occurrences of a pattern within a text by using an "offset table" to skip over characters in the text that are less likely to lead to a match.

Efficiency: The Sunday algorithm has an average time complexity of $O(n)$, but in the worst case, it could be $O(nm)$, where n is the length of the text, and m is the length of the pattern.

2.3 KMP

The Knuth-Morris-Pratt (KMP) algorithm is a classic string-searching algorithm designed for efficient single-pattern matching within a larger text. It uses a two-step process: preprocessing the pattern to create a "partial match table" or "prefix function" and then using this table to guide the pattern matching in the text.

Efficiency: The KMP algorithm has a time complexity of $O(n + m)$.

2.4 FSM

Finite State Machine (FSM) algorithms for single pattern matching are based on constructing a state machine that can efficiently scan through a text to locate instances of a specific pattern.

Efficiency: FSM-based algorithms typically process text in linear time $O(n)$ after preprocessing.

2.5 Rabin-Karp

The Rabin-Karp algorithm uses hashing to quickly find matching patterns, making it particularly useful when searching for multiple patterns or dealing with long texts.

Efficiency: The Rabin-Karp algorithm has an average-case time complexity of $O(n + m)$.

2.6 Gusfield Z

The Gusfield Z algorithm is based on the concept of Z-values or Z-array, which captures the information about how much of a string matches with itself at various positions.

Efficiency: The Z-algorithm is efficient with a time complexity of $O(n)$.

3 Methodology

We implemented the algorithms in C++ and measured their execution times using different pattern lengths on a large text. The performance tests were conducted for both small and long patterns.

4 Results

The following results were obtained from the performance tests:

4.1 Performance tests for small patterns

```
Brute-force took 1.7416e-05 seconds
Sunday took 2.583e-06 seconds
KMP took 3.4875e-05 seconds
FSM took 0.000169208 seconds
Rabin-Karp took 1.825e-05 seconds
Gusfield Z took 4.2084e-05 seconds
```

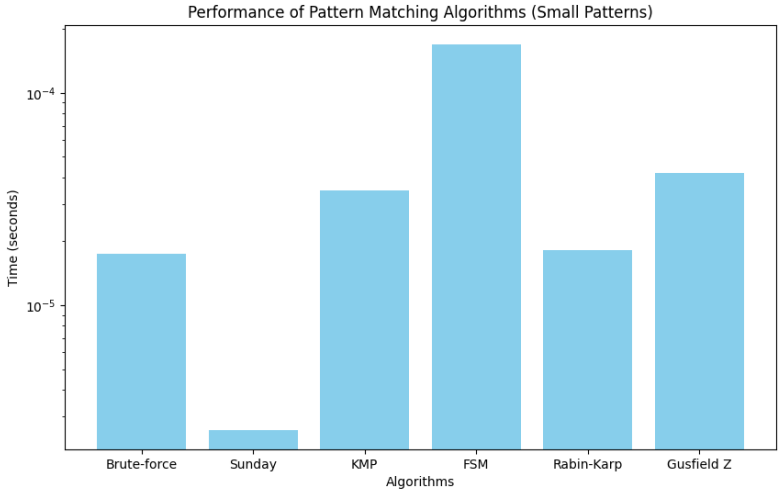

Figure 1: Comparison with short pattern.

From the results, we can see that the Sunday algorithm is the fastest for small patterns, followed by Rabin-Karp and Brute-force. The FSM algorithm is the least efficient.

4.2 Performance tests for long patterns

```
Brute-force took 0.00177487 seconds
Sunday took 0.000216667 seconds
KMP took 0.00351417 seconds
FSM took 0.00304475 seconds
Rabin-Karp took 0.00179875 seconds
Gusfield Z took0.00394033 seconds
```

For long patterns, the Sunday algorithm again proves to be the fastest, followed by Brute-force and Rabin-Karp. The Gusfield Z algorithm is the least efficient in this case.

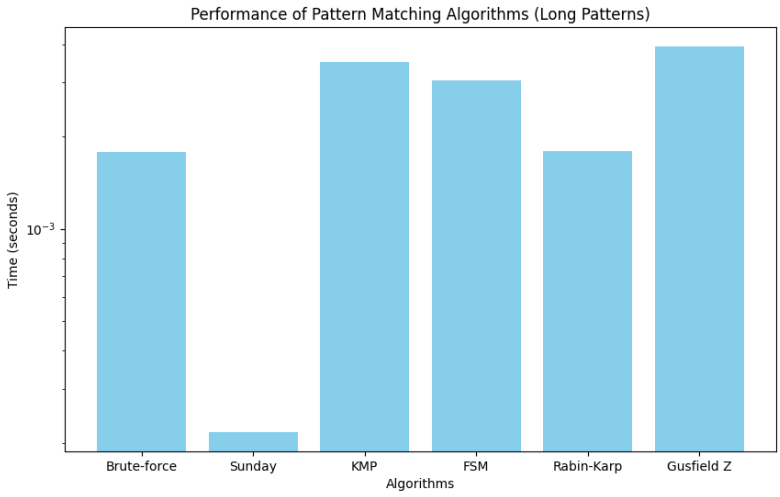

Figure 2: Comparison with long pattern.

4.3 Detailed Comparisons

- **Sunday vs. Gusfield Z:**

```
Sunday: 0.0002115 seconds
Gusfield Z: 0.00390796 seconds
```

  The Sunday algorithm is at least twice as fast as the Gusfield Z algorithm.

- **KMP vs. Rabin-Karp:**

```
KMP: 0.00357996 seconds
Rabin-Karp: 0.0017815 seconds
```

  The Rabin-Karp algorithm is significantly faster than the KMP algorithm.

- **Rabin-Karp vs. Sunday:**

```
Rabin-Karp: 0.00178983 seconds
Sunday: 0.000211541 seconds
```

  The Sunday algorithm is at least twice as fast as the Rabin-Karp algorithm.

- **Brute-force vs. FSM:**

```
Brute-force: 0.00174 seconds
FSM: 0.00301725 seconds
```

  The Brute-force algorithm is at least twice as fast as the FSM algorithm.

5 Conclusions

Based on the performance tests conducted, we can conclude the following:

- The Sunday algorithm is the most efficient for both short and long patterns.
- The FSM algorithm is the least efficient for both short and long patterns.
- The Rabin-Karp algorithm generally performs well, especially for long patterns.
- The Brute-force algorithm, while simple, performs better than FSM for larger patterns.
- The KMP algorithm, despite its optimal theoretical complexity, is outperformed by Rabin-Karp and Sunday in practical scenarios.
- The Gusfield Z algorithm, although efficient in theory, tends to be slower in practice compared to other algorithms like Sunday.

These results highlight the importance of choosing the right algorithm based on the specific use case and the characteristics of the text and pattern involved.