**Slide 0: Introduction**

**Slide 1:**

Here I have a snippet from our Introduction:

**“**The Rabin-Karp String Search Algorithm is a powerful and versatile method for pattern matching. It works by employing a rolling hash function to quickly compare the hash values of substrings in the text with the hash value of the target pattern. When a hash match is found, the algorithm performs a character-by-character comparison to confirm the match, allowing it to find all occurrences of the pattern in linear time complexity.”

Now what is Hashing?

Hashing is a process that transforms input data into a fixed-size string of characters, typically for creating a unique identifier or for data retrieval in constant time. It's what makes the Rabin-Karp String Search algorithm efficient and so interesting. It will be discussed throughout the whole presentation.

**Slide 2:**

In our report The following 3 String Search Algorithms are compared with Rabin-Karp.

Knuth-Morris-Pratt (KMP) Algorithm: It pre-processes the pattern, creating a partial match table to skip unnecessary comparisons. Ideal for single pattern searches, it averages O(n) time complexity with a worst case of O(n\*m).

Boyer-Moore Algorithm: Focuses on right-to-left character comparison, reducing comparisons significantly. Best for larger alphabets/texts, averaging O(n) time complexity with a worst case of O(n\*m).

Brute-Force Algorithm: Straightforward, comparing pattern and text character by character. Less efficient but serves as a performance baseline. Best/Average/Worst Case of O(n\*m) time complexity.

**Slide 3:**

Here I have another snippet this time from our Justification of Superiority: section

“Compared to the other string search algorithms Rabin-Karp is the only one to make use of hashing to efficiently find patterns within a text. This is important because it allows for quicker identification of patterns by converting substrings into hash values, enabling rapid comparison between the hashes of the pattern and potential matches within the text. This method significantly enhances the speed of pattern matching, especially with larger texts or complex patterns, reducing the computational load compared to linear search algorithms.”

Primary points of the section:

Rabin-Karp also excels in multiple pattern searches within a single text compared to the other string search algorithms.

And finally, while the time complexity is explored in greater detail later in the report it is included in brevity in this section Average case time complexity is O(n+m) with a worst case of O(n\*m). “This positions it between other string search algorithms in terms of time complexity, while having superior functionality.”