The Rabin-Karp string search algorithm is a pattern matching algorithm used to find occurrences of a substring (or pattern) within a larger text or string. It was developed by Michael O. Rabin and Richard M. Karp in 1987. The algorithm is known for its efficiency in searching for patterns in text, particularly in situations where multiple patterns need to be searched for simultaneously.

The Rabin-Karp algorithm employs a technique called hashing to compare substrings of the text with the pattern. Here's a high-level overview of how it works:

1. Preprocessing:

- Calculate the hash value of the pattern.

- Calculate the hash value of the first substring of the text, which has the same length as the pattern.

2. Search Process:

- Compare the hash value of the current substring in the text with the hash value of the pattern. If they match, it means there is a potential match between the pattern and the current substring.

- If the hash values match, perform a character-by-character comparison of the pattern and the current substring to confirm the match.

- If the hash values don't match, move the sliding window one character to the right in the text and calculate the hash value for the new substring. Repeat the hash comparison and character comparison until you find a match or reach the end of the text.

3. Handling Hash Collisions:

- Hash collisions can occur when different substrings have the same hash value. To handle this, the algorithm uses a rolling hash function that efficiently updates the hash value as the window slides.

The key advantages of the Rabin-Karp algorithm include its ability to search for multiple patterns in a single pass and its average-case time complexity of (theta)(N+M), where N is the length of the text and M is the length of the pattern. However, the worst-case time complexity can be O(N\*M) if hash collisions are not handled properly.

The Rabin-Karp algorithm is particularly useful when searching for multiple patterns in the same text or when you need to perform approximate string matching by allowing a certain level of mismatch tolerance. Additionally, it has applications in plagiarism detection, file comparison, and DNA sequence matching, among others.