String-related problems on LeetCode often require a good understanding of various key concepts and techniques. Here are some of the most popular and widely used ones:

1. **Two Pointers Technique**

* Utilize two pointers to traverse the string simultaneously from different ends or at different speeds.
* Commonly used in problems involving palindrome checks, substring matching, or sliding window problems.

1. **Sliding Window Technique**

* Maintain a window of elements and slide it through the string to efficiently solve substring problems.
* Useful for tasks like finding the longest substring with unique characters.

1. **HashMaps/Hashing**

* Use hash functions and hash maps to efficiently store and retrieve information about characters or substrings.
* Often employed in anagrams, finding duplicates, or tracking character frequencies.

1. **Sorting**

* Sort strings or characters to simplify comparison or identification of patterns.
* Useful in problems related to anagrams or finding the lexicographically smallest/largest substring.

1. **Dynamic Programming**

* Solve problems by breaking them into subproblems and building solutions incrementally.
* Applied in cases of finding the longest common subsequence, palindrome partitions, or edit distance.

1. **Regular Expressions**

* Leverage regular expressions for pattern matching and validation.
* Useful in problems requiring complex string pattern recognition.

1. **KMP Algorithm**

* Knuth-Morris-Pratt algorithm for efficient pattern matching.
* Particularly helpful for finding occurrences of a pattern in a string.

1. **Trie (Prefix Tree)**

* A tree-like data structure for efficiently storing and searching strings.
* Commonly used in problems involving autocomplete, spell checking, and IP address storage.

1. **Binary Search**

* Apply binary search on sorted strings or characters to efficiently locate elements.
* Useful in problems like searching a rotated array or finding the minimum in rotated sorted arrays.

1. **Manacher's Algorithm**

* Efficient algorithm for finding the longest palindromic substring.
* Particularly helpful in tasks requiring palindrome identification.

1. **Greedy Algorithms**

* Make locally optimal choices at each stage with the hope of finding a global optimum.
* Applied in problems like string compression or tasks where a greedy approach is suitable.

Understanding and mastering these concepts and techniques will greatly enhance your ability to tackle a wide range of string-related problems on LeetCode and other coding platforms.

Here's a list of widely used algorithms for solving string-related problems on LeetCode:

**Brute Force**

Simple approach of checking all possible substrings or combinations.

Used when the problem size is small and more optimized algorithms are not required.

**Two Pointers Technique**

Efficiently solve problems by maintaining two pointers in the string.

Commonly used in palindrome checks, substring matching, or sliding window problems.

**Sliding Window Technique**

Maintain a window of elements and slide it through the string.

Useful for substring problems, especially when looking for the longest substring with unique characters.

**KMP Algorithm**

Knuth-Morris-Pratt algorithm for efficient pattern matching.

Particularly helpful for finding occurrences of a pattern in a string.

**Rabin-Karp Algorithm**

A rolling hash algorithm used for substring search.

Efficient for finding multiple occurrences of a pattern in a string.

**Manacher's Algorithm**

Efficient algorithm for finding the longest palindromic substring.

Useful in tasks requiring palindrome identification.

**Trie (Prefix Tree)**

A tree-like data structure for storing and searching strings.

Commonly used in problems involving autocomplete, spell checking, and IP address storage.

**Boyer-Moore Algorithm**

Efficient string searching algorithm.

Particularly useful for searching in large texts.

**Edit Distance (Dynamic Programming)**

Dynamic programming approach to find the minimum number of operations (insertions, deletions, substitutions) to transform one string into another.

**Longest Common Subsequence (LCS - Dynamic Programming)**

Find the longest common subsequence between two strings.

**Z Algorithm**

Efficient pattern matching algorithm that finds all occurrences of a pattern in a string.

**Suffix Array and LCP (Longest Common Prefix) Array**

Data structures used for efficiently solving problems related to substrings.

**Greedy Algorithms**

Make locally optimal choices at each stage with the hope of finding a global optimum.

Applied in problems like string compression or tasks where a greedy approach is suitable.

**Regular Expressions**

Leverage regular expressions for pattern matching and validation.

Useful in problems requiring complex string pattern recognition.

Understanding these algorithms and when to apply them will greatly enhance your ability to tackle a variety of string-related problems on LeetCode and other coding platforms.