Assignment –3

Explanation

KMP Algorithm

The Knuth-Morris-Pratt (KMP) algorithm is an efficient string matching algorithm that improves upon the naive approach by avoiding unnecessary comparisons. The KMP algorithm achieves this through a preprocessing step that constructs an LPS (Longest Prefix Suffix) array for the pattern.

How Preprocessing Improves Search Time

1. **Naive Approach**:
    - The naive approach compares the pattern to the text starting at each position in the text. When a mismatch is found, it moves the pattern one position to the right and starts comparing again from the beginning of the pattern.
    - In the worst case, this results in a time complexity of O(m * n), where m is the length of the pattern and n is the length of the text.
2. **KMP Approach**:
    - The KMP algorithm preprocesses the pattern to create the LPS array. The LPS array indicates the longest proper prefix of the pattern that is also a suffix. This preprocessing helps to skip unnecessary comparisons.
    - When a mismatch occurs after some matches, the LPS array is used to determine the next positions to compare, instead of starting from the beginning of the pattern.
    - This reduces the number of comparisons significantly, resulting in a time complexity of O(m + n).

Preprocessing Step (LPS Array)

- The **computeLPSArray** method constructs the LPS array. It iterates through the pattern and uses previous LPS values to avoid redundant comparisons.
- When constructing the LPS array, if there is a mismatch, the algorithm uses the LPS value to skip characters that are known to match.

Searching Step

- The **KMPSearch** method uses the LPS array to perform the actual pattern search. It compares characters and uses the LPS array to skip characters when mismatches occur, leading to fewer overall comparisons.

By preprocessing the pattern to create the LPS array, the KMP algorithm reduces the number of character comparisons needed, making the pattern search more efficient than the naive approach.