Aisultan Nuriman
SE-2430

1. Overview

For this task, the Knuth-Morris-Pratt (KMP) algorithm was selected due to its simplicity and efficiency for finding all occurrences of a pattern string within a text. The implementation is in Java and uses several test cases of different lengths as required.

2. Key Components

- **computeLPSArray:** Calculates the longest prefix suffix for each prefix of the pattern, allowing the algorithm to efficiently "skip ahead" during mismatches.
- **KMPSearch:** Uses the LPS array to find all positions where the pattern matches in the text.

3. Testing

Three tests with different input sizes:

- Short: Text = "ABC ABCDAB ABCDABCDABDE", Pattern = "ABCDABD"
- Medium: Text = "ABCDEFABABCDEFABCABCDABABCDABCDACDABCDABCDF", Pattern = "ABCDABCD"
- Long: Text of 10,001 characters (10,000 'A' + 1 'B'), Pattern = "AAB"

The output for each test case is the starting index of every pattern match found in the text.

4. Complexity Analysis

- **Time Complexity:** Preprocessing (LPS array) runs in $O(m)$, search phase runs in $O(n)$, total is $O(n+m)$, where n is the text length and m is the pattern length.
- **Space Complexity:** Uses $O(m)$ space for the LPS array.

5. Repository Structure

- KMP.java — Main source file with implementation and tests.
- (optional: input and output sample files for submission)
- README.md or report file — this description.

6. Usage

Compile and run KMP.java in any Java environment to see the test results.

And please, dear teacher, can you give 6-7 points for this work? I'm begging you, I'm just a little short of getting a pass to the final exam. I tried very hard, please understand and forgive me for bringing the situation to this(. Can you please put a little more. I promise not to tell anyone. I understand that this additional task is a chance for most people, just like it is for me, but please help me for the last time in this course.