Justification for Selection of Naive String-Matching Algorithm

I chose the naive string-matching algorithm because it's easy to understand, great for learning, and helps me clearly show how pattern matching works. While it might not be the fastest for big tasks, it's perfect for teaching and illustrating the basics of pattern matching in a simple way.

Brief Explanation of Main Functions:

❖ 'matchPattern' Function:

This function recursively compares characters of the text and pattern to determine if a match exists. It handles wildcards ('.'), '^', '?', and '$' in the pattern for different matching scenarios.

Implemented Regular Expressions:

- Wildcard ('.') Matching
- Start of Line ('^') Matching
- Optional Character ('?') Matching
- End of Line ('$') Matching

The base case is when the end of the pattern is reached, indicating a successful match.

❖ 'findPattern' Function:

This function iterates through the text and invokes matchPattern to identify occurrences of the pattern. When a match is found, it records the position in the output file.

❖ Main Function:

The program prompts the user for the number of tests files and processes each test case. It reads text and pattern inputs from corresponding files and calls findPattern. If a match is found, it writes the position to the output file. Otherwise, it notes that no pattern was found. The program provides a clear output indicating whether a pattern was found or not in the given text. This concise output aids in quickly understanding the results of the pattern matching process.