Camel Case Matching LeetCode

Given an array of strings queries and a string pattern, return a boolean array answer where answer[i] is true if queries[i] matches pattern, and false otherwise.

A query word queries[i] matches pattern if you can insert lowercase English letters pattern so that it equals the query. You may insert each character at any position and you may not insert any characters.

Example 1:

Input: queries = ["FooBar","FooBarTest","FootBall","FrameBuffer","ForceFeedBack"], pattern = "FB"

Output: [true,false,true,true,false]

Explanation: "FooBar" can be generated like this "F" + "oo" + "B" + "ar".

"FootBall" can be generated like this "F" + "oot" + "B" + "all".

"FrameBuffer" can be generated like this "F" + "rame" + "B" + "uffer".

Example 2:

Input: queries = ["FooBar","FooBarTest","FootBall","FrameBuffer","ForceFeedBack"], pattern = "FoBa"

Output: [true,false,true,false,false]

Explanation: "FooBar" can be generated like this "Fo" + "o" + "Ba" + "r".

"FootBall" can be generated like this "Fo" + "ot" + "Ba" + "ll".

Example 3:

Input: queries = ["FooBar","FooBarTest","FootBall","FrameBuffer","ForceFeedBack"], pattern = "FoBaT"

Output: [false,true,false,false,false]

Explanation: "FooBarTest" can be generated like this "Fo" + "o" + "Ba" + "r" + "T" + "est".

Approach 1: Function to perform CamelCase matching for a list of queries with a given pattern

Function Purpose:

Perform CamelCase matching for a list of queries with a given pattern.

Explanation:

- **TrieNode Class:**

  - Represents a node in the trie with children nodes, an end-of-word flag, and the original word associated with the node.
- **Trie Class:**
  - Manages the trie and provides methods to insert words and check if a given string matches the CamelCase pattern.
- **insert Method:**
  - Inserts each character of a word into the trie.
- **isCamelCase Method:**
  - Checks if a given string matches the CamelCase pattern by traversing the trie.
- **camelMatch Function:**
  - Creates a trie and inserts the pattern.
  - Checks CamelCase matching for each query using the trie.

Time Complexity:

- **Trie Insertion: O(P), where P is the length of the pattern.**
- **CamelCase Matching per Query: O(L), where L is the average length of a query.**
- **Overall Time Complexity: O(P + (N * L)), where N is the number of queries and L is the average length of a query.**

Space Complexity:

- **Trie Storage: O(P), where P is the length of the pattern.**
- **Overall Space Complexity: O(P + (N * L)), where N is the number of queries and L is the average length of a query.**