Replace Word from Sentence LeetCode

In English, we have a concept called **root**, which can be followed by some other word to form another longer word - let's call this word **successor**. For example, when the **root** "an" is followed by the **successor** word "other", we can form a new word "another".

Given a dictionary consisting of many **roots** and a sentence consisting of words separated by spaces, replace all the **successors** in the sentence with the **root** forming it. If a **successor** can be replaced by more than one **root**, replace it with the **root** that has **the shortest length**.

Return *the sentence* after the replacement.

Example: **Input:** dictionary = ["cat","bat","rat"], sentence = "the cattle was rattled by the battery"

Output: "the cat was rat by the bat"

Example 2:

Input: dictionary = ["a","b","c"], sentence = "aadsfasf absbs bbab cadsfafs"

Output: "a a b c"

Approach 1: Function to replace words in a sentence with their shortest prefixes

Function Purpose:

Replace words in a sentence with their shortest prefixes using a trie-based approach.

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 get the shortest prefix for a given word.
- **insert Method:**
    - Inserts each word into the trie.
- **getWord Method:**
    - Retrieves the shortest prefix for a given word from the trie.
- **replaceWords Function:**
    - Creates a trie and inserts dictionary words.

- Splits the sentence into words and replaces each word with its shortest prefix.

Time Complexity:

- **Trie Insertion: O(N * L), where N is the number of words in the dictionary, and L is the average length of a word.**
- **Word Replacement: O(M * L), where M is the number of words in the sentence, and L is the average length of a word.**
- **Overall Time Complexity: O(N * L + M * L)**

Space Complexity:

- **Trie Storage: O(N * L), where N is the number of words in the dictionary, and L is the average length of a word.**
- **Words Vector: O(M * L), where M is the number of words in the sentence, and L is the average length of a word.**
- **Overall Space Complexity: O(N * L + M * L)**