Problem 1639: Number of Ways to Form a Target String Given a Dictionary

Problem Information

Difficulty: Hard
Acceptance Rate: 56.75%
Paid Only: No
Tags: Array, String, Dynamic Programming

Problem Description

You are given a list of strings of the **same length** `words` and a string `target`.

Your task is to form `target` using the given `words` under the following rules:

* `target` should be formed from left to right. * To form the `ith` character (**0-indexed**) of `target`, you can choose the `kth` character of the `jth` string in `words` if `target[i] = words[j][k]`. * Once you use the `kth` character of the `jth` string of `words`, you **can no longer** use the `xth` character of any string in `words` where `x <= k`. In other words, all characters to the left of or at index `k` become unusuable for every string. * Repeat the process until you form the string `target`.

Notice that you can use **multiple characters** from the **same string** in `words` provided the conditions above are met.

Return _the number of ways to form`target` from `words`_. Since the answer may be too large, return it **modulo** `109 + 7`.

Example 1:

Input: words = ["acca","bbbb","caca"], target = "aba" **Output:** 6 **Explanation:** There are 6 ways to form target. "aba" -> index 0 ("_a_ cca"), index 1 ("b _b_ bb"), index 3 ("cac _a_ ") "aba" -> index 0 ("_a_ cca"), index 2 ("bb _b_ b"), index 3 ("cac _a_ ") "aba" -> index 0 ("_a_ cca"), index 1 ("b _b_ bb"), index 3 ("acc _a_ ") "aba" -> index 0 ("_a_ cca"), index 2 ("bb _b_ b"), index 3 ("acc _a_ ") "aba" -> index 1 ("c _a_ ca"), index 2 ("bb _b_ b"), index 3 ("acc _a_ ") "aba" -> index 1 ("c _a_ ca"), index 2 ("bb _b_ b"), index 3 ("cac _a_ ")

Example 2:

Input: words = ["abba","baab"], target = "bab" **Output:** 4 **Explanation:** There are 4 ways to form target. "bab" -> index 0 ("_b_ aab"), index 1 ("b _a_ ab"), index 2 ("ab _b_ a") "bab" -> index 0 ("_b_ aab"), index 1 ("b _a_ ab"), index 3 ("baa _b_ ") "bab" -> index 0 ("_b_ aab"), index 2 ("ba _a_ b"), index 3 ("baa _b_ ") "bab" -> index 1 ("a _b_ ba"), index 2 ("ba _a_ b"), index 3 ("baa _b_ ")

Constraints:

* `1 <= words.length <= 1000` * `1 <= words[i].length <= 1000` * All strings in `words` have the same length. * `1 <= target.length <= 1000` * `words[i]` and `target` contain only lowercase English letters.

Code Snippets

C++:

```
class Solution {
public:
int numWays(vector<string>& words, string target) {

}
};
```

Java:

```
class Solution {
public int numWays(String[] words, String target) {

}
}
```

Python3:

```
class Solution:
def numWays(self, words: List[str], target: str) -> int:
```