Problem 411: Minimum Unique Word Abbreviation

Problem Information

Difficulty: Hard
Acceptance Rate: 40.30%
Paid Only: Yes
Tags: Array, String, Backtracking, Bit Manipulation

Problem Description

A string can be **abbreviated** by replacing any number of **non-adjacent** substrings with their lengths. For example, a string such as `"substitution"` could be abbreviated as (but not limited to):

* `"s10n"` (`"s _ubstitutio_ n"`) * `"sub4u4"` (`"sub _stit_ u _tion_ "`) * `"12"` (`"_substitution_ "`) * `"su3i1u2on"` (`"su _bst_ i _t_ u _ti_ on"`) * `"substitution"` (no substrings replaced)

Note that `"s55n"` (`"s _ubsti_ _tutio_ n"`) is not a valid abbreviation of `"substitution"` because the replaced substrings are adjacent.

The **length** of an abbreviation is the number of letters that were not replaced plus the number of substrings that were replaced. For example, the abbreviation `"s10n"` has a length of `3` (`2` letters + `1` substring) and `"su3i1u2on"` has a length of `9` (`6` letters + `3` substrings).

Given a target string `target` and an array of strings `dictionary`, return _an**abbreviation** of _`target` _with the**shortest possible length** such that it is **not an abbreviation** of **any** string in _`dictionary` _. If there are multiple shortest abbreviations, return any of them_.

Example 1:

Input: target = "apple", dictionary = ["blade"] **Output:** "a4" **Explanation:** The shortest abbreviation of "apple" is "5", but this is also an abbreviation of "blade". The next shortest abbreviations are "a4" and "4e". "4e" is an abbreviation of blade while "a4" is not. Hence, return "a4".

Example 2:

Input: target = "apple", dictionary = ["blade","plain","amber"] **Output:** "1p3" **Explanation:** "5" is an abbreviation of both "apple" but also every word in the dictionary. "a4" is an abbreviation of "apple" but also "amber". "4e" is an abbreviation of "apple" but also "blade". "1p3", "2p2", and "3l1" are the next shortest abbreviations of "apple". Since none of them are abbreviations of words in the dictionary, returning any of them is correct.

Constraints:

* `m == target.length` * `n == dictionary.length` * `1 <= m <= 21` * `0 <= n <= 1000` * `1 <= dictionary[i].length <= 100` * `log2(n) + m <= 21` if `n > 0` * `target` and `dictionary[i]` consist of lowercase English letters. * `dictionary` does not contain `target`.

Code Snippets

C++:

```
class Solution {
public:
string minAbbreviation(string target, vector<string>& dictionary) {

}
};
```

Java:

```
class Solution {
public String minAbbreviation(String target, String[] dictionary) {

}
}
```

Python3:

```
class Solution:
def minAbbreviation(self, target: str, dictionary: List[str]) -> str:
```