Problem 288: Unique Word Abbreviation

Problem Information

Difficulty: Medium
Acceptance Rate: 27.29%
Paid Only: Yes
Tags: Array, Hash Table, String, Design

Problem Description

The **abbreviation** of a word is a concatenation of its first letter, the number of characters between the first and last letter, and its last letter. If a word has only two characters, then it is an **abbreviation** of itself.

For example:

* `dog --> d1g` because there is one letter between the first letter `'d'` and the last letter `'g'`. * `internationalization --> i18n` because there are 18 letters between the first letter `'i'` and the last letter `'n'`. * `it --> it` because any word with only two characters is an **abbreviation** of itself.

Implement the `ValidWordAbbr` class:

* `ValidWordAbbr(String[] dictionary)` Initializes the object with a `dictionary` of words. * `boolean isUnique(string word)` Returns `true` if **either** of the following conditions are met (otherwise returns `false`): * There is no word in `dictionary` whose **abbreviation** is equal to `word`'s **abbreviation**. * For any word in `dictionary` whose **abbreviation** is equal to `word`'s **abbreviation** , that word and `word` are **the same**.

Example 1:

Input ["ValidWordAbbr", "isUnique", "isUnique", "isUnique", "isUnique", "isUnique"] [[["deer", "door", "cake", "card"]], ["dear"], ["cart"], ["cane"], ["make"], ["cake"]] **Output** [null, false, true, false, true, true] **Explanation** ValidWordAbbr validWordAbbr = new ValidWordAbbr(["deer", "door", "cake", "card"]); validWordAbbr.isUnique("dear"); // return false, dictionary word "deer" and word "dear" have the same abbreviation "d2r" but are not the

same. validWordAbbr.isUnique("cart"); // return true, no words in the dictionary have the abbreviation "c2t". validWordAbbr.isUnique("cane"); // return false, dictionary word "cake" and word "cane" have the same abbreviation "c2e" but are not the same. validWordAbbr.isUnique("make"); // return true, no words in the dictionary have the abbreviation "m2e". validWordAbbr.isUnique("cake"); // return true, because "cake" is already in the dictionary and no other word in the dictionary has "c2e" abbreviation.

Constraints:

* `1 <= dictionary.length <= 3 * 104` * `1 <= dictionary[i].length <= 20` * `dictionary[i]` consists of lowercase English letters. * `1 <= word.length <= 20` * `word` consists of lowercase English letters. * At most `5000` calls will be made to `isUnique`.

Code Snippets

C++:

```
class ValidWordAbbr {
public:
ValidWordAbbr(vector<string>& dictionary) {

}

bool isUnique(string word) {

}
};

/**
* Your ValidWordAbbr object will be instantiated and called as such:
* ValidWordAbbr* obj = new ValidWordAbbr(dictionary);
* bool param_1 = obj->isUnique(word);
*/
```

Java:

```
class ValidWordAbbr {

public ValidWordAbbr(String[] dictionary) {

}
```

```
public boolean isUnique(String word) {

}
}

/**
* Your ValidWordAbbr object will be instantiated and called as such:
* ValidWordAbbr obj = new ValidWordAbbr(dictionary);
* boolean param_1 = obj.isUnique(word);
*/
```

Python3:

```
class ValidWordAbbr:

def __init__(self, dictionary: List[str]):


def isUnique(self, word: str) -> bool:



# Your ValidWordAbbr object will be instantiated and called as such:
# obj = ValidWordAbbr(dictionary)
# param_1 = obj.isUnique(word)
```