# Problem 527: Word Abbreviation

## Problem Information

**Difficulty:** Hard
**Acceptance Rate:** 62.53%
**Paid Only:** Yes
**Tags:** Array, String, Greedy, Trie, Sorting

## Problem Description

Given an array of **distinct** strings `words`, return _the minimal possible**abbreviations** for every word_.

The following are the rules for a string abbreviation:

1. The **initial** abbreviation for each word is: the first character, then the number of characters in between, followed by the last character. 2. If more than one word shares the **same** abbreviation, then perform the following operation: * **Increase** the prefix (characters in the first part) of each of their abbreviations by `1`. * For example, say you start with the words `["abcdef","abndef"]` both initially abbreviated as `"a4f"`. Then, a sequence of operations would be `["a4f","a4f"]` -> `["ab3f","ab3f"]` -> `["abc2f","abn2f"]`. * This operation is repeated until every abbreviation is **unique**. 3. At the end, if an abbreviation did not make a word shorter, then keep it as the original word.

**Example 1:**

**Input:** words = ["like","god","internal","me","internet","interval","intension","face","intrusion"] **Output:** ["l2e","god","internal","me","i6t","interval","inte4n","f2e","intr4n"]

**Example 2:**

**Input:** words = ["aa","aaa"] **Output:** ["aa","aaa"]

**Constraints:**

* `1 <= words.length <= 400` * `2 <= words[i].length <= 400` * `words[i]` consists of lowercase English letters. * All the strings of `words` are **unique**.

## Code Snippets

### C++:

```
class Solution {
public:
vector<string> wordsAbbreviation(vector<string>& words) {

}
};
```

### Java:

```
class Solution {
public List<String> wordsAbbreviation(List<String> words) {

}
}
```

### Python3:

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