527. Word Abbreviation

Hard | 276 | 186 | Add to List | Share

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. Begin with the first character, and then the number of characters abbreviated, followed by the last character.
2. If there is any conflict and more than one word shares the same abbreviation, a longer prefix is used instead of only the first character until making the map from word to abbreviation become unique. In other words, a final abbreviation cannot map to more than one original word.
3. If the abbreviation does not make the word shorter, then keep it as the original.

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**.

Accepted 20,791 | Submissions 36,148

Seen this question in a real interview before? Yes No

Companies

0 ~ 6 months | 6 months ~ 1 year | 1 year ~ 2 years

Google | 3 | Uber | 2 | Snapchat

Related Topics

Array | String | Greedy | Trie | Sorting

Similar Questions

Valid Word Abbreviation — Easy

Minimum Unique Word Abbreviation — Hard

```java
class Solution {
    public List<String> wordsAbbreviation(List<String> words) {
        Map<String, List<IndexedWord>> groups = new HashMap();
        String[] ans = new String[words.size()];

        int index = 0;
        for (String word: words) {
            String ab = abbrev(word, 0);
            if (!groups.containsKey(ab))
                groups.put(ab, new ArrayList());
            groups.get(ab).add(new IndexedWord(word, index));
            index++;
        }

        for (List<IndexedWord> group: groups.values()) {
            TrieNode trie = new TrieNode();
            for (IndexedWord iw: group) {
                TrieNode cur = trie;
                for (char letter: iw.word.substring(1).toCharArray()) {
                    if (cur.children[letter - 'a'] == null)
                        cur.children[letter - 'a'] = new TrieNode();
                    cur.count++;
                    cur = cur.children[letter - 'a'];
                }
            }

            for (IndexedWord iw: group) {
                TrieNode cur = trie;
                int i = 1;
                for (char letter: iw.word.substring(1).toCharArray()) {
                    if (cur.count == 1) break;
                    cur = cur.children[letter - 'a'];
                    i++;
                }
                ans[iw.index] = abbrev(iw.word, i-1);
            }
        }

        return Arrays.asList(ans);
    }

    public String abbrev(String word, int i) {
        int N = word.length();
        if (N - i <= 3) return word;
        return word.substring(0, i+1) + (N - i - 2) + word.charAt(N-1);
    }
}

class TrieNode {
    TrieNode[] children;
    int count;
    TrieNode() {
        children = new TrieNode[26];
        count = 0;
    }
}
class IndexedWord {
    String word;
    int index;
    IndexedWord(String w, int i) {
        word = w;
        index = i;
    }
}
```