720 Longest Word in Dictionary

2018年5月4日 10:34

Given a list of strings words representing an English Dictionary, find the longest word in words that can be built one character at a time by other words in words. If there is more than one possible answer, return the longest word with the smallest lexicographical order.

If there is no answer, return the empty string.

Example 1:

Input:

words = ["w","wo","wor","worl", "world"]

Output: "world"

Explanation:

The word "world" can be built one character at a time by "w", "wo", "wor", and "worl".

Example 2:

Input:

words = ["a", "banana", "app", "appl", "ap", "apply", "apple"]

Output: "apple"

Explanation:

Both "apply" and "apple" can be built from other words in the dictionary. However, "apple" is lexicographically smaller than "apply".

Note:

- All the strings in the input will only contain lowercase letters.
- The length of words will be in the range [1, 1000].
- The length of words[i] will be in the range [1, 30].

来自 <https://leetcode.com/problems/longest-word-in-dictionary/description/>

给出一个字符串数组words组成的一本英语词典。从中找出最长的一个单词，该单词是由words词典中其他单词逐步添加一个字母组成。若其中有多个可行的答案，则返回答案中字典序最小的单词。

若无答案，则返回空字符串。

注意:

- 所有输入的字符串都只包含小写字母。
- words数组长度范围为[1,1000]。
- words[i]的长度范围为[1,30]。

Solution for Python3:

```
class Solution1:
        def longestWord(self, words):
            """
            :type words: List[str]
            :rtype: str
            """
            ans = ""
            wordset = set(words)
            for word in words:
               if len(word) > len(ans) or len(word) == len(ans) and word < ans:
                   if all(word[:k] in wordset for k in range(1,len(word))):
                       ans = word
            return ans

    class Solution2:
        def longestWord(self, words):
            """
            :type words: List[str]
            :rtype: str
            """
            wordset = set(words)
            words.sort(key = lambda c : (-len(c), c))
            for word in words:
                if all(word[:k] in wordset for k in range(1,len(word))):
                    return word
            return ""

    class Solution3:
        def longestWord(self, words):
            """
            :type words: List[str]
            :rtype: str
            """
            return min((set(itertools.accumulate(w)) - set(words), -len(w), w) for w in words + [''])[2]
```

Solution for C++:

```
class Solution2 {
public:
    string longestWord(vector<string>& words) {
        string ans = "";
        set<string> wordset(words.begin(), words.end());
        for (string word : words) {
            if (word.length() > ans.length() || word.length() == ans.length() && word < ans) {
                bool flag = true;
                for (int k = 1; k < word.length(); k++) {
                    if (!wordset.count(word.substr(0,k))) {
                        flag = false;
                        break;
                    }
                }
                if (flag)
                    ans = word;
            }
        }
        return ans;
    }
};

class Solution2 {
public:
    string longestWord(vector<string>& words) {
        set<string> wordset(words.begin(), words.end());
        sort(words.begin(), words.end(), [](const string& a, const string& b){return a.length() ==
b.length() ?  a < b : a.length() > b.length();});
        for (string word : words) {
            bool flag = true;
            for (int k = 1; k < word.length(); k++) {
                if (!wordset.count(word.substr(0,k))) {
                    flag = false;
                    break;
                }
            }
            if (flag)
                return word;
        }
        return "";
    }
};
```