Problem 472: Concatenated Words

Problem Information

Difficulty: Hard
Acceptance Rate: 49.62%
Paid Only: No
Tags: Array, String, Dynamic Programming, Depth-First Search, Trie, Sorting

Problem Description

Given an array of strings `words` (**without duplicates**), return _all the**concatenated words** in the given list of_ `words`.

A **concatenated word** is defined as a string that is comprised entirely of at least two shorter words (not necessarily distinct) in the given array.

Example 1:

Input: words = ["cat","cats","catsdogcats","dog","dogcatsdog","hippopotamuses","rat","ratcatdogcat"] **Output:** ["catsdogcats","dogcatsdog","ratcatdogcat"] **Explanation:** "catsdogcats" can be concatenated by "cats", "dog" and "cats"; "dogcatsdog" can be concatenated by "dog", "cats" and "dog"; "ratcatdogcat" can be concatenated by "rat", "cat", "dog" and "cat".

Example 2:

Input: words = ["cat","dog","catdog"] **Output:** ["catdog"]

Constraints:

* `1 <= words.length <= 104` * `1 <= words[i].length <= 30` * `words[i]` consists of only lowercase English letters. * All the strings of `words` are **unique**. * `1 <= sum(words[i].length) <= 105`

Code Snippets

C++:

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

}
};
```

Java:

```
class Solution {
public List<String> findAllConcatenatedWordsInADict(String[] words) {

}
}
```

Python3:

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