Problem 336: Palindrome Pairs

Problem Information

Difficulty: Hard
Acceptance Rate: 36.74%
Paid Only: No
Tags: Array, Hash Table, String, Trie

Problem Description

You are given a **0-indexed** array of **unique** strings `words`.

A **palindrome pair** is a pair of integers `(i, j)` such that:

* `0 <= i, j < words.length`, * `i != j`, and * `words[i] + words[j]` (the concatenation of the two strings) is a palindrome.

Return _an array of all the**palindrome pairs** of _`words`.

You must write an algorithm with `O(sum of words[i].length)` runtime complexity.

Example 1:

Input: words = ["abcd","dcba","lls","s","sssll"] **Output:** [[0,1],[1,0],[3,2],[2,4]] **Explanation:** The palindromes are ["abcddcba","dcbaabcd","slls","llssssll"]

Example 2:

Input: words = ["bat","tab","cat"] **Output:** [[0,1],[1,0]] **Explanation:** The palindromes are ["battab","tabbat"]

Example 3:

Input: words = ["a",""] **Output:** [[0,1],[1,0]] **Explanation:** The palindromes are ["a","a"]

Constraints:

* `1 <= words.length <= 5000` * `0 <= words[i].length <= 300` * `words[i]` consists of lowercase English letters.

Code Snippets

C++:

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

}
};
```

Java:

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

}
}
```

Python3:

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