단어 리스트에서 words[i] + words[j]가 팰린드롬이 되는 모든 인덱스 조합 (i, j)를 구하라.

Input: words = ["abcd","dcba","lls","s","sssll"]
Output: [[0,1],[1,0],[3,2],[2,4]]

1.브루트포스

```
class Solution:
    def palindromePairs(self, words: List[str]) -> List[List[int]]:
        def isPalindrome(word):
            return word == word[::-1]

        result = []
        for i, word1 in enumerate(words):
            for j, word2 in enumerate(words):
                if i!= j and isPalindrome(word1 + word2):
                    result.append((i, j))
        return result
```

- 시간 초과

2.Trie 활용

```
import collections
from typing import List


class TrieNode:
    def __init__(self):
        self.children = collections.defaultdict(TrieNode)
        self.word_id = -1
        self.palindrome_word_ids = []


class Trie:
    def __init__(self):
        self.root = TrieNode()

    @staticmethod
    def is_palindrome(word: str) -> bool:
        return word[::] == word[::-1]

    def insert(self, index, word) -> None:
        node = self.root
        for i, char in enumerate(reversed(word)):
            if self.is_palindrome(word[0:len(word) - i]):
                node.palindrome_word_ids.append(index)
            node = node.children[char]
        node.word_id = index

    def search(self, index, word) -> List[List[int]]:
        result = []
```

```
        node = self.root
        while word:
            if node.word_id >= 0:
                if self.is_palindrome(word):
                    result.append([index, node.word_id])
            if not word[0] in node.children:
                return result
            node = node.children[word[0]]
            word = word[1:]
        if node.word_id >= 0 and node.word_id != index:
            result.append([index, node.word_id])
        for palindrome_word_id in node.palindrome_word_ids:
            result.append([index, palindrome_word_id])
        return result


class Solution:
    def palindromePairs(self, words: List[str]) -> List[List[int]]:
        trie = Trie()
        for i, word in enumerate(words):
            trie.insert(i, word)
        results = []
        for i, word in enumerate(words):
            results.extend(trie.search(i, word))
        return results
```