1181. Before and After Puzzle

My Submissions (/contest/biweekly-contest-8/problems/before-and-after-puzzle/submissions/)

Back to Contest (/contest/biweekly-contest-8/)

Given a list of `phrases`, generate a list of Before and After puzzles.

A *phrase* is a string that consists of lowercase English letters and spaces only. No space appears in the start or the end of a phrase. There are no consecutive spaces in a phrase.

Before and After puzzles are phrases that are formed by merging two phrases where the **last word of the first phrase** is the same as the **first word of the second phrase**.

Return the Before and After puzzles that can be formed by every two phrases `phrases[i]` and `phrases[j]` where `i != j`. Note that the order of matching two phrases matters, we want to consider both orders.

You should return a list of **distinct** strings **sorted lexicographically**.

User Accepted:	628
User Tried:	759
Total Accepted:	631
Total Submissions:	1736
Difficulty:	Medium

Example 1:

```
Input: phrases = ["writing code","code rocks"]
Output: ["writing code rocks"]
```

Example 2:

```
Input: phrases = ["mission statement",
                  "a quick bite to eat",
                  "a chip off the old block",
                  "chocolate bar",
                  "mission impossible",
                  "a man on a mission",
                  "block party",
                  "eat my words",
                  "bar of soap"]
Output: ["a chip off the old block party",
         "a man on a mission impossible",
         "a man on a mission statement",
         "a quick bite to eat my words",
         "chocolate bar of soap"]
```

Example 3:

```
Input: phrases = ["a","b","a"]
Output: ["a"]
```

Constraints:

- `1 <= phrases.length <= 100`
- `1 <= phrases[i].length <= 100`

Discuss (https://leetcode.com/problems/before-and-after-puzzle/discuss)

Python3

```python
class Solution:
    # pythonic, 40 ms
    def beforeAndAfterPuzzles1(self, phrases: List[str]) -> List[str]:
        phrases = [phrase.split() for phrase in phrases]
        total = [phrases[i] + phrases[j][1:]
                for i in range(len(phrases))
                for j in range(len(phrases))
                if i != j and phrases[i][-1] == phrases[j][0]]
        return sorted(set([" ".join(phrase) for phrase in total]))

    # split() at the begining, 60 ms
    def beforeAndAfterPuzzles1(self, phrases: List[str]) -> List[str]:
        phrases = [phrase.split() for phrase in phrases]
        total = []
        for i in range(len(phrases)):
            for j in range(len(phrases)):
                if i != j and phrases[i][-1] == phrases[j][0]:
                    total.append(phrases[i] + phrases[j][1:])
        return sorted(set([" ".join(phrase) for phrase in total]))

    # Done in contest, 108 ms
    def beforeAndAfterPuzzles(self, phrases: List[str]) -> List[str]:
        total = []
        for i in range(len(phrases)):
            for j in range(len(phrases)):
                phrase1 = phrases[i].split()
                phrase2 = phrases[j].split()
                if i != j and phrase1[-1] == phrase2[0]:
                     total.append(" ".join(phrase1 + phrase2[1:]))
        return sorted(set(total))
```

☐ **Custom Testcase** Run Code Submit