Problem 893: Groups of Special-Equivalent Strings

Problem Information

Difficulty: Medium
Acceptance Rate: 73.29%
Paid Only: No
Tags: Array, Hash Table, String, Sorting

Problem Description

You are given an array of strings of the same length `words`.

In one **move** , you can swap any two even indexed characters or any two odd indexed characters of a string `words[i]`.

Two strings `words[i]` and `words[j]` are **special-equivalent** if after any number of moves, `words[i] == words[j]`.

* For example, `words[i] = "zzxy"` and `words[j] = "xyzz"` are **special-equivalent** because we may make the moves `"zzxy" -> "xzzy" -> "xyzz"`.

A **group of special-equivalent strings** from `words` is a non-empty subset of words such that:

* Every pair of strings in the group are special equivalent, and * The group is the largest size possible (i.e., there is not a string `words[i]` not in the group such that `words[i]` is special-equivalent to every string in the group).

Return _the number of**groups of special-equivalent strings** from _`words`.

Example 1:

Input: words = ["abcd","cdab","cbad","xyzz","zzxy","zzyx"] **Output:** 3 **Explanation:** One group is ["abcd", "cdab", "cbad"], since they are all pairwise special equivalent, and none of the other strings is all pairwise special equivalent to these. The other two groups are

["xyzz", "zzxy"] and ["zzyx"]. Note that in particular, "zzxy" is not special equivalent to "zzyx".

Example 2:

Input: words = ["abc","acb","bac","bca","cab","cba"] **Output:** 3

Constraints:

* `1 <= words.length <= 1000` * `1 <= words[i].length <= 20` * `words[i]` consist of lowercase English letters. * All the strings are of the same length.

Code Snippets

C++:

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

}
};
```

Java:

```
class Solution {
public int numSpecialEquivGroups(String[] words) {

}
}
```

Python3:

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