Problem 839: Similar String Groups

Problem Information

Difficulty: Hard
Acceptance Rate: 55.80%
Paid Only: No
Tags: Array, Hash Table, String, Depth-First Search, Breadth-First Search, Union Find

Problem Description

Two strings, `X` and `Y`, are considered similar if either they are identical or we can make them equivalent by swapping at most two letters (in distinct positions) within the string `X`.

For example, `"tars"` and `"rats"` are similar (swapping at positions `0` and `2`), and `"rats"` and `"arts"` are similar, but `"star"` is not similar to `"tars"`, `"rats"`, or `"arts"`.

Together, these form two connected groups by similarity: `{"tars", "rats", "arts"}` and `{"star"}`. Notice that `"tars"` and `"arts"` are in the same group even though they are not similar. Formally, each group is such that a word is in the group if and only if it is similar to at least one other word in the group.

We are given a list `strs` of strings where every string in `strs` is an anagram of every other string in `strs`. How many groups are there?

Example 1:

Input: strs = ["tars","rats","arts","star"] **Output:** 2

Example 2:

Input: strs = ["omv","ovm"] **Output:** 1

Constraints:

* `1 <= strs.length <= 300` * `1 <= strs[i].length <= 300` * `strs[i]` consists of lowercase letters only. * All words in `strs` have the same length and are anagrams of each other.

Code Snippets

C++:

```
class Solution {
public:
int numSimilarGroups(vector<string>& strs) {

}
};
```

Java:

```
class Solution {
public int numSimilarGroups(String[] strs) {

}
}
```

Python3:

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