Problem 839: Similar String Groups

Problem Information

Difficulty: Hard
Acceptance Rate: 0.00%
Paid Only: No

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:
```

Python:

```
class Solution(object):
def numSimilarGroups(self, strs):
"""
:type strs: List[str]
:rtype: int
"""
```

JavaScript:

```
/**
* @param {string[]} strs
* @return {number}
*/
var numSimilarGroups = function(strs) {

};
```

TypeScript:

```
function numSimilarGroups(strs: string[]): number {

};
```

C#:

```
public class Solution {
public int NumSimilarGroups(string[] strs) {

}
}
```

C:

```
int numSimilarGroups(char** strs, int strsSize) {

}
```

Go:

```
func numSimilarGroups(strs []string) int {

}
```

Kotlin:

```
class Solution {
fun numSimilarGroups(strs: Array<String>): Int {

}
}
```

Swift:

```
class Solution {
func numSimilarGroups(_ strs: [String]) -> Int {

}
}
```

Rust:

```
impl Solution {
pub fn num_similar_groups(strs: Vec<String>) -> i32 {

}
}
```

Ruby:

```
# @param {String[]} strs
# @return {Integer}
def num_similar_groups(strs)

end
```

PHP:

```
class Solution {

/**
* @param String[] $strs
* @return Integer
*/
function numSimilarGroups($strs) {

}
}
```

Dart:

```
class Solution {
int numSimilarGroups(List<String> strs) {

}
}
```

Scala:

```
object Solution {
def numSimilarGroups(strs: Array[String]): Int = {

}
}
```

Elixir:

```
defmodule Solution do
@spec num_similar_groups(strs :: [String.t]) :: integer
def num_similar_groups(strs) do

end
end
```

Erlang:

```
-spec num_similar_groups(Strs :: [unicode:unicode_binary()]) -> integer().
num_similar_groups(Strs) ->
.
```

Racket:

```
(define/contract (num-similar-groups strs)
(-> (listof string?) exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Similar String Groups
* Difficulty: Hard
* Tags: array, string, graph, hash, search
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

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

}
};
```

Java Solution:

```
/**
* Problem: Similar String Groups
* Difficulty: Hard
* Tags: array, string, graph, hash, search
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

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

}
```

```
}
```

Python3 Solution:

```
"""
Problem: Similar String Groups
Difficulty: Hard
Tags: array, string, graph, hash, search

Approach: Use two pointers or sliding window technique
Time Complexity: O(n) or O(n log n)
Space Complexity: O(n) for hash map
"""

class Solution:
def numSimilarGroups(self, strs: List[str]) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def numSimilarGroups(self, strs):
"""
:type strs: List[str]
:rtype: int
"""
```

JavaScript Solution:

```
/**
* Problem: Similar String Groups
* Difficulty: Hard
* Tags: array, string, graph, hash, search
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

/**
```

```
* @param {string[]} strs
* @return {number}
*/
var numSimilarGroups = function(strs) {

};
```

TypeScript Solution:

```
/**
* Problem: Similar String Groups
* Difficulty: Hard
* Tags: array, string, graph, hash, search
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

function numSimilarGroups(strs: string[]): number {

};
```

C# Solution:

```
/*
* Problem: Similar String Groups
* Difficulty: Hard
* Tags: array, string, graph, hash, search
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

public class Solution {
public int NumSimilarGroups(string[] strs) {

}
}
```

C Solution:

```
/*
* Problem: Similar String Groups
* Difficulty: Hard
* Tags: array, string, graph, hash, search
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

int numSimilarGroups(char** strs, int strsSize) {

}
```

Go Solution:

```
// Problem: Similar String Groups
// Difficulty: Hard
// Tags: array, string, graph, hash, search
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) for hash map

func numSimilarGroups(strs []string) int {

}
```

Kotlin Solution:

```
class Solution {
fun numSimilarGroups(strs: Array<String>): Int {

}
}
```

Swift Solution:

```
class Solution {
func numSimilarGroups(_ strs: [String]) -> Int {
```

```
}
}
```

Rust Solution:

```
// Problem: Similar String Groups
// Difficulty: Hard
// Tags: array, string, graph, hash, search
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) for hash map

impl Solution {
pub fn num_similar_groups(strs: Vec<String>) -> i32 {

}
}
```

Ruby Solution:

```
# @param {String[]} strs
# @return {Integer}
def num_similar_groups(strs)

end
```

PHP Solution:

```
class Solution {

/**
* @param String[] $strs
* @return Integer
*/
function numSimilarGroups($strs) {

}
}
```

Dart Solution:

```
class Solution {
int numSimilarGroups(List<String> strs) {

}
}
```

Scala Solution:

```
object Solution {
def numSimilarGroups(strs: Array[String]): Int = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec num_similar_groups(strs :: [String.t]) :: integer
def num_similar_groups(strs) do

end
end
```

Erlang Solution:

```
-spec num_similar_groups(Strs :: [unicode:unicode_binary()]) -> integer().
num_similar_groups(Strs) ->
.
```

Racket Solution:

```
(define/contract (num-similar-groups strs)
(-> (listof string?) exact-integer?)
)
```