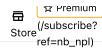
(/) Explore(/explore/) Problems(/problemset/all/)

Interview

contest/) Ontest Discuss(/discuss/)









2157. Groups of Strings

My Submissions (/contest/weekly-contest-278/problems/groups-of-strings/submissions/)

Back to Contest (/contest/weekly-contest-278/)

You are given a **0-indexed** array of strings words. Each string consists of **lowercase English letters** only. No letter occurs more than once in any string of words.

Two strings s1 and s2 are said to be **connected** if the set of letters of s2 can be obtained from the set of letters of s1 by any **one** of the following operations:

- Adding exactly one letter to the set of the letters of s1.
- Deleting exactly one letter from the set of the letters of s1.
- Replacing exactly one letter from the set of the letters of s1 with any letter, including
 itself

User Accepted:	217
User Tried:	1116
Total Accepted:	232
Total Submissions:	3704
Difficulty:	Hard

The array words can be divided into one or more non-intersecting **groups**. A string belongs to a group if any **one** of the following is true:

- It is connected to at least one other string of the group.
- It is the **only** string present in the group.

Note that the strings in words should be grouped in such a manner that a string belonging to a group cannot be connected to a string present in any other group. It can be proved that such an arrangement is always unique.

Return an array ans of size 2 where:

- ans [0] is the total number of groups words can be divided into, and
- ans [1] is the size of the largest group.

Example 1:

Example 2:

Constraints:

- 1 <= words.length <= 2 * 10⁴
- 1 <= words[i].length <= 26
- words[i] consists of lowercase English letters only.
- No letter occurs more than once in words [i].

Discuss (https://leetcode.com/problems/groups-of-strings/discuss)

```
JavaScript
                                                                                               4ħ
                                                                                                     \mathcal{C}
1 ▼ function DJSet(n) {
        // parent[i] < 0, -parent[i] is the group size which root is i. example: (i -> parent[i] ->
2
    parent[parent[i]] -> parent[parent[parent[i]]] ...)
        // parent[i] >= 0, i is not the root and parent[i] is i's parent. example: (...
 3
    parent[parent[i]]] -> parent[parent[i]] -> parent[i] -> i)
 4
        let parent = Array(n).fill(-1);
5
        return { find, union, count, equiv, getParent }
6
        function find(x) {
 7
            return parent[x] < 0 ? x : parent[x] = find(parent[x]);</pre>
8
9 •
        function union(x, y) {
10
            x = find(x);
            y = find(y);
11
12 ▼
            if (x != y) {
13
                 if (parent[x] < parent[y])[x, y] = [y, x];
14
                parent[x] += parent[y];
15
                parent[y] = x;
16
17
            return x == y;
18
        }
        function count() { // total connected groups (value < 0)</pre>
19 •
20
             return parent.filter(v => v < 0).length;
21
22 🔻
        function equiv(x, y) {
23
            return find(x) == find(y);
24
        function getParent() {
25 ▼
26
            return parent;
27
        }
   }
28
29
30 v const groupStrings = (words) => {
        let n = words.length, ds = new DJSet(n), pre = Array(n).fill(0), m = new Map();
31
32 ▼
        for (let i = 0; i < n; i++) {
33 ▼
            for (const c of words[i]) {
34
                pre[i] |= 1 << c.charCodeAt() - 97;</pre>
35
36
            if (m.has(pre[i])) ds.union(i, m.get(pre[i]));
37
            m.set(pre[i], i);
38
        }
39 ▼
        for (let i = 0; i < n; i++) {
40 ▼
            for (let j = 0; j < 26; j++) {
41
                let toggle = pre[i] ^ 1 << j;</pre>
42
                 if (m.has(toggle)) ds.union(i, m.get(toggle));
43
44 ▼
            for (let j = 0; j < 26; j++) {
45 ▼
                 for (let k = 0; k < 26; k++) {
46
                     let bitOfOneIJ = pre[i] & (1 << j), bitOfOneIK = pre[i] & (1 << k);</pre>
```

```
47 ▼
                    if (bitOfOneIJ && !bitOfOneIK) {
48
                         let toggle = pre[i] ^ (1 << j) ^ (1 << k);
49
                         if (m.has(toggle)) ds.union(i, m.get(toggle));
50
                    }
51
                }
52
            }
53
        }
54
        return [ds.count(), -(Math.min(...ds.getParent()))];
55
   };
```

☐ Custom Testcase

Use Example Testcases

Run



Submission Result: Accepted (/submissions/detail/630733556/) ?

More Details ➤ (/submissions/detail/630733556/)

Share your acceptance!

Copyright © 2022 LeetCode

Help Center (/support) | Jobs (/jobs) | Bug Bounty (/bugbounty) | Online Interview (/interview/) | Students (/student) | Terms (/terms) |

Privacy Policy (/privacy)

United States (/region)