



2157. Groups of Strings

[My Submissions \(/contest/weekly-contest-278/problems/groups-of-strings/submissions/\)](/contest/weekly-contest-278/problems/groups-of-strings/submissions/)
[Back to Contest \(/contest/weekly-contest-278/\)](/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.

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:

Input: `words = ["a","b","ab","cde"]`

Output: `[2,3]`

Explanation:

- `words[0]` can be used to obtain `words[1]` (by replacing 'a' with 'b'), and `words[2]` (by adding 'b').
- `words[1]` can be used to obtain `words[0]` (by replacing 'b' with 'a'), and `words[2]` (by adding 'a').
- `words[2]` can be used to obtain `words[0]` (by deleting 'b'), and `words[1]` (by deleting 'a').
- `words[3]` is not connected to any string in `words`.

Thus, `words` can be divided into 2 groups `["a","b","ab"]` and `["cde"]`. The size of the largest group is 3.

Example 2:

Input: `words = ["a","ab","abc"]`

Output: `[1,3]`

Explanation:

- `words[0]` is connected to `words[1]`.
- `words[1]` is connected to `words[0]` and `words[2]`.
- `words[2]` is connected to `words[1]`.

Since all strings are connected to each other, they should be grouped together.

Thus, the size of the largest group is 3.

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

Constraints:

- $1 \leq \text{words.length} \leq 2 * 10^4$
- $1 \leq \text{words}[i].\text{length} \leq 26$
- $\text{words}[i]$ consists of lowercase English letters only.
- No letter occurs more than once in $\text{words}[i]$.

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

JavaScript



```

1 function DJSet(n) {
2     // parent[i] < 0, -parent[i] is the group size which root is i. example: (i -> parent[i] ->
parent[parent[i]] -> parent[parent[parent[i]]] ...)
3     // parent[i] >= 0, i is not the root and parent[i] is i's parent. example: (...
parent[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]);
8     }
9     function union(x, y) {
10         x = find(x);
11         y = find(y);
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     }
19     function count() { // total connected groups (value < 0)
20         return parent.filter(v => v < 0).length;
21     }
22     function equiv(x, y) {
23         return find(x) == find(y);
24     }
25     function getParent() {
26         return parent;
27     }
28 }
29
30 const groupStrings = (words) => {
31     let n = words.length, ds = new DJSet(n), pre = Array(n).fill(0), m = new Map();
32     for (let i = 0; i < n; i++) {
33         for (const c of words[i]) {
34             pre[i] |= 1 << c.charCodeAt() - 97;
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;
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);

```

```
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](#)[Submit](#)

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 \(/region\)](#)