

5962. Longest Palindrome by Concatenating Two Letter Words

My Submissions (/contest/biweekly-contest-69/problems/longest-palindrome-by-concatenating-two-letter-words/submissions/)

Back to Contest (/contest/biweekly-contest-69/)

You are given an array of strings `words` . Each element of `words` consists of **two** lowercase English letters.

Create the **longest possible palindrome** by selecting some elements from `words` and concatenating them in **any order**. Each element can be selected **at most once**.

Return *the length of the longest palindrome that you can create*. If it is impossible to create any palindrome, return `0` .

A **palindrome** is a string that reads the same forward and backward.

User Accepted:	0
User Tried:	0
Total Accepted:	0
Total Submissions:	0
Difficulty:	Medium

Example 1:

Input: words = ["lc","cl","gg"]
Output: 6
Explanation: One longest palindrome is "lc" + "gg" + "cl" = "lcggcl", of length 6.
Note that "clggcl" is another longest palindrome that can be created.

Example 2:

Input: words = ["ab","ty","yt","lc","cl","ab"]
Output: 8
Explanation: One longest palindrome is "ty" + "lc" + "cl" + "yt" = "tylcclyt", of length 8.
Note that "lcyttycl" is another longest palindrome that can be created.

Example 3:

Input: words = ["cc","ll","xx"]
Output: 2
Explanation: One longest palindrome is "cc", of length 2.
Note that "ll" is another longest palindrome that can be created, and so is "xx".

Constraints:

- 1 <= words.length <= 10⁵
- words[i].length == 2
- words[i] consists of lowercase English letters.

JavaScript

📄

↺

⚙️


```
1 const counter = (a_or_s) => { let m = new Map(); for (const x of a_or_s) m.set(x, m.get(x) + 1 || 1); return m; };
2 const reverse = (s) => { let res = ""; for (let i = s.length - 1; i >= 0; i--) { res += s[i]; } return res; };
3
4 const longestPalindrome = (a) => {
5   let m = counter(a), res = 0;
6   // pr(m);
7   for (const [s, occ] of m) {
8     let rs = reverse(s), n = s.length;
9     // pr("\n", s, rs, res);
10    if (m.has(rs)) {
11      let rsocc = m.get(rs);
12      if (s == rs) {
13        if (occ % 2 == 0) {
14          res += occ * n;
15          m.delete(s);
16        } else {
```

```
17         let remove = occ - 1;
18         res += remove * n;
19         m.set(s, 1);
20     }
21 } else {
22     if (occ < rsocc) {
23         m.delete(s);
24         m.set(rs, rsocc - occ);
25         res += occ * n * 2;
26     } else if (occ > rsocc) {
27         m.delete(rs);
28         m.set(s, occ - rsocc);
29         res += rsocc * n * 2;
30     } else {
31         m.delete(s);
32         m.delete(rs);
33         res += occ * n * 2;
34     }
35 }
36 }
37 // pr("step2", res, m);
38 }
39 let middle = 0;
40 for (const [s, ] of m) {
41     if (counter(s).size == 1) {
42         middle = Math.max(middle, s.length);
43     }
44 }
45 return res + middle;
46 };
```

☐ Custom Testcase[Use Example Testcases](#)[Run](#)[Submit](#)**Submission Result: Accepted** (/submissions/detail/615598411/) ?[More Details](#) (/submissions/detail/615598411/)

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)