



Reverse Shuffle Merge ★

[Problem](#)
[Submissions](#)
[Leaderboard](#)
[Editorial](#)
[Topics](#)

Given a string, A , we define some operations on the string as follows:

- a. $reverse(A)$ denotes the string obtained by reversing string A . Example: $reverse("abc") = "cba"$
- b. $shuffle(A)$ denotes any string that's a permutation of string A . Example: $shuffle("god") \in ['god', 'gdo', 'ogd', 'odg', 'dgo', 'dog']$
- c. $merge(A1, A2)$ denotes any string that's obtained by interspersing the two strings $A1$ & $A2$, maintaining the order of characters in both. For example, $A1 = "abc"$ & $A2 = "def"$, one possible result of $merge(A1, A2)$ could be "abcdef", another could be "abdecf", another could be "adbecf" and so on.

Given a string s such that $s \in merge(reverse(A), shuffle(A))$ for some string A , find the lexicographically smallest A .

For example, $s = abab$. We can split it into two strings of ab . The reverse is ba and we need to find a string to shuffle in to get $abab$. The middle two characters match our reverse string, leaving the a and b at the ends. Our shuffle string needs to be ab . Lexicographically $ab < ba$, so our answer is ab .

Function Description

Complete the reverseShuffleMerge function in the editor below. It must return the lexicographically smallest string fitting the criteria.

reverseShuffleMerge has the following parameter(s):

- s : a string

Input Format

A single line containing the string s .

Constraints

- s contains only lower-case English letters, `ascii[a-z]`
- $1 \leq |s| \leq 10000$

Output Format

Find and return the string which is the lexicographically smallest valid A .

Sample Input 0

```
eggegg
```

Sample Output 0

```
egg
```

Explanation 0

Split "eggegg" into strings of like character counts: "egg", "egg"

$reverse("egg") = "gge"$

$shuffle("egg")$ can be "egg"

"eggegg" belongs to the merge of ("gge", "egg")

The merge is: **eggegg**.

'egg' < 'gge'

**Sample Input 1**

abcdefgabcdefg

Sample Output 1

agfedcb

Explanation 1

Split the string into two strings with like characters: **abcde***fg* and **abcde***fg*.

Reverse **abcde***fg* = **gfedcb***a*

Shuffle **gfedcb***a* can be **bcde***fga*

Merge to **abcdefgabcde***fg*

Sample Input 2

aeiouuoiea

Sample Output 2

aeiou

Explanation 2

Split the string into groups of like characters: **aeiou**

Reverse **aeiou** = **uoiea**

These merge to **aeiouuoiea**

[Change Theme](#)

JavaScript (Node.js)



```
--
27 // Complete the reverseShuffleMerge function below.
28 function reverseShuffleMerge(s) {
29     let map={};
30     s = s.split('').reverse('').join('')
31     for(let item of s.split('')){
32         map[item]=map[item]?map[item]+1:1;
33     }
34     let ref={}
35     for(let key in map){
36         ref[key] = map[key]/2
37     }
38     let solution = [],i=0;
39     while (solution.length<s.length/2){
40         let min_char_pos = -1
41
42         while(true){
43             let c=s[i];
44             if(ref[c]>0&&(min_char_pos<0||c<s[min_char_pos])){
45                 min_char_pos = i;
46             }
47             map[c] -= 1;
48             if(map[c] < ref[c]){
49                 break
50             }
51             i++;
52         }
53         solution.push(s[i]);
54     }
55     return solution.join('');
56 }
```

Line: 41 Col: 1

[Upload Code as File](#) ☐ [Test against custom input](#)[Run Code](#)[Submit Code](#)

Congratulations!

You have passed the sample test cases. Click the submit button to run your code against all the test cases.

✔ Sample Test case 0

✔ Sample Test case 1

✔ Sample Test case 2

Input (stdin)

1 | eggegg

Your Output (stdout)

1 | egg

Expected Output

1 | egg

Download

Download