Reverse Shuffle Merge ☆

Problem Submissions Leaderboard Editorial Topics

RATE THIS CHALLENGE

☆ ☆ ☆ ☆ ☆

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") ∈ ['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 ∈ 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: $abcdefg$ and $abcdefg$.
Reverse $abcdefg$ = $gfedcba$
Shuffle $agfedcb$ can be $bcdefga$
Merge to **a**bcdefga$bcdefg$

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 $aeiou$**uoiea**

Change Theme C++

```
// Complete the reverseShuffleMerge function below.
//https://www.youtube.com/watch?v=_QQe2TQ2o_4&ab_channel=KuldipGhotane
string reverseShuffleMerge(string s) {
    int n = s.size();
    int unused[26]={0};
    int used[26]={0};
    int required[26]={0};
    char res[10000];
    int j=0;
    //filling array mapping char freq
    for(int i=0; i<n; i++)
        unused[s[i]-'a']++;
    for(int i=0; i<26; i++)
        required[i]= unused[i]/2;
     // last character
    char ch = s[n-1];
    int ch_position = ch-'a'; // index present in above arrays
    res[j++]=ch;
    unused[ch_position]--;
    used[ch_position]++;
    //rest of char     //add ---- req is smaller than pres
    // ch smaller     //ch bigger
    for(int i=n-2 ; i>=0; i--)
    {   ch =s[i];
        ch_position = ch-'a';
```

```
            // to add or not
        if(used[ch_position]< required[ch_position])
        {   //add char
            if(ch>res[j-1])
            {   res[j++]=ch;
                unused[ch_position]--;
                used[ch_position]++;
            }
            else{
                //check bigger ele -- we re  //pop
                while(j>0 && ch<res[j-1] && used[res[j-1]-'a']-1 + unused[res[j-1]-'a']
>= required[res[j-1]-'a'])
                {   used[res[--j]-'a']--;
                }
                res[j++]=ch;
                unused[ch_position]--;
                used[ch_position]++;
            }
        }
        else
        {// rejecting / discarding the perticulr char
            unused[ch_position]--;
        }
    }
 return res;
}
```

Line: 5 Col: 40

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) Download

```
1 eggegg
```

Your Output (stdout)

```
1 egg
```

Expected Output Download

```
1 egg
```