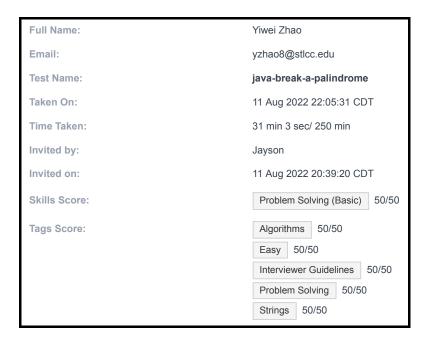


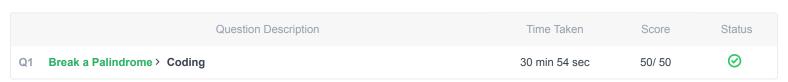
You can view this report online at: https://www.hackerrank.com/x/tests/1416996/candidates/43352591/report

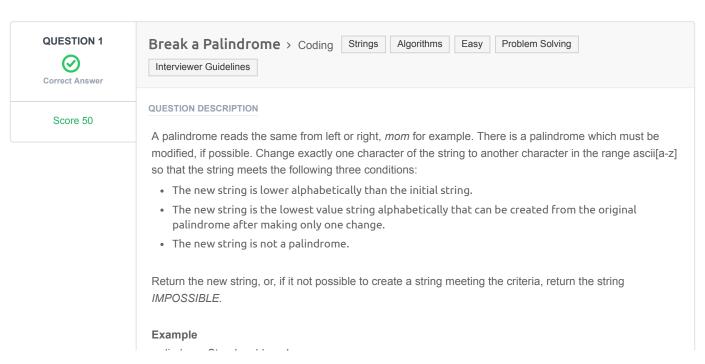




Recruiter/Team Comments:

No Comments.





palindromeStr = 'aaabbaaa

- Possible strings lower alphabetically than 'aaabbaaa' after one change are ['aaaabaaa', 'aaabaaaa'].
- 'aaaabaaa' is not a palindrome and is the lowest string that can be created from palindromeStr.

Function Description

Complete the function breakPalindrome in the editor below.

breakPalindrome has the following parameter(s): string palindromeStr: the original string

Returns:

string: the resulting string, or IMPOSSIBLE if one cannot be formed

Constraints

- 1 ≤ length of *palindromeStr* ≤ 1000
- palindromeStr is a palindrome
- palindromeStr contains only lowercase English letters

▼ Input Format For Custom Testing

Locked stub code in the editor reads a single string, *palindromeStr*, from stdin and passes it to the function.

▼ Sample Case 0

Sample Input For Custom Testing

```
STDIN Function
-----
bab → palindromeStr = 'bab'
```

Sample Output

```
aab
```

Explanation

- Possible strings lower alphabetically than 'bab' after one change are ['aab', 'baa'].
- 'aab' is not a palindrome and is the lowest string that can be created from palindromeStr.

▼ Sample Case 1

Sample Input For Custom Testing

```
STDIN Function
----
aaa → palindromeStr = 'aaa'
```

Sample Output

```
IMPOSSIBLE
```

Explanation

- There are no strings lower alphabetically than 'aaa' after one change.
- So, it is IMPOSSIBLE to create a string lower than 'aaa'.

▼ Sample Case 2

Sample Input For Custom Testing

```
STDIN Function
-----
acca → palindromeStr = 'acca'
```

Sample Output

aaca

Explanation

- Possible strings lower alphabetically than 'acca' after one change are ['abca', 'aaca', 'acba', 'acaa'].
- 'aaca' is not a palindrome and is the lowest string that can be created from palindromeStr.

INTERVIEWER GUIDELINES

▼ Hint 1

Try to think if we want to make it lexicographically smallest, we must replace the first character which we can replace with a smaller one.

▼ Hint 2

We should definitely replace a character with 'a' only as it would make it minimal.

▼ Hint 3

Other than an all 'a' string, what are the possible cases when it is impossible to make a non-palindromic smaller string?

Ans - For odd lengths string, if there are all 'a' except the center character as replacing the center character would still result in palindrome.

▼ Solution

Concepts Covered: Problem Solving, Greedy Algorithms

The problem tests the candidate's ability to use the basic knowledge of Greedy algorithms and solving it by greedily selecting the optimal strategy.

Optimal Solution:

We need to find the first character in the string which is not 'a' and replace it. We need it only for the first n/2 characters as it is a palindrome.

```
def breakPalindrome(palindromeStr):
    # store in a mutable structure
    sa = list(palindromeStr)
    # only consider 1st half (it's a palindrome initially)
    n = len(sa)//2 # returns integer floor of division
    # if any character is not 'a', replace it with 'a'
    # and return the new string
    for i in range(n):
        if not sa[i] == 'a':
            sa[i] = 'a'
            return ''.join(sa)
    # all characters are 'a' or the string is 1 character (always a palindrome),
    # so it's not possible
    return 'IMPOSSIBLE'
```

Brute Force:

We can iterate over the string and try changing each character to any other character. For the new string, we can check the following two conditions:

- 1. The new string should not be a palindrome.
- 2. The new string should be alphabetically smaller than the original string.

We take the alphabetically smallest string as our answer among all such strings. If there is no string

satisfying the above two conditions we return "IMPOSSIBLE" as the answer.

Time Complexity - O(N x N x 26)

▼ Complexity Analysis

Time Complexity - O(N) - We iterate over the array only once and therefore complexity is of the order O(N).

Space Complexity - O(1) - No extra space is required.

▼ Follow up Question

Suppose you are allowed to rearrange the characters of the string now. What is the smallest possible string with at most one replacement which is also smaller than the original string and is not a palindrome?

We can sort the characters of the string. A normal sort function would take $O(N \log N)$ time. We can do counting sort instead which runs in O(N) time and since there are only 26 characters in O(1) space. We can repeat the same above algorithm now. Note that here we need to check for non-'a' character till n and not just n/2.

Psuedo Code -

```
string breakPalindrome(string s) {
   int count[26];
   memset(count, 0, sizeof count);
   for(auto i:s)
       count[i - 'a']++;
    s.clear();
    for (int i=0; i<26; i++)
        while(count[i] > 0)
            s.push_back(i + 'a');
            count[i]--;
    int n = s.size();
    for(int i=0; i<n; i++)
        if(s[i]!='a')
            s[i] = 'a';
            return s;
    return "IMPOSSIBLE";
}
```

CANDIDATE ANSWER

Language used: Java 15

```
class Result {

/*
    * Complete the 'breakPalindrome' function below.

*
    * The function is expected to return a STRING.
    * The function accepts STRING palindromeStr as parameter.

*/

public static String breakPalindrome(String palindromeStr) {
```

```
ArrayList<String> possibleStrings = new ArrayList<String>();
           //step1: get all possible strings which satisfy the 1st condition: the
15 new string is lower alphabetically than the initial string
           for (int i = 0; i < palindromeStr.length(); i ++) {</pre>
               char letter = palindromeStr.charAt(i);
               //compare the letter with a (97), if the letter is great than a,
19 substitue it with one smaller letter, and decrement the letter'value by 1
               while (letter > 97) {
                   char newLetter = (char) (letter -1);
                   String newString;
                   if (i < palindromeStr.length() - 1) {</pre>
                        newString = palindromeStr.substring(0,i) + newLetter +
                   palindromeStr.substring(i+1);
                    } else {
                        newString = palindromeStr.substring(0,i) + newLetter;
                   possibleStrings.add(newString);
                   letter --;
               }
           }
           //step2: find all the palindrome strings, and reset their value to
35 null
           for (int i = 0; i < possibleStrings.size(); i ++) {</pre>
               String str = possibleStrings.get(i);
               while (str.length()>1) {
           //check 1st char and the last char, if they are equal, delete the 1st
41 char and the last char
                    if (str.charAt(0) == str.charAt(str.length()-1)) {
                        str = str.substring(1, str.length()-1);
45
                   else {
           //when the 1st char and the last char are not equal, leave the loop
47
                       break:
                   }
               }
           //if the string is palindrome, then set its value to null
               if (str.length() <=1) {</pre>
                   System.out.print("reach here");
                   possibleStrings.set(i, null);
                   System.out.print(possibleStrings.get(i));
           //if there are more than zero string satisfy 1st condition, and the
61 1st string is not null, assign the 1st string to lowestStr, then compare it
62 with other string to find the lowest value string
           if (possibleStrings.size()>0 && possibleStrings.get(0) != null) {
               String lowestStr = possibleStrings.get(0);
               for (int i = 1; i < possibleStrings.size(); i++) {</pre>
                    //if string is null(palindrome), then skip this interation
                   if (possibleStrings.get(i) == null) {
                       continue;
                    else{
                    for (int j =0; j < palindromeStr.length(); j ++) {</pre>
                       if (lowestStr.charAt(j) <</pre>
73 possibleStrings.get(i).charAt(j)) {
                            break;
                        } else if (lowestStr.charAt(j) >
```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
Testcase 0	Easy	Sample case	Success	1	0.155 sec	35.4 KB
Testcase 1	Easy	Sample case	Success	1	0.1113 sec	34.8 KB
Testcase 2	Easy	Sample case	Success	1	0.0738 sec	31.4 KB
Testcase 3	Easy	Sample case	Success	4	0.1123 sec	31.4 KB
Testcase 4	Easy	Hidden case	Success	4	0.111 sec	33.1 KB
Testcase 5	Easy	Hidden case	Success	4	0.0899 sec	34.4 KB
Testcase 6	Easy	Hidden case	Success	5	0.1316 sec	34.8 KB
Testcase 7	Easy	Hidden case	Success	5	0.1271 sec	35.3 KB
Testcase 8	Hard	Sample case	Success	5	0.0673 sec	31.3 KB
Testcase 9	Hard	Hidden case	Success	5	0.1273 sec	35.5 KB
Testcase 10	Hard	Hidden case	Success	5	0.0918 sec	35.5 KB
Testcase 11	Hard	Hidden case	Success	5	0.121 sec	34.6 KB
Testcase 12	Hard	Hidden case	Success	5	0.0966 sec	36.1 KB

No Comments

PDF generated at: 12 Aug 2022 03:38:24 UTC