Average Rating: 3.52 (66 votes)

Approach #1: Greedy [Accepted]

Intuition

A palindrome consists of letters with equal partners, plus possibly a unique center (without a partner). The letter `i` from the left has its partner `i` from the right. For example in `'abcba'`, `'aa'` and `'bb'` are partners, and `'c'` is a unique center.

Imagine we built our palindrome. It consists of as many partnered letters as possible, plus a unique center if possible. This motivates a greedy approach.

Algorithm

For each letter, say it occurs `v` times. We know we have `v // 2 * 2` letters that can be partnered for sure. For example, if we have `'aaaaa'`, then we could have `'aaaa'` partnered, which is `5 // 2 * 2 = 4` letters partnered.

At the end, if there was any `v % 2 == 1`, then that letter could have been a unique center. Otherwise, every letter was partnered. To perform this check, we will check for `v % 2 == 1` and `ans % 2 == 0`, the latter meaning we haven't yet added a unique center to the answer.

Java Python Copy

```python
class Solution:
    def longestPalindrome(self, s):
        ans = 0
        for v in collections.Counter(s).itervalues():
            ans += v / 2 * 2
            if ans % 2 == 0 and v % 2 == 1:
                ans += 1
        return ans
```

Complexity Analysis

- Time Complexity: $O(N)$, where N is the length of `s`. We need to count each letter.
- Space Complexity: $O(1)$, the space for our count, as the alphabet size of `s` is fixed. We should also consider that in a bit complexity model, technically we need $O(\log N)$ bits to store the count values.

Report Article Issue

Python3 Autocomplete

```python
class Solution:
    def longestPalindrome(self, s: str) -> int:

        di = {}
        for i in s:

            if i not in di:
                di.update({i: 1})
            else :
                di[i] += 1

        print(di)
        output,c = 0,0
        for i in di.values() :

            if i%2 ==0 :
                output += i
            else :
                c=1
                output += i-1

        return (output + c)
```

Testcase Run Code Result Debugger

Accepted Runtime: 42 ms

Your input "abccccdd"

stdout {'a': 1, 'b': 1, 'c': 4, 'd': 2}

Output 7 Diff

Expected 7