409 Longest Palindrome

2018年4月11日 9:33

Given a string which consists of lowercase or uppercase letters, find the length of the longest palindromes that can be built with those letters.
This is case sensitive, for example "Aa" is not considered a palindrome here.
Note:
Assume the length of given string will not exceed 1,010.
Example:
Input:
"abccccdd"
Output:
7
Explanation:
One longest palindrome that can be built is "dccaccd", whose length is 7.

来自 <https://leetcode.com/problems/longest-palindrome/description/>
给定一个包含大写字母和小写字母的字符串，找到通过这些字母构造成的最长的回文串。
在构造过程中，请注意区分大小写。比如 "Aa" 不能当做一个回文字符串。
注意:
假设字符串的长度不会超过 1010。

Solution for Python3:

```
class Solution1:
    def longestPalindrome(self, s):
        """
        :type s: str
        :rtype: int
        """
        d = {}
        for i in s:
            if not d.get(i):
                d[i] = 1
            else:
                d[i] += 1
        res = 0
        for i in d.items():
                res += i[1] // 2 * 2
        if res < len(s):
            return res + 1
       return res

class Solution2:
    def longestPalindrome(self, s):
        """
        :type s: str
        :rtype: int
        """
        odd = 0
        for c in range(ord('A'), ord('z') + 1):
            odd += s.count(chr(c)) & 1
        return len(s) - odd + int(odd > 0)

class Solution3:
```

```python
    def longestPalindrome(self, s):
        """
        :type s: str
        :rtype: int
        """
        import collections
        odd = sum(v & 1 for v in collections.Counter(s).values())
        return len(s) - odds + bool(odd)

class Solution4:
    def longestPalindrome(self, s):
        """
        :type s: str
        :rtype: int
        """
        import collections
        # 1->二进制：0000 ... 0000 0001
        #~1->二进制：1111 ... 1111 1110
        evenpart = sum(v & ~1 for v in collections.Counter(s).values())
        return evenpart + (evenpart < len(s))
```

Solution for C++:

```cpp
class Solution1 {
public:
    int longestPalindrome(string s) {
        int odd = 0;
        for (char c = 'A'; c < = 'z'; c++) {
            odd += count(s.begin(), s.end(), c) & 1;
        }
        return s.length() - odd + (odd > 0);
    }
};

class Solution2 {
public:
    int longestPalindrome(string s) {
        if (s.empty() || s.length() == 0)
            return 0;
        unordered_set<char> set;
        int cnt = 0; //成对出现的个数
        for (int i = 0; i < s.length(); i++) {
            if (set.count(s[i])) {
                set.erase(s[i]);
                cnt++;
            } else {
                set.insert(s[i]);
            }
        }
        if (s.empty())
            return cnt * 2;
        return cnt * 2 + 1;
    }
};
```