LeetCode409\_LongestPalindrome\_最长回文串\_Easy

# LeetCode409\_LongestPalindrome\_最长回文串\_Easy

## 题目介绍

<https://leetcode.com/problems/longest-palindrome/description/>

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.

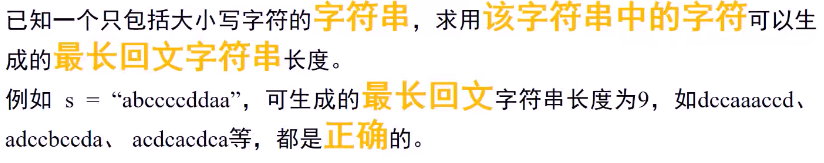
Note:Assume the length of given string will not exceed 1,010.

Example:

Input: "abccccdd"

Output: 7

Explanation:One longest palindrome that can be built is "dccaccd", whose length is 7.



## 思路分析

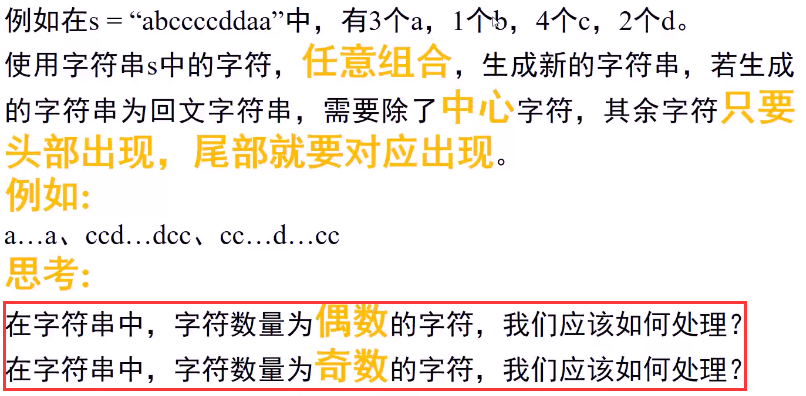
\* 思路分析：利用哈希表解决。

\* 首先只有字母，则建立一个大小为52的哈希表即可，为了安全，若出现非字母，故设置哈希表大小为53；

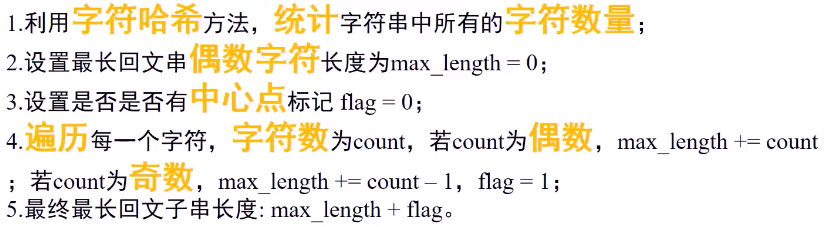
\* 利用哈希表统计字母出现个数。对于出现偶数个的字母，完全可以对称出现，直接加到总数中；

\* 因为回文字符串，只有最中间出现一个单个的字母，因此首先将统计时，先对奇数减1，设置flag标志

\* 是否出现过奇数个字符，若有最后再加1。



具体思路：



## Java代码

public int **longestPalindrome**(String s) {

if (s == null||s.length()==0) return 0;

char[] chs = s.toCharArray();

int hashLen = 53,count = 0;

int[] hashTableCount = new int[hashLen];//第53位存放非字母字符数目

for(char c:chs) hashTableCount[hash(c)]++;

boolean oddFlag = false;

for(int i = 0;i < hashLen;i++){

**count += hashTableCount[i];**

if((hashTableCount[i]&1)==1){//奇数

count--;//减1

**oddFlag = true;//存在奇数**

}

}

**return oddFlag?count+1:count;**

}

//求解字母的hash值

public int hash(char c){

if('A'<= c && c <='Z') return c-'A';

if('a'<= c && c <='z') return c-'a'+ 26;

return 53;

}