# 题目

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。

示例 1:

输入:

"abccccdd"

输出:

7

解释:

我们可以构造的最长的回文串是"dccaccd", 它的长度是 7。

# 分析

## 方法一：哈希表

**思路：**

1、用哈希表统计每个字符出现的次数

2、判断奇偶性：

若字符出现次数为偶数，则肯定能够组成回文串，计入累加器

若为奇数，添加个数-1并计入累加器，并且标记存在中心字符

**代码：**

class Solution {

public:

int longestPalindrome(string s) {

unordered\_map<char, int> hash;

for (auto x : s) hash[x] ++;

int res = 0, center = 0;

for (auto item : hash){

if (item.second % 2 == 0)

res += item.second;

else{

res += item.second - 1；

center = 1;

}

}

return res + center;

}

};

复杂度分析：

时间复杂度：最多需要遍历输字符串中n个字符，所以时间复杂度为O(n)。  
 空间复杂度：根据unordered\_map的空间复杂度为O(n)。

**另一种写法：**

class Solution {

public:

int longestPalindrome(string s) {

unordered\_map<char, int> count;

int ans = 0;

for (char c : s)

++count[c];

for (auto p : count) {

int v = p.second;

ans += v / 2 \* 2;

if (v % 2 == 1 and ans % 2 == 0)

++ans;

}

return ans;

}

};

**复杂度分析：**

时间复杂度：O(N)，其中N为字符串s的长度。我们需要遍历每个字符一次。

空间复杂度：O(S)，其中S为字符集大小。在Java代码中，我们使用了一个长度为128的数组，存储每个字符出现的次数，这是因为字符的ASCII值的范围为[0, 128)。而由于题目中保证了给定的字符串s只包含大小写字母，因此我们也可以使用哈希映射（HashMap）来存储每个字符出现的次数，例如Python和C++的代码。如果使用哈希映射，最多只会存储52个（即小写字母与大写字母的数量之和）键值对。