# Leetcode 题解 - 字符串

* [Leetcode 题解 - 字符串](#leetcode-题解---字符串)
  + [1. 字符串循环移位包含](#1-字符串循环移位包含)
  + [2. 字符串循环移位](#2-字符串循环移位)
  + [3. 字符串中单词的翻转](#3-字符串中单词的翻转)
  + [4. 两个字符串包含的字符是否完全相同](#4-两个字符串包含的字符是否完全相同)
  + [5. 计算一组字符集合可以组成的回文字符串的最大长度](#5-计算一组字符集合可以组成的回文字符串的最大长度)
  + [6. 字符串同构](#6-字符串同构)
  + [7. 回文子字符串个数](#7-回文子字符串个数)
  + [8. 判断一个整数是否是回文数](#8-判断一个整数是否是回文数)
  + [9. 统计二进制字符串中连续 1 和连续 0 数量相同的子字符串个数](#9-统计二进制字符串中连续-1-和连续-0-数量相同的子字符串个数)

## 1. 字符串循环移位包含

[编程之美 3.1](#)

s1 = AABCD, s2 = CDAA  
Return : true

给定两个字符串 s1 和 s2，要求判定 s2 是否能够被 s1 做循环移位得到的字符串包含。

s1 进行循环移位的结果是 s1s1 的子字符串，因此只要判断 s2 是否是 s1s1 的子字符串即可。

## 2. 字符串循环移位

[编程之美 2.17](#)

s = "abcd123" k = 3  
Return "123abcd"

将字符串向右循环移动 k 位。

将 abcd123 中的 abcd 和 123 单独翻转，得到 dcba321，然后对整个字符串进行翻转，得到 123abcd。

## 3. 字符串中单词的翻转

[程序员代码面试指南](#)

s = "I am a student"  
Return "student a am I"

将每个单词翻转，然后将整个字符串翻转。

## 4. 两个字符串包含的字符是否完全相同

242. Valid Anagram (Easy)

[Leetcode](https://leetcode.com/problems/valid-anagram/description/) / [力扣](https://leetcode-cn.com/problems/valid-anagram/description/)

s = "anagram", t = "nagaram", return true.  
s = "rat", t = "car", return false.

可以用 HashMap 来映射字符与出现次数，然后比较两个字符串出现的字符数量是否相同。

由于本题的字符串只包含 26 个小写字符，因此可以使用长度为 26 的整型数组对字符串出现的字符进行统计，不再使用 HashMap。

public boolean isAnagram(String s, String t) {  
 int[] cnts = new int[26];  
 for (char c : s.toCharArray()) {  
 cnts[c - 'a']++;  
 }  
 for (char c : t.toCharArray()) {  
 cnts[c - 'a']--;  
 }  
 for (int cnt : cnts) {  
 if (cnt != 0) {  
 return false;  
 }  
 }  
 return true;  
}

## 5. 计算一组字符集合可以组成的回文字符串的最大长度

409. Longest Palindrome (Easy)

[Leetcode](https://leetcode.com/problems/longest-palindrome/description/) / [力扣](https://leetcode-cn.com/problems/longest-palindrome/description/)

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

使用长度为 256 的整型数组来统计每个字符出现的个数，每个字符有偶数个可以用来构成回文字符串。

因为回文字符串最中间的那个字符可以单独出现，所以如果有单独的字符就把它放到最中间。

public int longestPalindrome(String s) {  
 int[] cnts = new int[256];  
 for (char c : s.toCharArray()) {  
 cnts[c]++;  
 }  
 int palindrome = 0;  
 for (int cnt : cnts) {  
 palindrome += (cnt / 2) \* 2;  
 }  
 if (palindrome < s.length()) {  
 palindrome++; // 这个条件下 s 中一定有单个未使用的字符存在，可以把这个字符放到回文的最中间  
 }  
 return palindrome;  
}

## 6. 字符串同构

205. Isomorphic Strings (Easy)

[Leetcode](https://leetcode.com/problems/isomorphic-strings/description/) / [力扣](https://leetcode-cn.com/problems/isomorphic-strings/description/)

Given "egg", "add", return true.  
Given "foo", "bar", return false.  
Given "paper", "title", return true.

记录一个字符上次出现的位置，如果两个字符串中的字符上次出现的位置一样，那么就属于同构。

public boolean isIsomorphic(String s, String t) {  
 int[] preIndexOfS = new int[256];  
 int[] preIndexOfT = new int[256];  
 for (int i = 0; i < s.length(); i++) {  
 char sc = s.charAt(i), tc = t.charAt(i);  
 if (preIndexOfS[sc] != preIndexOfT[tc]) {  
 return false;  
 }  
 preIndexOfS[sc] = i + 1;  
 preIndexOfT[tc] = i + 1;  
 }  
 return true;  
}

## 7. 回文子字符串个数

647. Palindromic Substrings (Medium)

[Leetcode](https://leetcode.com/problems/palindromic-substrings/description/) / [力扣](https://leetcode-cn.com/problems/palindromic-substrings/description/)

Input: "aaa"  
Output: 6  
Explanation: Six palindromic strings: "a", "a", "a", "aa", "aa", "aaa".

从字符串的某一位开始，尝试着去扩展子字符串。

private int cnt = 0;  
  
public int countSubstrings(String s) {  
 for (int i = 0; i < s.length(); i++) {  
 extendSubstrings(s, i, i); // 奇数长度  
 extendSubstrings(s, i, i + 1); // 偶数长度  
 }  
 return cnt;  
}  
  
private void extendSubstrings(String s, int start, int end) {  
 while (start >= 0 && end < s.length() && s.charAt(start) == s.charAt(end)) {  
 start--;  
 end++;  
 cnt++;  
 }  
}

## 8. 判断一个整数是否是回文数

9. Palindrome Number (Easy)

[Leetcode](https://leetcode.com/problems/palindrome-number/description/) / [力扣](https://leetcode-cn.com/problems/palindrome-number/description/)

要求不能使用额外空间，也就不能将整数转换为字符串进行判断。

将整数分成左右两部分，右边那部分需要转置，然后判断这两部分是否相等。

public boolean isPalindrome(int x) {  
 if (x == 0) {  
 return true;  
 }  
 if (x < 0 || x % 10 == 0) {  
 return false;  
 }  
 int right = 0;  
 while (x > right) {  
 right = right \* 10 + x % 10;  
 x /= 10;  
 }  
 return x == right || x == right / 10;  
}

## 9. 统计二进制字符串中连续 1 和连续 0 数量相同的子字符串个数

696. Count Binary Substrings (Easy)

[Leetcode](https://leetcode.com/problems/count-binary-substrings/description/) / [力扣](https://leetcode-cn.com/problems/count-binary-substrings/description/)

Input: "00110011"  
Output: 6  
Explanation: There are 6 substrings that have equal number of consecutive 1's and 0's: "0011", "01", "1100", "10", "0011", and "01".

public int countBinarySubstrings(String s) {  
 int preLen = 0, curLen = 1, count = 0;  
 for (int i = 1; i < s.length(); i++) {  
 if (s.charAt(i) == s.charAt(i - 1)) {  
 curLen++;  
 } else {  
 preLen = curLen;  
 curLen = 1;  
 }  
  
 if (preLen >= curLen) {  
 count++;  
 }  
 }  
 return count;  
}