# leetcode个人刷题笔记

|  |  |
| --- | --- |
| 撰写人 | 刘恒辉 |
| 版本 |  |
| 时间 | 2019.7.5 |

1. 原题目(13)

罗马数字包含以下七种字符: I， V， X， L，C，D 和 M。

例如， 罗马数字 2 写做 II ，即为两个并列的 1。12 写做 XII ，即为 X + II 。 27 写做  XXVII, 即为 XX + V + II 。

通常情况下，罗马数字中小的数字在大的数字的右边。但也存在特例，例如 4 不写做 IIII，而是 IV。数字 1 在数字 5 的左边，所表示的数等于大数 5 减小数 1 得到的数值 4 。同样地，数字 9 表示为 IX。这个特殊的规则只适用于以下六种情况：

I 可以放在 V (5) 和 X (10) 的左边，来表示 4 和 9。

X 可以放在 L (50) 和 C (100) 的左边，来表示 40 和 90。

C 可以放在 D (500) 和 M (1000) 的左边，来表示 400 和 900。

给定一个罗马数字，将其转换成整数。输入确保在 1 到 3999 的范围内。

1. 题目大意

根据输入的罗马数字字符，输出相应的阿拉伯数字。

1. 思路分析

本人的思路：

首先构建一个替换函数，用来替换单个罗马数字字符。

然后在转换函数中针对特殊情况来进行调整，最后将各个字符对应数字的和输出即可。

1. 具体代码

本人的代码：

int replace(char \* str)

{

int num = 0;

switch (\*str)

{

case 'I':

num = 1;

break;

case 'V':

num = 5;

break;

case 'X':

num = 10;

break;

case 'L':

num = 50;

break;

case 'C':

num = 100;

break;

case 'D':

num = 500;

break;

case 'M':

num = 1000;

break;

}

return num;

}

int romanToInt(char \* s)

{

int sum = 0,len=strlen(s);

char \* p;

while (len--)

{

sum += replace(&s[len]);

if (s[len] == 'I' &&( s[len + 1] == 'V'|| s[len + 1] == 'X'))

{

sum -= 2;

}

if (s[len] == 'X' && (s[len + 1] == 'L' || s[len + 1] == 'C'))

{

sum -= 20;

}

if (s[len] == 'C' && (s[len + 1] == 'D' || s[len + 1] == 'M'))

{

sum -= 200;

}

}

return sum;

}

1. 来源<https://leetcode-cn.com/problems/roman-to-integer/>
2. 原题目(387)

给定一个字符串，找到它的第一个不重复的字符，并返回它的索引。如果不存在，则返回 -1。

1. 题目大意

识别字符串中的第一个唯一的字符并且返回他的序号

1. 思路分析

本人的思路：

构建一个容量位26的数组，每遍历字符串中的一个字母，就在数组中的特定位置的值覆盖为一，覆盖两次或以上为二，第二次遍历新建数组，将值为一的数组序号返回。

1. 具体代码

本人的代码：

int firstUniqChar(char \* s)

{

char memo[26] = { 0 };

for (int i = 0; s[i]; i++)

{

memo[s[i] - 97] = memo[s[i] - 97] == 0 ? 1 : 2;

}

for (int i = 0; s[i]; i++)

{

if (memo[s[i] - 97] == 1)

return i;

}

return -1;

}

1. 来源<https://leetcode-cn.com/problems/first-unique-character-in-a-string/comments/>
2. 原题目(344)

编写一个函数，其作用是将输入的字符串反转过来。输入字符串以字符数组 char[] 的形式给出。

不要给另外的数组分配额外的空间，你必须原地修改输入数组、使用 O(1) 的额外空间解决这一问题。

你可以假设数组中的所有字符都是 ASCII 码表中的可打印字符。

1. 题目大意

将字符串倒序。

1. 思路分析

本人的思路：

设置容器变量temp，循环n/2次，将第i个与第n-1-i个进行交换。

1. 具体代码

本人的代码：

void reverseString(char \* s , int sSize)

{

char temp;

int time=sSize%2==1?sSize/2+1:sSize/2;

for (int i = 0; i < time; i++)

{

temp = s[i];

s[i] = s[sSize - 1 - i];

s[sSize - 1 - i]=temp;

}

}

1. 来源<https://leetcode-cn.com/problems/reverse-string/submissions/>
2. 原题目(242)

给定两个字符串 s 和 t ，编写一个函数来判断 t 是否是 s 的字母异位词。

1. 题目大意

**说明:**  
你可以假设字符串只包含小写字母。

**进阶:**  
如果输入字符串包含 unicode 字符怎么办？你能否调整你的解法来应对这种情况？

1. 思路分析

本人的思路：

将两个字符串按从小到大排序，如果两个只是顺序不同，但是包含字符完全匹配，那么排序后的两个字符串一定是相同的。

1. 具体代码

本人的代码：

int cmp(const void \* a , const void \* b)

{

return(\*(char \*)a - \*(char \*)b);

}

bool isAnagram(char \* s , char \* t)

{

int s1 = strlen(s) , t1 = strlen(t);

qsort(s , s1 , sizeof(s[0]) , cmp);

qsort(t , t1 , sizeof(t[0]) , cmp);

return !strcmp(s , t);

}

1. 来源<https://leetcode-cn.com/problems/valid-anagram/>

一、原题目(171)

给定一个Excel表格中的列名称，返回其相应的列序号。

1. 题目大意

根据特定的运算规则，将字符串转化为数字。

三、思路分析

本人的思路：

将26进制转化为10进制为所求。

四、具体代码

本人的代码：

int titleToNumber(char \* s){

int num=0,len=strlen(s);

for (int i = 0; i < len; ++i)

{

num=num\*26+(s[i]-65)+1;

}

return num;

}

五、来源<https://leetcode-cn.com/problems/excel-sheet-column-number/>

一、原题目(9)

判断一个整数是否是回文数。回文数是指正序（从左向右）和倒序（从右向左）读都是一样的整数。

二、题目大意

一个数从前往后读和从后往前读一样就是回文数。

三、思路分析

本人的思路：

使样本不断除以十，然后按照顺序合成一个新的数，如果两数相等就是回文数。

四、具体代码

本人的代码：

bool isPalindrome(int x)

{

long temp=0;

int i = 0;

int X = x;

if (x < 0)

return false;

else

while (x)

{

temp = (temp + x % 10) \* 10;

x /= 10;

}

temp /= 10;

printf("%d\n" , temp);

return X==temp?true:false;

}

五、来源<https://leetcode-cn.com/problems/palindrome-number/>

一、原题目(28)

实现 strStr() 函数。

给定一个 haystack 字符串和一个 needle 字符串，在 haystack 字符串中找出 needle 字符串出现的第一个位置 (从0开始)。如果不存在，则返回  -1。

二、题目大意

如果第二个字符串存在在第一个字符串中，那么将第一个字符串中与第二个字符串匹配的第一个字符所在的位置输出。

三、思路分析

本人的思路：

运用KMP算法来解决字符匹配的问题。

四、具体代码

本人的代码：

int strStr(char \* haystack , char \* needle)

{

int needlelen = 0;

while (needle[needlelen] != '\0')

{

needlelen++;

}

if (needlelen == 0)

{

return 0;

}

// 求next数组

int \* next = (int \*)malloc(sizeof(int) \* needlelen);

int p\_len = needlelen;

int i = 0;

int j = -1;

next[0] = -1;

while (i < p\_len - 1)

{

if (j == -1 || needle[i] == needle[j])

{

i++;

j++;

next[i] = j;

}

else

j = next[j];

}

//进行KMP

int s\_len = 0;

while (haystack[s\_len] != '\0')

{

s\_len++;

}

p\_len = needlelen;

i = 0;

j = 0;

while (i < s\_len && j < p\_len)

{

if (j == -1 || haystack[i] == needle[j])

{

i++;

j++;

}

else

j = next[j];

}

if (j == p\_len)

{

return i - j;

}

return -1;

}

五、来源<https://leetcode-cn.com/problems/implement-strstr/>