# leetcode个人刷题笔记

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

一、原题目(7)

给出一个 32 位的有符号整数，你需要将这个整数中每位上的数字进行反转。

注意:

假设我们的环境只能存储得下 32 位的有符号整数，则其数值范围为 [−2^31,  2^31 − 1]。请根据这个假设，如果反转后整数溢出那么就返回 0。

二、题目大意

将数字各位顺序颠倒，保留原有的正负号。

三、思路分析

本人的思路：

不断得到数字除以十的余数，重新构建一个数据，该数据加上正负号即为所求，但是需要考虑数据溢出情况，所以在循环时加上溢出与否的判断即可

四、具体代码

本人的代码：

int reverse(int x)

{

int rev = 0;

while (x != 0)

{

int pop = x % 10;

x /= 10;

if (rev > INT\_MAX / 10 || (rev == INT\_MAX / 10 && pop > 7)) return 0;

if (rev < INT\_MIN / 10 || (rev == INT\_MIN / 10 && pop < -8)) return 0;

rev = rev \* 10 + pop;

}

return rev;

}

五、来源<https://leetcode-cn.com/problems/reverse-integer/>

1. 原题目(383)

给定一个赎金信 (ransom) 字符串和一个杂志(magazine)字符串，判断第一个字符串ransom能不能由第二个字符串magazines里面的字符构成。如果可以构成，返回 true ；否则返回 false。

(题目说明：为了不暴露赎金信字迹，要从杂志上搜索各个需要的字母，组成单词来表达意思。)

注意：

你可以假设两个字符串均只含有小写字母。

1. 题目大意

判断第二个字符串中是否包含第一个字符串。

三、思路分析

本人的思路：

将第二个字符串作为模板字符串，第一个字符串依次拿出字符与第二个比较，当匹配一次，就删除第二个字符串中相应的字符。如果能遍历完第一个字符串中所有字符，就返回true

四、具体代码

本人的代码：

bool fun(char \* str\_1 , char \* str\_2)//1为完整字符串，2为目标字符，搜索字符串1中是否由2

{

for (int i = 0; i < strlen(str\_1); i++)

{

if (str\_1[i] == \*str\_2)

{

return true;

str\_1[i] = ' ';

break;

}

}

return false;

}

bool canConstruct(char \* ransomNote , char \* magazine)

{

int len\_ran = strlen(ransomNote);

int len\_mag = strlen(magazine);

int sign = 1;

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

{

if (!fun(magazine , &ransomNote[i]))

{

sign = 0; break;

}

}

return sign == 1 ? true : false;

}

五、来源<https://leetcode-cn.com/problems/ransom-note/>

1. 原题目(3)

给定一个字符串，请你找出其中不含有重复字符的 **最长子串**的长度。

二、题目大意

将字符串中最长连续且字符各不相同的长度输出。

三、思路分析

本人的思路：

循环字符串长度，一次比较每一个字符，循环结束后会得到一个最长且有效的的长度。

四、具体代码

本人的代码：

int contains(char \* s , char c , int len)

{

int i = 0;

while (s[i] != NULL && i < len)

{

if (s[i] == c)

{

return 1;

}

i++;

}

return 0;

}

int lengthOfLongestSubstring(char \* s)

{

int len = 0;

while (s[len] != NULL)

{

// 计算字符串s的长度

len++;

}

int i = 0;

int j = 1;

int maxLen = 0;

// 如果从i位开始的字符串到末尾的长度小于最大字符串的长度，则不需要继续计算

while (s[i] != NULL && (len - i > maxLen))

{

while (s[j] != NULL && (contains(&s[i] , s[j] , j - i) == 0))

{

// 遍历子串，如果前面几位都不包含当前的字符，则继续遍历；如果包含，则退出循环，计算下一个子串

// 如果之前计算过的子串，则不继续计算

// 例如，abcdabcd，当计算以首位开始的子串最长长度为abcd的字符串，因为bcd是abcd的子串，则不需要重复计算

// 计算以b开头的子串时，直接判断上一次遍历j=4，即a是否属于bcd

j++;

}

if (j - i > maxLen)

{

maxLen = j - i;

}

i++;

}

return maxLen;

}

五、来源

<https://leetcode-cn.com/problems/longest-substring-without-repeating-characters/comments/>

一、原题目(345)

编写一个函数，以字符串作为输入，反转该字符串中的元音字母。

二、题目大意

将字符串反转，仅限于元音字母。

三、思路分析

本人的思路：

设置两个标签，分别从字符串前后开始移动，将此时两个元音字母进行交换。

四、具体代码

本人的代码：

bool isvowel(char ch)

{

if (

(ch == 'a') || (ch == 'o') || (ch == 'e') || (ch == 'i') || (ch == 'u') ||

(ch == 'A') || (ch == 'O') || (ch == 'E') || (ch == 'I') || (ch == 'U')

)

return true;

else

return false;

}

char \* reverseVowels(char \* S)

{

char \* save = S;

char \* start = S;

char \* end = S + strlen(S) - 1;

char ch;

while (start <= end)

{

if (isvowel(\*start) && isvowel(\*end)) // 如果都是字符，交换他们

{

ch = \*start;

\*start = \*end;

\*end = ch;

start++;

end--;

}

else

{

if (!isvowel(\*start))

start++;

if (!isvowel(\*end))

end--;

}

}

return save;

}

五、来源<https://leetcode-cn.com/problems/reverse-vowels-of-a-string/>

一、原题目(392)

给定字符串 s 和 t ，判断 s 是否为 t 的子序列。

你可以认为 s 和 t 中仅包含英文小写字母。字符串 t 可能会很长（长度 ~= 500,000），而 s 是个短字符串（长度 <=100）。

字符串的一个子序列是原始字符串删除一些（也可以不删除）字符而不改变剩余字符相对位置形成的新字符串。（例如，"ace"是"abcde"的一个子序列，而"aec"不是）。

示例 1:

s = "abc", t = "ahbgdc"

返回 true.

示例 2:

s = "axc", t = "ahbgdc"

返回 false.

二、题目大意

判断字符串s的字母中是否在字符串t中都存在。

三、思路分析

本人的思路：

将标记指向字符串s，然后在与字符串t比较时，适时的移动两个字符串，当移动到结束符时即为结束。

四、具体代码

本人的代码：

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

{

while (\*s && \*t)

{

if (\*s == \*t)

{

s++;

}

t++;

}

if (\*s == '\0')

{

return true;

}

else

{

return false;

}

}

五、来源<https://leetcode-cn.com/problems/is-subsequence/>

一、原题目()

二、题目大意

三、思路分析

本人的思路：

四、具体代码

本人的代码：

五、来源