13. 罗马数字转整数

已解答

简单 相关标签 相关企业 提示 Aあ

罗马数字包含以下七种字符: `I`， `V`， `X`， `L`，`C`，`D` 和 `M`。

字符	数值
I	1
V	5
X	10
L	50
C	100
D	500
M	1000

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。

```
int ans = 0;
int n = s.length();
for (int i = 0; i < n; ++i) {
    int value = symbolValues[s[i]];
    if (i < n - 1 && value < symbolValues[s[i + 1]])
    {
        ans -= value;
    }
    else
    {
        ans += value;
    }
}
return ans;
```

58. 最后一个单词的长度

已解答

简单 相关标签 相关企业

给你一个字符串 `s`，由若干单词组成，单词前后用一些空格字符隔开。返回字符串中 **最后一个** 单词的长度。

单词 是指仅由字母组成、不包含任何空格字符的最大子字符串。

```
int index = s.size() - 1;
    while (s[index] == ' ')
    {
        index--;
    }
    int wordLength = 0;
    while (index >= 0 && s[index] != ' ') {
        wordLength++;
        index--;
    }
    return wordLength;
```

66. 加一

已解答

简单 相关标签 相关企业

给定一个由 **整数** 组成的 **非空** 数组所表示的非负整数，在该数的基础上加一。

最高位数字存放在数组的首位， 数组中每个元素只存储**单个**数字。

你可以假设除了整数 0 之外，这个整数不会以零开头。

```
for(int i=digits.size()-1; i>=0; i--)
{
    digits[i]++;
    if(digits[i] == 10)  digits[i] = 0;
    else  return digits;
}
digits.insert(digits.begin(), 1);
return digits;
```

简单 相关标签 相关企业 提示 A文

假设你正在爬楼梯。需要 `n` 阶你才能到达楼顶。

每次你可以爬 `1` 或 `2` 个台阶。你有多少种不同的方法可以爬到楼顶呢?

以上是动态规划的转移方程，下面我们来讨论边界条件。我们是从第 0 级开始爬的，所以从第 0 级爬到第 0 级我们可以看作只有一种方案，即 $f(0)=1$；从第 0 级到第 1 级也只有一种方案，即爬一级，$f(1)=1$。这两个作为边界条件就可以继续向后推导出第 n 级的正确结果。我们不妨写几项来验证一下，根据转移方程得到 $f(2)=2$，$f(3)=3$，$f(4)=5$，……，我们把这些情况都枚举出来，发现计算的结果是正确的。

我们不难通过转移方程和边界条件给出一个时间复杂度和空间复杂度都是 $O(n)$ 的实现，但是由于这里的 $f(x)$ 只和 $f(x-1)$ 与 $f(x-2)$ 有关，所以我们可以用「滚动数组思想」把空间复杂度优化成 $O(1)$。**下面的代码中给出的就是这种实现。**

```
class Solution {
public:
    int climbStairs(int n) {
        int p = 0, q = 0, r = 1;
        for (int i = 1; i <= n; ++i) {
            p = q;
            q = r;
            r = p + q;
        }
        return r;
    }
};
```

69. x 的平方根

已解答

简单 相关标签 相关企业 提示

给你一个非负整数 `x` ，计算并返回 `x` 的 **算术平方根** 。

由于返回类型是整数，结果只保留 **整数部分** ，小数部分将被 **舍去** 。

注意： 不允许使用任何内置指数函数和算符，例如 `pow(x, 0.5)` 或者 `x ** 0.5` 。

```
    int i = 1;
    int j = x;
    int ans = 0;
    while (i <=j){
        int mid = i + (j-i)/2;
        if (mid <= x/mid)
        {
            i = mid +1;
            ans = mid;
        }
        else
            j = mid-1;
    }

    return ans;
}
```

67. 二进制求和

已解答

简单 相关标签 相关企业

给你两个二进制字符串 `a` 和 `b` ，以二进制字符串的形式返回它们的和。

```
int al = a.size();
        int bl = b.size();
        while(al < bl)
        {
            a = '0' + a;
            ++ al;
        }
        while(al > bl)
        {
            b = '0' + b;
            ++ bl;
        }
        for(int j = a.size() - 1; j > 0; -- j)
        {
            a[j] = a[j] - '0' + b[j];
            if(a[j] >=  '2')
            {
                a[j] = (a[j] - '0') % 2 + '0';
                a[j-1] = a[j-1] + 1;
            }
        }
        a[0] = a[0] - '0' + b[0];
        if(a[0] >= '2')
        {
            a[0] = (a[0] - '0') % 2 + '0';
            a = '1' + a;
        }
        return a;
```

14. 最长公共前缀

已解答

简单 相关标签 相关企业

编写一个函数来查找字符串数组中的最长公共前缀。

如果不存在公共前缀，返回空字符串 `""`。

```cpp
class Solution {
public:
    string longestCommonPrefix(vector<string>& strs)
    {
        string s;
        s = strs[0];
        for (int i = 1; i < strs.size(); ++i)
            for (int j = 0; j < s.size(); ++j)
                if (s[j] != strs[i][j])
                    s.erase(j);
        return s;
    }
};
```