07 整数转罗马数字&罗马数字转整数

更新时间：2019-08-13 09:55:16

> 耐心是一切聪明才智的基础。
>
> ——柏拉图

刷题内容

> 难度: **Easy**
>
> 原题链接：https://leetcode-cn.com/problems/integer-to-roman/。

内容描述

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；
给定一个整数，将其转为罗马数字。输入确保在 1 到 3999 的范围内。

示例 1:
输入: 3
输出: "III"

示例 2:
输入: 4
输出: "IV"

示例 3:
输入: 9
输出: "IX"

示例 4:
输入: 58
输出: "LVIII"
解释: L = 50, V = 5, III = 3.

示例 5:
输入: 1994
输出: "MCMXCIV"
解释: M = 1000, CM = 900, XC = 90, IV = 4.

解题方案

思路 1：时间复杂度: O(N) 空间复杂度: O(1)

首先，根据题意，我们可以知道罗马数字有如下符号和其对应的数值：

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

详细的罗马数字计数规则如下：

- 相同的数字连写，所表示的数等于这些数字相加得到的数，例如：III = 3；
- 小的数字在大的数字右边，所表示的数等于这些数字相加得到的数，例如：VIII = 8；
- 小的数字，限于（I、X 和 C）在大的数字左边，所表示的数等于大数减去小数所得的数，例如：IV = 4，这条规

则好目前与本题无关；

- 正常使用时，连续的数字重复不得超过三次；
- 在一个数的上面画横线，表示这个数扩大 1000 倍（本题只考虑 3999 以内的数，所以用不到这条规则）；
- 从前向后遍历罗马数字，如果某个数比前一个数小，则加上该数；反之，减去前一个数的两倍然后加上该数。

我们可以将所有罗马数字的不同符号及对应整数放在字典中。但由于题目限制，正常使用时连续的数字重复不能超过三次，所以对于 400、40、4 或者是 900、90、9 这种情况我们直接单独列出来。

> 注：之所以将 900、90、9 这三种情况列出是因为罗马数字中并没有单独表示 400 的字符存在，而且不允许DCD这样的情况出现。

将罗马数字和对应整数以及各种特殊情况整合到字典中后，首先将字典按照对应罗马数字的对应整数值进行排序并遍历，每一项中的键 `symbol` 对应罗马数字符号 ，值 `val` 对应整数。

如果 `num` 大于 `val` 则进入 `while` 循环，并将当前的罗马数字字符 `symbol` 拼接到最后结果 `roman` 中，然后 `num - val` ，每执行一次 while 循环重新判断 `num >= val` 。如果不满足则跳出 while 循环，执行下一次 for 循环。具体流程参考：

```
以整数 3568 为例，3568 的对应罗马数字为：MMMDLXVIII

第一次 for 循环 ————> symbol = 'M' val = 1000 , num >= val成立，进入 while 循环：
 第一次 while 循环：
  将 'M' 拼接到 roman 中， num 减去当前 val ，此时 roman 为 'M',num 为 2568
 第二次 while 循环：
  将 'M' 拼接到 roman 中，num 减去当前 val ，此时 roman 为 'MM',num 为 1568
 第三次 while 循环：
  将 'M' 拼接到 roman 中，num 减去当前 val ，此时 roman 为 'MMM',num 为 568     num >= val 不成立，退出 while 循环。
第二次 for 循环 ————> symbol = 'CM' val = 900 , num >= val不成立，不能进入 while 循环：
第三次 for 循环 ————> symbol = 'D' val = 500 , num >= val成立，进入 while 循环：
 第一次 while 循环：
    将 'D' 拼接到 roman 中， num 减去当前 val ，此时 roman 为 'MMMD',num 为 68
       num >= val 不成立，退出 while 循环。
第四次 for 循环 ————> symbol = 'CD' val = 400 , num >= val不成立，不能进入 while 循环：
.............
第七次 for 循环 ————> symbol = 'L' val = 50 , num >= val成立，进入 while 循环：
 第一次 while 循环：
    将 'L' 拼接到 roman 中， num 减去当前 val ，此时 roman 为 'MMMDL',num 为 18
       num >= val 不成立，退出 while 循环。
.............依次执行后得出结果为 MMMDLXVIII 。
```

> 代码讲解以 Python 为例，各语言之间解法会有差异，需同学们自行通读自己使用的语言代码，在这里就不一一详细解读了。

下面是具体代码：

Python beats 74.06%

```python
class Solution:
    def intToRoman(self, num: int) -> str:
        lookup = {
            'M': 1000,
            'CM': 900,
            'D': 500,
            'CD': 400,
            'C': 100,
            'XC': 90,
            'L': 50,
            'XL': 40,
            'X': 10,
            'IX': 9,
            'V': 5,
            'IV': 4,
            'I': 1
        }
        roman = ''
        # 因为dict本身是无序的，这里做了一个排序的操作，否则可能会出现IIII这种状况。
        for symbol, val in sorted(lookup.items(), key = lambda t: t[1])[::-1]:
            while num >= val:
                roman += symbol
                num -= val
        return roman
```

Java beats 47.36%

```java
class Solution {
    public String intToRoman(int num) {
        // 初始化了一个一一对应的map，方便后面取出符号。
        String[][] lookup = {
                {"", "I", "II", "III", "IV", "V", "VI", "VII", "VIII", "IX"},
                {"", "X", "XX", "XXX", "XL", "L", "LX", "LXX", "LXXX", "XC"},
                {"", "C", "CC", "CCC", "CD", "D", "DC", "DCC", "DCCC", "CM"},
                {"", "M", "MM", "MMM"}
        };
        String ret = "";
        for (int i = 0; i < 4; i++) {
            ret = lookup[i][num % 10] + ret;
            num /= 10;
        }
        return ret;
    }
}
```

Go beats 99.49%

```go
func intToRoman(num int) string {
    // 初始化了一个一一对应的map，方便后面取出符号。
    lookupSymbol := []string{"M", "CM", "D", "CD", "C", "XC", "L", "XL", "X", "IX", "V", "IV", "I"}
    lookupNum := []int{1000, 900, 500, 400, 100, 90, 50, 40, 10, 9, 5, 4, 1}
    roman := ""
    for i, symbol := range lookupSymbol {
        val := lookupNum[i]
        for num >= val {
            roman += symbol
            num -= val
        }
    }
    return roman
}
```

c++ beats 95.02%

```
class Solution {
public:
    // 初始化了一个一一对应的map，方便后面取出符号。
    string a[4][10] = {
        {"", "I", "II", "III", "IV", "V", "VI", "VII", "VIII", "IX"},
        {"", "X", "XX", "XXX", "XL", "L", "LX", "LXX", "LXXX", "XC"},
        {"", "C", "CC", "CCC", "CD", "D", "DC", "DCC", "DCCC", "CM"},
        {"", "M", "MM", "MMM"}
    };
    string intToRoman(int num) {
        string ret = "";
        for (int i = 0;i < 4;i++) {
            ret = a[i][num % 10] + ret;
            num /= 10;
        }
        return ret;
    }
};
```

罗马数字转整数

> 难度：**Easy**
>
> 原题链接：https://leetcode-cn.com/problems/roman-to-integer/。

刚才上面的题目要求我们将整数转为罗马数字，其实还有一道题和这道题很相似，那就是罗马数字转整数。这道题和刚才那道题差不多，所以在这里我就不放题目内容了，大家跟着我一起，来把这道题做一下就行。

思路 1：时间复杂度: O(N) 空间复杂度: O(1)

这道题同样有将整数限制在 `1 - 3999` 之间。我们从前往后扫描，用一个临时变量来分段记录数字。因为有这样一条规则：小的数字，限于（I、X 和 C）在大的数字左边，所表示的数等于大数减去小数所得的数，例如：`IV = 4`。所以如果当前罗马数字的值比前面一个大，说明这一段的值应当是减去上一个值。否则，应将当前值加入到最后结果中并开始下一次记录，例如:`VI = 5 + 1, II = 1 +1`。

下面来看具体代码：

Python beats 67.57%

```python
class Solution:
    def romanToInt(self, s: str) -> int:
        # 初始化了一个一一对应的map，方便后面取出符号。
        lookup = {
            'I': 1,
            'V': 5,
            'X': 10,
            'L': 50,
            'C': 100,
            'D': 500,
            'M': 1000
        }
        res = 0
        for i in range(len(s)):
            # 判断当前的值是否比前一个大
            if i > 0 and lookup[s[i]] > lookup[s[i-1]]:
                # 如果当前值比前一个大则减去
                res += lookup[s[i]] - 2 * lookup[s[i-1]]
            else:
                res += lookup[s[i]]
        return res
```

Java beats 81.66%

```java
class Solution {
    public int romanToInt(String s) {
        // 初始化了一个一一对应的map，方便后面取出符号。
        String[][] lookup = {
                {"", "I", "II", "III", "IV", "V", "VI", "VII", "VIII", "IX"},
                {"", "X", "XX", "XXX", "XL", "L", "LX", "LXX", "LXXX", "XC"},
                {"", "C", "CC", "CCC", "CD", "D", "DC", "DCC", "DCCC", "CM"},
                {"", "M", "MM", "MMM"}
        };
        int ret = 0;
        int base = 1000;
        int x = 3;
        int y = 3;
        int pos = 0;
        while (pos < s.length()){
            if (pos + lookup[x][y].length() <= s.length()) {
                boolean wrong = false;
                for (int i = 0; i < lookup[x][y].length(); i++) {
                    if (lookup[x][y].charAt(i) != s.charAt(pos + i)){
                        wrong = true;
                        break;
                    }
                }
                if (!wrong) {
                    pos += lookup[x][y].length();
                    ret += base * y;
                }
            }
            y--;
            if (y == 0) {
                base /= 10;
                x--;
                y = 9;
            }
        }
        return ret;
    }
}
```

Go beats 92.98%

```go
func romanToInt(s string) int {
    // 初始化了一个一一对应的map，方便后面取出符号。
    lookup := make(map[byte]int)
    lookup['I'] = 1
    lookup['V'] = 5
    lookup['X'] = 10
    lookup['L'] = 50
    lookup['C'] = 100
    lookup['D'] = 500
    lookup['M'] = 1000

    res := 0
    for i, _ := range s {
        if i > 0 && lookup[s[i]] > lookup[s[i-1]] {
            res += lookup[s[i]] - 2 * lookup[s[i-1]]
        } else {
            res += lookup[s[i]]
        }
    }
    return res
}
```

c++ beats 98.60%

```cpp
class Solution {
public:
    // 初始化了一个一一对应的map，方便后面取出符号。
    string a[4][10] = {
        {"", "I", "II", "III", "IV", "V", "VI", "VII", "VIII", "IX"},
        {"", "X", "XX", "XXX", "XL", "L", "LX", "LXX", "LXXX", "XC"},
        {"", "C", "CC", "CCC", "CD", "D", "DC", "DCC", "DCCC", "CM"},
        {"", "M", "MM", "MMM"}
    };
    int romanToInt(string s) {
        int ret = 0;
        int base = 1000;
        int x = 3, y = 3;
        int pos = 0;
        while (pos < s.size()) {
            if (pos + a[x][y].size() <= s.size()) {
                bool wrong = false;
                for (int i = 0;i < a[x][y].size();i++) {
                    if (a[x][y][i] != s[pos + i]) {
                        wrong = true;
                        break;
                    }
                }
                if (!wrong) {
                    pos += a[x][y].size();
                    ret += base * y;
                }
            }
            y--;
            if (y == 0) {
                base /= 10;
                x --;
                y = 9;
            }
        }
        return ret;
    }
};
```

小结

这个小节我们做了两道题。虽然这两道题差不多，但还是有差异的。整数转罗马数字明显更难一些，罗马数字转整数则相对比较简单。建议同学们自己多刷几遍，书读百遍，其义自见。

}

06 回文数

08 最长公共前缀