06 回文数

更新时间：2019-08-12 10:42:10

> 读书而不思考，等于吃饭而不消化。
>
> ——波尔克

更多一手资源请+V：Andyqclu
qq：3118617541

刷题内容

> **难度: Easy**
>
> 原题链接：https://leetcode-cn.com/problems/palindrome-number/。

内容描述

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。

示例 1:
输入: 121
输出: true

示例 2:
输入: -121
输出: false
解释: 从左向右读, 为 -121 。 从右向左读, 为 121- 。因此它不是一个回文数。

示例 3:
输入: 10
输出: false
解释: 从右向左读, 为 01 。因此它不是一个回文数。

进阶:你能不将整数转为字符串来解决这个问题吗？

题目详解

这道题的题目描述得很清楚：**判断是否是回文数**。如果是则返回 `true` ，如果不是则返回 `false`。要注意以下两点：

- 考虑下负数的情况，如果是负数直接返回 `false`，因为负数不可能是回文数；
- 考虑有没有 leading-zero 的情况（leading-zero指得是数字首位是 0 的情况，如0111、022），这种情况也直接返回 `false` 即可。

解题方案

思路 1：时间复杂度: O(N) 空间复杂度: O(N)

这道题目还是比较简单的，我们先来看下代码的执行流程：

- 判断 `x` 是否为负数，如果是负数直接返回；
- 反转 `x`，如果反转之后的值与原来的值不同直接返回 `false`；
- 如果不为负数，同时与反转后的值相等则返回 `true`。

下面来看下具体的代码：

Python beats 98.35%

```python
class Solution:
    def isPalindrome(self, x: int) -> bool:
        if x < 0: # 排除小于0的数
            return False
        elif x != int(str(x)[::-1]): # 通过字符串进行反转，对比数字是否相等就行
            return False
        else:
            return True
```

更多一手资源请+V：Andyqclu
qq：3118617541

Python 中 `int(str(x)[::-1])` 会直接将字符串 `x` 反转，最后判断是否与 `x` 相等。相等则返回 `true`，不相等则返回 `false`。

Java beats 74.06%

```java
class Solution {
    public boolean isPalindrome(int x) {
        if (x < 0) { // 排除小于0的数
            return false;
        }
        String str = String.valueOf(x);
        int n = str.length();
        for (int i = 0; i < n; i++) {
            if (str.charAt(i) != str.charAt(n - 1 - i)) { // 通过字符串前后对应字符比较，对比数字是否相等就行
                return false;
            }
        }
        return true;
    }
}
```

因为 Java 和 Python 语言上的差异，所以在 Java 版本的代码中我并没有将字符串反转。在 Java 中通过对比字符串前后对应字符来比较结果，判断是否是回文数。

例如：12321
首先将 12321 转为字符串。循环取出首尾对应字符。
1 2 3 2 1
第一次循环 str.charAt(i) 取出第一位上的 1，str.charAt(n - 1 - i) 取出最后一位的 1（n为字符串长度）；
第二次循环取出第二位的 2 和倒数第二位上的 2；
依次循环，如果每次取出的字符相等则为回文数，不相等则不是回文数。

Go beats 99.67%

```
func isPalindrome(x int) bool {
 if x < 0 { // 排除小于0的数
  return false
 }
 xStr := strconv.Itoa(x)
 xStrReverse := make([]rune, 0)
 for i, _ := range xStr {
  xStrReverse = append(xStrReverse, rune(xStr[len(xStr)-1-i]))
 }
 for i := 0; i < len(xStr); i += 1 { // 通过字符串进行反转，对比数字是否相等就行
  if rune(xStr[i]) != xStrReverse[i] {
   return false
  }
 }
 return true
}
```

C++ beats: 94.73%

```
class Solution {
public:
    bool isPalindrome(int x) {
        if (x < 0) { // 排除小于0的数
            return false;
        }
        char c[20];
        sprintf(c, "%d", x);
        int n = strlen(c);
        for (int i = 0;i < n;i++) {
            if (c[i] != c[n - 1 - i]) { // 通过字符串首尾比较，对比数字是否相等就行
                return false;
            }
        }
        return true;
    }
};
```

更多一手资源请+V：Andyqclu
qq：3118617541

思路 1 的时间和空间复杂度都是 O(N)，那么有没有办法来优化一下呢？

答案是：有。还记得内容描述中的最后一句话吗？

你能不将整数转为字符串来解决这个问题吗？下面我们不把整数转为字符串来解决一下这个问题。

思路 2 (满足Follow up)：时间复杂度: O(1) 空间复杂度: O(1)

刚才我们在思路 1 中通过将整数转为字符串处理，现在我们不转了，直接用整数类型来判断是否是回文数。首先负数肯定不是回文数，这个不用多说。其次就是如果一个数字为正整数，而且能够被 10 整除，那么这个数字也不是回文数，因为回文数的首位肯定不是 0 。

那么不把整数转为字符串该怎么做呢？还记得在 2.2 小节整数反转么？

没错，我们直接把整数反转过来，与原来的值比较即可。如果印象不深刻的同学可以去重新看下2.2小节。

这样做大大降低了思路 1 的时间和空间复杂度。下面我们来看下具体代码：

Python beats 84.41%

```python
class Solution:
    def isPalindrome(self, x: int) -> bool:
        # 负数肯定不是palindrome
# 如果一个数字是一个正数，并且能被10整除，那它肯定也不是palindrome，因为首位肯定不是0
        if x < 0 or (x != 0 and x % 10 == 0):
            return False
        rev, y = 0, x
        while x > 0:
            rev = rev * 10 + x % 10
            x //= 10
        return y == rev
```

Java beats 89.85%

```java
class Solution {
    public boolean isPalindrome(int x) {
        if (x < 0) { // 负数肯定不是palindrome
            return false;
        }
        int temp = x;
        // 翻转之后的数字可能超过整型的范围
        long y = 0;
        while ( x != 0) {
            y = y * 10 + x % 10;
            x /= 10;
        }
        return  temp == y;
    }
}
```

更多一手资源请+V：Andyqclu
qq：3118617541

Go beats 97.72%

```go
func isPalindrome(x int) bool {
    // 负数肯定不是palindrome
    // 如果一个数字是一个正数，并且能被10整除，那它肯定也不是palindrome，因为首位肯定不是 0
    if x < 0 || (x != 0 && x % 10 == 0) {
        return false
    }
    rev, y := 0, x
    for x > 0 {
        rev = rev * 10 + x % 10
        x /= 10
    }
    return y == rev
}
```

C++ beats 99.69%

```
class Solution {
public:
    bool isPalindrome(int x) {
        if (x < 0) { // 负数肯定不是palindrome
            return false;
        }
        int temp = x;
        //这里会溢出
        long long y = 0;
        while (x != 0) {
            y = y * 10 + x % 10;
            x /= 10;
        }
        return temp == y;
    }
};
```

小结

做题是首先要考虑到各种特殊情况，但是特殊情况也不一定要特殊处理，有时候只是递归调用一下原函数即可。

同时，在用思路 2 解题的时候，要随时考虑**溢出**的情况！！！

}

05 整数反转

07 整数转罗马数字&罗马数字转整数

更多一手资源请+V：Andyqclu
qq：3118617541