「小算法」回文数与数值合法性检验

原创 夕小瑶 夕小瑶的卖萌屋 2018-06-12

喵喵喵，小夕最近准备复习一下数学和基础算法，尽量每篇推送下面会附带点数学和基础算法的小文章。说不定哪天就用（考）到了呢(￣▽￣)注意哦，与头条位的文章推送不同，「小公式」和「小算法」里的小标题之间可能并无逻辑关联。

回文数

链接： https://leetcode.com/problems/palindrome-number/description/

判断一个整数是否是回文数是leetcode上的一个简单算法题。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。

例1:

```
输入：121
输出：true
```

例2:

```
输入：-121
输出：false
解释：从右往左读为121-。
```

例3:

```
输入：10
输出：false
```

解这个题的话，一个很自然而简单的想法就是将整数转换为字符串，并检查字符串是否为回文。但是，这需要额外的非常量空间来创建问题描述中所不允许的字符串。

第二个想法是将数字本身反转，然后将反转后的数字与原始数字进行比较，如果它们是相同的，那么这个数字就是回文。但是，如果反转后的数字大于 int.MAX，会发生数值溢出啦！

不过，按照第二个想法，为了避免数字反转可能导致的溢出问题，为什么不考虑只反转 int 数字的一半？毕竟如果该数字是回文，其后半部分反转后肯定与原始数字的前半部分相同的呀。

所以直接上代码（原谅我用python写\(//▽//)\）

```python
class Solution(object):
    def isPalindrome(self, x):
        """
        :type x: int
        :rtype: bool
        """
        if x < 0 or (x != 0 and x % 10 == 0):
            return False
        invx = 0
        while invx < x:
            invx = invx * 10 + x % 10
            x //= 10
        return invx == x or invx // 10 == x
```

当然，得益于python自带大数运算的特性（即不存在撞枪int.MAX的情况），在python里直接用第二种方法解也是可以acc的。

```
class Solution(object):
    def isPalindrome(self, x):
        """
        :type x: int
        :rtype: bool
        """
        if x < 0:
            return False
        rawx = x
        invx = 0
        while x > 0:
            invx = invx * 10 + x % 10
            x //= 10
        return invx == rawx
```

数值合法性检验

链接： https://leetcode.com/problems/valid-number/description/

这个算法题是一个对初学者写到吐，对有基础的人写着玩儿的一道题。题目很简单，就是判断用户的输入（字符串形式）是不是一个合法的数值。这里合法的数值其实就是我们平常所说的数字啦，比如 `123` , `-1.0` , `+423` , `.4234` , `2.345e21` 都是合法数值。当然这里还允许用户在合法数值的前后插入若干空格。

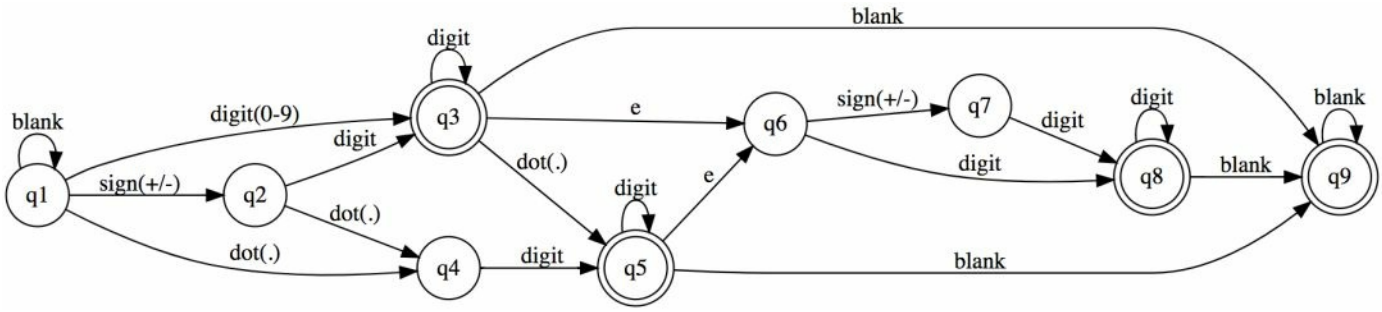
这个问题如果直接用 `if else while` 生写的话会写到吐，而且极难debug。但是这个问题一个机智的做法就是直接上**确定有限状态自动机（deterministic finite automation, DFA）**。

DFA是由

1. 一个非空有限的状态集合Q
2. 一个输入字母表$\sum$（非空有限的字符集合）
3. 一组转移函数Δ（如$\delta(q,\sigma)=p,(p,q\in Q,\sigma\in\Sigma)$）
4. 一个开始状态$s\in Q$
5. 一个接受（终止）状态的集合$F\subsetneq Q$

所组成的5元组。这个在编译原理、NLP基础里都有讲，忘了的同学自行补上哦。

所以很自然的这个题目画出的状态机如下图



有了状态机，代码就变得简洁多了。这里贴出python实现（来自leetcode讨论区，小夕在此题悲剧）

```python
class Solution(object):
  def isNumber(self, s):
    """
    :type s: str
    :rtype: bool
    """
    #define a DFA
    state = [{},
          {'blank': 1, 'sign': 2, 'digit':3, '.':4},
          {'digit':3, '.':4},
          {'digit':3, '.':5, 'e':6, 'blank':9},
          {'digit':5},
          {'digit':5, 'e':6, 'blank':9},
          {'sign':7, 'digit':8},
          {'digit':8},
          {'digit':8, 'blank':9},
          {'blank':9}]
    currentState = 1
    for c in s:
      if c >= '0' and c <= '9':
        c = 'digit'
      if c == ' ':
        c = 'blank'
      if c in ['+', '-']:
        c = 'sign'
      if c not in state[currentState].keys():
        return False
      currentState = state[currentState][c]
    if currentState not in [3,5,8,9]:
      return False
    return True
```

嗯，就酱┐(￣▽￣)┌