剑指 Offer 46. 把数字翻译成字符串

Monday, June 6, 2022 10:59 AM

https://leetcode.cn/problems/ba-shu-zi-fan-yi-cheng-zi-fu-chuan-lcof/
中等

给定一个数字，我们按照如下规则把它翻译为字符串：

0 翻译成 “a” ，
1 翻译成 “b”，……，
11 翻译成 “l”，……，
25 翻译成 “z”。
一个数字可能有多个翻译。请编程实现一个函数，用来计算一个数字有多少种不同的翻译方法。

示例 1:
输入: 12258
输出: 5
解释: 12258有5种不同的翻译，分别是
"bccfi", "bwfi", "bczi", "mcfi"和"mzi"
0 <= num < 2^{31}

我的思路（错误）：动规

1. 将数字转成字符串
2. 动规：
   dp[i]:
      截止到下标i的翻译种类数。
      dp[0]初始化为1
   递推公式：
      //i=1开始循环，截取当前数字与前一数字，小于26则截止当前种类数为前者+1；
      dp[i]=dp[i-1]+1;
      //否则 与未增加时类数一致。
      dp[i]=dp[i-1];

code：

```
public int translateNum(int num) {
    if(num<10) return 1;
    String s=String.valueOf(num);
    int[] dp=new int[s.length()];
    dp[0]=1;
    for(int i=1;i<s.length();i++){
       if(Integer.parseInt(s.substring(i-1,i+1))<=25){
          dp[i]=dp[i-1]+1;
       }else{
          dp[i]=dp[i-1];
       }
    }
```

```
        return dp[s.length()-1];
    }
```

解法错误!

1212时，1 12 21 12 为4、实则有5种 1 2 1 2 12 1 2、 12 12、 1 21 2、1 2 12

数字中有 0 未考虑，506： 5 0 6、5 06为2，实则只有5 0 6一种。

正确思路：

1. 只有2位数时，若小于26则 为 2种
2. 三位数及以上时，
   - 前一位为1或前一位为2且当前位小于等于5，此时：dp[i]=dp[i-1]+dp[i-2];
     //含义：截止当前，种数为 当前两位数分开时:dp[i-1] 加上 当前两数不分开时:dp[i-2] 的类数之和。

（与跳阶梯类似，每次能跳两步或一步，第三层时：dp[2]=dp[0]+dp[1] 即：到达第三层的方法等于到达第一层的方法+到达第二层的方法）

   - 否则：dp[i]=dp[i-1];
     //含义：此时只能分开，故当前新增一位数与没新增时种类一致，故为dp[i-1]。

code：

```
public int translateNum(int num) {
    // 将 int 变量 num 转换成字符串
    String s = String.valueOf(num);
    // dp[i] 表示前 i 位可以解码的总数
    int[] dp = new int[s.length()];
    // 只有一位数 只存在1种分法
    dp[0] = 1;
    // 通过 for 循环填充 dp 数组
    for (int i = 1; i < s.length(); i++) {
        //只在前一位数为1 或 前一位数为2且当前数小于等于5时，考虑两种情况
        if (s.charAt(i - 1) == '1' || (s.charAt(i - 1) == '2' && (s.charAt(i) <= '5'))) {
            //只有两位数时例外，不用考虑dp[i-2]
            if (i == 1) {
                dp[i] = 2;
            } else {
            //大于两位数时： dp[i] = dp[i - 1] + dp[i - 2];
                dp[i] = dp[i - 1] + dp[i - 2];
            }
        } else {
        // 前一位数不为 1 时，或 前一位数为 2 且当前数大于5时，增加当前数无异于不增加
        dp[i] = dp[i - 1];
      }
    }
    return dp[s.length() - 1];
```

}