

剑指46 把数字翻译成字符串

本质还是二维动态规划，依然是使用2个变量交替前进优化空间使用。

但是状态转移方程有不同情况，正向的话思考有点纠结，第一次做耗时很多。

$$\text{num} = x_1x_2 \dots x_{i-2}x_{i-1}x_i \dots x_{n-1}x_n$$

(例如: $12258 = x_1x_2x_3x_4x_5$)



- 设 $x_1x_2 \dots x_{i-2}$ 的翻译方案数量为 $f(i-2)$
- 设 $x_1x_2 \dots x_{i-2}x_{i-1}$ 的翻译方案数量为 $f(i-1)$



- 当整体翻译 $x_{i-1}x_i$ 时, $x_1x_2 \dots x_{i-2}x_{i-1}x_i$ 的方案数为 $f(i-2)$
- 当单独翻译 x_i 时, $x_1x_2 \dots x_{i-2}x_{i-1}x_i$ 的方案数为 $f(i-1)$



方案数的递推关系:

$$f(i) = \begin{cases} f(i-2) + f(i-1) & \text{, 若数字 } x_{i-1}x_i \text{ 可被翻译} \\ f(i-1) & \text{, 若数字 } x_{i-1}x_i \text{ 不可被翻译} \end{cases}$$

```
1 // my answer
2 class Solution {
3 public:
4     int translateNum(int num) {
5         int dpshort = 1, dpmid = 1, dplong = 1;
6         while (num/10 != 0) {
7             if (num % 100 <= 25 && num % 100 >= 10) {
8                 dplong = dpmid + dpshort;
9             }
10            else {
11                dplong = dpmid;
12            }
13            dpshort = dpmid;
14            dpmid = dplong;
15            num /= 10;
16        }
17        return dplong;
18        //从最后1个数字到倒数第n个数字。
```

```
19 // dp[n] = dp[n-1] + dp[n-2] （从包含倒数第n个数字向右的二位数 <=25 且 >=10 ）
20 // dp[n] = dp[n-1] （二位数 > 25）
21 }
22 };
```