LeetCode P790 解题总结（DP）

题目

有两种形状的瓷砖：一种是 2x1 的多米诺形，另一种是形如 "L" 的托米诺形。两种形状都可以旋转。

XX <- 多米诺

XX <- "L" 托米诺
X
给定 N 的值，有多少种方法可以平铺 2 x N 的面板？返回值 mod 10^9 + 7。

（平铺指的是每个正方形都必须有瓷砖覆盖。两个平铺不同，当且仅当面板上有四个方向上的相邻单元中的两个，使得恰好有一个平铺有一个瓷砖占据两个正方形。）

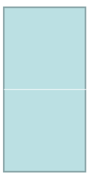
示例:
输入: 3
输出: 5
解释:
下面列出了五种不同的方法，不同字母代表不同瓷砖：
XYZ XXZ XYY XXY XYY
XYZ YYZ XZZ XYY XXY
提示：

N 的范围是 [1, 1000]

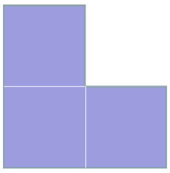
解题过程

题目分析

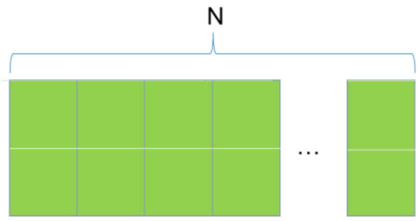
对题目进行解读，有 2 种瓷砖，其中一种规格是 1x2 的长方形：



另一种规格是 L 形的 3 块瓷砖形状：

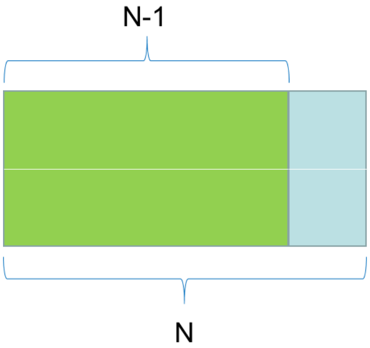


问题是对于 2xN 长度的区域，用以上的两种瓷砖，有多少种铺法？

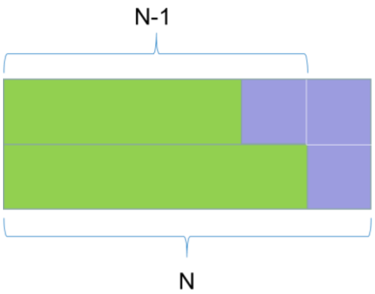


假设长度为 N 的区域铺满的情况，可能有以下几种情况：

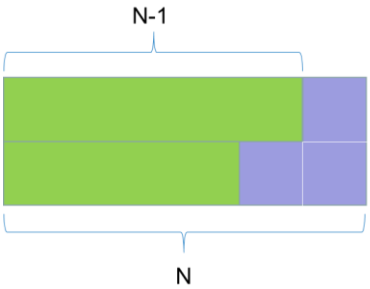
a) 前面的长度为 N-1 的区域完全铺满，再加上 1x2 类型的瓷砖竖着铺即铺满长度为 N 的区域



b) 前面的长度为 N-1 的区域没有完全铺满，遗留向上的缺口，此时加上 L 形的瓷砖铺上即可铺满长度为 N 的区域：

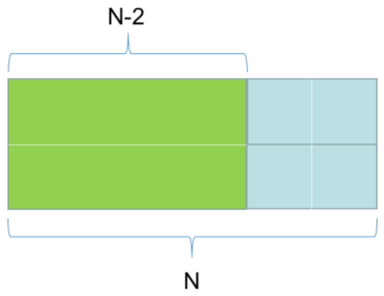


c) 前面的长度为 N-1 的区域没有完全铺满，遗留向下的缺口，此时加上 L 形的瓷砖铺上即可铺满长度为 N 的区域



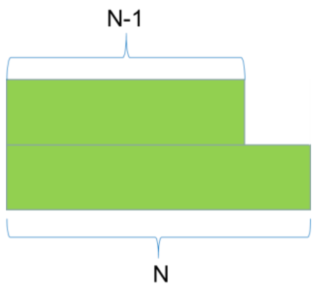
d) 前面长度为 N-2 的区域完全铺满，此时加上两块 1x2 的区域横着铺即可铺满（如果竖着

铺的话与 a 中描述的铺法重复)

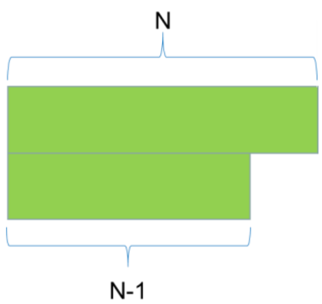


递推公式

通过以上分析后，定义状态转移方程，设：
$f(n,0)$表示长度为 n 的区域完全铺满的方法数
$f(n,1)$表示长度为 n 的区域没有完全铺满，遗留向上的缺口的方法数，即：



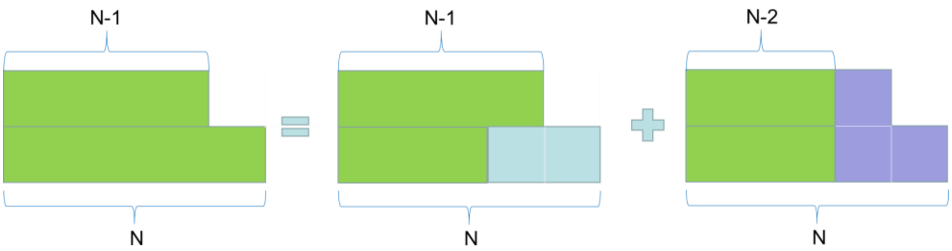
的铺法。
$f(n,2)$表示长度为 n 的区域没有完全铺满，遗留向下的缺口的方法数，即：



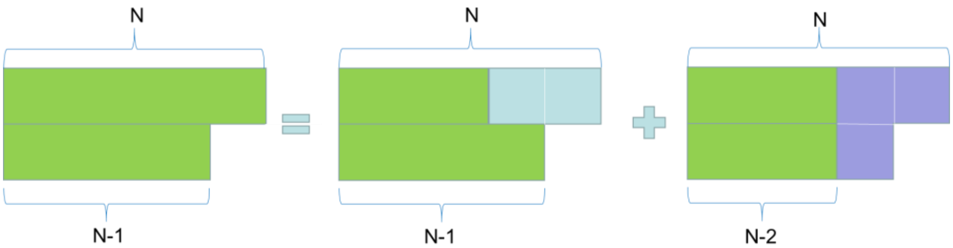
的铺法。那么需要求的值即为$f(n,0)$，可得如下公式：

$$f(n,0)=f(n-1,0)+f(n-1,2)+f(n-1,1)+f(n-2,0)$$

同理：
$f(n,1)=f(n-1,2)+f(n-2,0)$，即：

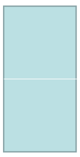


$f(n,2) = f(n-1,1) + f(n-2,0)$，即



初始化条件:

$f(1,0) = 1$,长度为 1 的完整铺法，只有一种:

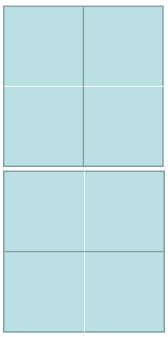


$f(1,1) = 0$

$f(1,2) = 0$

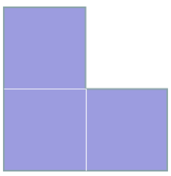
长度为 1，且留缺口的铺法，无论向上留，还是向下留都没有。

$f(2,0) = 2$，长度为 2 的完整铺法，有两种，并排竖着铺和并排横着铺:



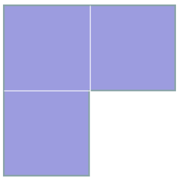
$f(2,1) = 1$

长度为 2，缺口向上的铺法，只有一种:



$f(2,2) = 1$

长度为 2，缺口向下的铺法，只有一种:



由底向上递归即可求得最终值。

代码

```
public int numTilings(int N) {
    long[][] tilings = new long[3][3];
    final int MOD = 1000000007;

    tilings[1][0] = 1;
    tilings[1][1] = 0;
    tilings[1][2] = 0;
    tilings[2][0] = 2;
    tilings[2][1] = 1;
    tilings[2][2] = 1;

    for (int i = 3; i <= N; i++) {
        int cur = i%3;
        int last = (i-1)%3;
        int lastlast = (i-2)%3;
        tilings[cur][0] = (tilings[last][0] + tilings[last][2] + tilings[last][1] + tilings[lastlast][0])%MOD;
        tilings[cur][1] = (tilings[last][2] + tilings[lastlast][0])%MOD;
        tilings[cur][2] = (tilings[last][1] + tilings[lastlast][0])%MOD;
    }
    return (int) tilings[N%3][0];
}
```