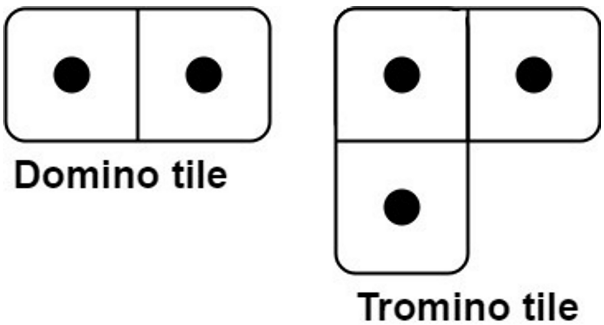
Friday, December 10, 2021 1:12 PM

790. Domino and Tromino Tiling

Medium

942424Add to ListShare

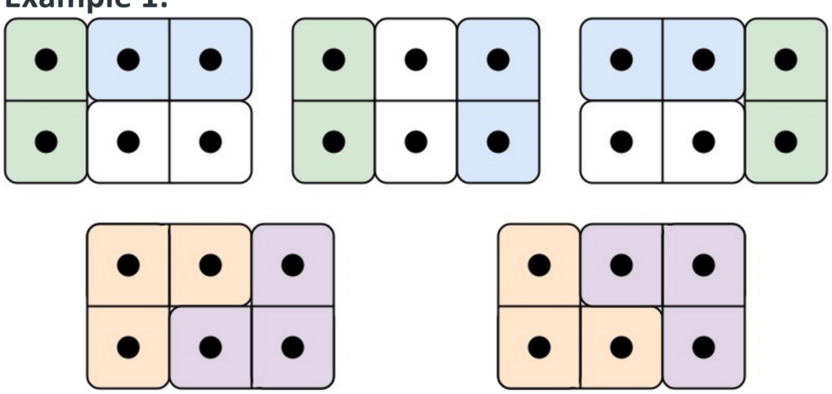
You have two types of tiles: a 2 x 1 domino shape and a tromino shape. You may rotate these shapes.



Given an integer n, return *the number of ways to tile an* 2 x n *board*. Since the answer may be very large, return it **modulo** 109 + 7.

In a tiling, every square must be covered by a tile. Two tilings are different if and only if there are two 4-directionally adjacent cells on the board such that exactly one of the tilings has both squares occupied by a tile.

Example 1:



Input: n = 3

Output: 5

Explanation: The five different ways are show above.

Example 2:

Input: n = 1

Output: 1

Constraints:

- 1 <= n <= 1000

Accepted
29,096
Submissions
66,491
Seen this question in a real interview before?

From <https://leetcode.com/problems/domino-and-tromino-tiling/>

Lets try to construct our dp

dp[0] = 1 (since, in question range is from 1, so we can assume dp[0] to be 1, to construct our dp easily)

dp[1] = 1 , { 1 }

dp[2] = 2 , { 11, = }

dp[3] = **5** , { dp[2] + {1} ⇒ {111, =1}, dp[1] + {=} ⇒ {1=},
dp[0] + {└┐, ┌┘} = {└┐, ┌┘} }

dp[4] = 11, { dp[3] + {1}, dp[2] + {=}, dp[1] + {┌┘, ┌┘},
dp[0] + {└═┘, ┌═┐} }

So, from the above trend we can see that,

dp[n-1] & dp[n-2] are contributing one time in dp[n]
but from dp[n-3] ... dp[0], are contributing two times in dp[n]

So, we can now deduce a formula from it,

$$dp[n] = dp[n-1] + dp[n-2] + 2(dp[n-3] + \dots dp[0])$$

Also, $dp[n-1] = dp[n-2] + dp[n-3] + 2(dp[n-4] + \dots dp[0])$

$\therefore \; dp[n] - dp[n-1] = dp[n-1] + dp[n-3]$

$$\boxed{dp[n] = 2 * dp[n-1] + dp[n-3]}$$

Time Complexity : O(n)
Space Complexity : O(n)

```java
class Solution {
    public int numTilings(int n) {
        int dp[] = new int[n+4];

        int mod = (int)(Math.pow(10,9)+7);
        dp[1] = 1;
        dp[2] = 2;
        dp[3] = 5;

        for(int i=4;i<=n;i++){
            dp[i] = ((2*dp[i-1])%mod)+dp[i-3];
            dp[i] %= mod;
        }

        return dp[n];
    }
}
```