Domino and Tromino Tiling

We have two types of tiles: a 2x1 domino shape, and an "L" tromino shape. These shapes may be rotated.

```
XX  <- domino

XX  <- "L" tromino
X
```

Given N, how many ways are there to tile a 2 x N board?**Return your answer modulo 10^9 + 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:
Input: 3
Output: 5
Explanation:
The five different ways are listed below, different letters indicates different tiles:
XYZ XXZ XYY XXY XYY
XYZ YYZ XZZ XYY XXY

Note:

- N will be in range `[1, 1000]`.

Solution 1

The answer will be a recursive sequence as follow: `1, 1, 2, 5, 11, 24, 53, 117, 258, 569, 1255`
It grows at a speed about 2 times bigger each time.
If you write down this recursive sequence and do some calculations, you may find that:

`5 = 2 * 2 + 1`
`11 = 5 * 2 + 1`
`24 = 11 * 2 + 2`
`53 = 24 * 2 + 5`
`117 = 57 * 2 + 11`
`A[N] = A[N-1] * 2 + A[N-3]`

Once you notice it, the rest work will be easy, even it may be hard to prove this regular.

C++:

```
    int numTilings(int N) {
        int a = 0, b = 1, c = 1, c2, mod = 1e9 + 7;
        while (--N) {
            c2 = (c * 2 % mod + a) % mod;
            a = b;
            b = c;
            c = c2;
        }
        return c;
    }
```

Java:

```
    public int numTilings(int N) {
        int a = 0, b = 1, c = 1, c2, mod = 1000000007;
        while (--N > 0) {
            c2 = (c * 2 % mod + a) % mod;
            a = b;
            b = c;
            c = c2;
        }
        return c;
    }
```

Python:

```
    def numTilings(self, N):
        a, b, c = 0, 1, 1
        for i in range(N - 1): a, b, c = b, c, (c + c + a) % int(1e9 + 7)
        return c
```

written by lee215 original link here

Solution 2

Calculate the number of tilings with both cells in the final row (even) and with only a single cell in the final row (odd).
We can make a new even row from the previous even row and a horizontal domino, or the even row before that and 2 vertical dominos, or the previous odd row and a tromino.
We can make a new odd row from the previous odd row and a vertical domino, or the i - 2 even row and a tromino in 2 possible orientations.

Space complexity can be reduced by not storing all the even and odd lists.

```
class Solution(object):
    def numTilings(self, N):
        MOD = (10 ** 9) + 7

        even = [0, 1, 2, 5]
        odd = [0, 0, 2, 4]

        for i in range(4, N + 1):

            new_even = (even[i - 1] + even[i - 2] + odd[i - 1]) % MOD
            even.append(new_even)

            new_odd = (odd[i - 1] + 2 * even[i - 2]) % MOD
            odd.append(new_odd)

        return clean[N] % MOD
```

written by yorkshire original link here

Solution 3

```
class Solution {
    static final int mod = 1000000007;
    public int numTilings(int N) {
        int dp[] = new int[N + 10];
        dp[0] = 1;
        dp[1] = 1;
        dp[2] = 2;
        dp[3] = 5;
        for(int i = 4 ; i <= N; i++){
            dp[i] = dp[i - 1] * 2 % mod + dp[i - 3] % mod;
            dp[i] %= mod;
        }
        return dp[N];
    }
}
```

written by ngoc_lam original link here