Tiling $2 \times n$ Grid with L-Trimino and Domino

Jubair Ahammad Akter

Recurrence Definition

Let $f(n)$ be the number of ways to tile a $2 \times n$ grid using L-triminoes and dominoes. Let $g(n)$ be the number of ways to tile a $2 \times n$ grid with one missing square on the top-right corner.

$$f(n) = f(n-1) + f(n-2) + g(n-2)$$
$$g(n) = 2f(n-1) + g(n-1) \quad \text{(i)}$$

Relation Derivation

From (i), we can derive:

$$g(n-2) = f(n) - f(n-1) - f(n-2)$$

$$g(n-1) = f(n+1) - f(n) - f(n-1) \quad \text{(ii)}$$
$$g(n) = f(n+2) - f(n+1) - f(n) \quad \text{(iii)}$$

Substituting (ii) and (iii) into (i):

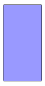
$$f(n+2) - f(n+1) - f(n) = 2f(n-1) + f(n+1) - f(n) - f(n-1)$$

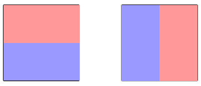
$$f(n+2) = 2f(n+1) + f(n-1)$$
$$\boxed{f(n) = 2f(n-1) + f(n-3)}$$

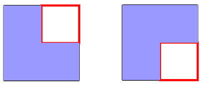
Base Cases

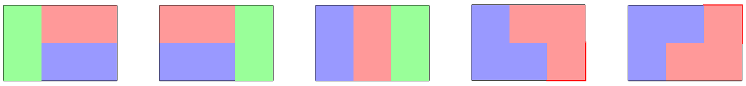
$$f(1) = 1, \quad f(2) = 2, \quad f(3) = 5$$
$$g(1) = 2, \quad g(2) = 4, \quad g(3) = 8$$

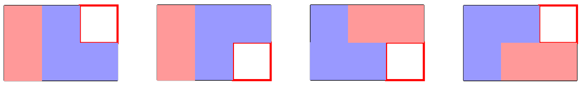
Visual Representations

1. **For** $f(1) = 1$ 

2. **For** $f(2) = 2$ 

3. **For** $g(1) = 2$ (Missing top left and right) 

4. **For** $f(3) = 5$ 

5. **For** $g(2) = 4$ (Missing top left and right) 

C++ Implementation

```
#include<bits/stdc++.h>
#define ll long long
ll mod = 1e9+7;
using namespace std;

void the_solver(int n){
    vector<ll> dp(n+1,0);
    dp[0]=1;
    dp[1]=1; dp[2]=2; dp[3]=5;
    if(n<=3){
        cout<<dp[n]<<endl;
        return;
    }
    for(int i=4;i<=n;i++){
        dp[i]=2*dp[i-1]+dp[i-3];
        dp[i]%=mod;
    }
    cout<<dp[n]<<endl;
}

int main(){
    int n=3;
    the_solver(n);
    return 0;
}
```

Explanation

The recurrence is based on the following placements:

- Placing a vertical domino at the right end contributes $f(n-1)$,
- Two horizontal dominoes stacked contribute another $f(n-1)$,
- An L-trimino covers three columns, contributing $f(n-3)$.

Hence, the total number of ways:

$$\boxed{f(n) = 2f(n-1) + f(n-3)}$$

n	$f(n)$	$g(n)$
1	1	2
2	2	4
3	5	8
4	11	18
5	24	40
6	53	88
7	117	194

— THE END —