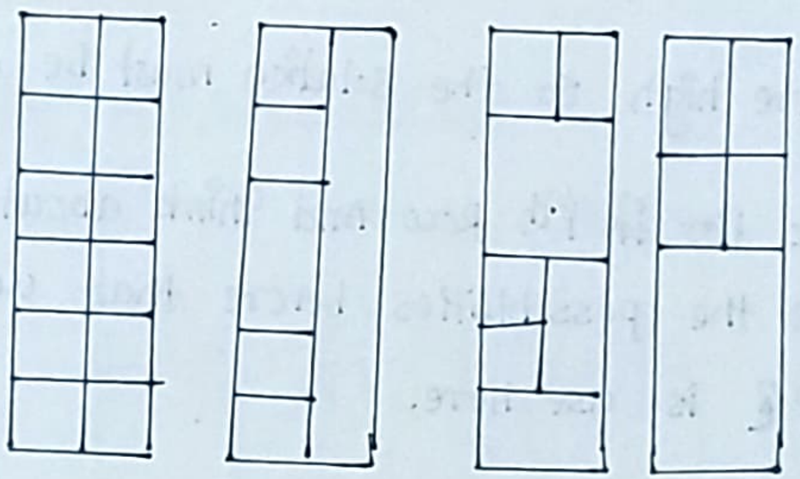
CSES 9 : Counting Towers. Day 11

problm : Your task is to build a tower whose width is 2 and height is n. You have an unlimited supply of blocks whose width and height are integers.

For Example : here are some possible slouti0n's for n=6.

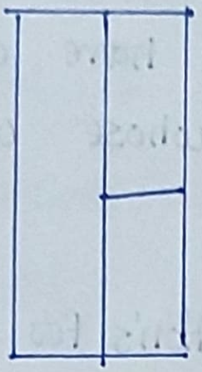


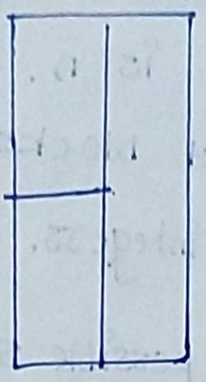
Given n, how many different towers can you build? Mirrored and rotated ~~to~~ towers are counted seperately if they look different.

Constraints :

- $1 \leq t \leq 100$.
- $1 \leq n \leq 10^6$

let's look what the meaning of mirrored and rotated travel.



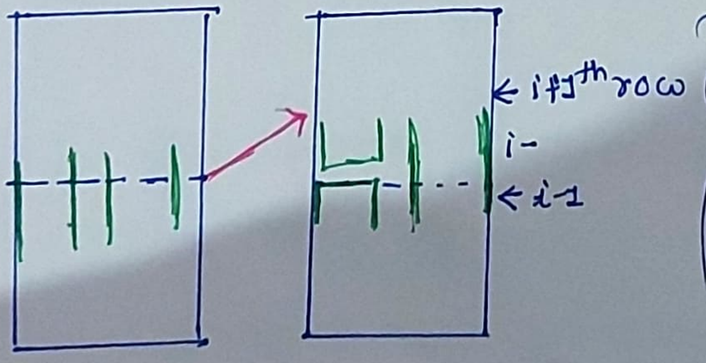


constrains are high so the solution must be optimised

→ We look for ~~#~~ ith Row And think about what are the possibilities before that we look why DP is use here.

→ let's try to understand what are the different cases are here.

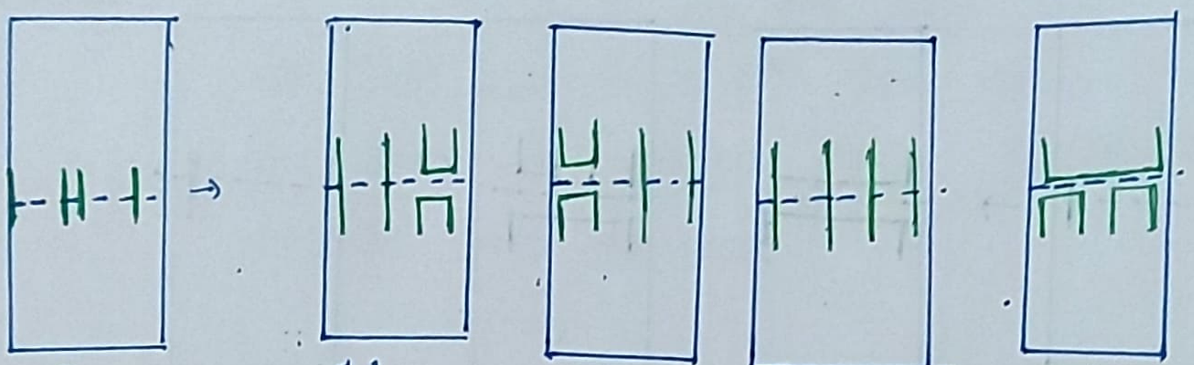
① two cells vertical which are trying to extend



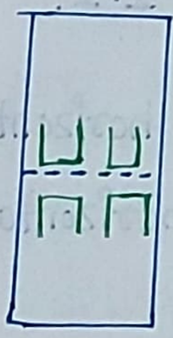
what we are doing there is two possibilities either we make portion for left block & extend right ~~or~~ and vice versa observation is if we portion left block so we have to create new block again ~~at~~ & at the i+1 th row the new block & previous block are extending → And we saw we are solving the Same problem for i-1 th row., so repitation is Occured.

Overall possibalities.

① two vertical row are extending



① 1.1. 1.2 1.3 1.4



1.5

this are the different possibilities when i-1 th row Are extending vertically.

in first on i th row we close right side of block and increase/extend the left block.

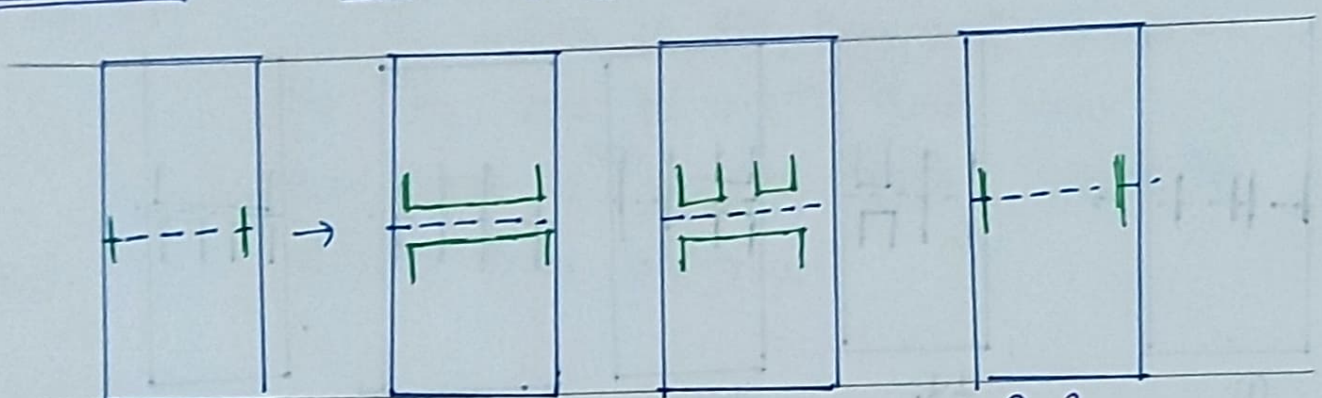
for. 1.2 & We close the left block and extend the right.

for 1.3. We extends both block

for 1.4. we close both of them and start horizontal

for. 1.5. We close both of them and start Again two vertical block.

② Horizontal cell trying to extends..



② 2.1 2.2. 2.3

For 1st you close the horizontal cell and start another horizontal cell.

For 2nd. Close the horizontal cell and start to vertical cell

For 3rd extend the horizontal cell.

If we look this entire things we can make a certain dp state.

State

dp[i][0] = no. of ways to fill the grid from the ith row to n-1 th row such that there is horizontal block trying to extends from ith row.

dp[i][1] = no. of ways to fill the grid from the ith row to n-1 th row such that there is two vertical block trying to extends from ith row

Transition

When we look figure. what are possibilities

dp[i][0] → have three. case.

dp[i+1][0], dp[i+1][1], dp[i+1][0]

-6

dp[i][1] → have 5 case.5

→ 4 of them are ~~dp[i]~~ dp[i+1][1] and last is dp[i+1][0].

So - transition are look like this.

$dp[i][0] = 2 \cdot dp[i+1][0] + dp[i+1][1]$.

$dp[i][1] = 4 \cdot dp[i+1][1] + dp[i+1][0]$.

base cases:

$dp[n][0] = 1 \rightarrow$ We reach to n^{th} Row and horizantal block are trying to extends

$dp[n][1] = 1 \rightarrow$ We are Reach n^{th} Row & vertical block trying to extends.

Final Sub-problem

Where We start the problem

$dp[1][0] + dp[1][1] \leftarrow$ which means one the zeroth row we put some thing and start the transition from first Row

Time Complexity

states × Avg. t.t

$$\underbrace{O(n) \times O(1)}_{}$$

$O(n) \rightarrow$ 1 test case.

$t * O(n) = O(t.n)$

Space Complexity

$O(n)$. $\leftarrow$ no. of. states.

code:

```
vector<vector<int>> dp (1e6 + 1, vector<int> (2));

int main(){
    int t;
    cin>>t;
    while(t--){
        int n;
        cin >> n;

        dp[n][0] = 1;   } Base case
        dp[n][1] = 1;
```

```
for(int i = n-2; i>=0; i-- )
   {
    dp[i][0] = (2LL * dp[i+1][0] + dp[i+1][1]) % MOD
    dp[i][1] = (4LL * dp[i+1][1] + dp[i+1][0]) % MOD

   }

 cout << (dp[1][0] + dp[1][1]) % MOD << endl;

}
```