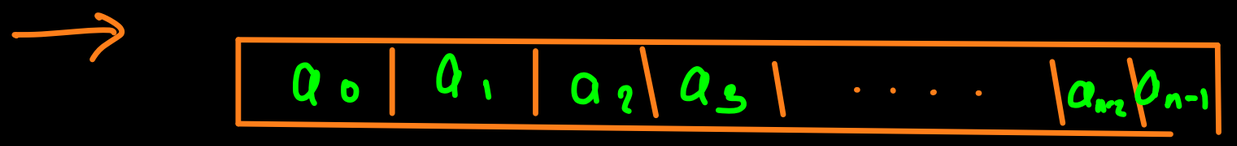
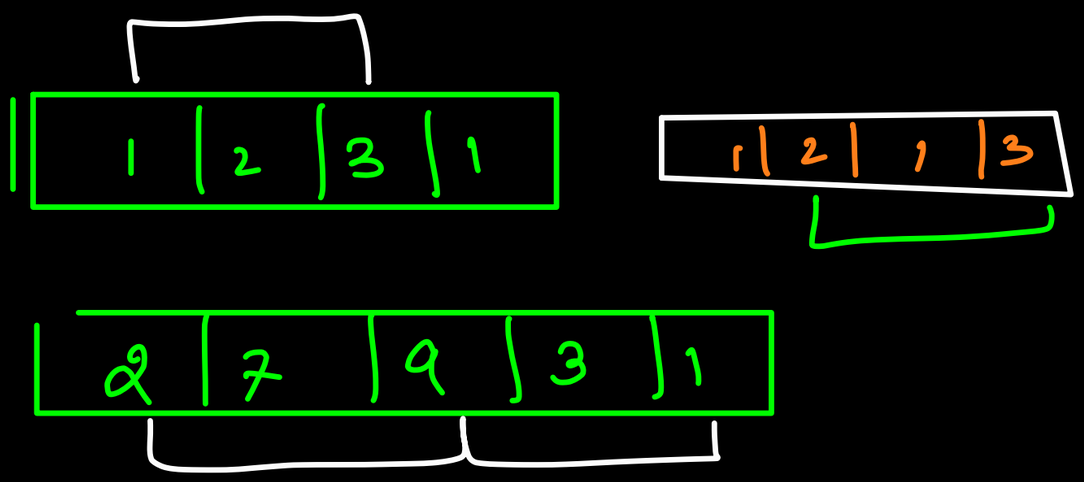
House Robber

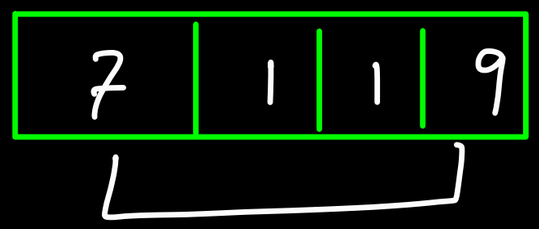


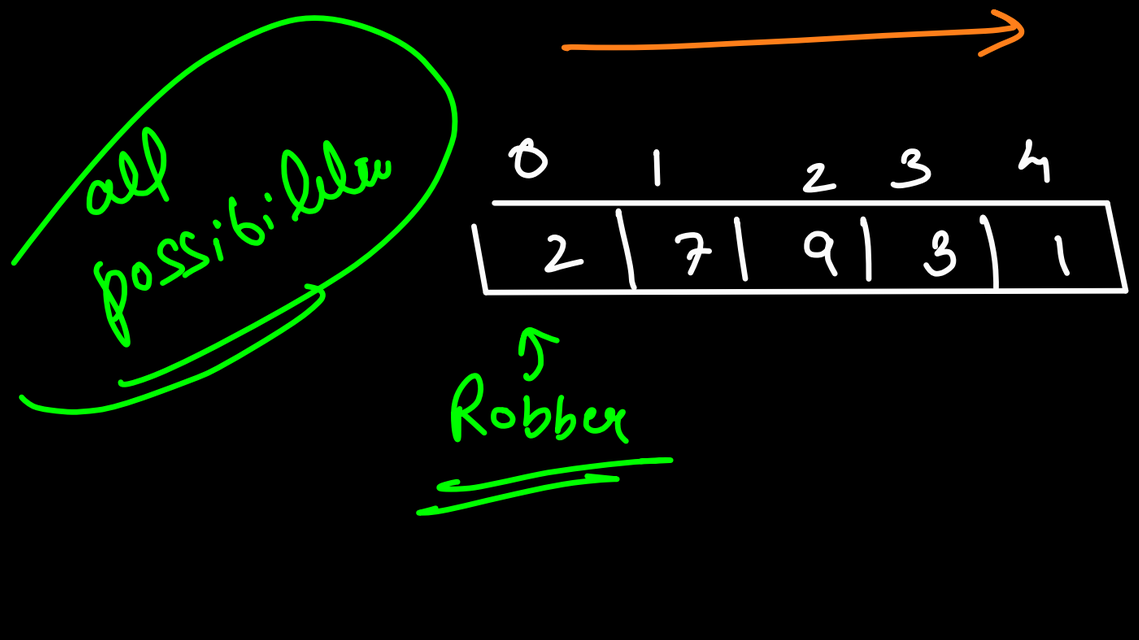
how much money each house of the street has.

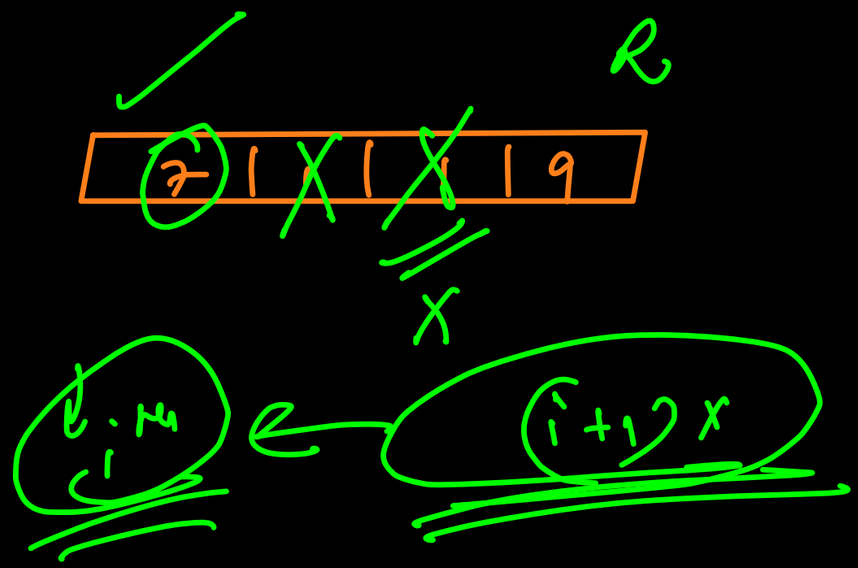
We cannot rob adjacent houses.



→ Sum of only even / odd indexes is a subA.







→ if the robber decides to rob the i^{th} house then the robber cannot rob the $(i+1)^{th}$ house. the next possible house is $(i+2)$

→ if the robber decides not to rob the i^{th} house then it has a choice that it can / cannot rob the $(i+1)^{th}$ house

TD

$f(i, arr)$ = Max profit if we start robbing from i^{th} house to $(n-1)^{th}$ house.

$= \max$ {

$arr[i] + f(i+2, arr)$ robbing the i^{th} house

$f(i+1, arr)$ not rob the i^{th} house

}

DP

ans → $f(0, arr)$

```
if(i >= arr.length)
return 0;
```

pick or not pick

10 dp

if (i >= arr.length) return 0;

0	1	2	3
7	1	1	9

f(i, arr)

only count factors

State

(0, arr)

✓ 7 — (2, arr) — 9

X — 10 — (1, arr)

(2, arr): ✓ 1 — (4, arr) — 8 ; X — (3, arr) — 9

(3, arr): ✓ 9 — (5, arr) — 16 ; X — (4, arr) — 7

(1, arr): ✓ 1 — (3, arr) — 9 ; X — (2, arr) — 9

(3, arr): ✓ 9 — (5, arr) — 10 ; X — (4, arr) — 1

(2, arr): ✓ 1 — (4, arr) — 1 ; X — (3, arr) — 9

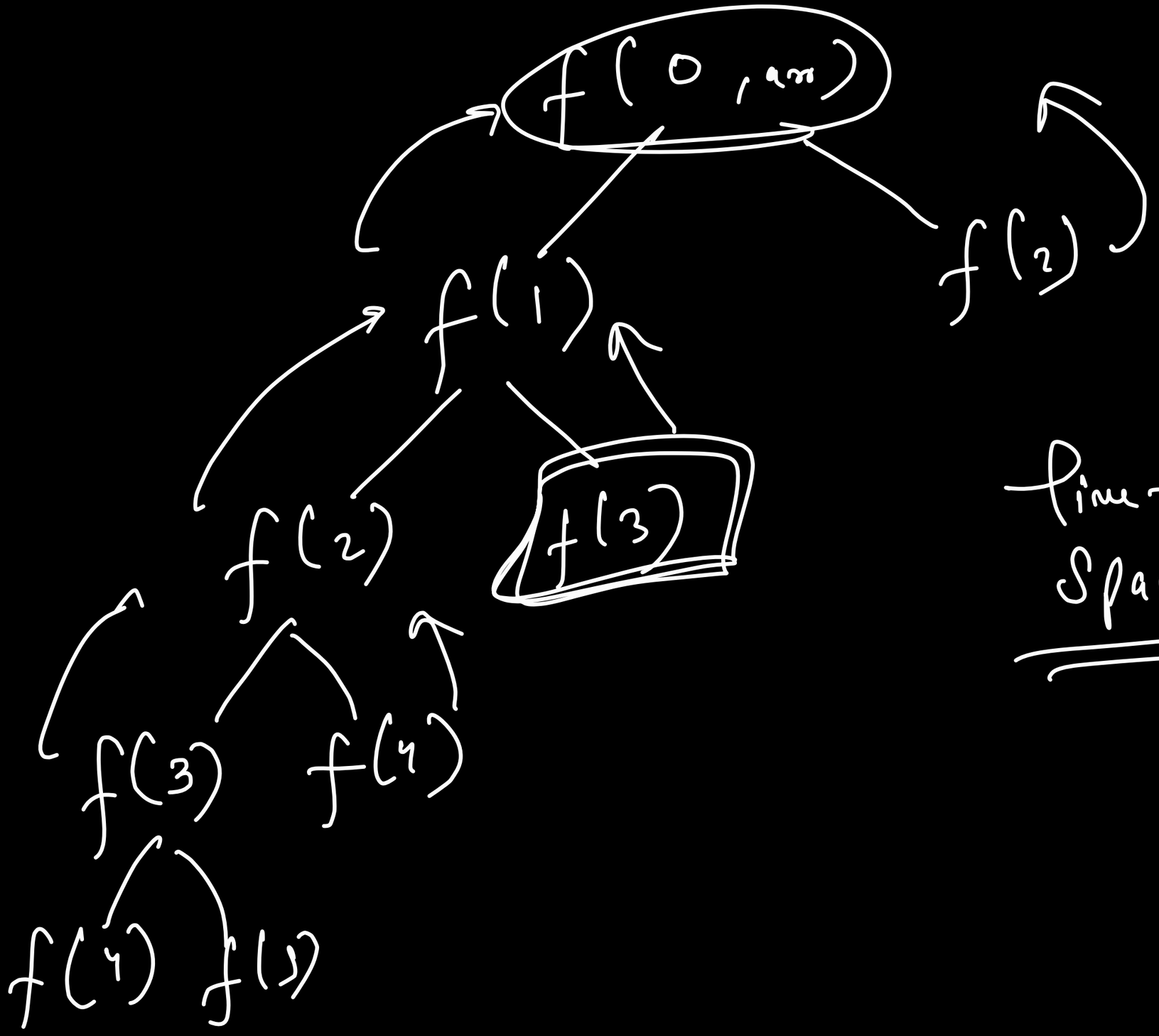
(3, arr): ✓ 9 — (5, arr) — 9 ; X — (4, arr) — 0

Base

• Overlapping in Sub

→ TD → memoization

BU → Tabulation



Time → O(n)

Space → O(n)

B.U

3

2	7	9	3	1

dp

0	1	2	3	4
12	10	10	3	1

$$\begin{cases} dp[n-1] = arr[n-1] \\ dp[n-2] = \max(arr[n-1], arr[n-2]) \end{cases}$$

Base Cases

ans → dp[0] → 12

$$f(i, arr) = \max\left(f(i+1, arr),\ arr[i] + f(i+2, arr)\right)$$

$$dp[i] = \max\left(dp[i+1],\ arr[i] + dp[i+2]\right)$$

$$dp[2] = \max\left(dp[3],\ 9 + dp[4]\right)$$

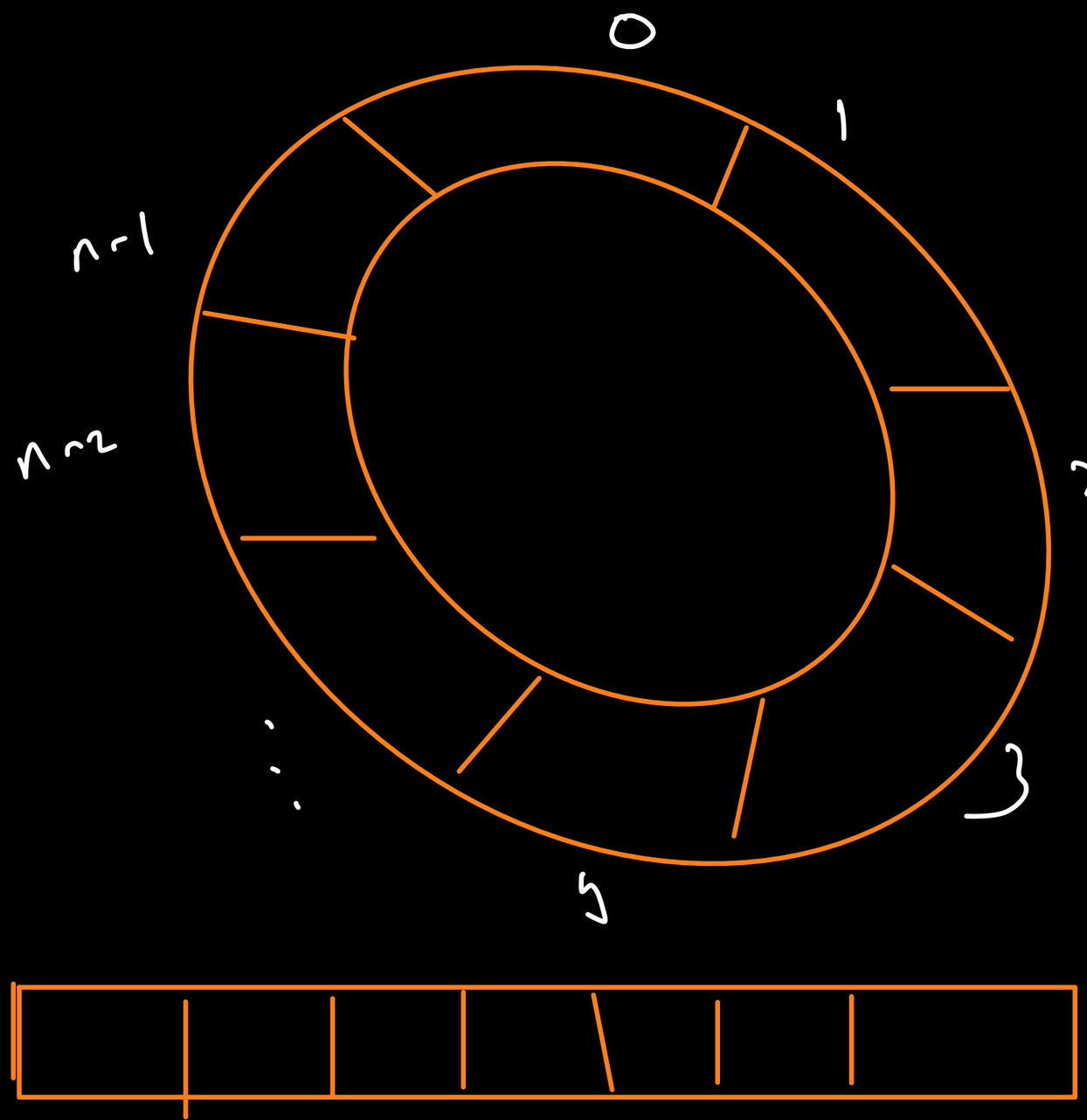
$$dp[1] = \max\left(dp[2],\ 7 + dp[3]\right)$$

```
for ( i = n-3 ; i>=0; i--) {
        dp[i] = max (dp[i+1], arr[i] + dp[i+2])
}
return dp[0]
```

$$dp[i] = \max(dp[i-1],\ a[i] + dp[i-2])$$

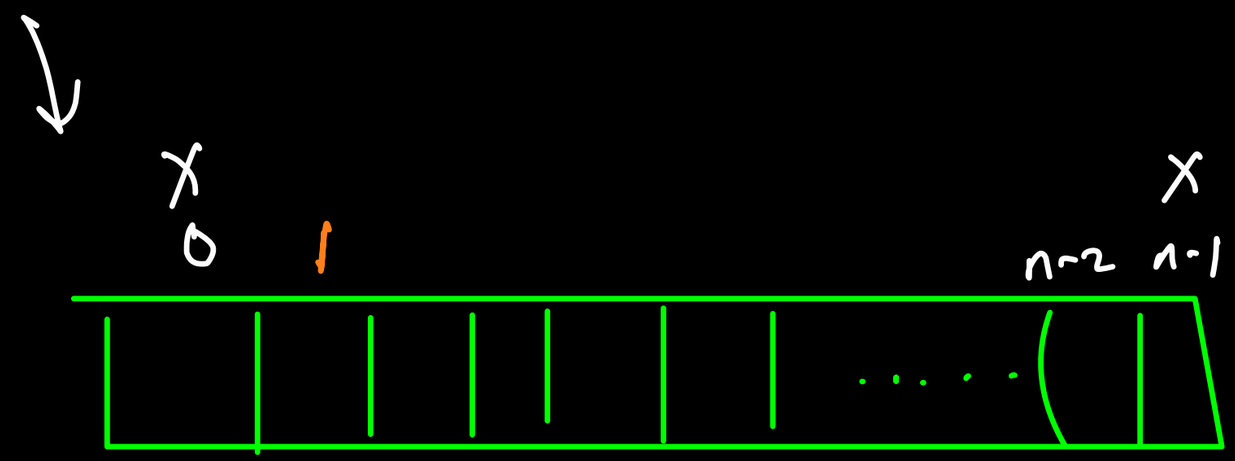
Time → O(n)

Space → O(n)



(7, 1, 1, 9)

1 2 3 1



Pivot → O(n)

Space

(0 — n-2) → 1d array

(1 — n-1) → 1d array

we already know how to solve for 1d array

max (case1, case2)

Vacation - Atcoder

	a	b	c
	0	1	2
0	10	20	30
1	5	10	90
2			
3			
4			

$n \times 3$

choosing the activity with highest happiness will be WA.

Try all possibilities

$$f(a, i) = a_i + \max\left(f(b, i+1), f(c, i+1)\right)$$

max happiness we can get by doing a activity on the i^{th} day

$$f(b,i) = b_i + \max\left(f(a,i+1),\ f(c,i+1)\right)$$

$$f(c,i) = c_i + \max\left(f(a,i+1),\ f(b,i+1)\right)$$

$$\max\left(f(a,i),\ f(b,i),\ f(c,i)\right)$$

$$\text{ans} \rightarrow \max\left(f(a,0),\ f(b,0),\ f(c,0)\right)$$

	0	1	2
0	10	40	70
1	20	50	60
2	30	60	90

3 - 1d array

	0	1	2
0	10 + 140 = 150	40 + 120 = 160	70 + 140 = 210
1	20 + 90 = 110	50 + 90 = 140	60 + 60 = 120
2	30	60	90

dp

n*3

a → 0
b → 1
c → 2

$dp[0,0] = a[0][0] + \max(dp[1,1], dp[2,1])$

$dp[1,0] = a[1][0] + \max(dp[0,1], dp[2,1])$

dp(i, j)

i^th activity on j^th day

$f(0,j) \rightarrow a[0,j] + \max(f(1,j-1), f(2,j-1))$

↓

in each col or j in

dog