Dynamic Programming

technique used to solve problems efficiently

* optimisation

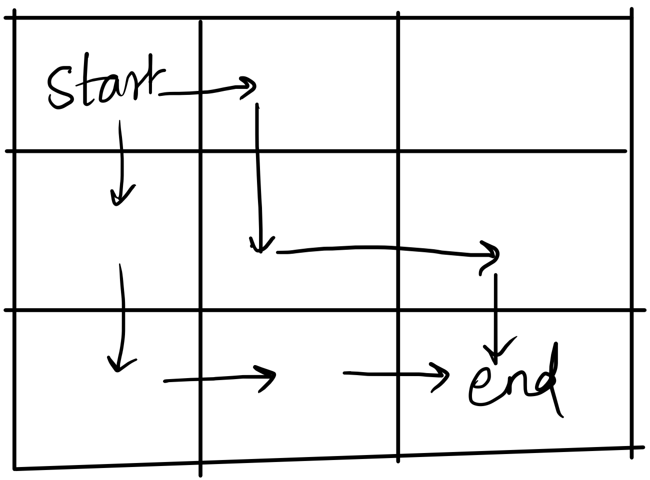
$$1 + 2 + 3 + 2 = 8$$

$$1 + 2 + 3 + 2 + 1 = 9$$

$$8 \quad + 1$$

store the result for future use

Number of paths

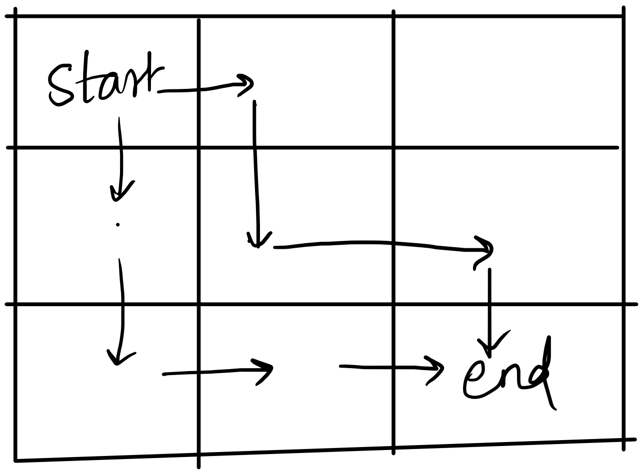


moves

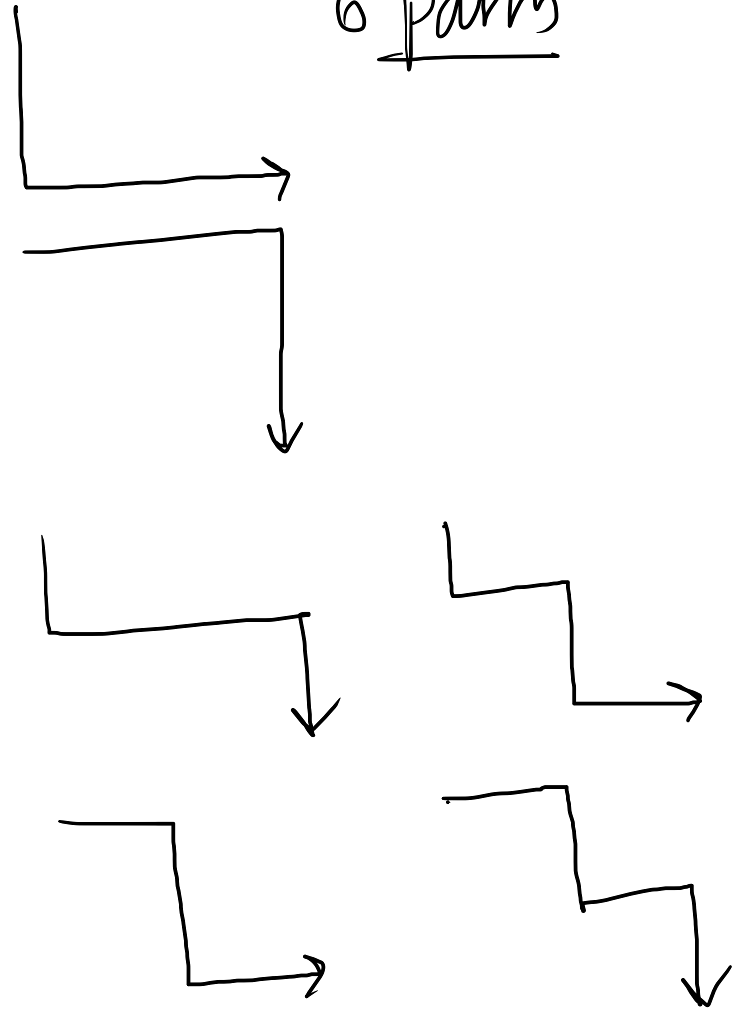
right $\rightarrow$

down $\downarrow$

Number of paths

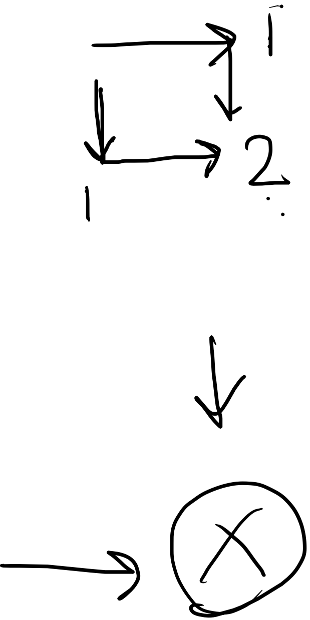


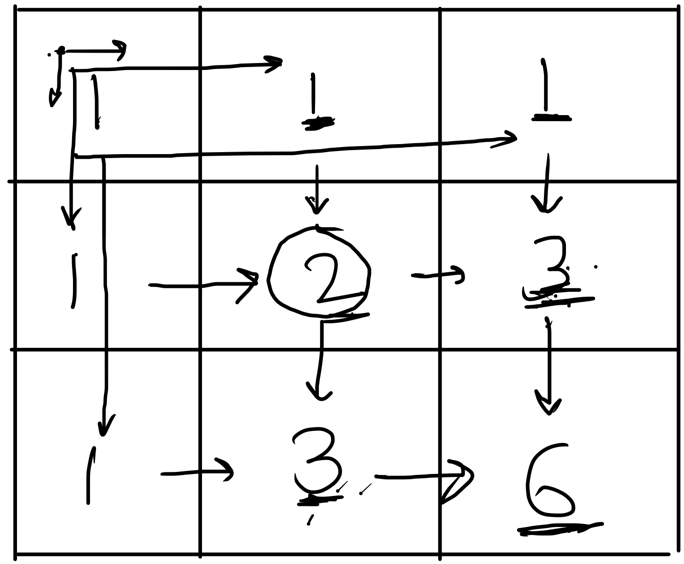
6 paths



number of paths (K) = nop (G) + nop (J)

A	B	C	D
E	F	G	H
I	J	K	L
M	N	O	P

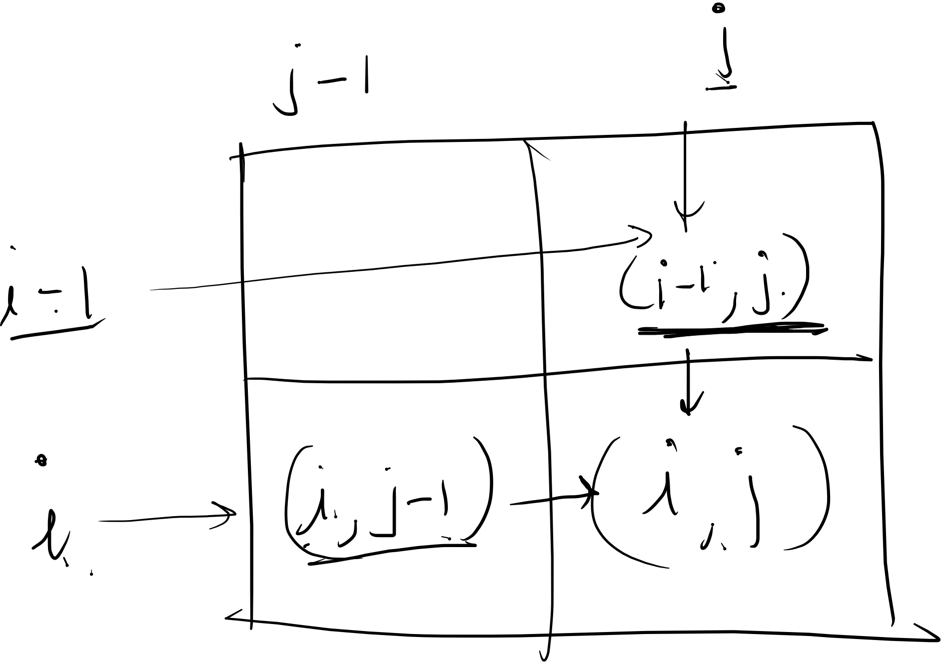

1
2
1


1 1 1
1 2 3
1 3 6

m

1	1	1	1
1	2	3	4
1	3	6	10
1	4	10	20

```
if (i==0 || j==0)
    m[i][j] = 1
```



$$m[i][j] = m[i-1][j] + m[i][j-1]$$

$m \times n$

$T.C = O(m \times n)$

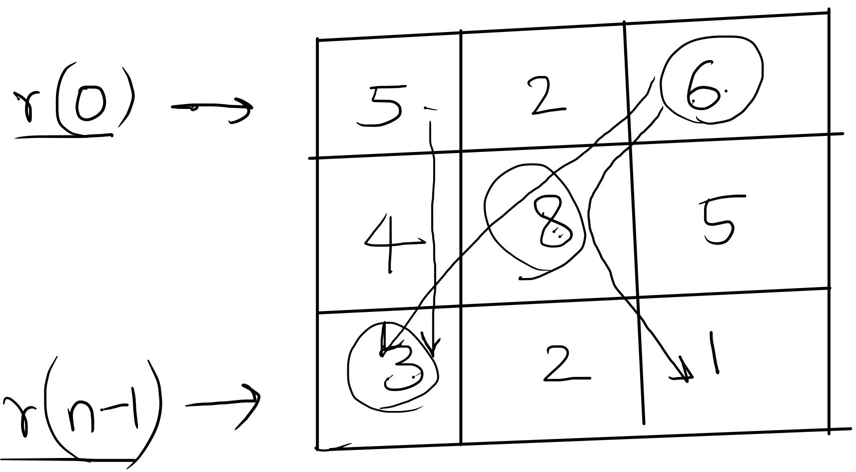
$S.C = O(m \times n)$

Maximum Path Sum

↓ down

↙ down left

↘ down right



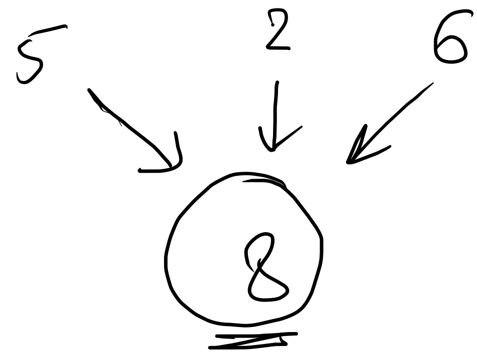
6 8 3

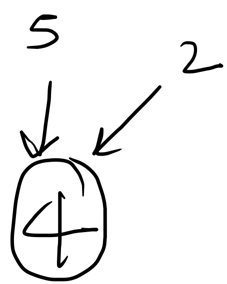
sum = 17

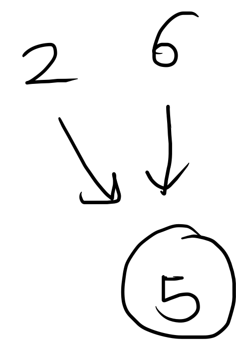
1	2	9
9	3	2
3	1	9

max Sum = $9 + 3 + 9 = 21$

5	2	6
4	8	5
3	2	1



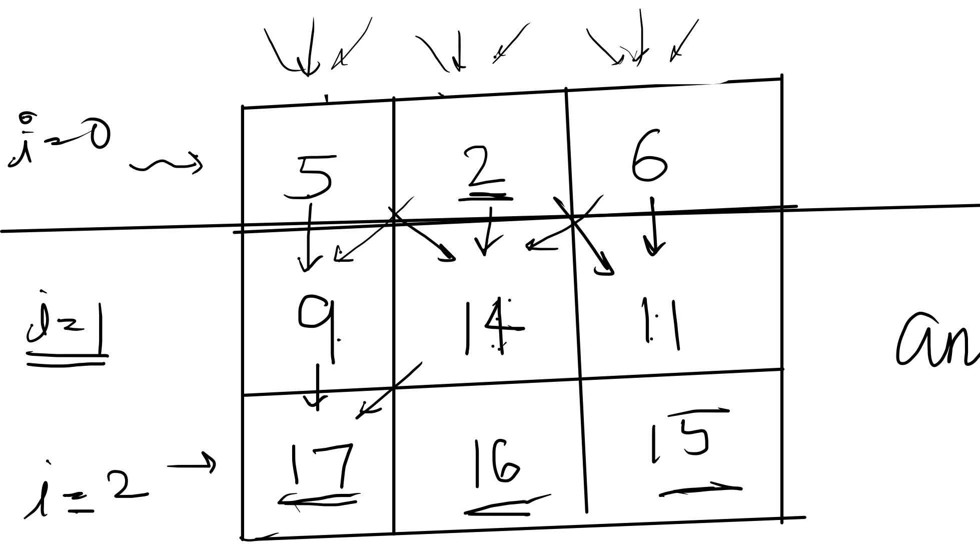




Maximum Path Sum

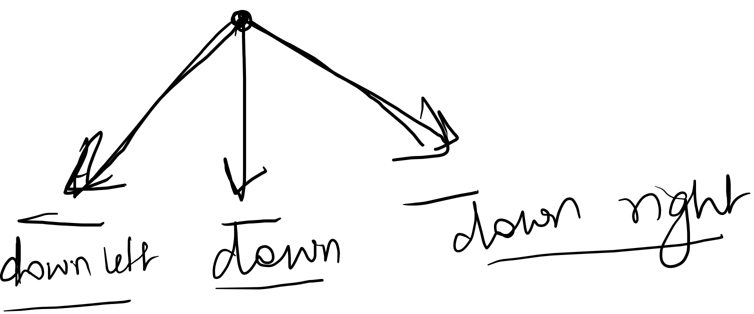
5	2	6
4	8	5
3	2	1

Maximum Path Sum

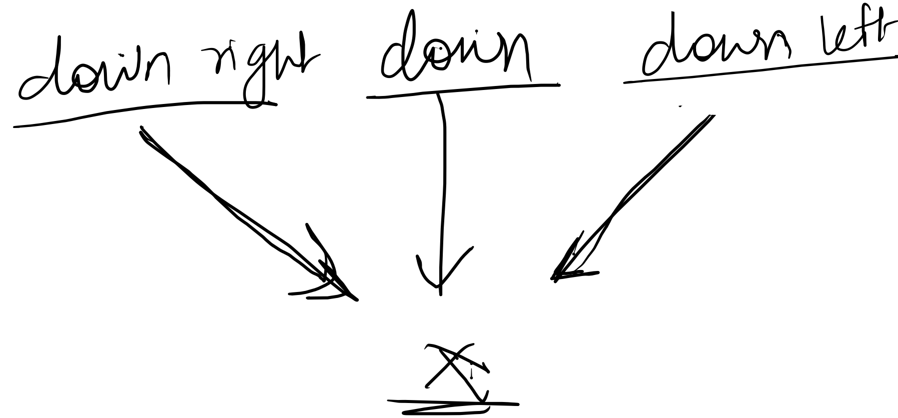


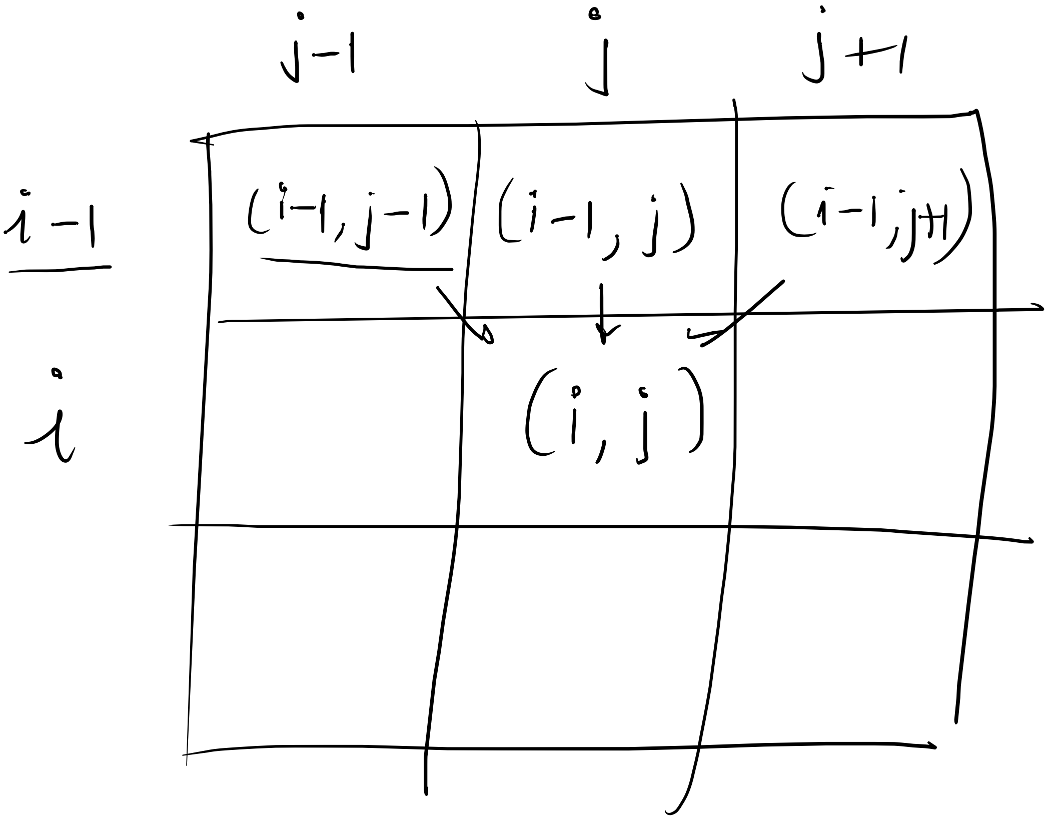
ans = max(last row elements)

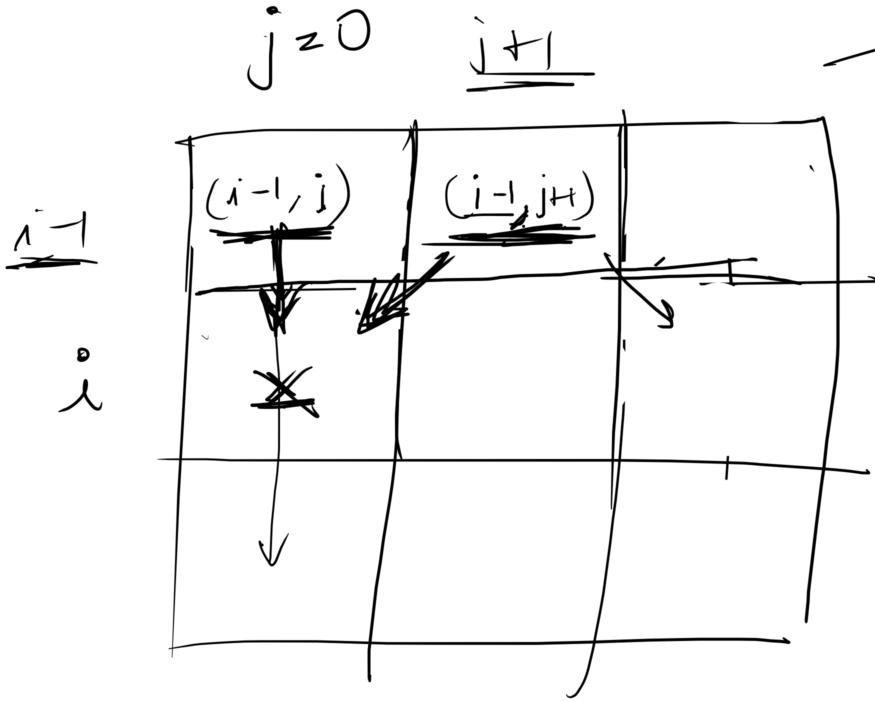
ways to go



ways to reach



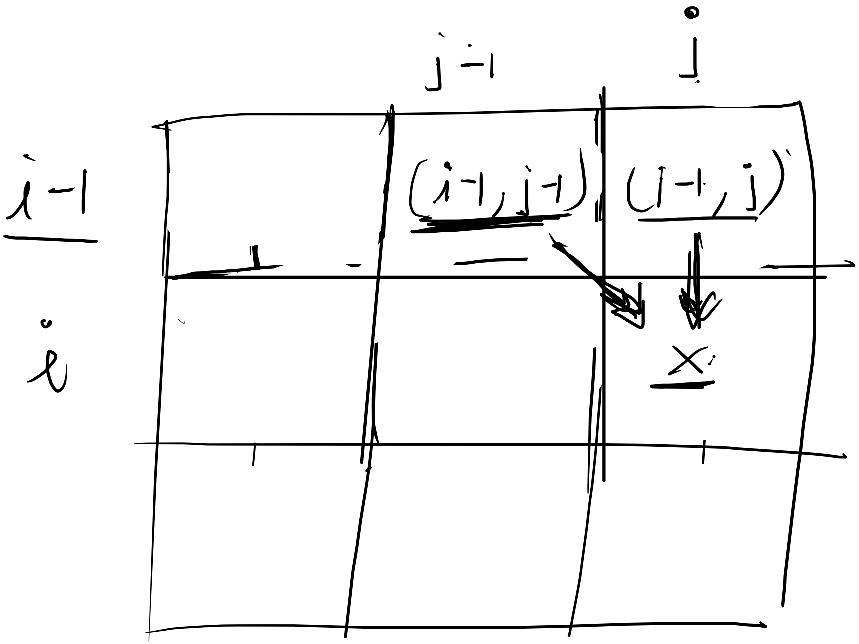

j-1
j
j+1
i-1
i
(i-1, j-1)
(i-1, j)
(i-1, j+1)
(i, j)



first column

$$\text{if } (j == 0)$$

$$\max(m[i-1][j], m[i-1][j+1])$$



Last column

if $(j == n-1)$

$$m[i][j] = \max\left(m[i-1][j-1],\ m[i-1][j]\right)$$

$$\max(\underline{a}, \underline{b}, \underline{c})$$

$$= \max(\underline{a}, \underline{\max(b, c)})$$

$$\max(5, \underline{7, 8})$$

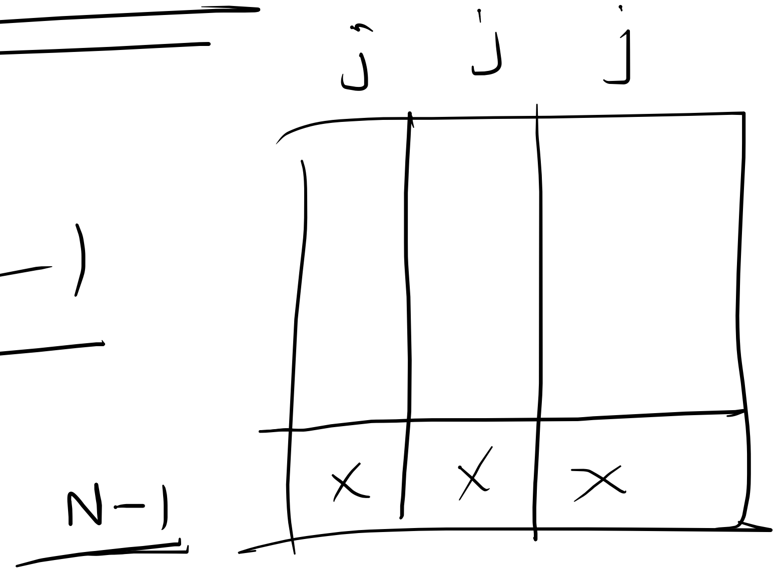
$$\max(5, \underline{\max(7, 8)})$$

$$\max(\underline{5}, \underline{8})$$

$$= 8$$

Last row

$i = n-1$



matrix $\rightarrow n \times n$

T.C $= O(n^2)$

SC $= O(n \times n)$

Aux space $= O(1)$

matrix → $n \times n$

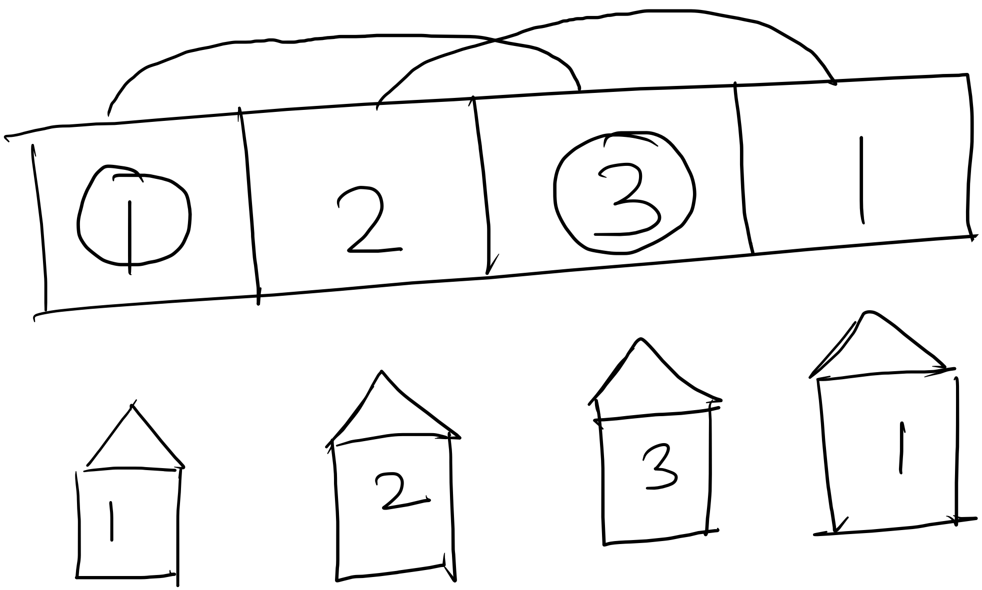
$T.C = O(n^2)$

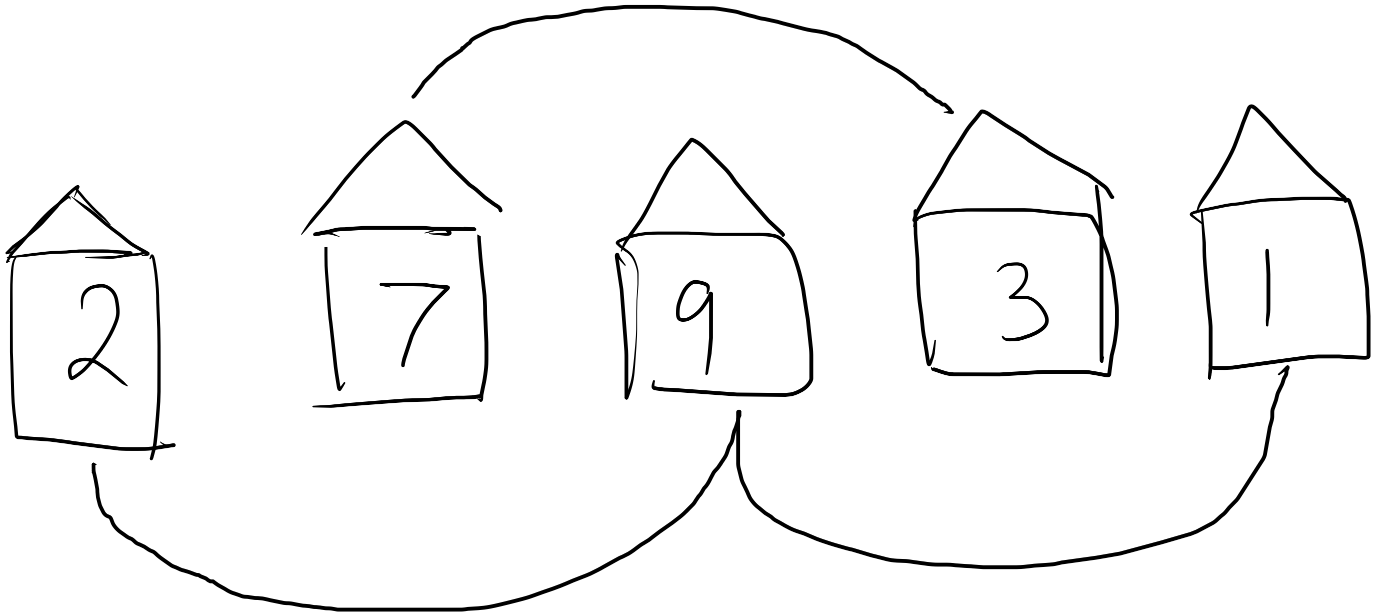
$SC = O(n \times n)$

Aux space $= O(1)$

House Robber

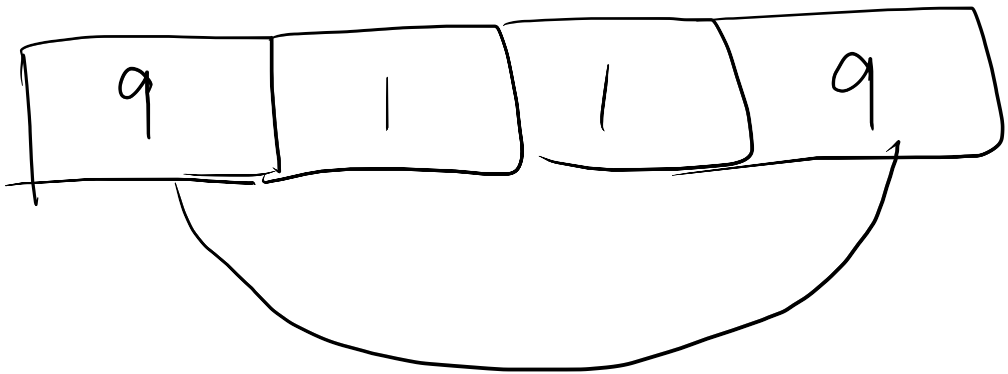
max = 1 + 3 = 4





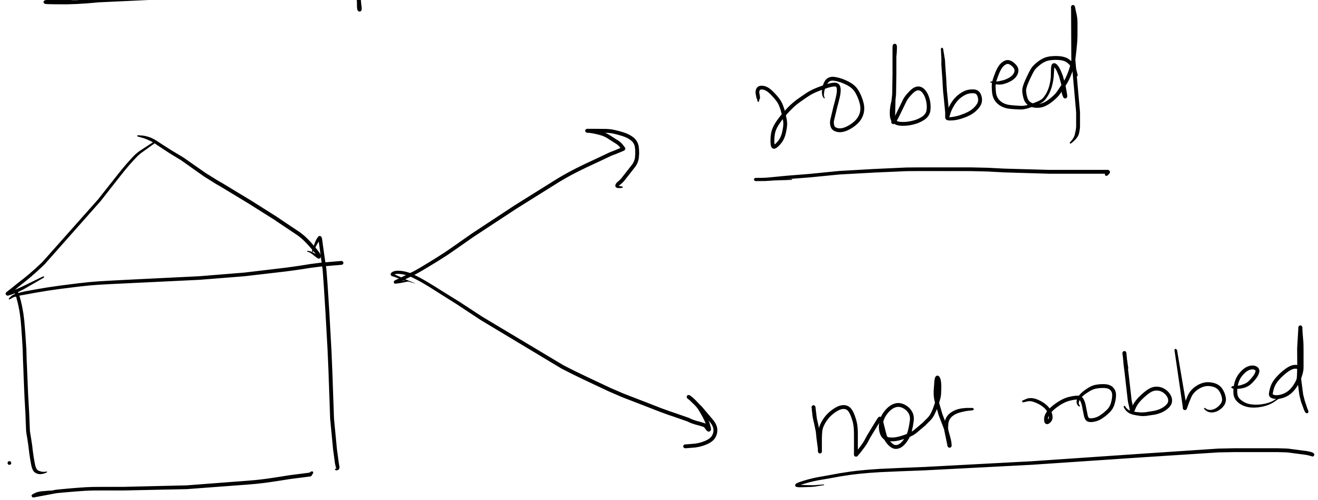
ans = 2 + 9 + 1 = 12

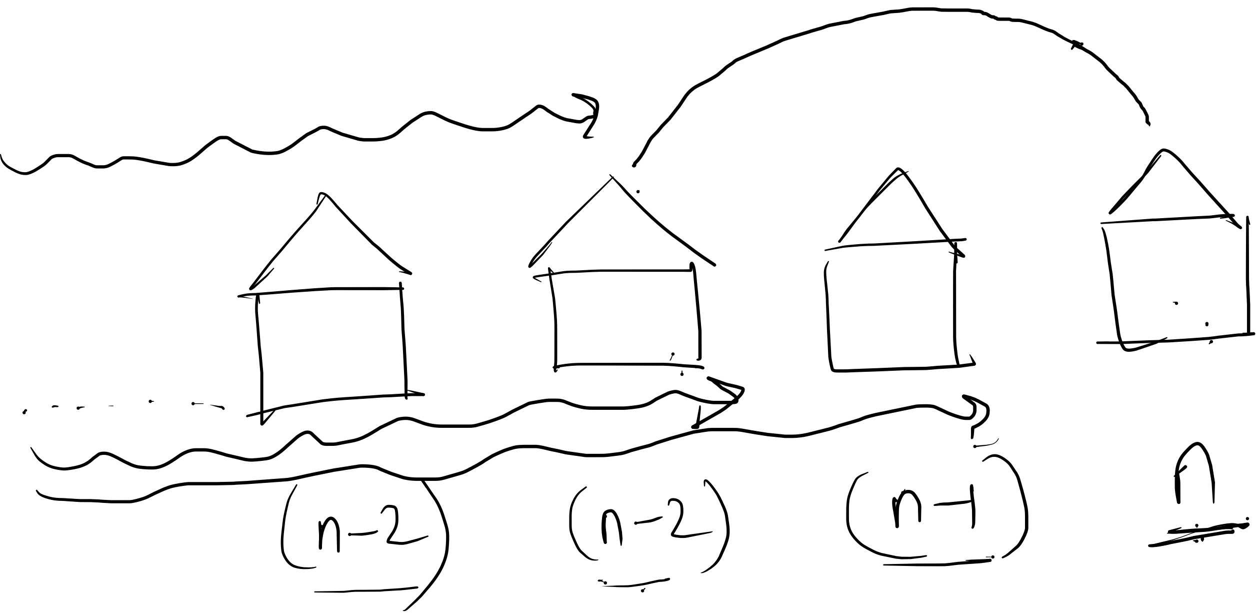
7 + 3 = 10



$9 + 9 = 18$

2 options



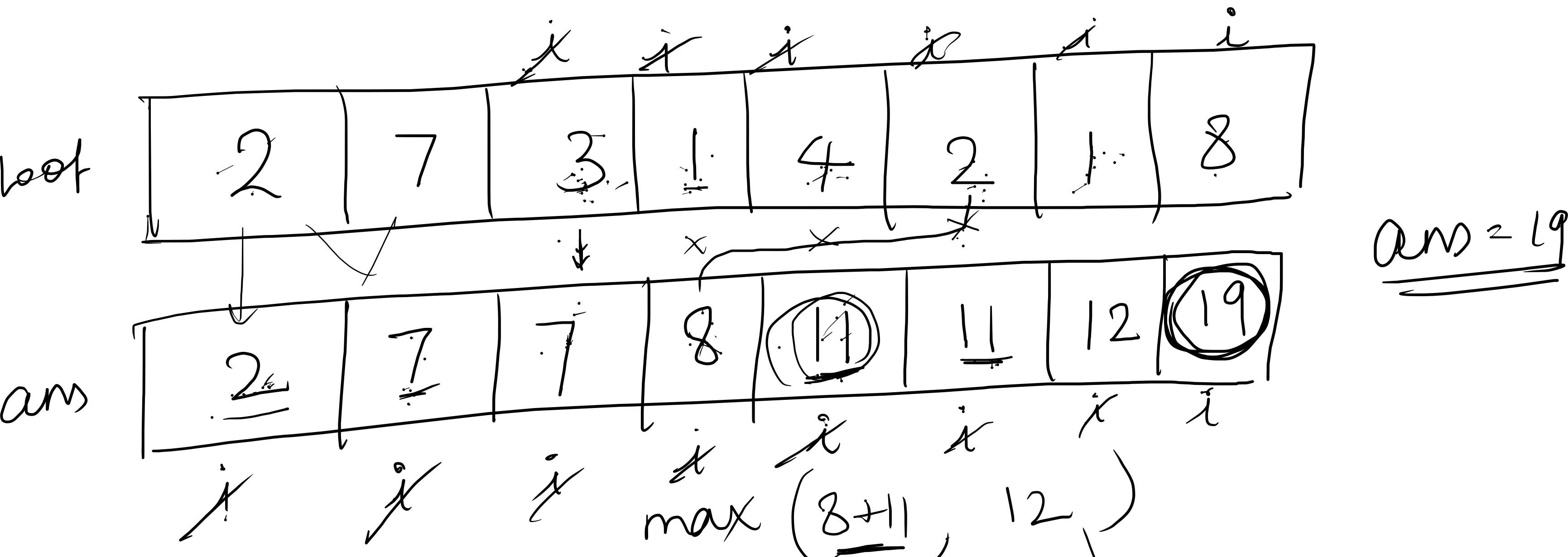


final loot = max(loot[n] + finalloot [n-2],
(till n) finalloot [n-1])

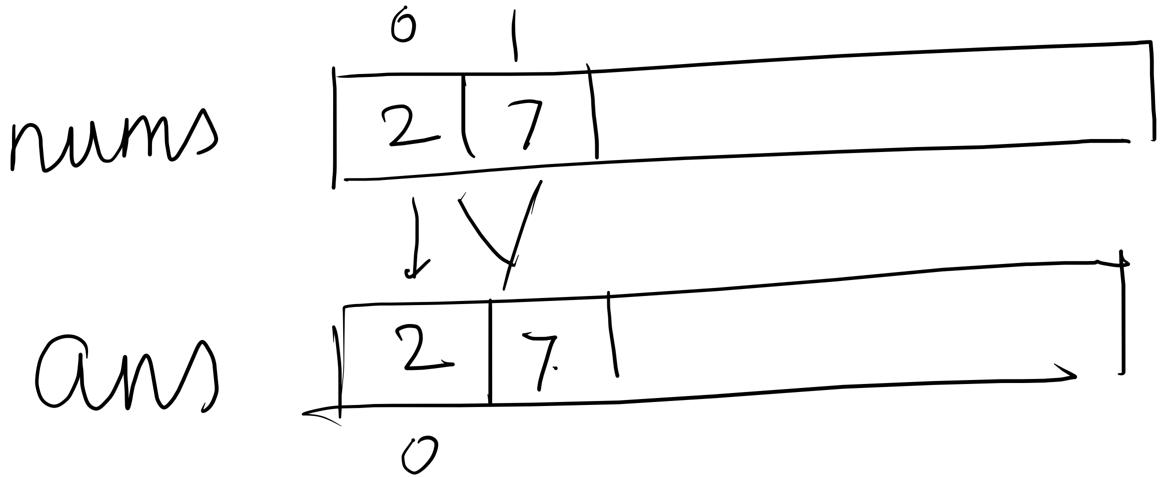
If last house is robbed

finalloot[n] = (loot[n] + finalloot[n-2])

If last house is not robbed = finalloot[n-1]

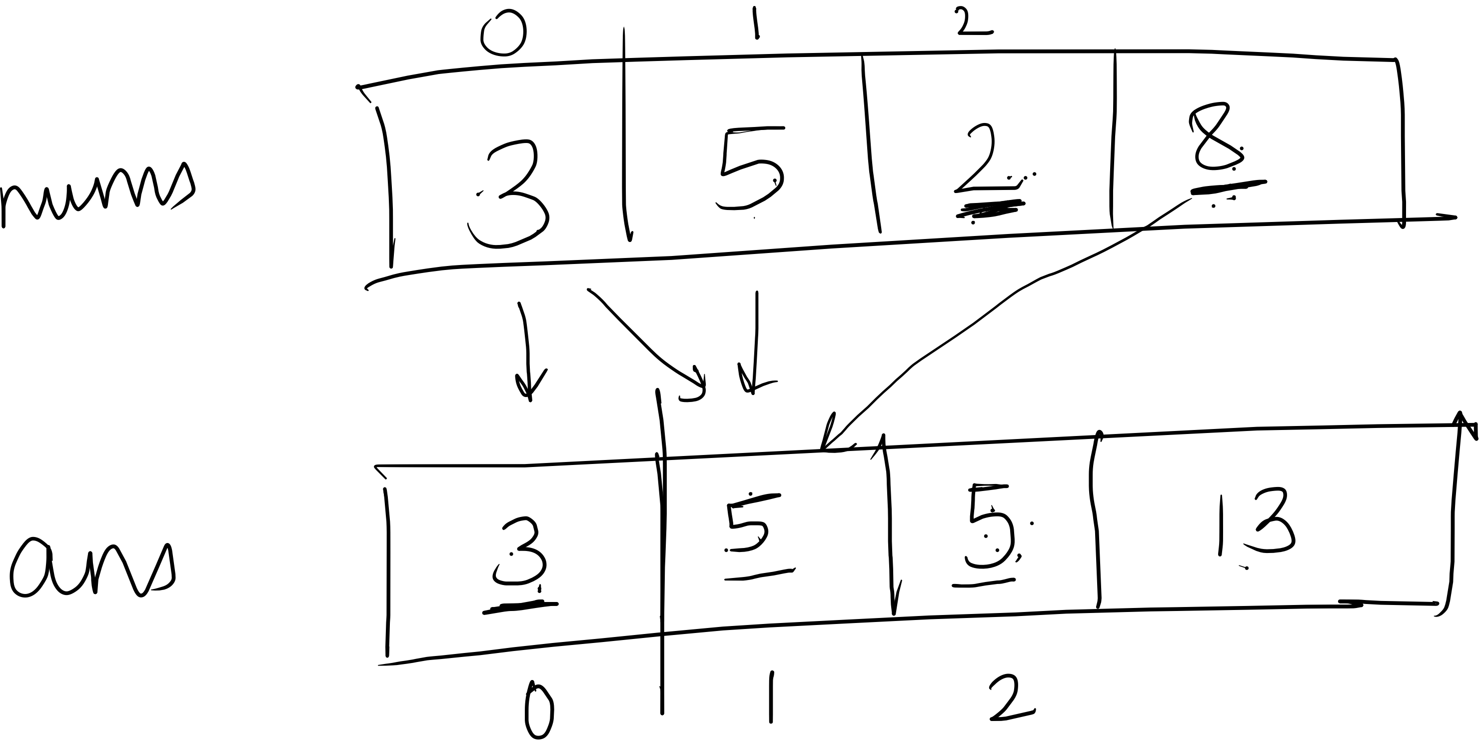


$$ans[i] = \max(\underbrace{loot[i] + ans[i-2]}_{\text{robbed}},\ \underbrace{ans[i-1]}_{\text{not robbed}})$$



$$ans[0] = nums[0]$$

$$ans[1] = max(nums[0], nums[1])$$



$\max(8+5, 5)$

T.C = $O(n)$

S.C = $O(n)$