

Dynamic Programming

→ technique used to solve problems efficiently

* optimisation

$$\underline{1 + 2 + 3 + 2 = 8}$$

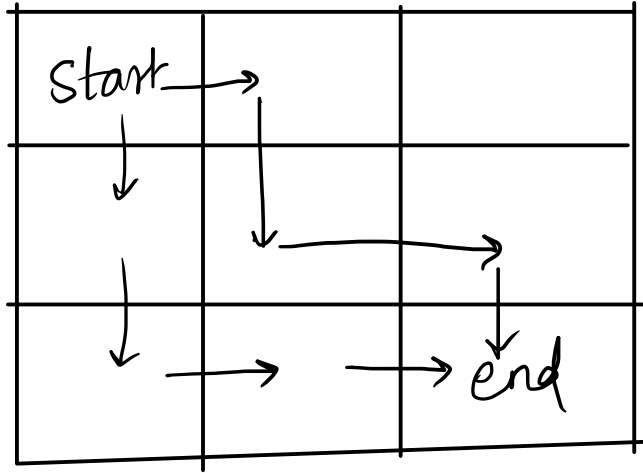
$$\underline{1 + 2 + 3 + 2} + 1 = 9$$

↓

8 + 1

store the result for future use

Number of paths

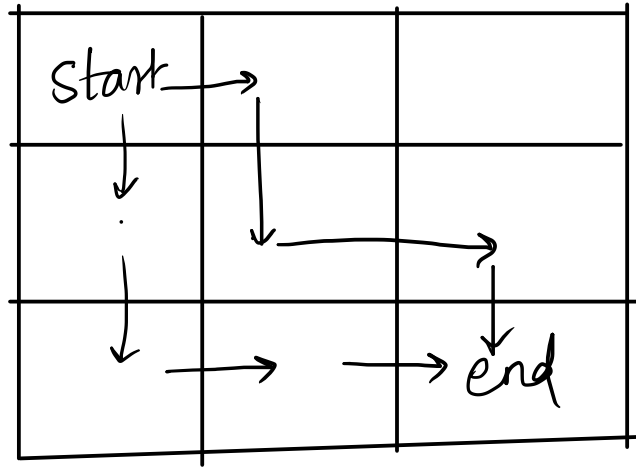


moves

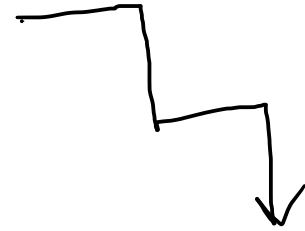
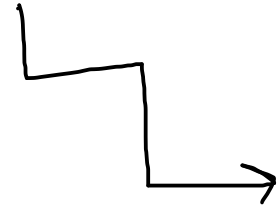
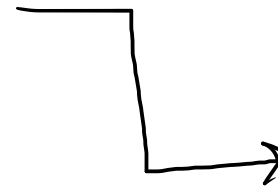
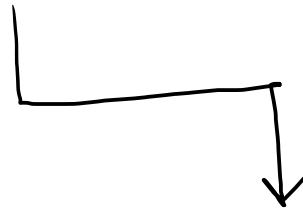
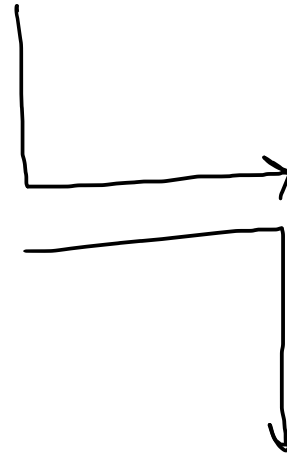
right →

down ↓

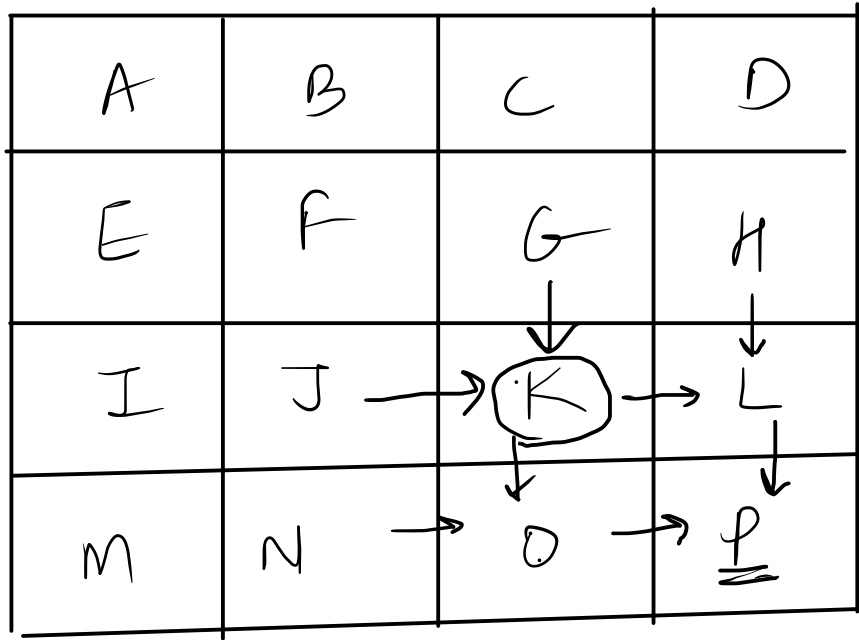
Number of paths



6 paths



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



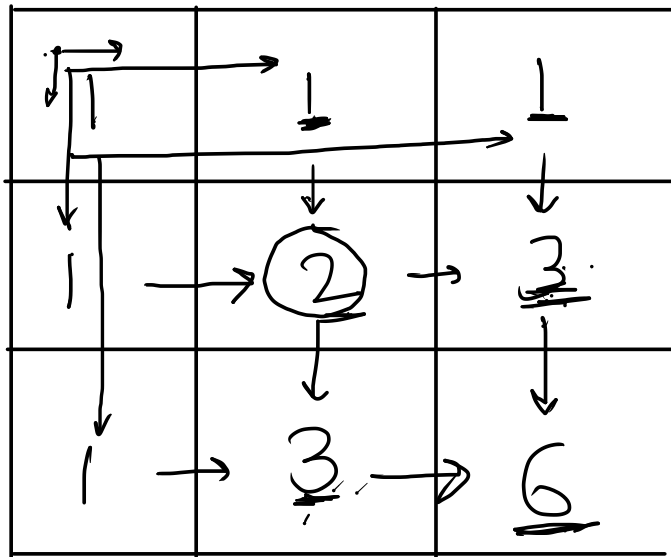
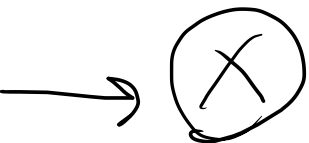
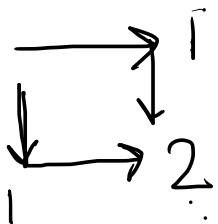


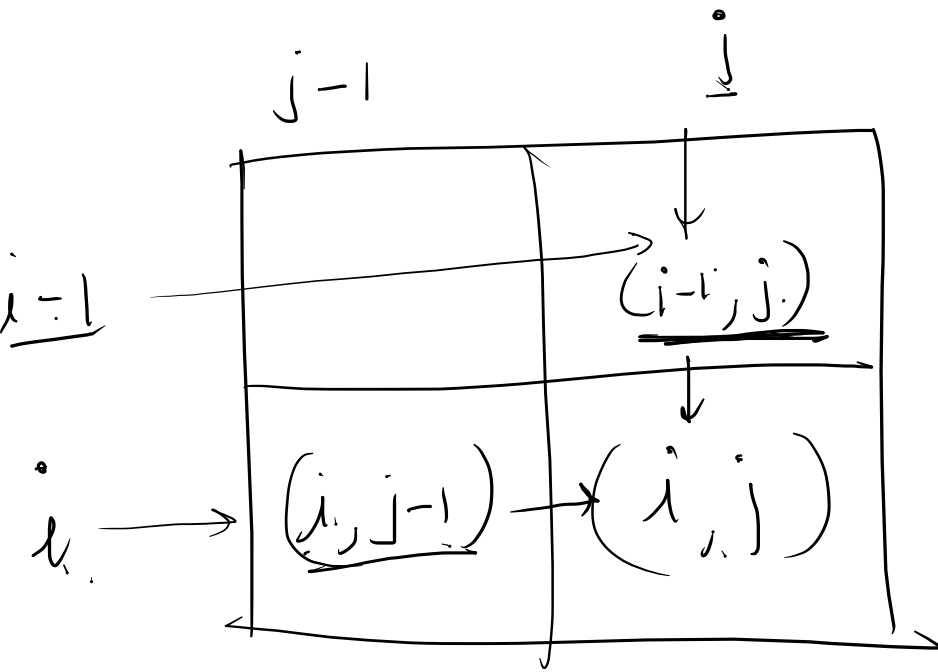
Diagram illustrating a 4x4 matrix with indices i and j ranging from 0 to 3. The matrix values are:

1	1	1	1
1	2 2	3	4
1	¹ 3	6 6	10
1	4	10	<u>20</u>

Annotations: A vertical line on the left is labeled m with arrows indicating the row index i . A horizontal line on top is labeled j with arrows indicating the column index.

if ($i=0$ || $j=0$)

$m[i][j] = 1$



$$\underline{m[i][j]} = \underline{m[i-1][j]} + \underline{m[i][j-1]}$$

$$\underline{T.C = O(m \times n)} \quad \underline{\underline{m \times n}}$$

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

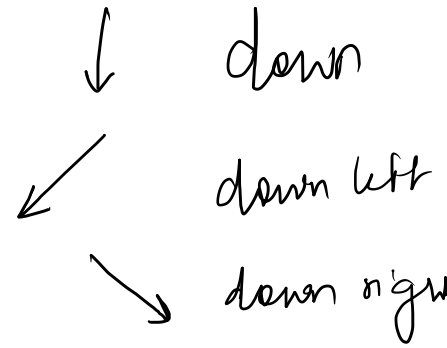
Maximum Path Sum

$r(0)$ \rightarrow

5	2	6
4	8	5
3	2	1

$r(n-1)$ \rightarrow

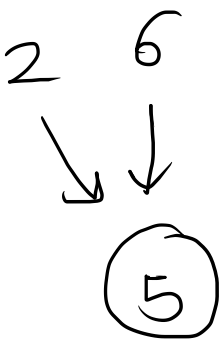
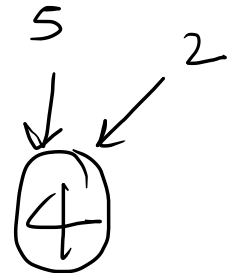
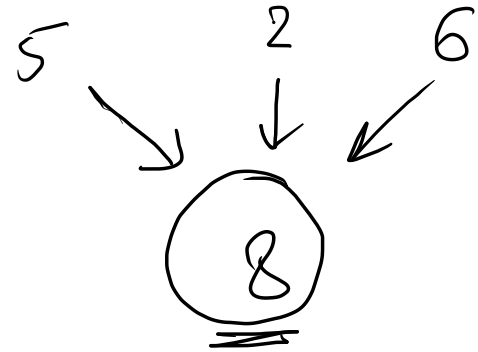
$$\begin{array}{r} 6 \ 8 \ 3 \\ \hline \text{sum} = 17 \end{array}$$



1	2	9
9	3	2
3	1	9

$$\text{max Sum} = 9 + 3 + 9 = 21$$

5	2	6
4	8	<u>5</u>
3	2	1



Maximum Path Sum

5	2	6
4	8	5
3	2	1

Maximum Path Sum

$i=0 \rightsquigarrow$

$\swarrow \searrow \swarrow \searrow$

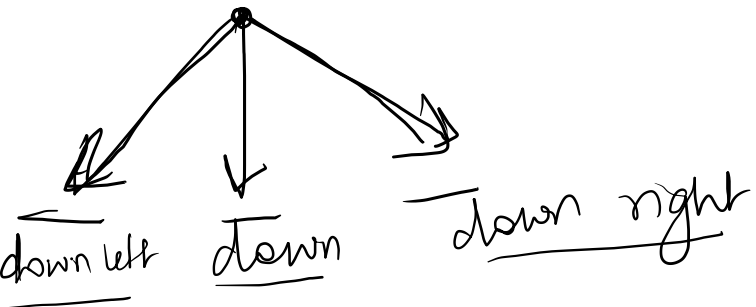
5	<u>2</u>	6
$\downarrow \swarrow$	$\downarrow \swarrow$	$\downarrow \swarrow$
<u>9</u>	14	11
$\downarrow \swarrow$		
<u>17</u>	<u>16</u>	<u>15</u>

$\underline{\underline{i=1}}$

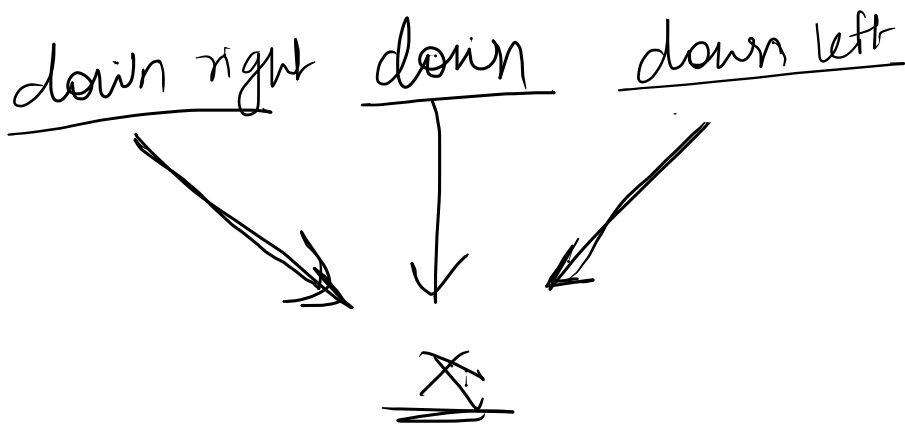
$i=2 \rightarrow$

Ans = max (last row elements)

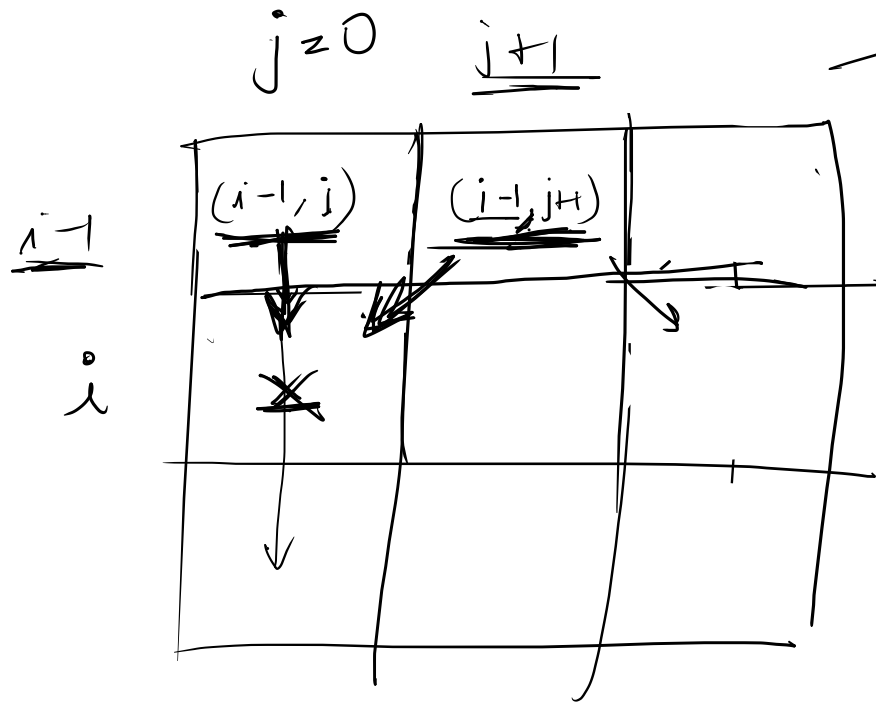
ways to go



ways to reach

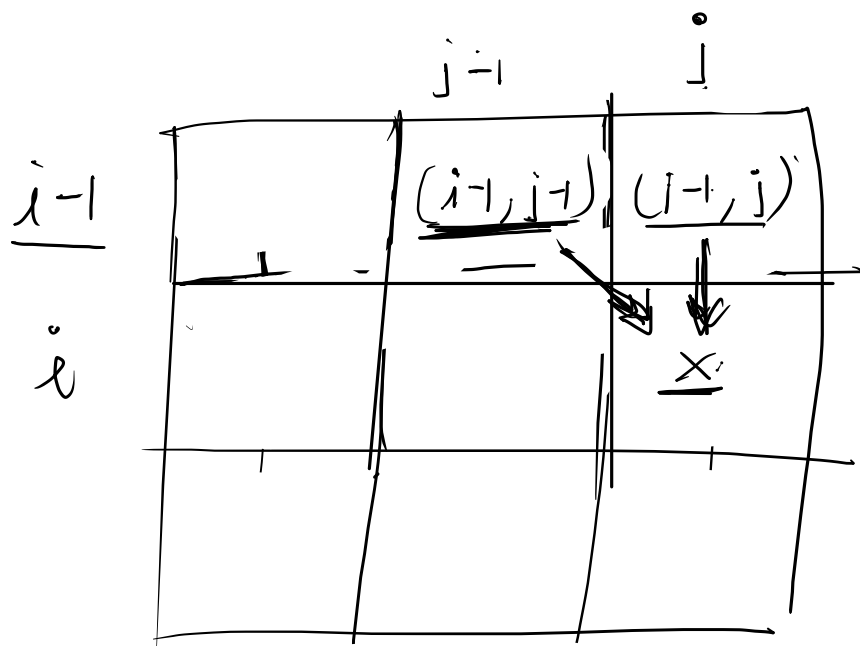


	$j-1$	j	$j+1$
$i-1$	<u>$(i-1, j-1)$</u>	$(i-1, j)$	$(i-1, j+1)$
i		(i, j)	



first column
if ($j = 0$)

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



last column

if $(j = n-1)$

$$m[i][j] = \max \left(\begin{array}{l} m[i-1][j-1], \\ m[i-1][j] \end{array} \right)$$

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

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

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

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

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

$$= 8$$

last row

$$\underline{j = n-1}$$

$$\underline{n-1}$$

j	j	j
x	x	x

matrix $\rightarrow n \times n$

$$\underline{T.C = O(n^2)}$$

$$\underline{S.C = O(n \times n)}$$

$$\underline{\text{Aux space} = \underline{\underline{O(1)}}}$$

matrix $\rightarrow n \times n$

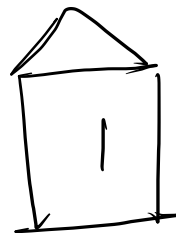
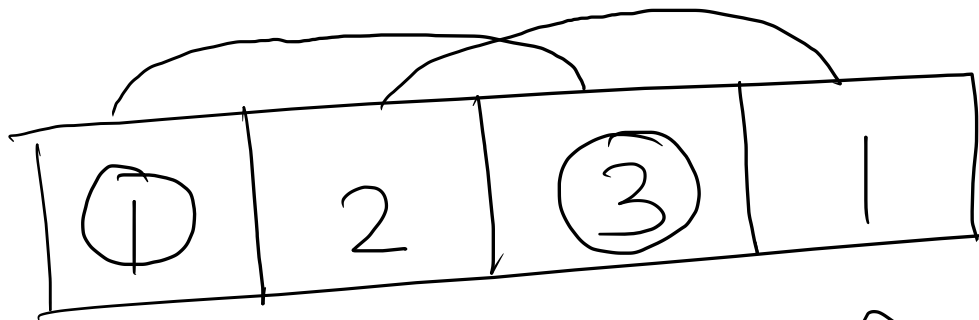
$$\underline{T.C = O(n^2)}$$

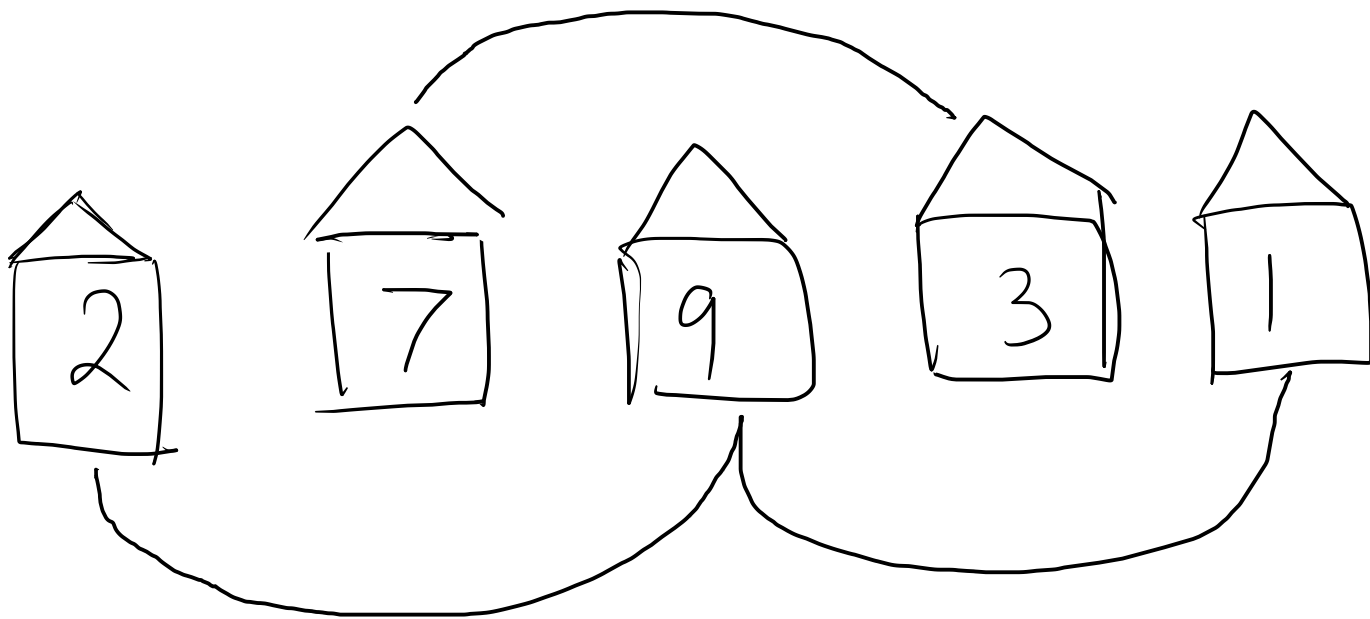
$$\underline{S.C = O(n \times n)}$$

$$\underline{\text{Aux space} = \underline{O(1)}}$$

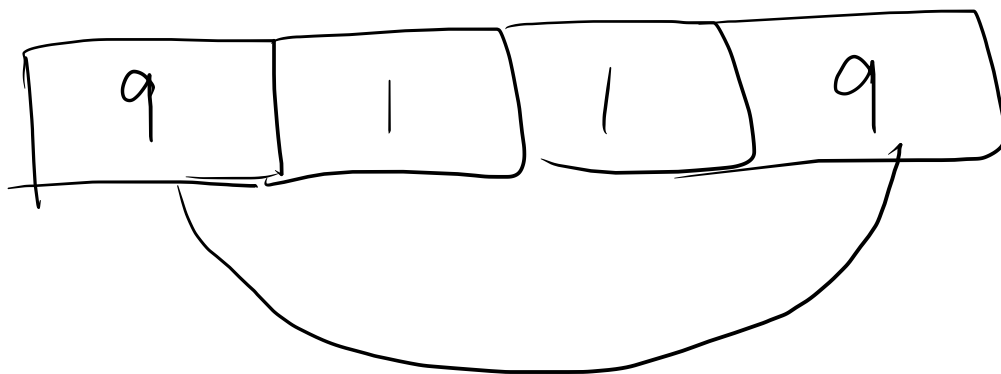
House Robber

$$\text{max} = 1 + 3 = 4$$



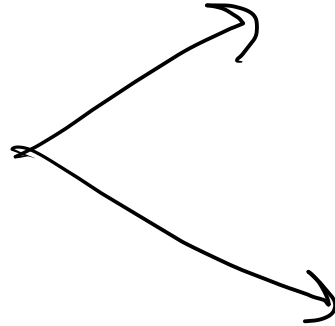
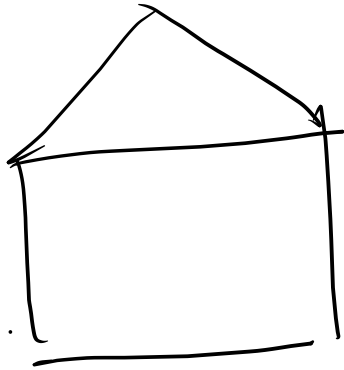


ans $\rightarrow 2 + 9 + 1 = \textcircled{12} \checkmark$
 $\rightarrow 7 + 3 = 10$



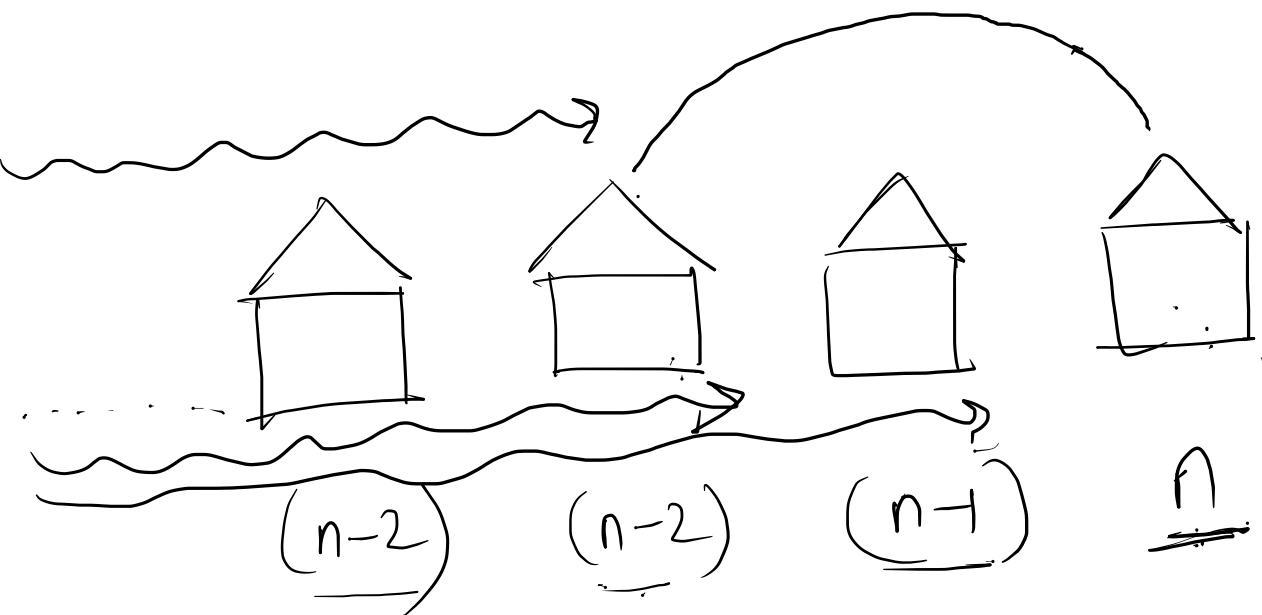
$$9 + 9 = 18$$

2 options



robbed

not robbed



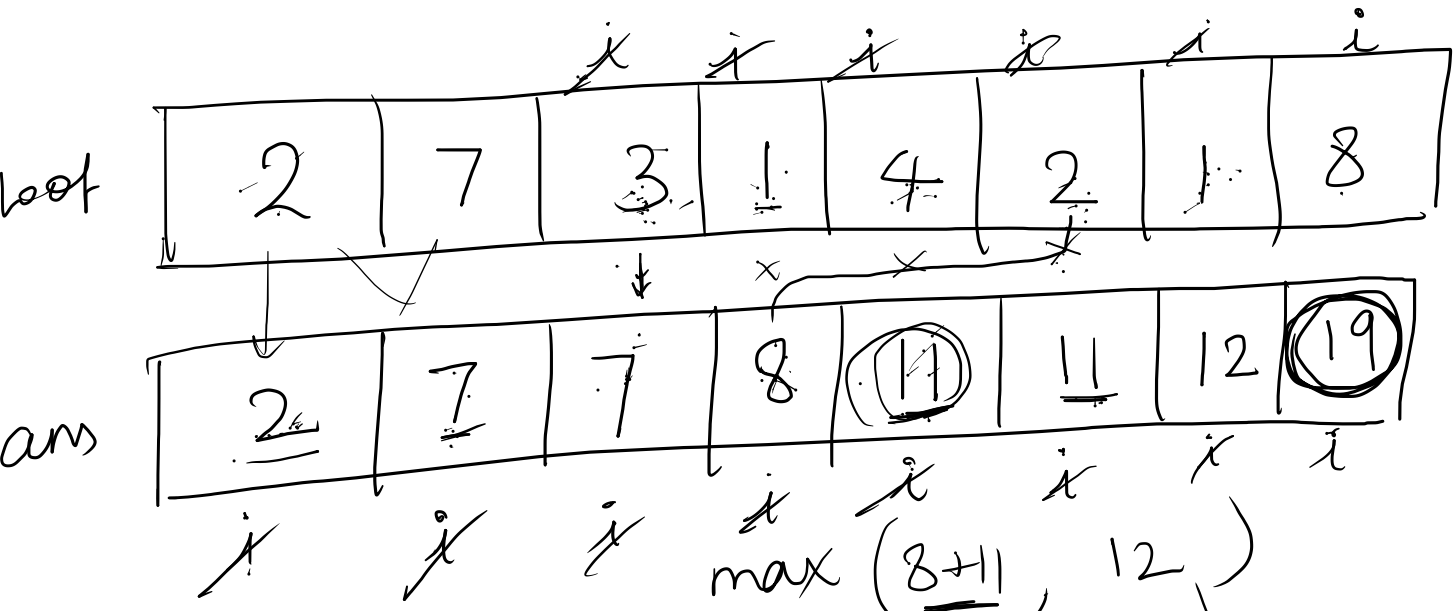
$$\text{final loot} = \max \left(\begin{array}{l} \text{loot}[n] + \text{final loot}[n-2], \\ \text{final loot}[n-1] \end{array} \right)$$

(till n)

If last
house is robbed

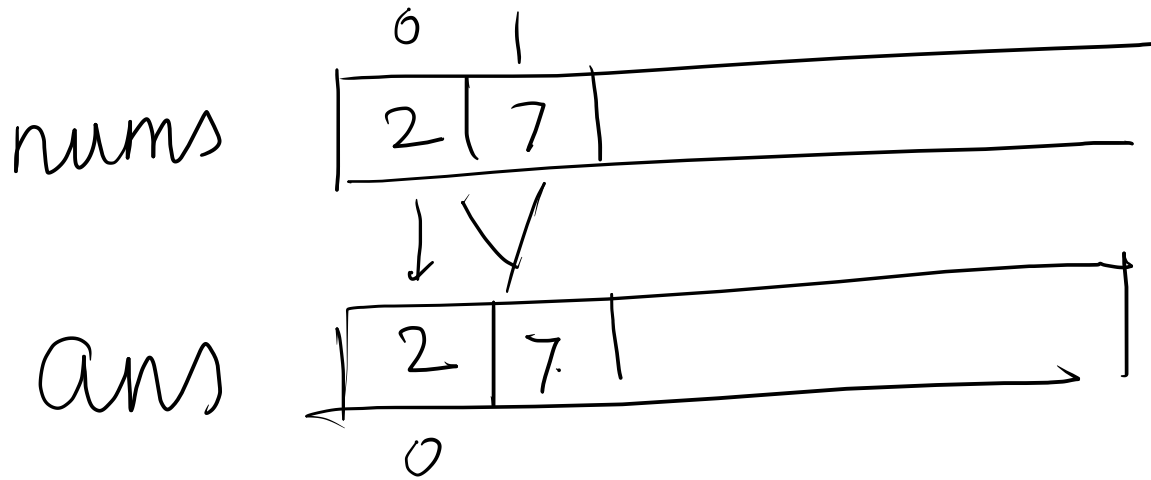
$$\text{finalloot}[n] = \text{loot}[n] + \text{finalloot}[n-2]$$

If last house
is not robbed = $\text{finalloot}[n-1]$



ans = 19

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



$$\text{ans}[0] = \text{nums}[0]$$

$$\text{ans}[1] = \max(\text{nums}[0], \text{nums}[1])$$

nums

0	1	2	
3	5	2	<u>8</u>

ans

<u>3</u>	<u>5</u>	<u>5</u>	13
0	1	2	

$$\max(8 + 5, 5)$$

$$\underline{T.C = O(n)}$$

$$\underline{S.C = O(n)}$$