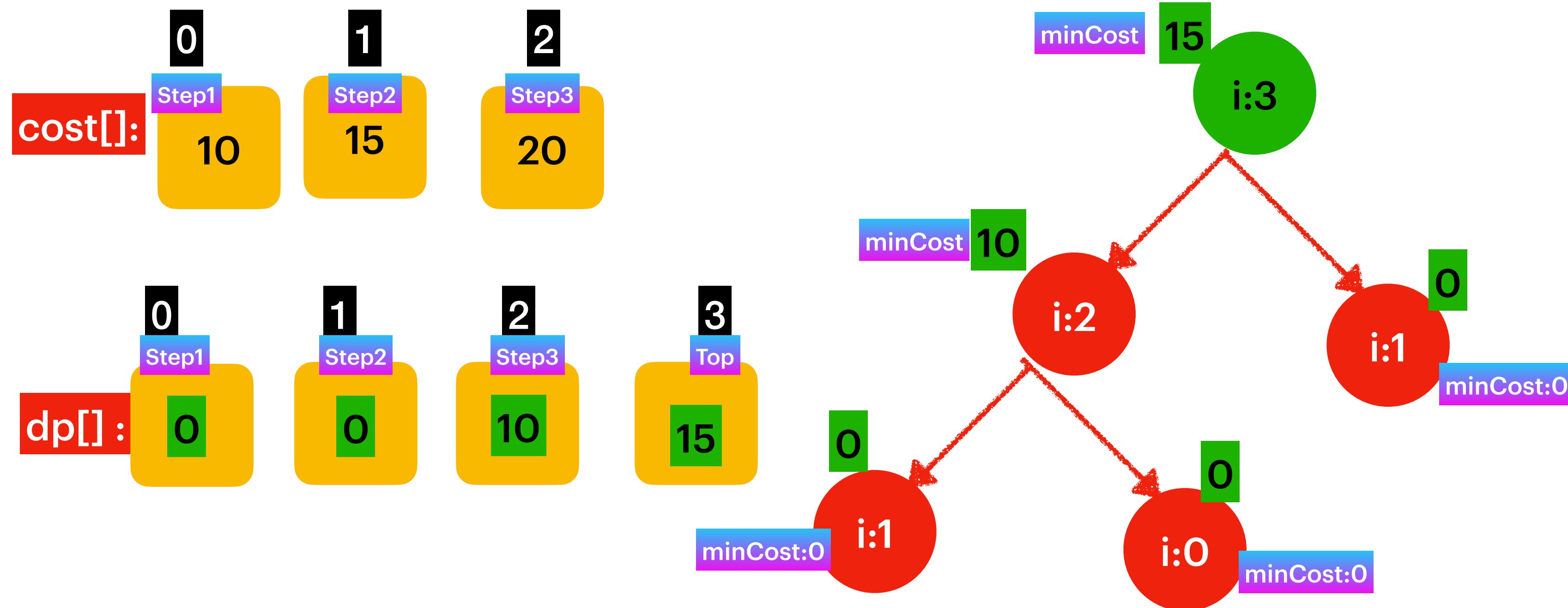


Memoization

Time Complexity : $O(n)$
Space Complexity : $O(n)$

```
Best Check :  
if( i <= 1)  
{  
    return 0;  
}
```



Recurrence Relation \rightarrow

We can reach i th step either from $i-1$ or $i-2$ steps.

$$dp[i] = \text{Math.min}(\text{cost}[i-1] + dp[i-1], \text{cost}[i-2] + dp[i-2])$$

198. House Robber

Medium

👍 11892

💬 255

❤ Add to List

🔗 Share

You are a professional robber planning to rob houses along a street. Each house has a certain amount of money stashed, the only constraint stopping you from robbing each of them is that adjacent houses have security systems connected and **it will automatically contact the police if two adjacent houses were broken into on the same night**.

Given an integer array `nums` representing the amount of money of each house, return *the maximum amount of money you can rob tonight **without alerting the police***.

Example 1:

Input: `nums = [1,2,3,1]`

Output: 4

Explanation: Rob house 1 (money = 1) and then rob house 3 (money = 3).

Total amount you can rob = 1 + 3 = 4.

Example 2:

Input: `nums = [2,7,9,3,1]`

Output: 12

Explanation: Rob house 1 (money = 2), rob house 3 (money = 9) and rob house 5 (money = 1).

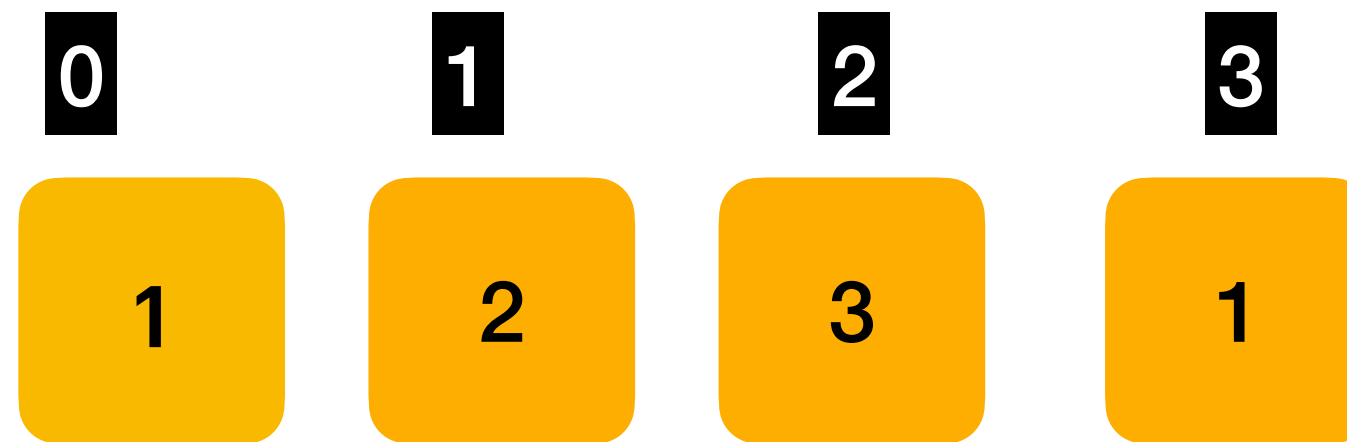
Total amount you can rob = 2 + 9 + 1 = 12.

Constraints:

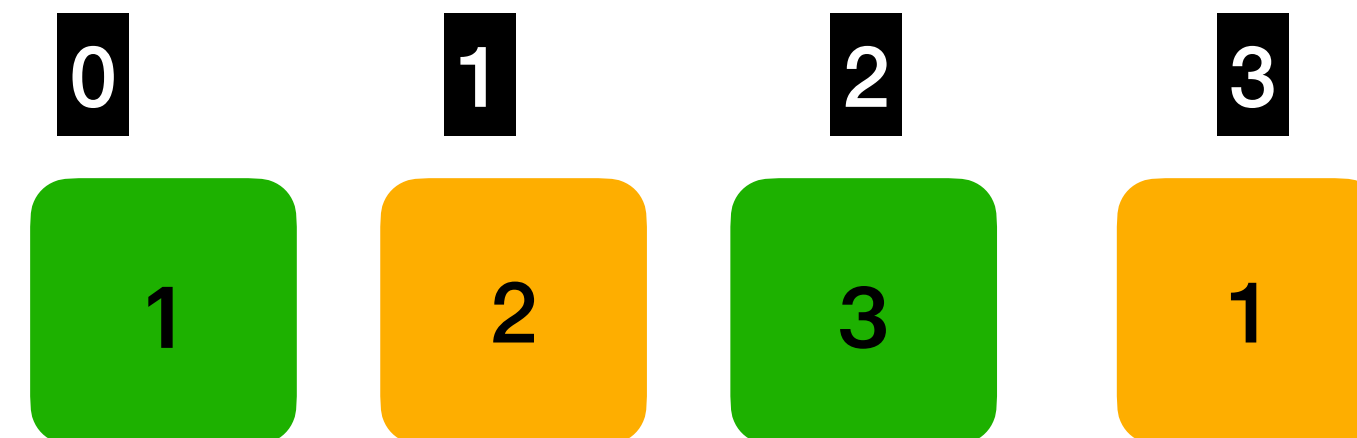
- `1 <= nums.length <= 100`
- `0 <= nums[i] <= 400`

Constraint : You can not rob from adjacent houses

Get the Max Profit without altering the police

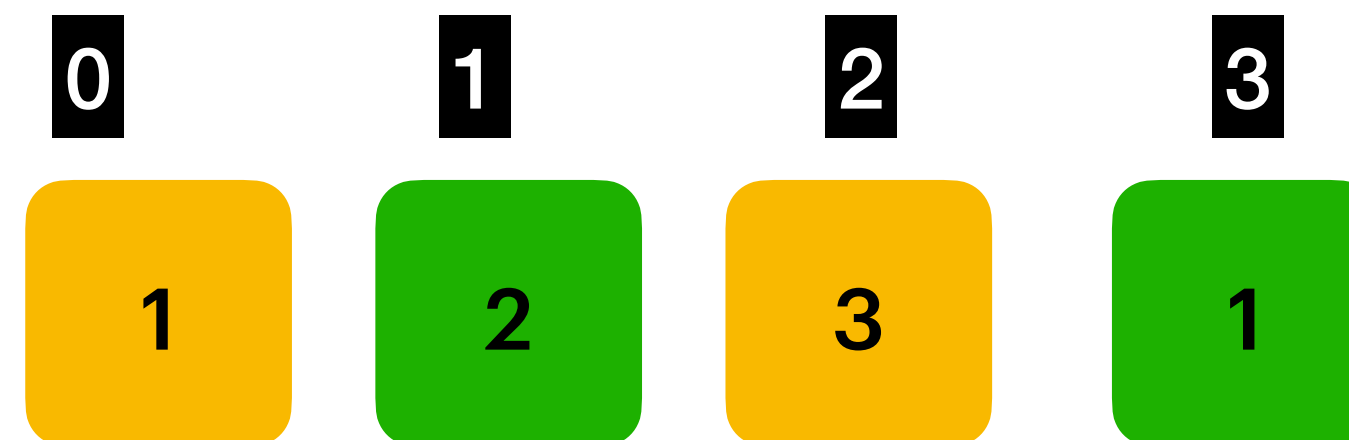


Case1



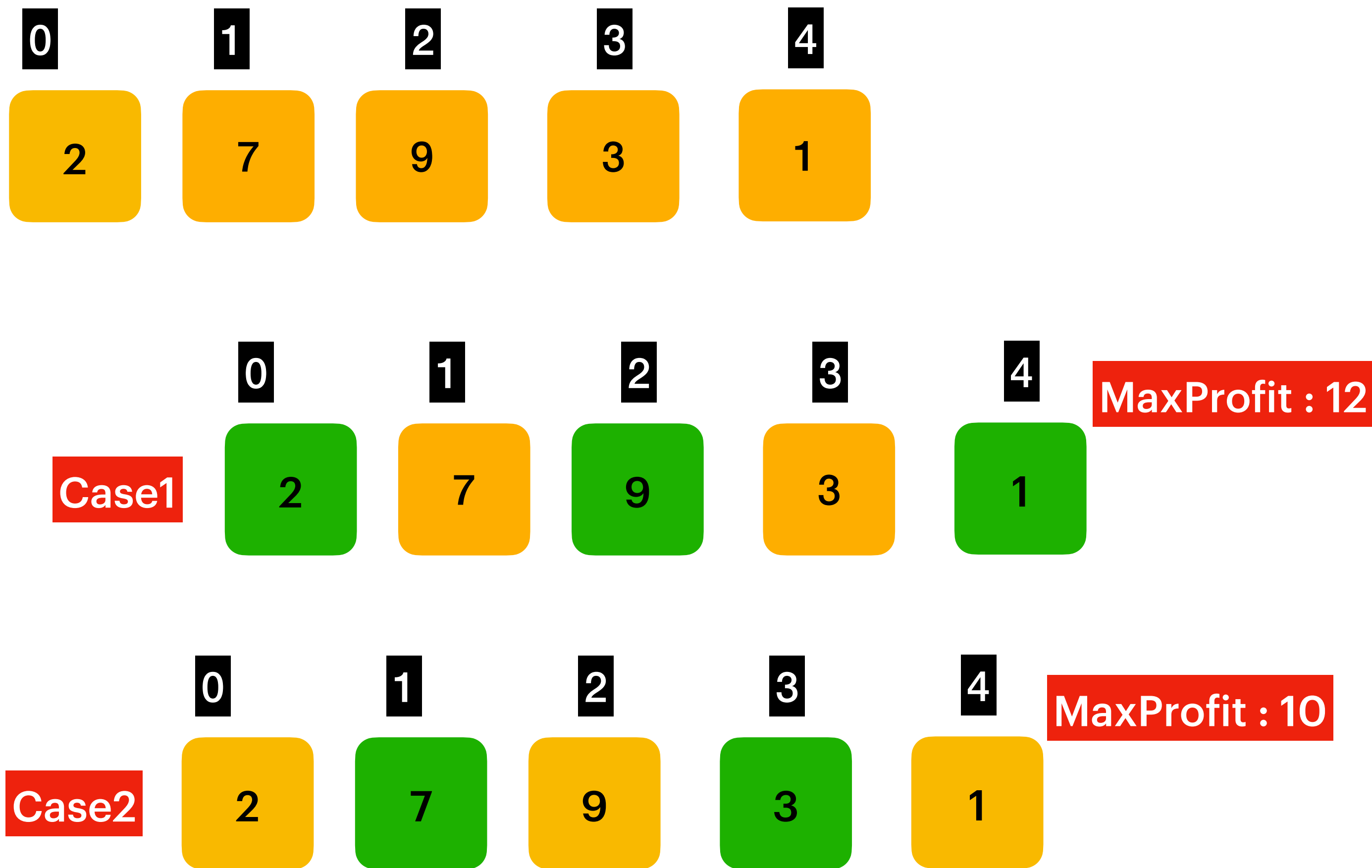
MaxProfit : 4

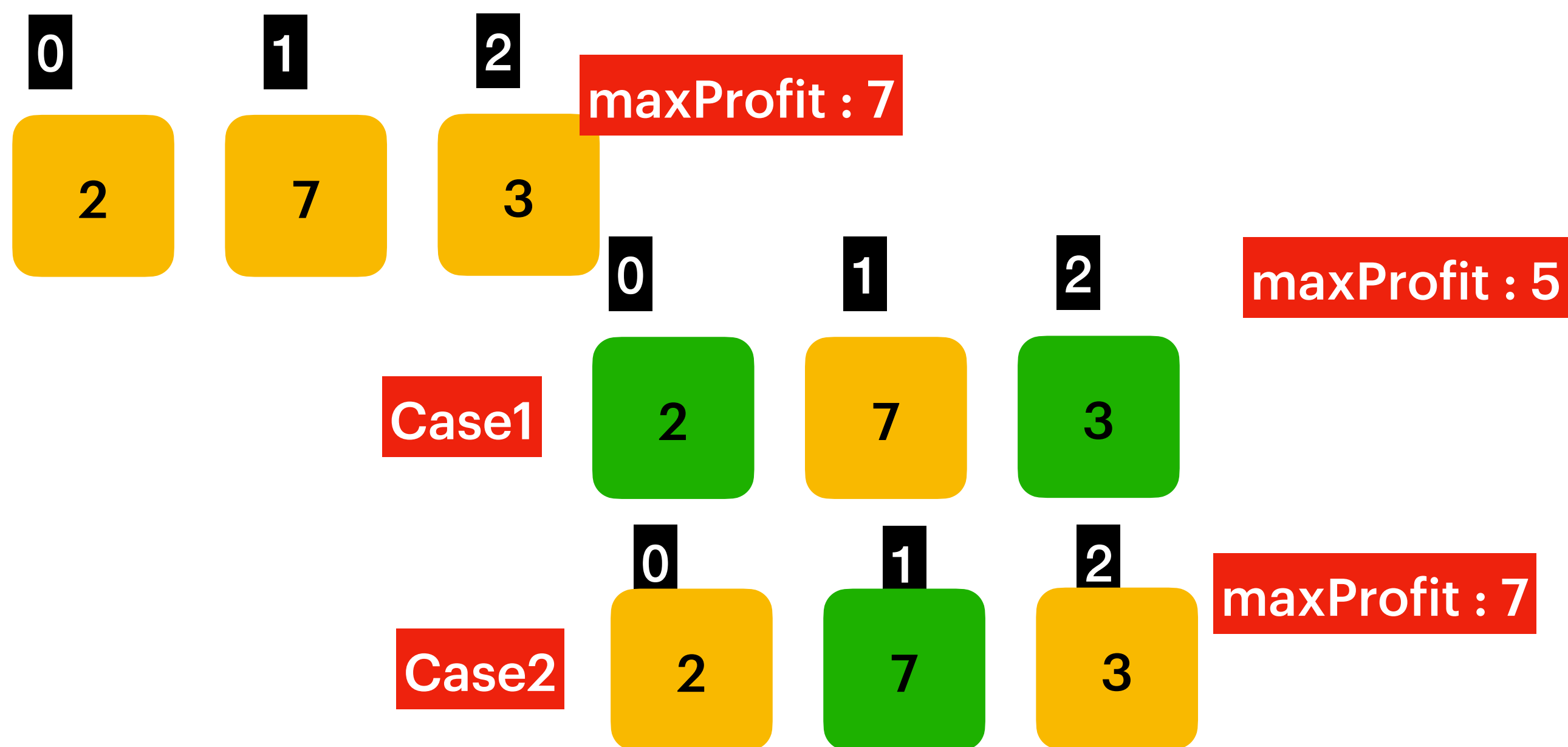
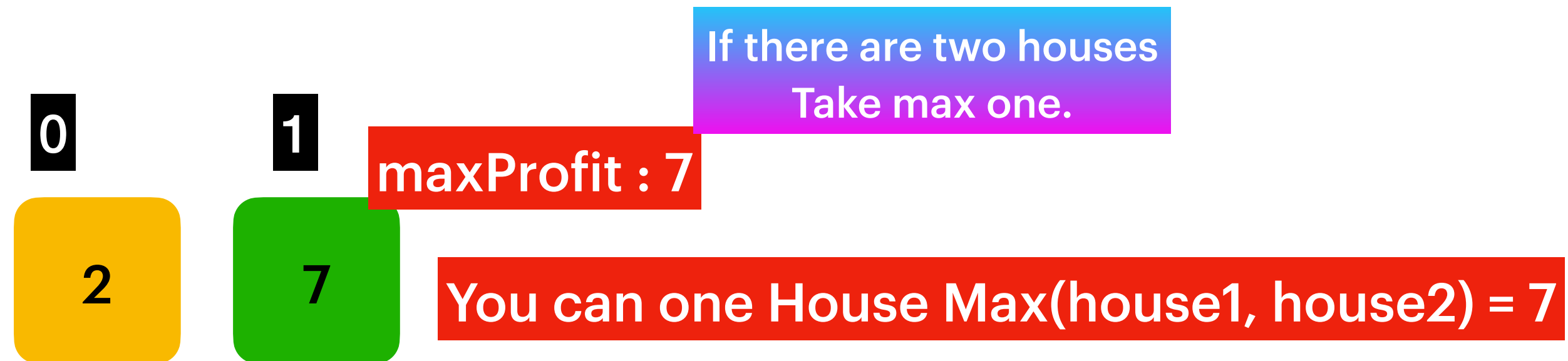
Case2



MaxProfit : 3

Should return Max Profit i.e 4





int[] dp = new int[n]

Used to store
subProblem results



If there is only one house
consider its profit.

SubProblem1 :



maxProfit : 2
dp[0] = 2

If there are two houses
Take max one.

SubProblem2 :



maxProfit : Max(2,7) = 7
dp[1] = 7

SubProblem3 :



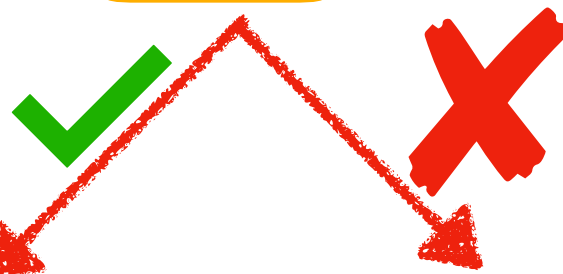
maxProfit : 11
dp[2] = 11

SubProblem4 :



maxProfit : 11
dp[3] = 11

SubProblem5 :



1+11 = 12

11

maxProfit : 12
dp[4] = 12

Tabulation

Time Complexity : O(n)
Space Complexity: O(n)

dp[i] = Math.max(nums[i] + dp[i-2] , dp[i-1])

Include



Exclude



nums[i] + dp[i-2] = 9+2 = 11

dp[i-1] = 7



3+7 = 10



11

Can we improve on Space for Tabulation Approach,
Yes we just need previous two subproblem results to solve current subproblem.
So we can swap between two variables. So that Space would be constant $O(1)$.

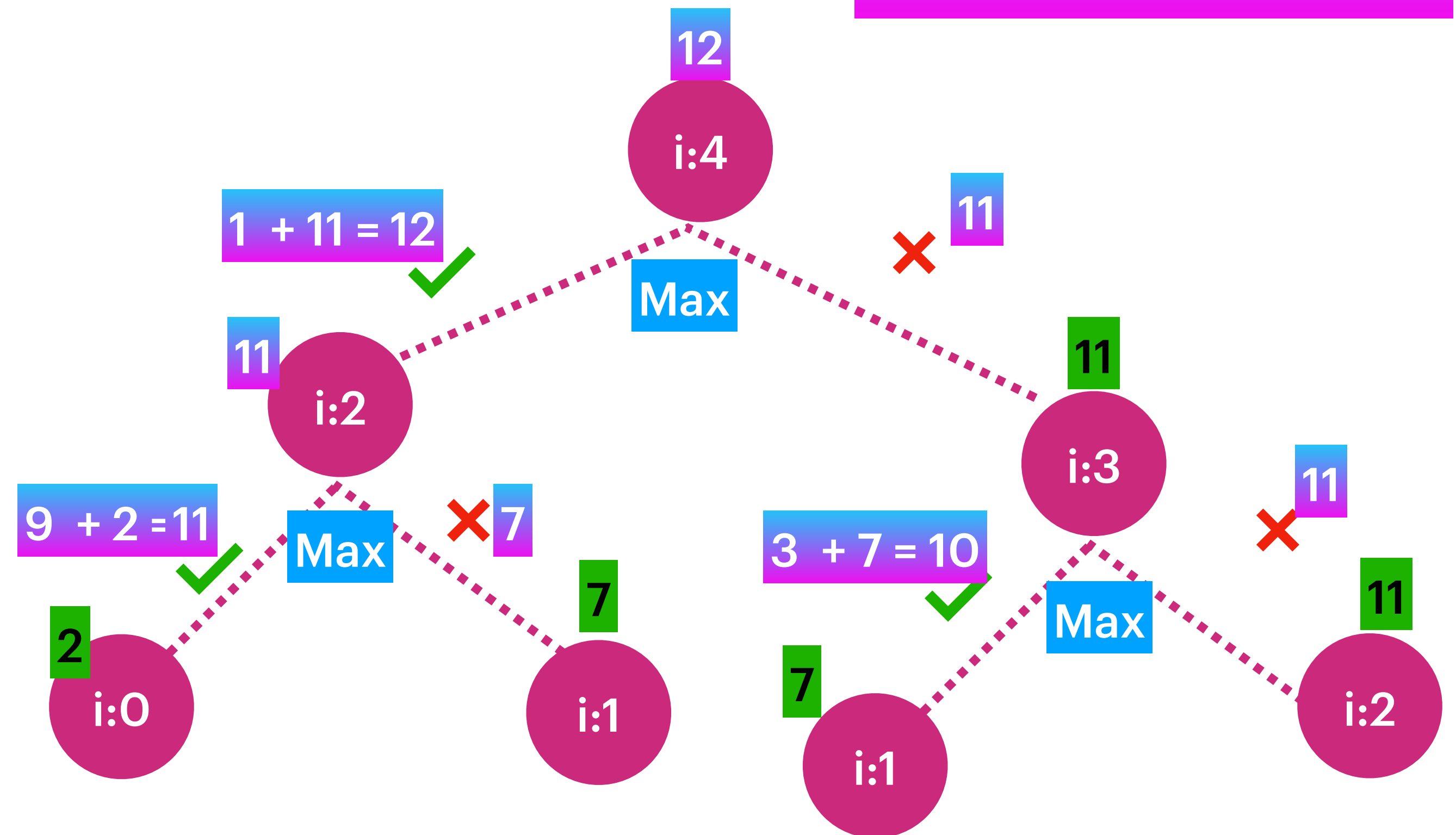
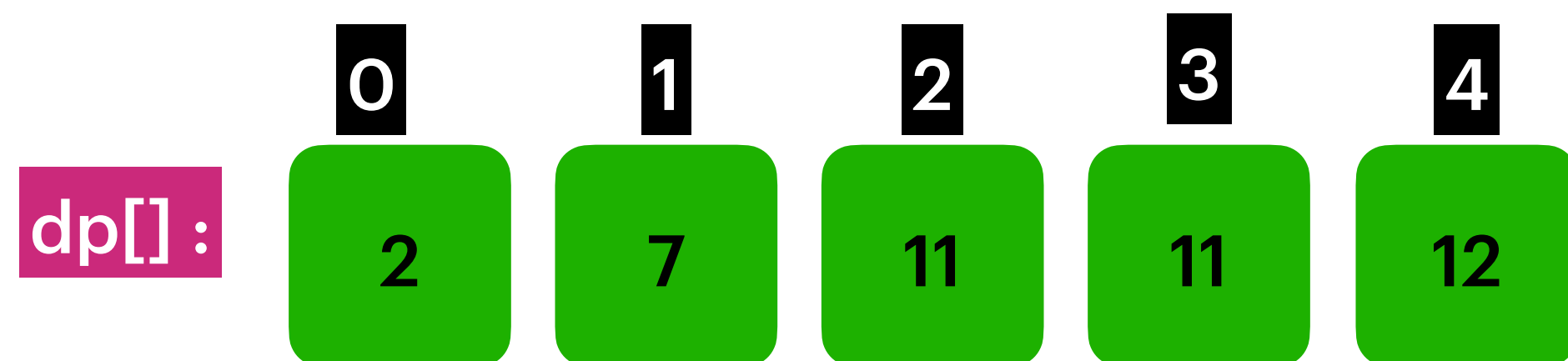
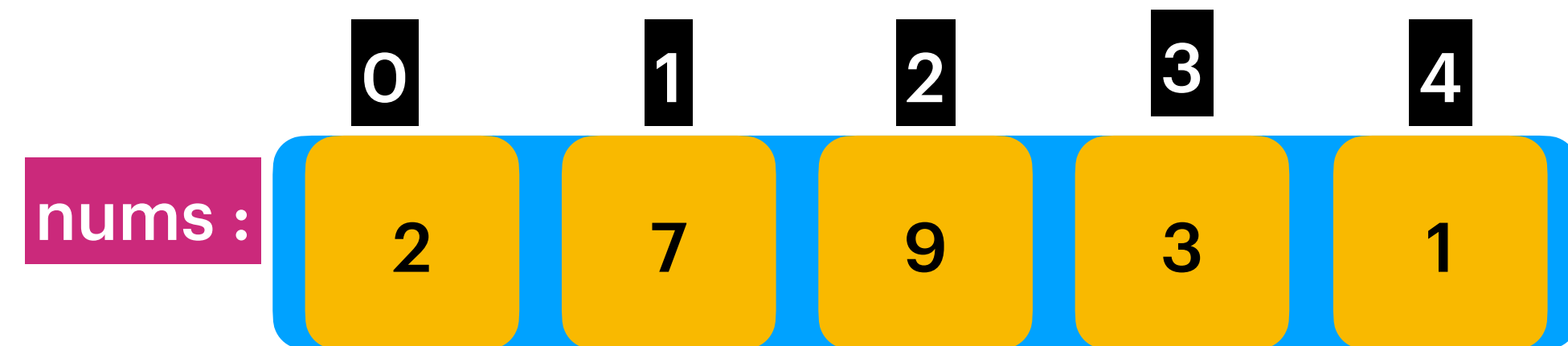
Memoization

Time Complexity : $O(n)$
Space Complexity: $O(n)$

Base checks :

```
if(currentHouseIndex == 0)
{
    return nums[0];
}

if(currentHouseIndex == 1)
{
    return Math.max(nums[0], nums[1]);
}
```



For Practice

213. House Robber II

Medium

👍 5290

💬 91

♡ Add to List

🔗 Share

You are a professional robber planning to rob houses along a street. Each house has a certain amount of money stashed. All houses at this place are **arranged in a circle**. That means the first house is the neighbor of the last one. Meanwhile, adjacent houses have a security system connected, and **it will automatically contact the police if two adjacent houses were broken into on the same night**.

Given an integer array `nums` representing the amount of money of each house, return *the maximum amount of money you can rob tonight **without alerting the police***.

Example 1:

Input: `nums = [2,3,2]`

Output: 3

Explanation: You cannot rob house 1 (money = 2) and then rob house 3 (money = 2), because they are adjacent houses.

Example 2:

Input: `nums = [1,2,3,1]`

Output: 4

Explanation: Rob house 1 (money = 1) and then rob house 3 (money = 3).
Total amount you can rob = 1 + 3 = 4.

Example 3:

Input: `nums = [1,2,3]`

Output: 3

Constraints:

- `1 <= nums.length <= 100`
- `0 <= nums[i] <= 1000`

