Q3: 198: House Robber

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.

Solution

1 Intuition

We can use dynamic programming to update the maximum sum of the amount of money since our task is to find the maximum total amount of money without alerting the police, which indicates that this process is accumulative.

2 Approach

Define a one-dimension array dp[i] as

$$dp[i] = \{\text{maximum amount of money from first } i \text{ house}\} \tag{1}$$

Hence, when the robber is facing the ith house, two strategies are available:

Case 1: The robber does not rob ith house, indicating that he has robbed the maximum amount of money of previous (i-1)th house. Thus, we can obtain:

$$dp[i] = dp[i-1] \tag{2}$$

Case 2: The robber robs ith house, indicating that the robber cannot rob (i-1)th house because they are adjacent. Thus, we can only consider the previous (i-2) th house.

Note that we don't consider the (i+1)th house since dynamic programming is developed from base case.

Hence, to get the maximum amount of money from first i house, we can obtain:

$$dp[i] = nums[i] + dp[i-2] \tag{3}$$

Combine two cases together so that we can obtain the final maximum amount of money, we have:

$$dp[i] = \max(dp[i-1], nums[i] + dp[i-2]) \tag{4}$$

3 Accepted Code

```cpp
class Solution {
public:
    int rob(vector<int>& nums) {
        int n = nums.size();
        if(n == 1) return nums[0];

        vector<int> dp(n,0);

        dp[0] = nums[0];
        dp[1] = max(nums[0], nums[1]);

        for(int i = 2; i < n; i++)
            dp[i] = max(dp[i - 1], nums[i] + dp[i - 2]);

        return dp[n - 1];
    }
};

```