## 198. House Robber

Medium 6352 184 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 system connected and **it will automatically contact the police if two adjacent houses were broken into on the same night**.

Given a list of non-negative integers representing the amount of money of each house, determine 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:**

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

Definition:
rob[i]: the maximum money you can rob tonight if you rob ith house.

Base case:
rob[0] = nums[0]
rob[1] = max(nums[0], nums[1])

Recurrence relation:
rob[i] = max(rob[i-1], rob[i-2] + nums[i])

Time complexity: O(n)
Space complexity: O(1)

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

        } else if (nums.size() == 2) {
            return max(nums[0], nums[1]);
        }


        // rob[0] = nums[0]
        int rob_i_prev = nums[0];
        // rob[1] = nums[1]
        int rob_i = max(nums[0], nums[1]);

        for (auto it = nums.begin() + 2; it != nums.end(); it++) {
            int rob_i_temp = rob_i;
            // rob[i] = max(rob[i-1], rob[i-2] + nums[i])
            rob_i = max(rob_i_temp, rob_i_prev + *it);
            // rob[i-2] = rob[i-1]
            rob_i_prev = rob_i_temp;
        }

        return rob_i;
    }
};
```