213. House Robber II

Solved

Medium | Topics | Companies | Hint

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.
```

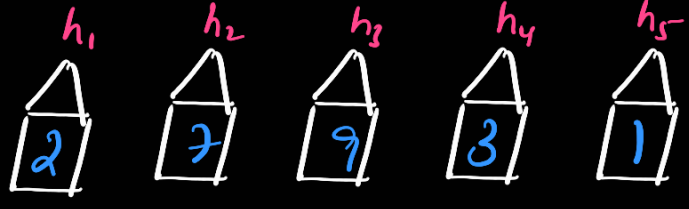
Example 3:

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

Constraints:

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

Same as House Robber, but here last and first house are also treated as adjacent



Observation:

its for sure that both h_1 and h_5 cannot be part of the solution at the same time.

So

The solution must be either

So we can use the same logic of house robber and implement.

```
class Solution {
public:
    int find(vector<int> &nums){
        int n = nums.size();
        int next2 = nums[n-1];
        int next1;

        if(n-2 >= 0)
            next1 = max(next2 , nums[n-2]);
        else
            return next2;

        for(int i = n-3;i>=0;i--){
            int curr = max(next1,nums[i] + next2);
            next2 = next1;
            next1 = curr;
        }

        return next1;
    }

    int rob(vector<int>& nums) {
        vector<int> temp1,temp2;

        if(nums.size() == 1)
            return nums[0];

        for(int i = 0;i<nums.size();i++)
        {
            if(i != 0)
                temp1.push_back(nums[i]);
            if(i != nums.size()-1)
                temp2.push_back(nums[i]);
        }

        return max(find(temp1),find(temp2));
    }
};
```

$h_{second} \rightarrow h_{last}$

$h_{first} \rightarrow h_{last-1}$

$T(n) : O(n) + O(n)$

$S(n) : O(n) + O(n)$

note: I directly made find function the most optimized one.

find() can be implemented in different ways as we did in house Robber.