1. 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 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.

**解1** dfs搜索。超时了。。。

class Solution {  
public:  
 int rob(vector<int>& nums) {  
 vector<bool>flag(nums.size(), false);  
 int money = 0, max\_money = 0;  
 dfs(nums, 0, flag, money, max\_money);  
 return max\_money;  
 }  
 void dfs(vector<int>& nums, int i,  
 vector<bool>& flag, int &money, int &max\_money){  
 if(i >= nums.size()){  
 if(money > max\_money)max\_money = money;  
 return;  
 }  
 if(i > 0){  
 if(flag[i-1] == false){  
 flag[i] = true;  
 money += nums[i];  
 dfs(nums, i+1, flag, money, max\_money);  
 flag[i] = false;  
 money -= nums[i];  
 }  
 dfs(nums, i+1, flag, money, max\_money);  
 }else{  
 flag[i] = true;  
 money += nums[i];  
 dfs(nums, i+1, flag, money, max\_money);  
 flag[i] = false;  
 money -= nums[i];  
 dfs(nums, i+1, flag, money, max\_money);  
 }  
 }  
};

**解2** 动态规划。dp[k]表示前k家能够抢到的最大金额，对于第k+1家：

* 抢：第k家就不能抢，因此dp[k+1] = dp[k-1] + A[k+1]
* 不抢：dp[k+1] = dp[k]

class Solution {  
public:  
 int rob(vector<int>& nums) {  
 if(nums.size() == 0)return 0;  
 if(nums.size() == 1)return nums[0];  
 int dp0 = nums[0], dp1 = max(nums[0], nums[1]);  
 for(int i = 2; i < nums.size(); ++i){  
 int tmp = max(dp1, dp0+nums[i]);  
 dp0 = dp1;  
 dp1 = tmp;  
 }  
 return dp1;  
 }  
};