动态规划： 213. 打家劫舍II

```cpp
class Solution {
public:
    int rob1(int start , int end,vector<int>& nums)
    {
        if(start > end) return 0;
        int n = nums.size();
        vector<int> f(n);
        vector<int> g(n);
        f[start] = nums[start];
        g[start] = 0;
        for(int i = start+1;i<= end;i++)
        {
            f[i] = g[i-1]+nums[i];
            g[i] = max(f[i-1],g[i-1]);
        }
        return max(f[end],g[end]);
    }
    int rob(vector<int>& nums) {
        int n = nums.size();

        return max(nums[0] + rob1(2,n-2,nums),rob1(1,n-1,nums));

    }
};
```

链表： BM11 链表相加(二)

```
class Solution {
    ListNode* ReverseList(ListNode* pHead) {
        if (pHead == NULL)
            return NULL;
        ListNode* cur = pHead;
        ListNode* pre = NULL;
        while (cur != NULL) {
            //断开链表，要记录后续一个
            ListNode* temp = cur->next;
            //当前的next指向前一个
            cur->next = pre;
            //前一个更新为当前
            pre = cur;
            //当前更新为刚刚记录的后一个
            cur = temp;
        }
        return pre;
    }
  public:
    /**
     * 代码中的类名、方法名、参数名已经指定，请勿修改，直接返回方法规定的值即可
     *
     *
     * @param head1 ListNode类
     * @param head2 ListNode类
     * @return ListNode类
     */
    ListNode* addInList(ListNode* head1, ListNode* head2) {
        if (head1 == nullptr) return head2;
        if (head2 == nullptr) return head1;

        head1 = ReverseList(head1);
        head2 = ReverseList(head2);

        ListNode* res = new ListNode(-1);
        ListNode* head = res;

        int carry = 0;
```

```
        while (head1 != NULL || head2 != NULL || carry != 0) {
            //链表不为空则取其值
            int val1 = head1 == NULL ? 0 : head1->val;
            int val2 = head2 == NULL ? 0 : head2->val;
            //相加
            int temp = val1 + val2 + carry;
            //获取进位
            carry = temp / 10;
            temp %= 10;
            //添加元素
            head->next = new ListNode(temp);
            head = head->next;
            //移动下一个
            if (head1 != NULL)
                head1 = head1->next;
            if (head2 != NULL)
                head2 = head2->next;
        }
        //结果反转回来
        return ReverseList(res->next);

    }
};
```