# 题目

给定一个由整数数组 A 表示的环形数组 C，求 C 的非空子数组的最大可能和。

在此处，环形数组意味着数组的末端将会与开头相连呈环状。（形式上，当0 <= i < A.length 时 C[i] = A[i]，且当 i >= 0 时 C[i+A.length] = C[i]）

此外，子数组最多只能包含固定缓冲区 A 中的每个元素一次。（形式上，对于子数组 C[i], C[i+1], ..., C[j]，不存在 i <= k1, k2 <= j 其中 k1 % A.length = k2 % A.length）

示例 1：

输入：[1,-2,3,-2]

输出：3

解释：从子数组 [3] 得到最大和 3

示例 2：

输入：[5,-3,5]

输出：10

解释：从子数组 [5,5] 得到最大和 5 + 5 = 10

示例 3：

输入：[3,-1,2,-1]

输出：4

解释：从子数组 [2,-1,3] 得到最大和 2 + (-1) + 3 = 4

示例 4：

输入：[3,-2,2,-3]

输出：3

解释：从子数组 [3] 和 [3,-2,2] 都可以得到最大和 3

示例 5：

输入：[-2,-3,-1]

输出：-1

解释：从子数组 [-1] 得到最大和 -1

提示：

-30000 <= A[i] <= 30000

1 <= A.length <= 30000

# 分析

## 方法一：动态规划

**思路：**

对于环形数组，分两种情况。

1、答案在数组中间，就是最大子序和。例如[1,-2,3,-2]；

2、答案在数组两边，例如[5,-3,5]最大的子序和就等于数组的总和SUM-最小的子序和。(一种特殊情况是数组全为负数，也就是SUM-最小子序和==0，最大子序和等于数组中最小的那个)。

**代码：**

class Solution {

public:

int maxSubarraySumCircular(vector<int>& nums) {

int sum=nums[0];

vector<int>dpmax(nums);

vector<int>dpmin(nums);

for(int i=1;i<nums.size();i++){

dpmax[i]=max(dpmax[i-1]+nums[i],nums[i]);

dpmin[i]=min(dpmin[i-1]+nums[i],nums[i]);

sum+=nums[i];

}

int maxv=\*max\_element(dpmax.begin(),dpmax.end());

int minv=\*min\_element(dpmin.begin(),dpmin.end());

return max(maxv,sum-minv==0?maxv:sum-minv);

}

};

**复杂度：**