# 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

类似题目：

53. 最大子数组和的动态规划解法

# 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：**

## 方法二：单调队列

## 方法三：数学分析

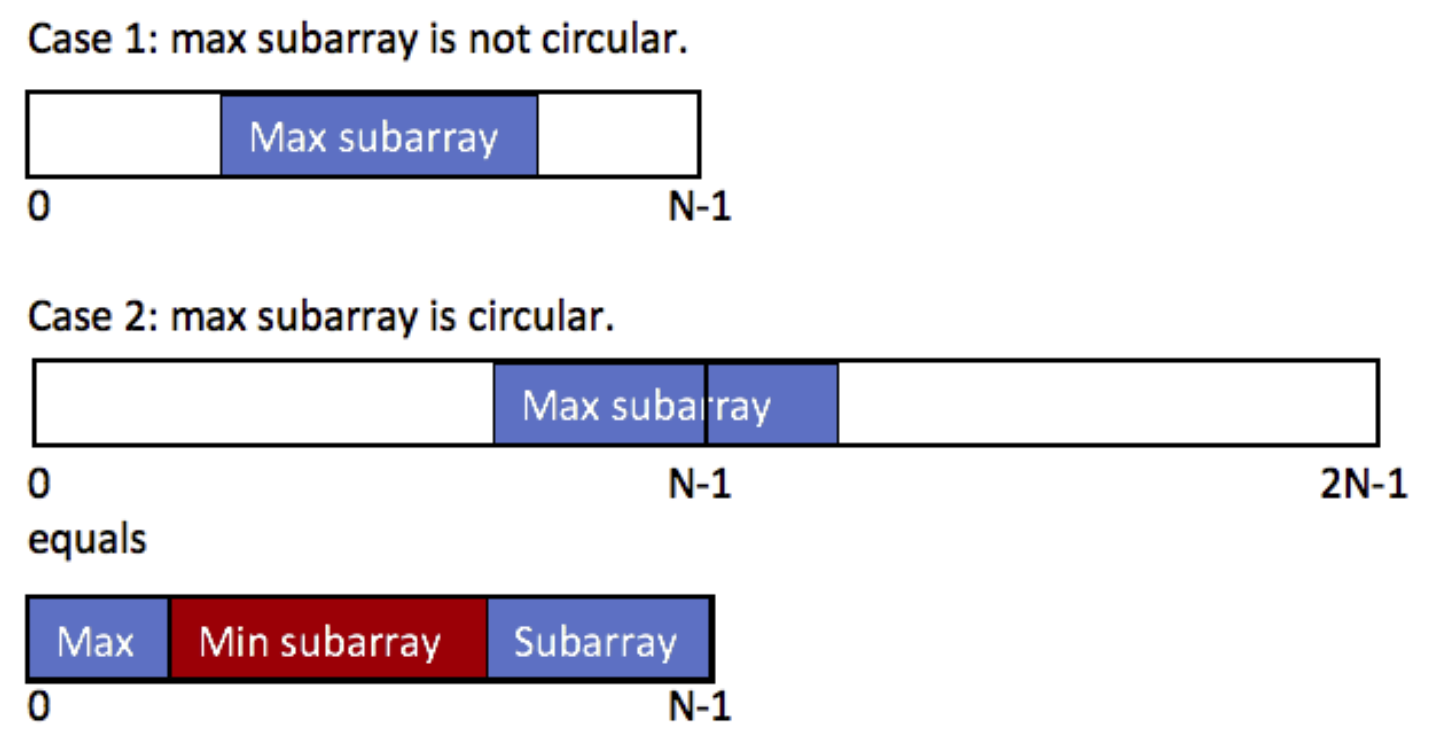
这题一共有两种情况（也就是相当于比53题多了一种最大子数组和是首尾连接的情况）：

下面的这个子数组指最大和的子数组

第一种情况：这个子数组不是环状的，就是说首尾不相连。

第二种情况：这个子数组一部分在首部，一部分在尾部，我们可以将这第二种情况转换成第一种情况

如下图：



所以这最大的环形子数组和 = max(最大子数组和，数组总和-最小子数组和)

证明：

证明一下第二种情况（最大子数组是环形的）

max(前缀数组+后缀数组)

= max(数组总和 - subarray) subarray指的是前缀数组和后缀数组中间的数组

= 数组总和 + max(-subarray) 数组总和是不变的，直接提出来

= 数组总和 - min(subarry) 这个都懂吧，把负号提出来，max变成min

极端情况：如果说这数组的所有数都是负数，那么上面的公式还需要变一下，因为这种情况，对于上面的第一种情况sum会等于数组中的最大值，而对二种情况sum=0（最小的子数组就是本数组，total-total=0）。所以多加一个case，判断最大子数组和是否小于0，小于0，直接返回该maxSubArray

代码：

class Solution {

public:

int maxSubarraySumCircular(vector<int>& A) {

int total = 0; // 数组的总和

int maxSum = A[0]; // 最大子数组和的初始值为第一个元素

int curMax = 0; // 当前子数组和的最大值

int minSum = A[0]; // 最小子数组和的初始值为第一个元素

int curMin = 0; // 当前子数组和的最小值

// 遍历数组中的每一个元素

for (int& a : A) {

curMax = max(curMax + a, a); // 更新当前子数组和的最大值

maxSum = max(maxSum, curMax); // 更新最大子数组和的值

curMin = min(curMin + a, a); // 更新当前子数组和的最小值

minSum = min(minSum, curMin); // 更新最小子数组和的值

total += a; // 计算数组的总和

}

// 如果最大子数组和大于零，则返回最大子数组和和 (总和 - 最小子数组和) 中的较大值；否则，返回最大子数组和本身

return maxSum > 0 ? max(maxSum, total - minSum) : maxSum;

}

};