Problem 918: Maximum Sum Circular Subarray

Problem Information

Difficulty: Medium
Acceptance Rate: 48.70%
Paid Only: No
Tags: Array, Divide and Conquer, Dynamic Programming, Queue, Monotonic Queue

Problem Description

Given a **circular integer array** \`nums\` of length \`n\`, return _the maximum possible sum of a non-empty**subarray** of _\`nums\`.

A **circular array** means the end of the array connects to the beginning of the array. Formally, the next element of \`nums[i]\` is \`nums[(i + 1) % n]\` and the previous element of \`nums[i]\` is \`nums[(i - 1 + n) % n]\`.

A **subarray** may only include each element of the fixed buffer \`nums\` at most once. Formally, for a subarray \`nums[i], nums[i + 1], ..., nums[j]\`, there does not exist \`i <= k1\`, \`k2 <= j\` with \`k1 % n == k2 % n\`.

Example 1:

Input: nums = [1,-2,3,-2] **Output:** 3 **Explanation:** Subarray [3] has maximum sum 3.

Example 2:

Input: nums = [5,-3,5] **Output:** 10 **Explanation:** Subarray [5,5] has maximum sum 5 + 5 = 10.

Example 3:

Input: nums = [-3,-2,-3] **Output:** -2 **Explanation:** Subarray [-2] has maximum sum -2.

Constraints:

* `n == nums.length` * `1 <= n <= 3 * 104` * `-3 * 104 <= nums[i] <= 3 * 104`

Code Snippets

C++:

```
class Solution {
public:
int maxSubarraySumCircular(vector<int>& nums) {

}
};
```

Java:

```
class Solution {
public int maxSubarraySumCircular(int[] nums) {

}
}
```

Python3:

```
class Solution:
def maxSubarraySumCircular(self, nums: List[int]) -> int:
```