918. Maximum Sum Circular Subarray

Problem Statement

The task is to find the maximum sum of a non-empty subarray within a given circular integer array, nums. A circular array means that the array is conceptually connected end-to-end, so the elements wrap around.

Intuition

To solve the maximum subarray problem for a circular array, we must consider two cases:

1. The maximum sum subarray is similar to the one found in a non-circular array (Kadane's algorithm is useful here).
2. The maximum sum subarray is the result of wrapping around the circular array.

For the first case, we use Kadane's algorithm to find the maximum sum subarray that does not wrap around.

For the second case, to handle subarrays that wrap, consider that the answer could be the total sum of the array minus the minimum sum subarray. This is akin to "selecting" the rest of the array that doesn't include the minimum sum subarray. To find the minimum sum subarray, we modify Kadane's algorithm. At the end, the maximum possible sum for the case when the subarray wraps around is computed as the total sum of the array minus the minimum sum subarray (sum(nums) - s2).

The edge case to consider is when all numbers in the array are negative. In this situation, Kadane's algorithm yields a subarray sum which is the maximum negative number (i.e., the least negative), and thus is the maximum sum we can achieve without wrapping around. Subtracting any subarray would result in a smaller sum, so we return the maximum subarray sum found without wrapping in this case.

Putting it all together, the final answer is the larger of the two possibilities: the maximum subarray sum found without wrapping (using Kadane's algorithm) or the total array sum minus the minimum subarray sum, unless all numbers are negative, in which case we return the maximum subarray sum without wrapping.

Implementation

A screenshot of a computer program

Description automatically generated