










# Max Circular Subarray Sum

 solved by	Senan
 Platform	GeeksForGeeks
 difficulty	Hard
 tags	Kadane
 language	C++
 solved on	@26/11/2024
 link	<a href="https://www.geeksforgeeks.org/problems/max-circular-subarray-sum-1587115620/1">https://www.geeksforgeeks.org/problems/max-circular-subarray-sum-1587115620/1</a>
 Completion	

## Intuition

The problem is to find the maximum sum of a circular subarray. This requires handling both normal subarrays and circular ones. For a circular subarray, the maximum sum can be calculated by excluding the minimum subarray sum from the total array sum. Special care is needed if all elements are negative, as the circular sum formula won't apply.

## Approach

- Normal Subarray Maximum:** Calculate the maximum sum of a normal subarray using Kadane's algorithm. This handles the case of non-circular subarrays.
- Normal Subarray Minimum:** Similarly, calculate the minimum sum of a normal subarray. This helps determine the contribution of the wrapped part of a circular subarray.
- Total Array Sum:** Calculate the sum of all elements in the array. This is needed to compute the circular subarray sum.
- Special Case:** If all elements are negative, the normal subarray maximum is the answer, since wrapping won't help.
- Circular Subarray Maximum:** Compute the circular subarray maximum by subtracting the minimum subarray sum from the total array sum.
- Final Answer:** Return the greater value between the normal subarray maximum and the circular subarray maximum.

## Complexity

### Time Complexity:

- $O(N)$ :** Two passes through the array are sufficient to compute both the maximum and minimum subarray sums, as well as the total sum.

### Space Complexity:

- $O(1)$ :** No additional space is used other than a few variables.

## Code

```

class Solution {
public:
    int circularSubarraySum(vector<int> &arr) {
        int n = arr.size();
        int maxSum = INT_MIN, minSum = INT_MAX;
        int sum;

        sum = 0;
        int totalSum = 0;
        for (int i = 0; i < n; i++) {
            sum += arr[i];
            totalSum += arr[i];
            maxSum = max(maxSum, sum);
            if (sum < 0) sum = 0;
        }

        sum = 0;
        for (int i = 0; i < n; i++) {
            sum += arr[i];
            minSum = min(minSum, sum);
            if (sum > 0) sum = 0;
        }

        if (maxSum < 0) return maxSum;
        return max(maxSum, totalSum - minSum);
    }
};

```