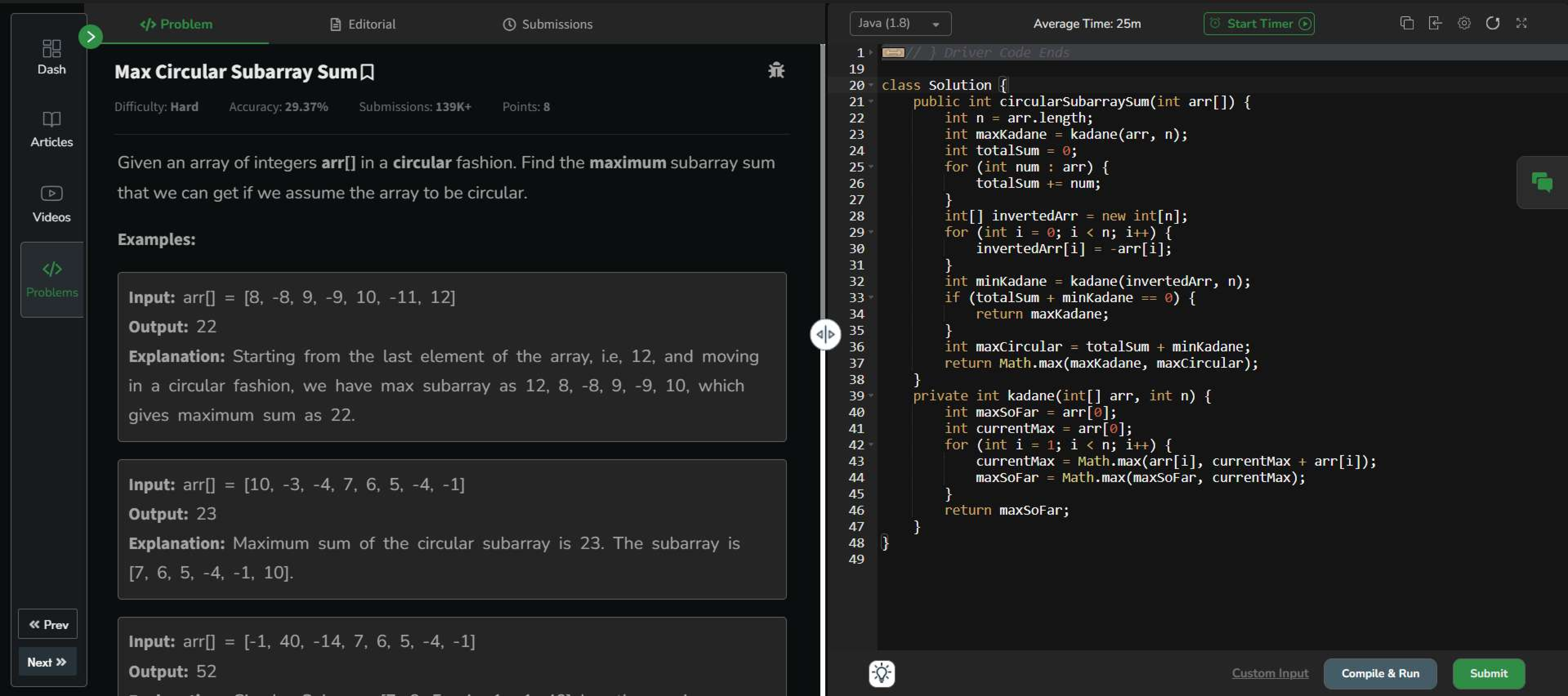

Max Circular Subarray Sum

Difficulty: **Hard** Accuracy: **29.37%** Submissions: **139K+** Points: **8**

Given an array of integers **arr[]** in a **circular** fashion. Find the **maximum** subarray sum that we can get if we assume the array to be circular.

Examples:

Input: arr[] = [8, -8, 9, -9, 10, -11, 12]
Output: 22
Explanation: Starting from the last element of the array, i.e, 12, and moving in a circular fashion, we have max subarray as 12, 8, -8, 9, -9, 10, which gives maximum sum as 22.

Input: arr[] = [10, -3, -4, 7, 6, 5, -4, -1]
Output: 23
Explanation: Maximum sum of the circular subarray is 23. The subarray is [7, 6, 5, -4, -1, 10].

Input: arr[] = [-1, 40, -14, 7, 6, 5, -4, -1]
Output: 52

```java
// } Driver Code Ends

class Solution {
    public int circularSubarraySum(int arr[]) {
        int n = arr.length;
        int maxKadane = kadane(arr, n);
        int totalSum = 0;
        for (int num : arr) {
            totalSum += num;
        }
        int[] invertedArr = new int[n];
        for (int i = 0; i < n; i++) {
            invertedArr[i] = -arr[i];
        }
        int minKadane = kadane(invertedArr, n);
        if (totalSum + minKadane == 0) {
            return maxKadane;
        }
        int maxCircular = totalSum + minKadane;
        return Math.max(maxKadane, maxCircular);
    }
    private int kadane(int[] arr, int n) {
        int maxSoFar = arr[0];
        int currentMax = arr[0];
        for (int i = 1; i < n; i++) {
            currentMax = Math.max(arr[i], currentMax + arr[i]);
            maxSoFar = Math.max(maxSoFar, currentMax);
        }
        return maxSoFar;
    }
}
```