# Kadane's Algorithm

**Problem Statement**

Given an array of N integers a1,a2,a3,....., aN find the maximum subarray(non-empty) sum of the given array.

**NOTE:** An array B is a subarray of an array A if B can be obtained from A by deleting several (possibly, zero, or all) elements from the beginning and several (possibly, zero or all) elements from the end. In particular, an array is a subarray of itself.

**For example:**

**Array A[]** = [-1, 2, -2, 5, 7, -3, 1]

**Maximum subarray sum** -> 12

Subarray(0-Based indexed) from index 1 to 4 -> [2, -2, 5, 7] and subarray(0-Based indexed) from index 3 to 4 -> [5, 7] have sum 12.

**Kadane's Algorithm**

The idea of Kadane's algorithm is to maintain a maximum subarray sum ending at every index 'i' of the given array and update the maximum sum obtained by comparing it with the maximum sum of the subarray ending at every index 'i'.

At any given index 'i' of the array, we can either:

- Append the element at index 'i' to the maximum sum subarray(so just add the element at index 'i' to the maximum you've found so far).
- Start a new subarray starting from index 'i'.

Appending an element at index 'i' to the maximum sum subarray obtained so far is beneficial if the sum till index 'i-1' is non-negative, otherwise it is better to start a new subarray starting from index 'i' and update the maximum sum obtained accordingly.

**For Example:** Consider the given array A[] = [1, -2, -3, 4, -1, 2, 1]. Element denotes the current element at index ‘i’, MaxSum is the maximum sum obtained so far till index ‘i’, Sum denotes the current sum obtained.

Initialize Sum to 0, MaxSum to INT_MIN
for i = 0
A[i] = 1,
Sum = Sum + A[i] = 1
MaxSum = max(MaxSum,Sum) = 1

for i = 1
A[i] = -2
Sum = Sum + A[i] = -1
MaxSum = max(MaxSum,Sum) = 1

Since Sum is negative, there is no point in appending the current sum obtained to the next element, so Sum = 0 i.e It is better to start a new subarray from the next index.

for i = 2
A[i] = -3
Sum = Sum + A[i] = -3
MaxSum = max(MaxSum,Sum) = 1

Again, since Sum is negative, there is no point in appending the current sum obtained to the next element, so Sum = 0 i.e It is better to start a new subarray from the next index.

for i = 3
A[i] = 4
Sum = Sum + A[i] = 4
MaxSum = max(MaxSum,Sum) = 4

for i = 4
A[i] = -1
Sum = Sum + A[i] = 3
MaxSum = max(MaxSum,Sum) = 4

Even if the element at A[i] is negative, the overall current sum is non-negative, so we retain the current sum to look for possible better options on appending the next elements. We have already updated the MaxSum for the subarray ending at index 3.

for i = 5

A[i] = 2
Sum = Sum + A[i] = 5
MaxSum = max(MaxSum,Sum) = 5

for i = 6
A[i] = 1
Sum = Sum + A[i] = 6
MaxSum = max(MaxSum,Sum) = 6

**Pseudocode:**

```
function Kadane(arr, N)

    //      Initializing curSum to 0 and maxSum to min value, denoting an empty
subarray
    curSum = 0
    maxSum = INT_MIN

    for idx = 0 to N-1
        curSum = curSum + arr[idx]
        //      Taking the max of maxSum and the curSum of the subarray
        maxSum = max(maxSum,curSum)

        //      Checking if the curSum becomes negative
        if curSum < 0
            curSum = 0

    return maxSum
```

**Time complexity: O(N)**, where N is the number of elements in the array, as we traverse the array once to get the maximum subarray sum.

**Space complexity: O(1)**, as no extra space is required.