### Kadane's Algorithm

**Company Tags**

[**Zoho**](https://practice.geeksforgeeks.org/explore/?company%5b%5d=Zoho) [**Flipkart**](https://practice.geeksforgeeks.org/explore/?company%5b%5d=Flipkart) [**Morgan Stanley**](https://practice.geeksforgeeks.org/explore/?company%5b%5d=Morgan%20Stanley) [**Accolite**](https://practice.geeksforgeeks.org/explore/?company%5b%5d=Accolite) [**Amazon**](https://practice.geeksforgeeks.org/explore/?company%5b%5d=Amazon) [**Microsoft**](https://practice.geeksforgeeks.org/explore/?company%5b%5d=Microsoft) [**Samsung**](https://practice.geeksforgeeks.org/explore/?company%5b%5d=Samsung)[**Snapdeal**](https://practice.geeksforgeeks.org/explore/?company%5b%5d=Snapdeal) [**24\*7 Innovation Labs**](https://practice.geeksforgeeks.org/explore/?company%5b%5d=24*7%20Innovation%20Labs) [**Citrix**](https://practice.geeksforgeeks.org/explore/?company%5b%5d=Citrix) [**D-E-Shaw**](https://practice.geeksforgeeks.org/explore/?company%5b%5d=D-E-Shaw) [**FactSet**](https://practice.geeksforgeeks.org/explore/?company%5b%5d=FactSet) [**Hike**](https://practice.geeksforgeeks.org/explore/?company%5b%5d=Hike) [**Housing.com**](https://practice.geeksforgeeks.org/explore/?company%5b%5d=Housing.com) [**MetLife**](https://practice.geeksforgeeks.org/explore/?company%5b%5d=MetLife) [**Ola Cabs**](https://practice.geeksforgeeks.org/explore/?company%5b%5d=Ola%20Cabs) [**Oracle**](https://practice.geeksforgeeks.org/explore/?company%5b%5d=Oracle)[**Payu**](https://practice.geeksforgeeks.org/explore/?company%5b%5d=Payu)[**Teradata**](https://practice.geeksforgeeks.org/explore/?company%5b%5d=Teradata) [**Visa**](https://practice.geeksforgeeks.org/explore/?company%5b%5d=Visa) [**Walmart**](https://practice.geeksforgeeks.org/explore/?company%5b%5d=Walmart) [**Adobe**](https://practice.geeksforgeeks.org/explore/?company%5b%5d=Adobe) [**Google**](https://practice.geeksforgeeks.org/explore/?company%5b%5d=Google) [**Arcesium**](https://practice.geeksforgeeks.org/explore/?company%5b%5d=Arcesium)

# Statement

Given an array **Arr[]**of **N** integers. Find the contiguous sub-array(containing at least one number) which has the maximum sum and return its sum.

**Example 1:**

**Input:**

N = 5

Arr[] = {1,2,3,-2,5}

**Output:**

9

**Explanation:**

Max subarray sum is 9

of elements (1, 2, 3, -2, 5) which

is a contiguous subarray.

**Example 2:**

**Input:**

N = 4

Arr[] = {-1,-2,-3,-4}

**Output:**

-1

**Explanation:**

Max subarray sum is -1

of element (-1)

### JAVA CODE

### import java.io.\*;

### import java.util.\*;

### class Main {

### 

### public static void main (String[] args) throws IOException {

### BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

### int t = Integer.parseInt(br.readLine().trim()); //Inputting the testcases

### while(t-->0){

### //size of array

### int n = Integer.parseInt(br.readLine().trim());

### int arr[] = new int[n];

### String inputLine[] = br.readLine().trim().split(" ");

### 

### //adding elements

### for(int i=0; i<n; i++){

### arr[i] = Integer.parseInt(inputLine[i]);

### }

### 

### Solution obj = new Solution();

### 

### //calling maxSubarraySum() function

### System.out.println(obj.maxSubarraySum(arr, n));

### }

### }

### }

### class Solution{

### long maxSubarraySum(int nums[], int n){

### 

### int maxSum = nums[0];

### int currSum = nums[0];

### for (int i = 1; i < n; i++) {

### currSum = Math.max(nums[i], currSum + nums[i]);

### maxSum = Math.max(maxSum, currSum);

### }

### return maxSum;

### }

### 

### 

### }