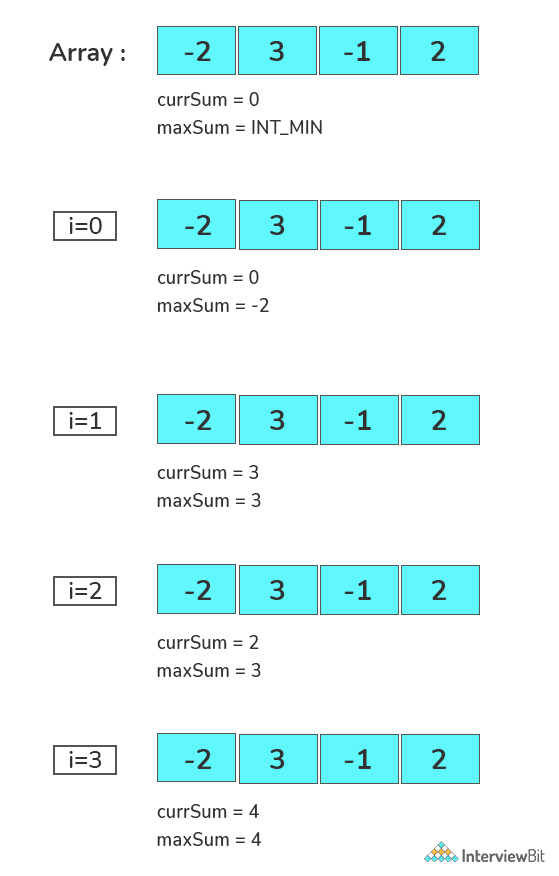
**Kadane’s Algorithm** is an iterative [**dynamic programming**](https://www.interviewbit.com/courses/programming/dynamic-programming/) algorithm. It calculates the maximum sum subarray ending at a particular position by using the maximum sum subarray ending at the previous position. Follow the below steps to solve the problem.

* Define two-variable currSum which stores maximum sum ending here and maxSum which stores maximum sum so far.
* Initialize currSum with 0 and maxSum with INT\_MIN.
* Now, iterate over the array and add the value of the current element to currSum and check
  + If currSum is greater than maxSum, update maxSum equals to currSum.
  + If currSum is less than zero, make currSum equal to zero.
* Finally, print the value of maxSum.

**Dry run of the above approach**



import java.util.\*;

import java.lang.\*;

import java.io.\*;

class Main {

public static int maximumSubarraySum(int[] arr) {

int n = arr.length;

int maxSum = Integer.MIN\_VALUE;

int currSum = 0;

for (int i = 0; i <= n - 1; i++) {

currSum += arr[i];

if (currSum > maxSum) {

maxSum = currSum;

}

if (currSum < 0) {

currSum = 0;

}

}

return maxSum;

}

public static void main(String args[]) {

// Your code goes here

int a[] = {1, 3, 8, -2, 6, -8, 5};

System.out.println(maximumSubarraySum(a));

}

}