

# **Design & Analysis of Algorithm (Lab)**

Name: Ananya

**SAPID: 590013832** 

**B-33** 

**Submitted to: Mr.Aryan Gupta** 

https://github.com/ananya438/DAALAB ANANYA-590013832

# Kadanes Algorithma and analyze its time complexity.

```
public class KadanesAlgorithm {
  public static int maxSubArraySum(int[] nums) {
    if (nums == null || nums.length == 0) {
       return 0:
    }
    int max_so_far = nums(0);
    int current_max = nums[0];
    for (int i = 1; i < nums.length; i++) \{
       current max = Math.max(nums[i], current max + nums[i]);
       max_so_far = Math.max(max_so_far, current_max);
    }
    return max_so_far;
  }
  public static void main(String[] args) {
    int[] arr1 = \{-2, 1, -3, 4, -1, 2, 1, -5, 4\};
    System.out.println("Maximum contiguous sum is " + maxSubArraySum(arr1));
    int[] arr2 = {1, 2, 3, -2, 5};
    System.out.println("Maximum contiguous sum is " + maxSubArraySum(arr2));
  }
```

```
刘 File Edit Selection View Go Run Terminal Help
                                                                                          Q First lectures
                                              ∨ FIRST ... 🗗 🖰 🗗 …
                           DAA > J Kadanes.java > 😝 Kadanes > 🕅 maxSubArraySum(int[])
                                 public class Kadanes {
      > Apna College
      V DAA
                                     public static int maxSubArraySum(int[] nums) {
       J celebrity_problem.cl...
                                         if (nums == null || nums.length == 0) {
       J celebrity_problem.ja...
                                             return 0;
       J HuffmanCoding.class
       int max_so_far = nums[0];
       J HuffmanComparato...
出
                                         int current_max = nums[0];
       KruskalMST.java
                                         for (int i = 1; i < nums.length; i++) {
83
       J merge_sort.class
                                             current_max = Math.max(nums[i], current_max + nums[i]);
                                             max_so_far = Math.max(max_so_far, current_max);
       return max so far;
       J Quick_Sort.java
                                      public static void main(String[] args) {
                                         int[] arr1 = {-2, 1, -3, 4, -1, 2, 1, -5, 4};
                                         System.out.println("Maximum contiguous sum is " + maxSubArraySum(arr1));
                                          System.out.println("Maximum contiguous sum is " + maxSubArraySum(arr2));
```

### 0/P:

```
PROBLEMS 3 OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\Users\nannu\Desktop\JAVA DSA\JAVA\First lectures> & 'C:\Program Files\Java\jdk-22\bin\java\nannu\AppData\Roaming\Code\User\workspaceStorage\0af2579802541dcb06e2e6ec0b1acf8d\redhat.java\jd Maximum contiguous sum is 6
Maximum contiguous sum is 9

PS C:\Users\nannu\Desktop\JAVA DSA\JAVA\First lectures>
```

**Test Case 1:** Standard Case (Positive and Negative Numbers)

**Input Array:** {-2, 1, -3, 4, -1, 2, 1, -5, 4}

**Expected Output: 6** 

Explanation: The maximum contiguous subarray is [4, -1, 2, 1], and its sum is 4 - 1 + 2 + 1 = 6.

```
nannu\AppData\Roaming\Code\User\workspaceStorage\0af2579802541dcb06e2e
Maximum contiguous sum is 6
Maximum contiguous sum is -1
PS C:\Users\nannu\Desktop\JAVA DSA\JAVA\First lectures>
```

#### **Test Case 2:** All Negative Numbers

```
Input Array: {-5, -2, -8, -1, -3}
```

#### **Expected Output: -1**

Explanation: When all numbers are negative, the maximum sum is the single largest number in the array. In this case, the largest number is -1.

```
nannu\AppData\Roaming\Code\User\workspaceStorage\0af2579802541dcb06e2e
Maximum contiguous sum is 6
Maximum contiguous sum is -1
PS C:\Users\nannu\Desktop\JAVA DSA\JAVA\First lectures>
```

### **Time Complexity Analysis**

#### O(n)

The algorithm uses a single for loop that runs through the array once.

Inside the loop, all operations are constant time, O(1).

Since it takes a fixed amount of time for each of the n elements, the total time is linear.

## **Space Complexity**

#### **O**(1)

The algorithm only uses a few variables to store the current and overall maximum sums.

The memory used does not depend on the size of the array. It's constant.