# **Leetcode 53 – Maximum Subarray**

## Problem Understanding

Given an integer array nums, find the **contiguous subarray** (containing at least one number) which has the **largest sum**, and return its sum.

This is the **maximum subarray sum** problem, also known as **Kadane’s Algorithm**.

### Key Intuition

You’re deciding at each position:

* Do I **extend** the previous subarray?
* Or **start fresh** from this element?

## Optimized Java Solution (Kadane’s Algorithm)

class Solution {

public int maxSubArray(int[] nums) {

int maxSoFar = nums[0];

int currentSum = nums[0];

for (int i = 1; i < nums.length; i++) {

currentSum = Math.max(nums[i], currentSum + nums[i]); // Extend or restart

maxSoFar = Math.max(maxSoFar, currentSum); // Track max

}

return maxSoFar;

}

}

## Dry Run Using Table

### Input:

nums = [-2, 1, -3, 4, -1, 2, 1, -5, 4]

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| i | num | currentSum | maxSoFar | Explanation |
| 0 | -2 | -2 | -2 | Initialization |
| 1 | 1 | max(1, -2+1) = 1 | 1 | Restart new subarray |
| 2 | -3 | max(-3, 1-3) = -2 | 1 | Bad drop, restart at -3 |
| 3 | 4 | max(4, -2+4) = 4 | 4 | New subarray with 4 |
| 4 | -1 | max(-1, 4-1) = 3 | 4 | Extend 4, still strong |
| 5 | 2 | max(2, 3+2) = 5 | 5 | Extend to 5 |
| 6 | 1 | max(1, 5+1) = 6 | 6 | Extend to 6 |
| 7 | -5 | max(-5, 6-5) = 1 | 6 | Still positive sum |
| 8 | 4 | max(4, 1+4) = 5 | 6 | Extend but doesn’t beat 6 |

✅ Final Answer: maxSoFar = 6

→ Subarray: [4, -1, 2, 1]

## Time and Space Complexity

| **Metric** | **Value** |
| --- | --- |
| Time | O(n) |
| Space | O(1) |

## Alternate Approaches

### 1. ****Brute Force (O(n²) or O(n³))****

* Try all subarrays, compute their sums.
* Very slow on large arrays.

### 2. ****Divide and Conquer (O(n log n))****

* Recursively find max subarray in:
  + Left half
  + Right half
  + Crossing middle
* Combine the results.
* More complex, rarely used in practice now.