(1) → Maximum subarray - #kadane #array

① → Question -

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

② → Examples -

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

→ 6 (4)

[4, -1, 2, 1]

nums = [1]

→ 1

nums = [5, 4, -1, 7, 8]

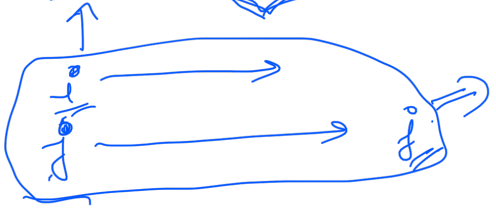
→ 23

③ → Intuition -

Take all subarrays; calc sum individually return the max.

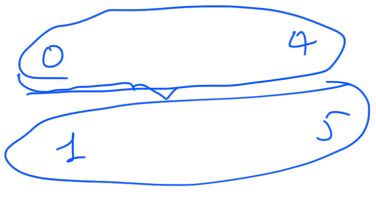
④ → Brute force soln -

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



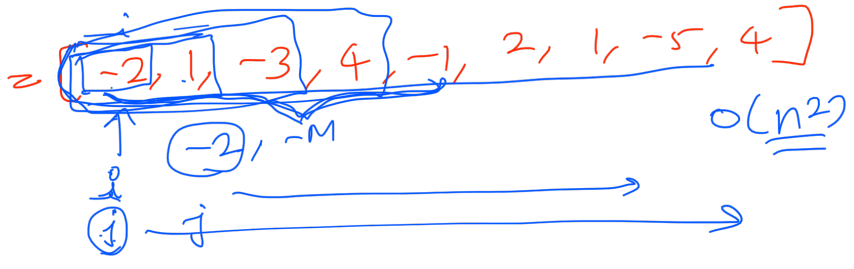
One optimisation

maxSum =

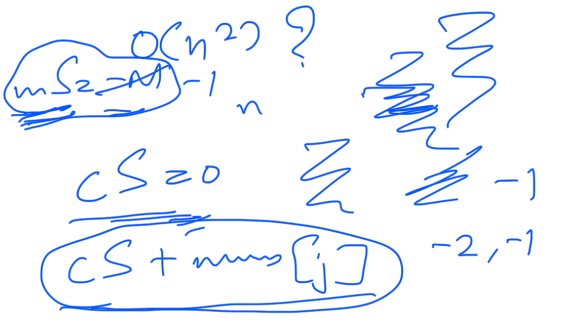


$O(n^2 \cdot n)$

$O(n^3)$



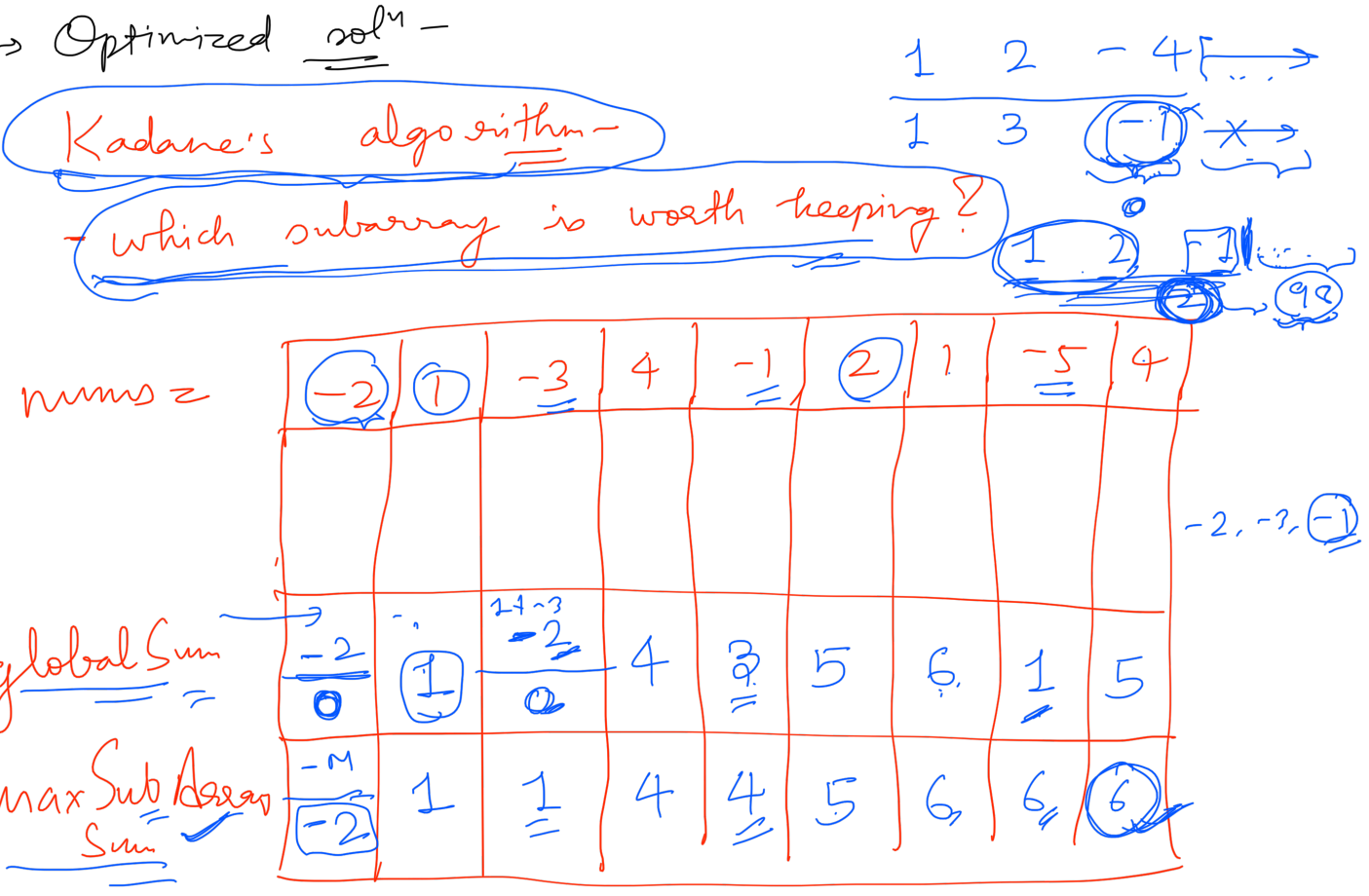
$O(n^2)$



⑤ Optimized soln –

Kadane's algorithm –

- which subarray is worth keeping?



⑥ Time complexity / Space Complexity –

T = O(n)

S = O(1)

⑦ Code walkthrough –

```
int maxSubArray(int[] nums) {
    int globalSum = 0;
    maxSubArraySum = MIN;
```

```
for (int idx = 0; idx < nums.length; idx++) {
    globalSum += nums[idx];
    maxSumArraySum = Math.max(
        maxSumArraySum, globalSum);
    if (globalSum < 0) {
        globalSum = 0;
    }
}
return maxSumArraySum;
}
```