**M - [53. Maximum Sub-array](https://leetcode.com/problems/maximum-subarray/)**

Given an integer array nums, find the subarray with the largest sum, and return its sum.

**Example 1:**

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

Output: 6

Explanation: The subarray [4,-1,2,1] has the largest sum 6.

**Example 2:**

Input: nums = [1]

Output: 1

Explanation: The subarray [1] has the largest sum 1.

**Example 3:**

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

Output: 23

Explanation: The subarray [5,4,-1,7,8] has the largest sum 23.

Constraints:

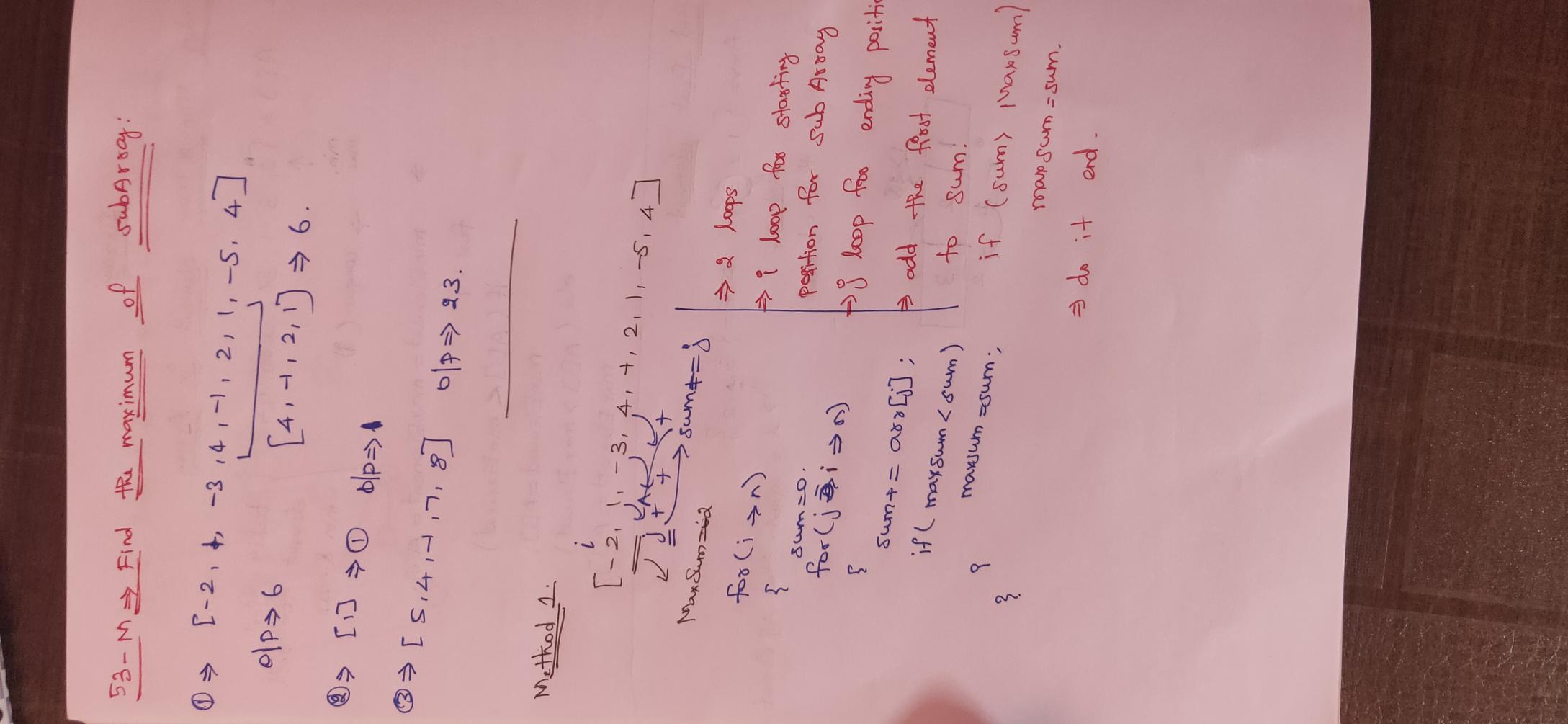
1 <= nums.length <= 105

-104 <= nums[i] <= 104

**Approach 1: => Brute Force**

**Procedure :**

* Use Two loops (i & j)
* i loop for start the sub-array to find Max\_Sum
* j loop is used to itrated to add the first of the sub\_array to last element of sub\_array
* First sum+=arr[j] , then **if(sum > max\_sum)** then **max\_sum=sum**
* After the loops , return max\_sum
* Cons : O(n^2) or TLE



**Solution:**

class Solution {

    public int maxSubArray(int[] nums) {

        int max\_sum=Integer.MIN\_VALUE;

        for(int i=0;i<nums.length;i++){

            int sum=0;

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

                sum+=nums[j];

                if(max\_sum<sum)

                    max\_sum=sum;

            }

        }

        return max\_sum;

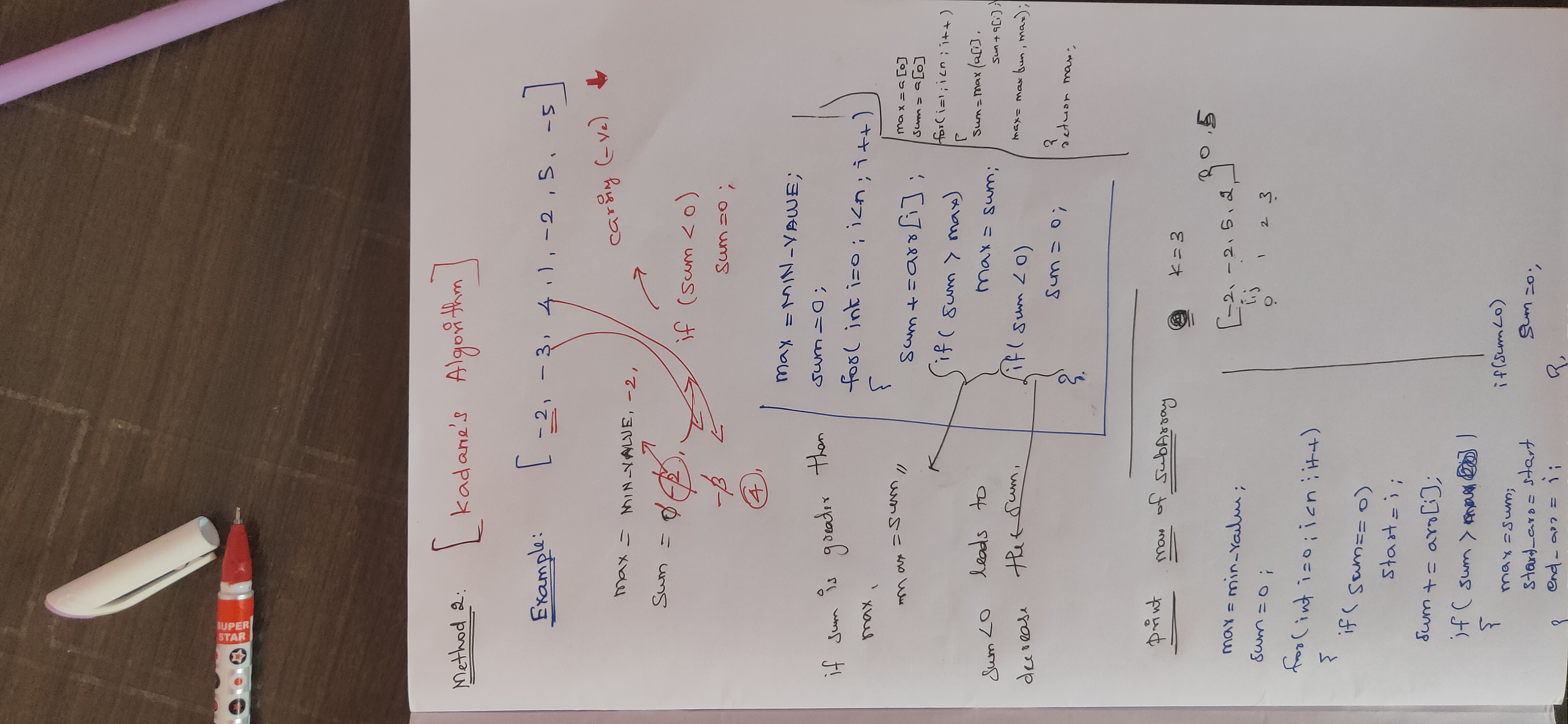
    }

}

**Approach 2: =>Kadane’s Algorithm**

**Procedure :**

* Lets take max and sum = 0
* Have a one for loop to iterate the elements until last element
* First add sum+=arr[i]
* then **if(sum > max\_sum)** then **max\_sum=sum**
* then **if(sum < 0)** then **sum =0 ,** because if sum<0 (negative sum) it leads to decrease the total sum ,



**Solution 1:**

class Solution {

    public int maxSubArray(int[] nums) {

        int max\_sum=Integer.MIN\_VALUE , sum =0;

        for(int i=0;i<nums.length;i++){

            sum+=nums[i];

            if(max\_sum<sum)

                max\_sum=sum;

            if(sum<0)

                sum=0;

        }

        return max\_sum;

    }

}

**Solution 2:**

class Solution {

    public int maxSubArray(int[] nums) {

        int maxSum = nums[0];

        int currentSum = nums[0];

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

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

            maxSum = Math.max(maxSum, currentSum);

        }

        return maxSum;

    }

}