M - 53. Maximum Sub-array

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:

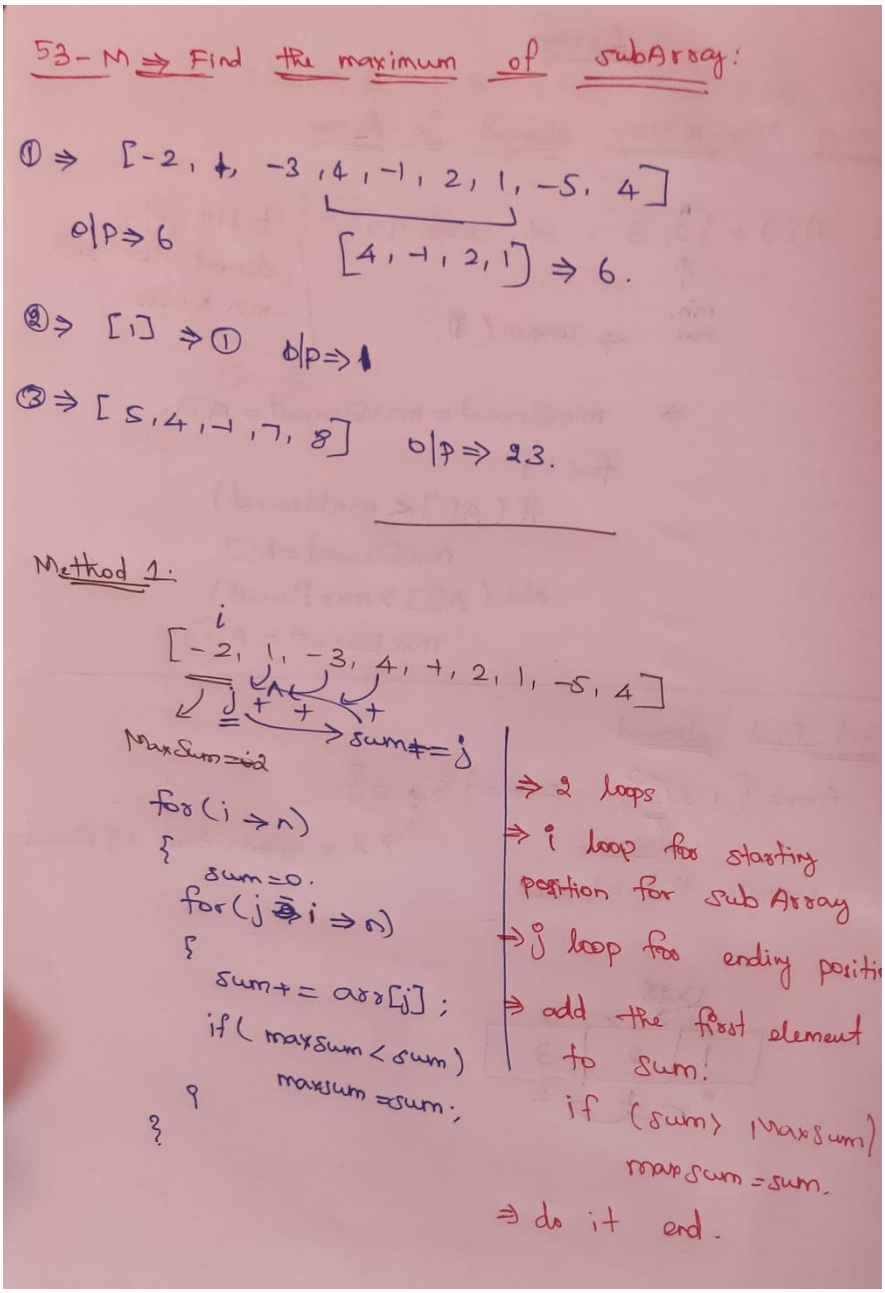
Input: nums = [5,4,-1,7,8]
Output: 23
Explanation: The subarray [5,4,-1,7,8] has the largest sum 23.

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



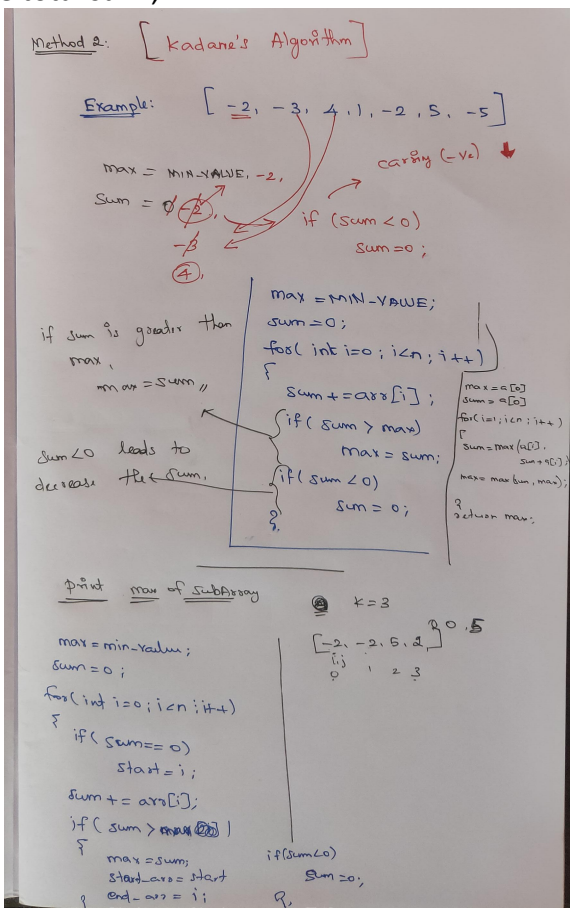
Solution:

```
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;
    }
}
```