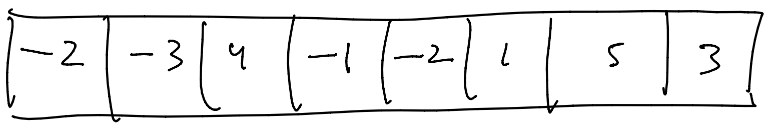
22. Maximum Subarray

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

A **subarray** is a **contiguous** part of an array.

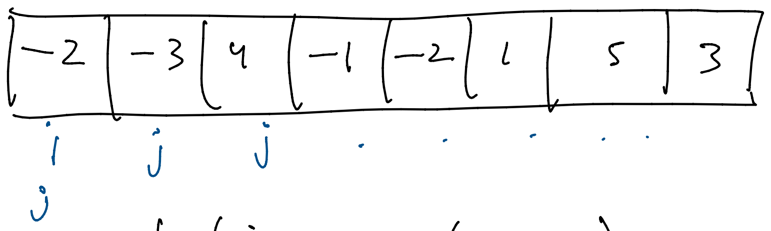
Example 1:

```
Input: nums = [-2,1,-3,4,-1,2,1,-5,4]
Output: 6
Explanation: [4,-1,2,1] has the largest sum = 6.
```



① Brute force

* Iterate over all the subarray
* Try to find out the max subarray



```
for (i → (0 − n−1)
    for (j → (i − n −1)
        for (k → (i...j))
            sum +=
            maxi = max (maxi, sum)
        ?
```

T.C ⇒ $O(N^3)$

}

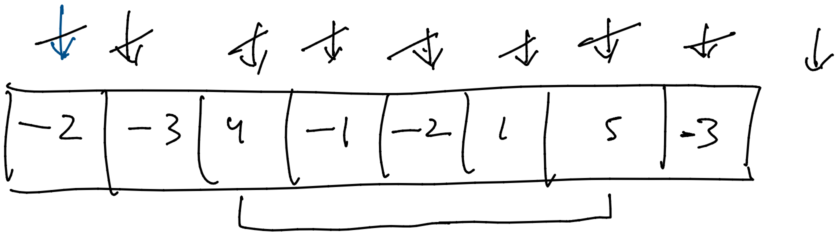
}

② Better Approach :

```
for(i → 0-n-1)
    for(j → i-n-1)
        sum += a[j]
    maxi = max(maxi, sum)
    }
```

$O(N^2)$

③ Kadane's Algorithm :

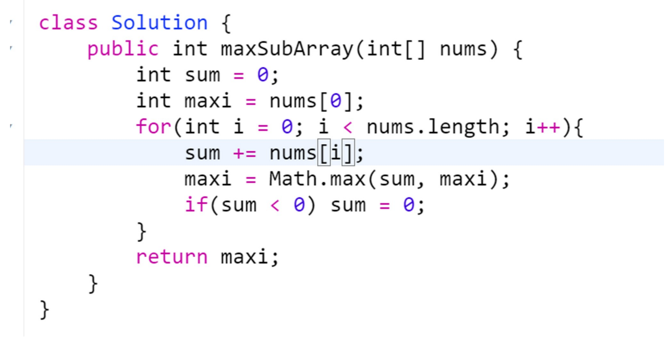


sum = ~~0~~ ~~-2~~ ~~0~~ ~~4~~ ~~3~~ ~~1~~ ~~2~~ ~~7~~ 4

maxi = a[0] // must have one element (given in the question)

= ~~-2~~ ~~0~~ ~~4~~ 7 ⇒ output

Carrying a -ve is of none so we change it to 0. because it decrease the value



```
class Solution {
    public int maxSubArray(int[] nums) {
        int sum = 0;
        int maxi = nums[0];
        for(int i = 0; i < nums.length; i++){
            sum += nums[i];
            maxi = Math.max(sum, maxi);
            if(sum < 0) sum = 0;
        }
        return maxi;
    }
}
```