53. 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.

Follow up: If you have figured out the O(n) solution, try coding another solution using the **divide and conquer** approach, which is more subtle.

C++ Solution

```cpp
#include <iostream>
#include <climits>
using namespace std;

int main() {
    int n = 5;
    int arr[5] = {1,2,3,4,5};
    int maxSum = INT_MIN;
    for(int st=0;st<n;st++){
        int curSum = 0;
        for(int end=st;end<n;end++){
            curSum += arr[end];
            maxSum = max(curSum,maxSum);
        }
    }
    cout <<"max subarray: "<<maxSum<<endl;
    return 0;
}
```

```
class Solution {
    public:
        int maxSubArray(vector<int>& nums) {
            int curSum = 0, maxSum = INT_MIN;
            for(int val : nums){
                curSum += val;
                maxSum = max(curSum,maxSum);
                if(curSum < 0){
                    curSum = 0;
                }
            }
            return maxSum;
        }
    };
```

Java Solution

```
class Solution {
    public int maxSubArray(int[] nums) {
        int curSum = 0, maxSum = Integer.MIN_VALUE;


        for(int val: nums){
            curSum += val;
            maxSum = Math.max(curSum,maxSum);
            if(curSum < 0){
                curSum = 0;
            }
        }
        return maxSum;
    }
}
```