53. Maximum Subarray

Easy

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

**Example:**

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

**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.

Dynamic Programming solution:

class Solution {

public:

int maxSubArray(vector<int>& nums) {

int dp[nums.size()];

dp[0]=nums[0];

int max=dp[0];

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

dp[i]=nums[i]+(dp[i-1]>0?dp[i-1]:0);

max=dp[i]>max?dp[i]:max;

}

return max;

}

};

class Solution {

public:

int maxSubArray(vector<int>& nums) {

int dp[nums.size()];

int max=dp[0]=nums[0];

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

if(dp[i-1]>0) dp[i]=dp[i-1]+nums[i];

else dp[i]=nums[i];

if(max<dp[i]) max=dp[i];

}

return max;

}

};

Success

[Details](https://leetcode.com/submissions/detail/206580331/)

Runtime: 12 ms, faster than 100.00% of C++ online submissions for Maximum Subarray.

Memory Usage: 6.8 MB, less than 0.75% of C++ online submissions for Maximum Subarray.