Maximum Subarray

Difficulty	Medium
Category	Greedy
Question	https://leetcode.com/problems/maximum-subarray/
Solution	https://youtu.be/5WZl3MMT0Eg
Status	Done

Question

> Given an integer array `nums`, find the subarray with the largest sum, and return *its sum*

Example

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.
```

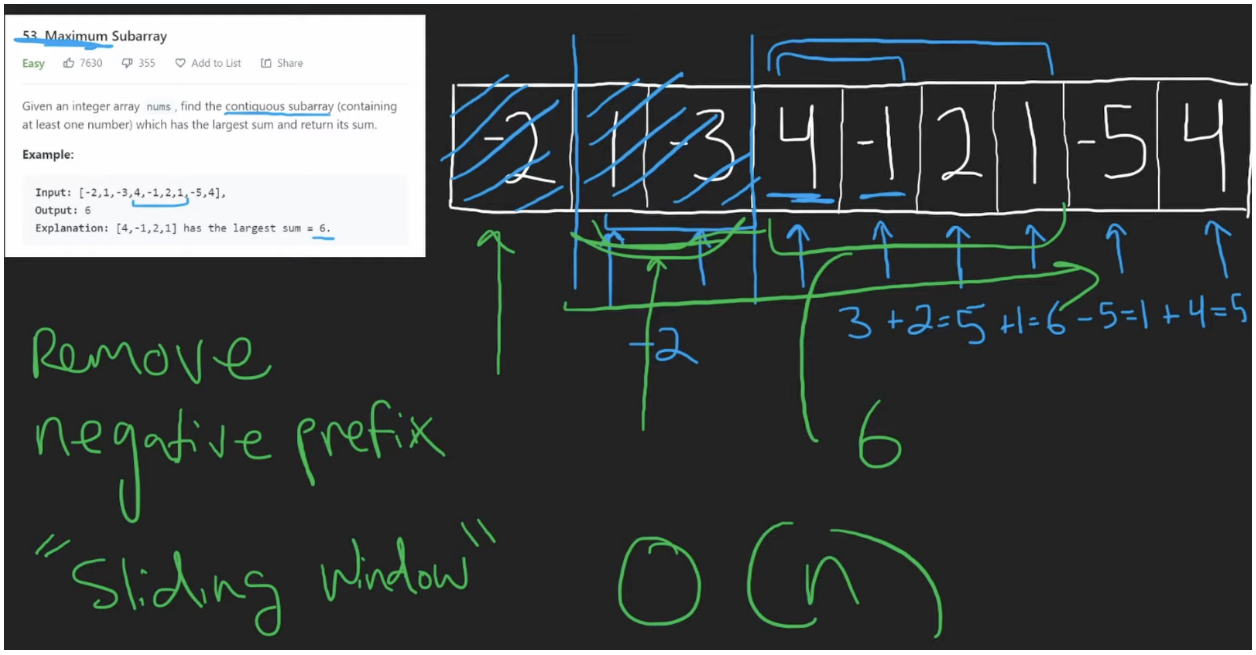
Example 3:

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

Idea

💡 Kadane’s Algo for finding the maximum subarray

💡 Dynamic programming: compute max sum for each prefix and keep track of the max



53. Maximum Subarray

Easy 7630 355 Add to List Share

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.
```

Solution

```
class Solution:
    def maxSubArray(self, nums: List[int]) -> int:
        # Initialize 'res' to the first element of the input array 'nums'.
        res = nums[0]

        # Initialize 'total' to 0, which represents the current subarray sum.
        total = 0

        # Iterate through the elements of the 'nums' array.
        for n in nums:
            total += n  # Add the current element 'n' to the 'total' to extend the subarray.

            # Update 'res' with the maximum of the current 'res' and 'total'.
            res = max(res, total)

            if total < 0:
                # If 'total' becomes negative, reset it to 0.
                # This ensures that negative subarrays are not considered in the maximum sum.
                total = 0

        return res  # Return the final value of 'res', which is the maximum subarray sum.
```

Explanation

Interview

Some technical interview questions based on the knowledge set in this coding problem might include:

- Can you explain the concept of dynamic programming and how it applies to this problem?
- How does the `dp` array work in the solution? Can you walk me through how it is updated for each element in the input array?
- What is the time complexity of the solution? Can you explain why?
- Are there any edge cases or inputs that could cause the solution to fail? How would you modify the solution to handle these cases?
- Can you think of any alternative approaches to solving this problem? How do they compare in terms of time and space complexity?