Problem 3410: Maximize Subarray Sum After Removing All Occurrences of One Element

Problem Information

Difficulty: Hard
Acceptance Rate: 21.70%
Paid Only: No
Tags: Array, Dynamic Programming, Segment Tree

Problem Description

You are given an integer array `nums`.

You can do the following operation on the array **at most** once:

* Choose **any** integer `x` such that `nums` remains **non-empty** on removing all occurrences of `x`. * Remove **all** occurrences of `x` from the array.

Return the **maximum** subarray sum across **all** possible resulting arrays.

Example 1:

Input: nums = [-3,2,-2,-1,3,-2,3]

Output: 7

Explanation:

We can have the following arrays after at most one operation:

* The original array is `nums = [-3, 2, -2, -1, _**3, -2, 3**_]`. The maximum subarray sum is `3 + (-2) + 3 = 4`. * Deleting all occurences of `x = -3` results in `nums = [2, -2, -1, **_3, -2, 3_**]`. The maximum subarray sum is `3 + (-2) + 3 = 4`. * Deleting all occurences of `x = -2` results in `nums = [-3, **_2, -1, 3, 3_**]`. The maximum subarray sum is `2 + (-1) + 3 + 3 = 7`. * Deleting all occurences of `x = -1` results in `nums = [-3, 2, -2, **_3, -2, 3_**]`. The maximum subarray

sum is `3 + (-2) + 3 = 4`. * Deleting all occurences of `x = 3` results in `nums = [-3, _**2**_ , -2, -1, -2]`. The maximum subarray sum is 2.

The output is `max(4, 4, 7, 4, 2) = 7`.

Example 2:

Input: nums = [1,2,3,4]

Output: 10

Explanation:

It is optimal to not perform any operations.

Constraints:

* `1 <= nums.length <= 105` * `-106 <= nums[i] <= 106`

Code Snippets

C++:

```
class Solution {
public:
long long maxSubarraySum(vector<int>& nums) {

}
};
```

Java:

```
class Solution {
public long maxSubarraySum(int[] nums) {

}
}
```

Python3:

```
class Solution:
def maxSubarraySum(self, nums: List[int]) -> int:
```