Problem 1746: Maximum Subarray Sum After One Operation

Problem Information

Difficulty: Medium
Acceptance Rate: 65.25%
Paid Only: Yes
Tags: Array, Dynamic Programming

Problem Description

You are given an integer array `nums`. You must perform **exactly one** operation where you can **replace** one element `nums[i]` with `nums[i] * nums[i]`.

Return _the**maximum** possible subarray sum after **exactly one** operation_. The subarray must be non-empty.

Example 1:

Input: nums = [2,-1,-4,-3] **Output:** 17 **Explanation:** You can perform the operation on index 2 (0-indexed) to make nums = [2,-1,**16** ,-3]. Now, the maximum subarray sum is 2 + -1 + 16 = 17.

Example 2:

Input: nums = [1,-1,1,1,-1,-1,1] **Output:** 4 **Explanation:** You can perform the operation on index 1 (0-indexed) to make nums = [1,**1** ,1,1,-1,-1,1]. Now, the maximum subarray sum is 1 + 1 + 1 + 1 = 4.

Constraints:

* `1 <= nums.length <= 105` * `-104 <= nums[i] <= 104`

Code Snippets

C++:

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

}
};
```

Java:

```
class Solution {
public int maxSumAfterOperation(int[] nums) {

}
}
```

Python3:

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