

# Problem 2036: Maximum Alternating Subarray Sum

## Problem Information

**Difficulty:** Medium

**Acceptance Rate:** 40.02%

**Paid Only:** Yes

**Tags:** Array, Dynamic Programming

## Problem Description

A **subarray** of a **0-indexed** integer array is a contiguous **non-empty** sequence of elements within an array.

The **alternating subarray sum** of a subarray that ranges from index `i` to `j` (**inclusive**, `0 ≤ i ≤ j < nums.length`) is `nums[i] - nums[i+1] + nums[i+2] - ... +/- nums[j]`.

Given a **0-indexed** integer array `nums`, return **the maximum alternating subarray sum** of any subarray of `nums`.

**Example 1.**

**Input:** `nums = [3,-1,1,2]` **Output:** `5` **Explanation:** The subarray `[3,-1,1]` has the largest alternating subarray sum. The alternating subarray sum is `3 - (-1) + 1 = 5`.

**Example 2.**

**Input:** `nums = [2,2,2,2,2]` **Output:** `2` **Explanation:** The subarrays `[2]`, `[2,2,2]`, and `[2,2,2,2,2]` have the largest alternating subarray sum. The alternating subarray sum of `[2]` is 2. The alternating subarray sum of `[2,2,2]` is `2 - 2 + 2 = 2`. The alternating subarray sum of `[2,2,2,2,2]` is `2 - 2 + 2 - 2 + 2 = 2`.

**Example 3.**

**Input:** `nums = [1]` **Output:** `1` **Explanation:** There is only one non-empty subarray, which is `[1]`. The alternating subarray sum is 1.

**\*\*Constraints:\*\***

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

## Code Snippets

### C++:

```
class Solution {  
public:  
    long long maximumAlternatingSubarraySum(vector<int>& nums) {  
  
    }  
};
```

### Java:

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

### Python3:

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