Problem 1911: Maximum Alternating Subsequence Sum

Problem Information

Difficulty: Medium
Acceptance Rate: 58.91%
Paid Only: No
Tags: Array, Dynamic Programming

Problem Description

The **alternating sum** of a **0-indexed** array is defined as the **sum** of the elements at **even** indices **minus** the **sum** of the elements at **odd** indices.

* For example, the alternating sum of `[4,2,5,3]` is `(4 + 5) - (2 + 3) = 4`.

Given an array `nums`, return _the**maximum alternating sum** of any subsequence of _`nums` _(after**reindexing** the elements of the subsequence)_.

A **subsequence** of an array is a new array generated from the original array by deleting some elements (possibly none) without changing the remaining elements' relative order. For example, `[2,7,4]` is a subsequence of `[4,_2_ ,3,_7_ ,2,1,_4_]` (the underlined elements), while `[2,4,2]` is not.

Example 1:

Input: nums = [_4_ ,_2_ ,_5_ ,3] **Output:** 7 **Explanation:** It is optimal to choose the subsequence [4,2,5] with alternating sum (4 + 5) - 2 = 7.

Example 2:

Input: nums = [5,6,7,_8_] **Output:** 8 **Explanation:** It is optimal to choose the subsequence [8] with alternating sum 8.

Example 3:

Input: nums = [_6_ ,2,_1_ ,2,4,_5_] **Output:** 10 **Explanation:** It is optimal to choose the subsequence [6,1,5] with alternating sum (6 + 5) - 1 = 10.

Constraints:

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

Code Snippets

C++:

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

}
};
```

Java:

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

}
}
```

Python3:

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