Problem 3509: Maximum Product of Subsequences With an Alternating Sum Equal to K

Problem Information

Difficulty: Hard
Acceptance Rate: 12.13%
Paid Only: No
Tags: Array, Hash Table, Dynamic Programming

Problem Description

You are given an integer array `nums` and two integers, `k` and `limit`. Your task is to find a non-empty **subsequence** of `nums` that:

* Has an **alternating sum** equal to `k`. * **Maximizes** the product of all its numbers _without the product exceeding_ `limit`.

Return the _product_ of the numbers in such a subsequence. If no subsequence satisfies the requirements, return -1.

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.

Example 1:

Input: nums = [1,2,3], k = 2, limit = 10

Output: 6

Explanation:

The subsequences with an alternating sum of 2 are:

* `[1, 2, 3]` * Alternating Sum: `1 - 2 + 3 = 2` * Product: `1 * 2 * 3 = 6` * `[2]` * Alternating Sum: 2 * Product: 2

The maximum product within the limit is 6.

Example 2:

Input: nums = [0,2,3], k = -5, limit = 12

Output: -1

Explanation:

A subsequence with an alternating sum of exactly -5 does not exist.

Example 3:

Input: nums = [2,2,3,3], k = 0, limit = 9

Output: 9

Explanation:

The subsequences with an alternating sum of 0 are:

* `[2, 2]` * Alternating Sum: `2 - 2 = 0` * Product: `2 * 2 = 4` * `[3, 3]` * Alternating Sum: `3 - 3 = 0` * Product: `3 * 3 = 9` * `[2, 2, 3, 3]` * Alternating Sum: `2 - 2 + 3 - 3 = 0` * Product: `2 * 2 * 3 * 3 = 36`

The subsequence `[2, 2, 3, 3]` has the greatest product with an alternating sum equal to `k`, but `36 > 9`. The next greatest product is 9, which is within the limit.

Constraints:

* `1 <= nums.length <= 150` * `0 <= nums[i] <= 12` * `-105 <= k <= 105` * `1 <= limit <= 5000`

Code Snippets

C++:

```
class Solution {
public:
int maxProduct(vector<int>& nums, int k, int limit) {

}
};
```

Java:

```
class Solution {
public int maxProduct(int[] nums, int k, int limit) {

}
}
```

Python3:

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