Problem 2552: Count Increasing Quadruplets

Problem Information

Difficulty: Hard
Acceptance Rate: 34.43%
Paid Only: No
Tags: Array, Dynamic Programming, Binary Indexed Tree, Enumeration, Prefix Sum

Problem Description

Given a **0-indexed** integer array `nums` of size `n` containing all numbers from `1` to `n`, return _the number of increasing quadruplets_.

A quadruplet `(i, j, k, l)` is increasing if:

* `0 <= i < j < k < l < n`, and * `nums[i] < nums[k] < nums[j] < nums[l]`.

Example 1:

Input: nums = [1,3,2,4,5] **Output:** 2 **Explanation:** - When i = 0, j = 1, k = 2, and l = 3, nums[i] < nums[k] < nums[j] < nums[l]. - When i = 0, j = 1, k = 2, and l = 4, nums[i] < nums[k] < nums[j] < nums[l]. There are no other quadruplets, so we return 2.

Example 2:

Input: nums = [1,2,3,4] **Output:** 0 **Explanation:** There exists only one quadruplet with i = 0, j = 1, k = 2, l = 3, but since nums[j] < nums[k], we return 0.

Constraints:

* `4 <= nums.length <= 4000` * `1 <= nums[i] <= nums.length` * All the integers of `nums` are **unique**. `nums` is a permutation.

Code Snippets

C++:

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

}
};
```

Java:

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

}
}
```

Python3:

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