Count The Number Of Fair Pairs

solved by	Senan
Platform	LeetCode
difficulty	Medium
# Serial	2563
tags	Sorting
language	C++
solved on	@13/11/2024
link	https://leetcode.com/problems/count-the-number-of-fair-pairs/
Completion	☑

Intuition

We need to count all pairs `(i, j)` in `nums` such that `i < j` and `lower <= nums[i] + nums[j] <= upper`. By sorting the array, we can use binary search to quickly find the range of valid values that form fair pairs with each element.

Approach

1. Sort `nums` to enable binary search.
2. Iterate over each element from the end of the sorted list.
    - For each element `nums[i]`, calculate the range `[nLower, nUpper]`:
        - `nUpper` is the maximum value for `nums[j]` such that `nums[i] + nums[j] <= upper`.
        - `nLower` is the minimum value for `nums[j]` such that `nums[i] + nums[j] >= lower`.
    - Use `lower_bound` and `upper_bound` to find the range of indices that satisfy `nLower <= nums[j] <= nUpper` for `j < i`.
3. The difference `ub - lb` gives the count of fair pairs for `nums[i]`, add it to `answer`.
4. Return `answer`.

Complexity

Time Complexity:

The time complexity is **O(n log n)** because sorting takes **O(n log n)** and each binary search operation inside the loop is **O(log n)**.

Space Complexity:

The space complexity is **O(1)** as no additional data structures are used.

Code

```
class Solution {
public:
    long long countFairPairs(vector<int>& nums, int lower, int upper) {
        sort(nums.begin(), nums.end());
        long long answer = 0;
```

```
        for(int i = nums.size() - 1; i >= 0; i--) {
            long long nUpper = upper - nums[i];
            long long nLower = lower - nums[i];
            auto ub = upper_bound(nums.begin(), nums.begin() + i, nUpper);
            auto lb = lower_bound(nums.begin(), nums.begin() + i, nLower);
            answer += ub - lb;
        }
        return answer;
    }
};
```