**Problem Name:** Three sum smaller

**Topics:**

**Companies:**

**Level:** Easy

**Language:** C++

**Problem Statement**:

**Input Format:**

**Output Format:**

**Constraints:**

**Examples:**

**Brute force Solution:**

**Explanation:**

**Code:**

// 2 pointers + linear scan: for each i, j, k triple, let i start from 0 to the end,

    // we need to find a way to calculate the (j, k) pairs such that the triple sum < target.

    // Let j start from i + 1, and k start from the end, each time we find (i, j, k) < target,

    // we will also find (i, j, k - 1), ..., (i, j, j + 1) < target, so there are totally

    // k - j pairs starting from j such that (i, j, k) < target.

    // Otherwise if (i, j, k) >= target, we reduce 1 from k, and check (i, j, k) again.

    // We keep doing this until j >= k.

**Time Complexity**: O(N3)

**Space Complexity: O(1)**

**Optimized Solution:**

Explanation: As you would know, every 3sum/2sum problem requires some form of sorting. Now as here we have to find the number of triplets, sorting makes sense as order doesnt matter.  
Now, iterate through the loop, specify two counters-> lo and hi, initialise them to i+1 and n-1 at the moment. We will be perform the two pointers on this counters.  
While lo<hi, calculate sum of current 3 indices- lo, hi and i. If the sum is < target, we know for any 3rd member of triplet in the sorted array, if we take sum, sum will be less than target. Example for moment lets say i=1, lo=2 and hi=5. And nums[i]+nums[lo]+nums[hi] came out to be <target, we know that for all indices less than hi, we will get a sum <target (on triplets nums[1]+nums[2]+nums[3] or nums[1]+nums[2]+nums[4]). So directly add lo-hi in the counter as we have that many number if triplets and increment lo++.  
If sum is >=target, we need to decrement hi-- as with current lo and i, we cant form sum<target.

**Code:**

**Time Complexity**: O(N2)

**Space Complexity:** O(1)