3Sum Smaller

Given an array of *n* integers *nums* and a *target*, find the number of index triplets `i, j, k` with `0 that satisfy the condition nums[i] + nums[j] + nums[k] .`

For example, given *nums* = `[-2, 0, 1, 3]`, and *target* = 2.

Return 2. Because there are two triplets which sums are less than 2:

```
[-2, 0, 1]
[-2, 0, 3]
```

Follow up:
Could you solve it in $O(n^2)$ runtime?

Solution 1

```
public class Solution {
    int count;

    public int threeSumSmaller(int[] nums, int target) {
        count = 0;
        Arrays.sort(nums);
        int len = nums.length;

        for(int i=0; i<len-2; i++) {
            int left = i+1, right = len-1;
            while(left < right) {
                if(nums[i] + nums[left] + nums[right] < target) {
                    count += right-left;
                    left++;
                } else {
                    right--;
                }
            }
        }

        return count;
    }
}
```

written by SOY original link here

Solution 2

After sorting, if `i, j, k` is a valid triple, then `i, j-1, k`, ..., `i, i+1, k` are also valid triples. No need to count them one by one.

```
def threeSumSmaller(self, nums, target):
    nums.sort()
    count = 0
    for k in range(len(nums)):
        i, j = 0, k - 1
        while i < j:
            if nums[i] + nums[j] + nums[k] < target:
                count += j - i
                i += 1
            else:
                j -= 1
    return count
```

written by StefanPochmann original link here

Solution 3

We sort the array first. Then, for each element, we use the two pointer approach to find the number of triplets that meet the requirements.

Let me illustrate how the two pointer technique works with an example:

```
target = 2

  i  lo    hi
[-2, 0, 1, 3]
```

We use a for loop (index i) to iterate through each element of the array. For each i, we create two pointers, lo and hi, where lo is initialized as the next element of i, and hi is initialized at the end of the array. If we know that nums[i] + nums[lo] + nums[hi] < target, then we know that since the array is sorted, we can replace hi with any element from lo+1 to nums.length-1, and the requirements will still be met. Just like in the example above, we know that since -2 + 0 + 3 < 2, we can replace hi (3) with 1, and it would still work. Therefore, we can just add hi - lo to the triplet count.

```
public class Solution {
    public int threeSumSmaller(int[] nums, int target) {
        int result = 0;
        Arrays.sort(nums);
        for(int i = 0; i <= nums.length-3; i++) {
            int lo = i+1;
            int hi = nums.length-1;
            while(lo < hi) {
                if(nums[i] + nums[lo] + nums[hi] < target) {
                    result += hi - lo;
                    lo++;
                } else {
                    hi--;
                }
            }
        }
        return result;
    }
}
```

written by kevinhsu original link here

From Leetcoder.