Two Sum I (LeetCode 1 - Easy)

Problem Description

Given an array of integers, return **indices** of the two numbers such that they add up to a specific target.

You may assume that each input would have ***exactly*** one solution, and you may not use the *same* element twice.

The input array is unsorted.

**Example:**

Given nums = [2, 7, 11, 15], target = 9,

Because nums[0] + nums[1] = 2 + 7 = 9, return [0, 1].

Problem Analysis

Use HashMap<Key, Value> 🡪 number as key, index as value.

One iteration 🡪 O(n)

Solution



Two Sum II – Input array is sorted (LeetCode167 - Medium)

Problem Description

Given an array of integers, return **indices** of the two numbers such that they add up to a specific target.

You may assume that each input would have ***exactly*** one solution, and you may not use the *same* element twice.

The input array is sorted.

**Example:**

Given nums = [2, 7, 11, 15], target = 9,

Because nums[0] + nums[1] = 2 + 7 = 9, return [0, 1].

Problem Analysis

Two pointers: low and high, collision.

Solution



Two Sum III (LintCode443 - Medium)

Problem Description

Given an array of integers, find **indices** of the two numbers such that they add up to a sum that is **larger than** specific target. Return number of such pairs.

**Example:**

Given nums = [2, 7, 11, 15], target = 23,

Because nums[2] + nums[3] = 11 + 15 = 26 > 23, return 1.

Problem Analysis

Sort first –> O(nlogn)

Two pointers: low and high, collision.

Solution



Triangle Count (LintCode382 - Hard)

Problem Description

Given an array of integers, use any three of these numbers as edges of a triangle, return how many triangles these numbers can form.

**Example:**

Given nums = [3, 4, 6, 7]

We can form three triangles: {3, 4, 6}, {3, 6, 7}, {4, 6, 7}, return 3

Problem Analysis

Sort first 🡪 O(nlogn)

Iterate through the array, assume the current integer is one of the edges, for the remaining numbers, use two pointers -> low and high, collision. 🡪 O(n^2)

Solution

