<https://leetcode.com/problems/3sum-closest/description/>

Given an integer array nums of length n and an integer target, find three integers in nums such that the sum is closest to target.

Return *the sum of the three integers*.

You may assume that each input would have exactly one solution.

**Example 1:**

Input: nums = [-1,2,1,-4], target = 1

Output: 2

Explanation: The sum that is closest to the target is 2. (-1 + 2 + 1 = 2).

**Example 2:**

Input: nums = [0,0,0], target = 1

Output: 0

Explanation: The sum that is closest to the target is 0. (0 + 0 + 0 = 0).

**Constraints:**

* 3 <= nums.length <= 500
* -1000 <= nums[i] <= 1000
* -104 <= target <= 104

**Attempt 1: 2023-02-27**

**Solution 1: Two Pointers but not Binary Search (30 min)**

class Solution {

public int threeSumClosest(int[] nums, int target) {

int len = nums.length;

int result = nums[0] + nums[1] + nums[len - 1];

Arrays.sort(nums);

for(int i = 0; i < len - 2; i++) {

int lo = i + 1;

int hi = len - 1;

while(lo < hi) {

int sum = nums[i] + nums[lo] + nums[hi];

if(Math.abs(sum - target) < Math.abs(result - target)) {

result = sum;

}

if(sum > target) {

hi--;

} else if(sum < target) {

lo++;

} else {

return sum;

}

}

}

return result;

}

}

Time complexity: O(n^2)

Space complexity: O(1)

**Refer to**

<https://leetcode.com/problems/3sum-closest/solutions/164611/logical-thinking-with-code-beats-99-33/>

**Logical Thinking**

If a candidate number nums[pre] is given, the problem is decreased to the **Two-sum Closest** problem - find two integers in nums such that the sum is closest to target - nums[pre]. We need to try all possible nums[pre].We'd better sort nums[] first. In this way, we can apply **Binary Search** in 2Sum Closest rather than **Exhaustive Search**.

**Trick**

The condition to terminate **Binary Search** is not 'no searching space' but 'not enough searching space' for we need to maintain two valid candidates in the searching space [lo, hi] both inclusive.

**Clear Code**

public int threeSumClosest(int[] nums, int target) {

if (nums == null || nums.length < 3) {

throw new IllegalArgumentException();

}

Arrays.sort(nums);

int n = nums.length, minDiff = Integer.MAX\_VALUE, sumClosest = -1;

for (int pre = 0; pre < n; pre++) {

int lo = pre + 1;

int hi = n - 1;

while (lo < hi) {

int curTarget = target - nums[pre];

int curSum = nums[lo] + nums[hi];

if (curSum < curTarget) {

if (curTarget - curSum < minDiff) {

minDiff = curTarget - curSum;

sumClosest = curSum + nums[pre];

}

lo++;

continue;

}

if (curSum > curTarget) {

if (curSum - curTarget < minDiff) {

minDiff = curSum - curTarget;

sumClosest = curSum + nums[pre];

}

hi--;

continue;

}

return curSum + nums[pre];

}

}

return sumClosest;

}