16 最接近的三数之和

Label：双指针

给定一个包括 n 个整数的数组 nums 和 一个目标值 target。找出 nums 中的三个整数，使得它们的和与 target 最接近。返回这三个数的和。假定每组输入只存在唯一答案。

```
输入：nums = [-1,2,1,-4], target = 1
输出：2
解释：与 target 最接近的和是 2 (-1 + 2 + 1 = 2) 。
```

- 定点 双重循环

```
class Solution {
    public int threeSumClosest(int[] nums, int target) {
        if (nums.length == 3) {
            return nums[0] + nums[1] + nums[2];
        }
        Arrays.sort(nums);
        int ans = nums[0] + nums[1] + nums[2];
        for (int i = 0; i < nums.length; i++) {  // 固定住第一个数
            int start = i + 1, end = nums.length - 1; // 遍历中间的数

            while (start < end) {
                int sum = nums[start] + nums[end] + nums[i];
                if(Math.abs(target - sum) < Math.abs(target - ans))  // 出现更小距离，用绝对值
                    ans = sum;
                if(sum > target) // sum 比target大，则右指针--
                    end--;
                else if(sum < target) // sum 比target小，则左指针++
                    start++;
                else
                    return ans;
            }
        }
        return ans;
    }
}
```

- 双重循环 剪枝

```
class Solution {
    public int threeSumClosest(int[] nums, int target) {
        Arrays.sort(nums);
        int result = nums[0] + nums[1] + nums[2];
        for (int i = 0; i < nums.length - 2; i++) {
            int left = i + 1;
            int right = nums.length - 1;
            while (left != right) {
                int min = nums[i] + nums[left] + nums[left + 1];
                if (target < min) {  // 最小值剪枝
                    if (Math.abs(result - target) > Math.abs(min - target))
                        result = min;
                    break;
                }
                int max = nums[i] + nums[right] + nums[right - 1];
                if (target > max) {
                    if (Math.abs(result - target) > Math.abs(max - target)) //?
                        result = max;
                    break;
                }
                int sum = nums[i] + nums[left] + nums[right];
                if (sum == target)
                    return sum;
                else if (Math.abs(sum - target) < Math.abs(result - target)) {
                    result = sum;

                }
                if (sum > target) {
                    right--;
                    while (left != right && nums[right] == nums[right + 1]) // 过滤连续相同值
                        right--;
                } else {
                    left++;
                    while (left != right && nums[left] == nums[left - 1])// 过滤连续相同值
                        left++;
                }
            }

            while (i < nums.length - 2 && nums[i] == nums[i + 1]) // i 同样过滤连续值
                i++;
        }

        return result;
    }
}
```