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。

例如，给定数组 `nums = [-1, 2, 1, -4]`, 和 `target = 1.`

与 `target` 最接近的三个数的和为 `2. (-1 + 2 + 1 = 2).`

思路，将数组排序，然后进行数组的两个和遍历，用sum标记当前最接近的和，通过比较sum和动态的两数的和来决定端口指向，把可能的数找出来，得到答案。

```
class Solution {
public:
    int threeSumClosest(vector<int>& nums, int target) {
        if(nums.size() <= 3)
        {
            int result = 0;
            for(int i = 0; i < nums.size(); i ++)
            {
                result += nums[i];
            }
            return result;
        }

        sort(nums.begin(), nums.end());
        int sum = nums[0] + nums[1] + nums[2];
        for(int i = 0; i < nums.size(); i ++)
        {
            int left = i + 1, right = nums.size() - 1;
            while(left < right)
            {
                int current = nums[left] + nums[right] + nums[i];
                if(abs(target - current) < abs(target - sum))
                {
                    sum = current;
                    if(sum == target)
                        return sum;
                }
                if(current > target)
                    right --;
                else
                    left ++;
            }
        }
        return sum;

    }
```

```
};
```