# 题目

给你一个 下标从 0 开始 的整数数组 nums ，其中 nums[i] 表示第 i 名学生的分数。另给你一个整数 k 。

从数组中选出任意 k 名学生的分数，使这 k 个分数间 最高分 和 最低分 的 差值 达到 最小化 。

返回可能的 最小差值 。

示例 1：

输入：nums = [90], k = 1

输出：0

解释：选出 1 名学生的分数，仅有 1 种方法：

- [90] 最高分和最低分之间的差值是 90 - 90 = 0

可能的最小差值是 0

示例 2：

输入：nums = [9,4,1,7], k = 2

输出：2

解释：选出 2 名学生的分数，有 6 种方法：

- [9,4,1,7] 最高分和最低分之间的差值是 9 - 4 = 5

- [9,4,1,7] 最高分和最低分之间的差值是 9 - 1 = 8

- [9,4,1,7] 最高分和最低分之间的差值是 9 - 7 = 2

- [9,4,1,7] 最高分和最低分之间的差值是 4 - 1 = 3

- [9,4,1,7] 最高分和最低分之间的差值是 7 - 4 = 3

- [9,4,1,7] 最高分和最低分之间的差值是 7 - 1 = 6

可能的最小差值是 2

提示：

1 <= k <= nums.length <= 1000

0 <= nums[i] <= 105

# 分析

## 方法一：排序

思路：

要想最小化选择的k名学生中最高分和最低分的差值，我们一定是在排好序后的数组中连续地进行选择。这是因为在选择时，如果跳过了某个下标i，那么在选择完毕后，将其中的最高分替换成nums[i]，最高分一定不会变大，与最低分的差值同样也不会变大。因此，一定存在有一种最优的选择方案，是连续选择了有序数组中的k个连续的元素。

这样一来，我们首先对数组nums进行升序排序，随后使用一个大小固定为k的滑动窗口在nums上进行遍历。记滑动窗口的左边界为i，那么右边界即为i+k−1，窗口中的k名学生最高分和最低分的差值即为nums[i+k−1]−nums[i]。

最终的答案即为所有nums[i+k−1]−nums[i]中的最小值。

代码：

class Solution {

public:

int minimumDifference(vector<int>& nums, int k) {

int n = nums.size();

sort(nums.begin(), nums.end());

int ans = INT\_MAX;

for (int i = 0; i + k - 1 < n; ++i) {

ans = min(ans, nums[i + k - 1] - nums[i]);

}

return ans;

}

};

复杂度分析

时间复杂度：O(nlogn)，其中n是数组nums的长度。排序需要的时间为O(nlogn)，后续遍历需要的时间为O(n)。

空间复杂度：O(logn)，即为排序需要使用的栈空间。