# 第一次算法设计作业

姓名： 刘宇骋 学号： 2021141440374

【说明】共两道题，至少完成一题，全完成可加分。解答应包括解题思路（方法、时间复杂度分析）、代码以及按结果验证中的输入得到的输出截图。作业提交时请只保留选做的题，删除空白题。

【题一】考虑在长度为 n 的数组 A 中查找第 k 小的元素的问题。请设计一个分治算法来解决这个问题。

**输入：**

一个包含 n 个整数的数组 nums。

一个整数 k，表示要找到第 k 小的元素。

**输出：**

第 k 小的元素。

【示例1】

输入：nums = [3, 2, 1, 5, 4]

k = 2

输出：The 2 th smallest element is: 2

【示例2】

输入：nums =[9, 7, 2, 5, 8, 3]

k=4

输出：The 4 th smallest element is: 7

【解题思路】

我们可以改进快速排序算法来解决这个问题：在分解的过程当中，我们会对子数组进行划分，如果划分得到的 q 正好就是我们需要的下标，就直接返回 a[q]；否则，如果 q 比目标下标小，就递归右子区间，否则递归左子区间。这样就可以把原来递归两个区间变成只递归一个区间，提高了时间效率。这就是「快速选择」算法。

我们知道快速排序的性能和「划分」出的子数组的长度密切相关。直观地理解如果每次规模为 n 的问题我们都划分成 1 和 n−1，每次递归的时候又向 n−1 的集合中递归，这种情况是最坏的，时间代价是 O(n ^ 2)。我们可以引入随机化来加速这个过程，它的时间代价的期望是 O(n)。需要注意的是，这个时间复杂度只有在 随机数据 下才成立，而对于精心构造的数据则可能表现不佳。因此我们这里并没有真正地使用随机数，而是使用双指针的方法，这种方法能够较好地应对各种数据。

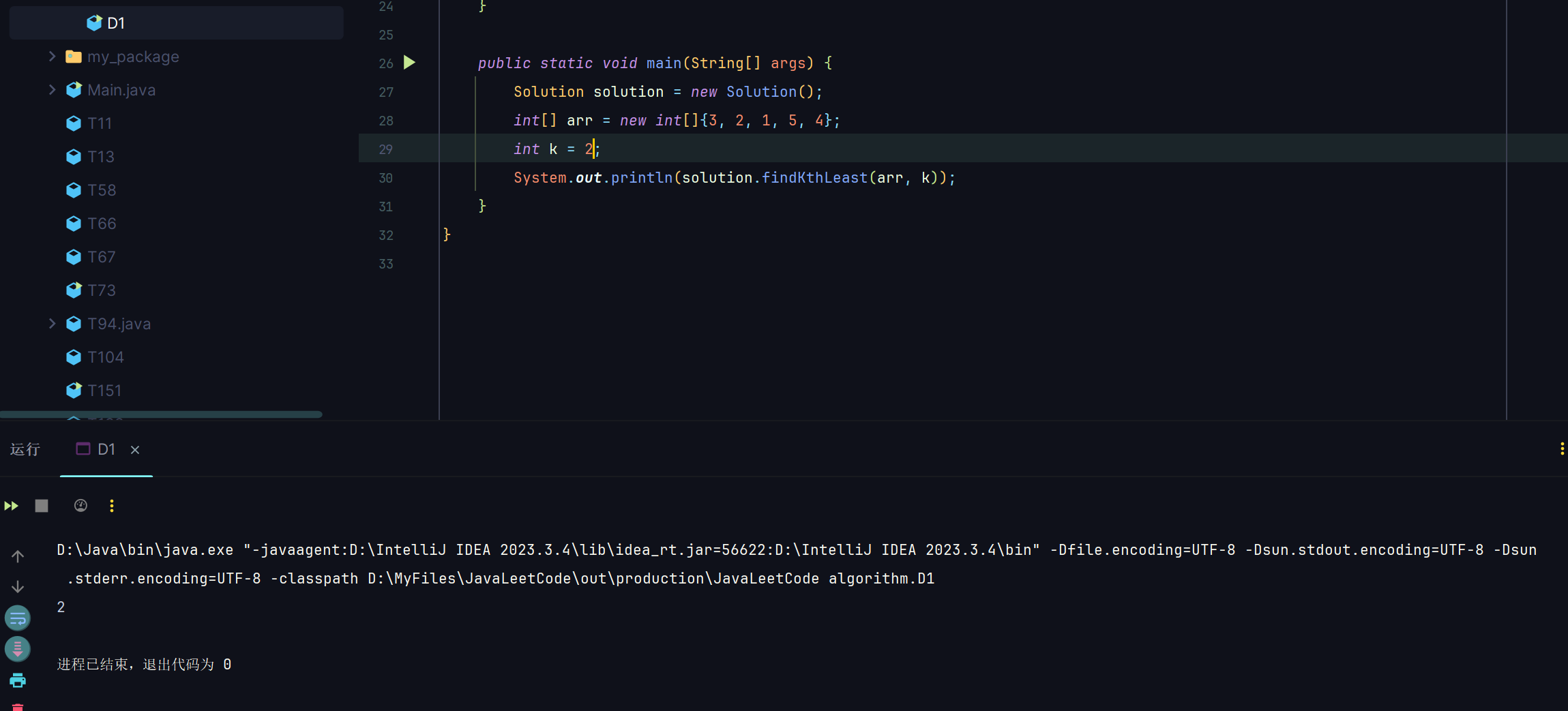
复杂度分析：时间复杂度：O(n)，如上文所述。

空间复杂度：O(log n)，递归使用栈空间的空间代价的期望为 O(log n)。

【代码】

package algorithm;  
  
public class D1 {  
 static class Solution {  
 int quickselect(int[] nums, int l, int r, int k) {  
 if (l == r) return nums[k - 1];  
 int x = nums[l], i = l - 1, j = r + 1;  
 while (i < j) {  
 do i++; while (nums[i] < x);  
 do j--; while (nums[j] > x);  
 if (i < j){  
 int tmp = nums[i];  
 nums[i] = nums[j];  
 nums[j] = tmp;  
 }  
 }  
 if (k <= j) return quickselect(nums, l, j, k);  
 else return quickselect(nums, j + 1, r, k);  
 }  
 public int findKthLeast(int[] \_nums, int k) {  
 int n = \_nums.length;  
 return quickselect(\_nums, 0, n - 1, k);  
 }  
 }  
  
 public static void main(String[] args) {  
 Solution solution = new Solution();  
 int[] arr = new int[]{9,7, 2, 5, 8, 3};  
 int k = 4;  
 System.out.println(solution.findKthLeast(arr, k));  
 }  
}

【结果验证】1、输入：nums = [3, 2, 1, 5, 4] , k=2



1. 输入: nums =[9, 7, 2, 5, 8, 3] , k=4

