学习情况表

|  |  |  |  |
| --- | --- | --- | --- |
| **姓名** | 李煦 | **学号** | 2021902007 |
| **学院** | 信息工程学院 | **专业** | 电子信息类 |

（请在下面表格本周学习情况）

|  |
| --- |
| 学习情况简述 |
| 1. **学习内容** 2. **快速排序的思想和过程** 3. **快速排序的第一版实现和其存在的问题** 4. **深入学习快速排序法** 5. **为快速排序法添加随机化** 6. **使用插入排序法优化快速排序法** 7. **双路快速排序法和三路快速排序法** 8. **学习收获（思考）** 9. **了解快速排序法的思想** 10. **快速排序法的代码实现** 11. **快速排序法的优化思路** 12. **两路快排和三路快排的思想及实现** |
| 本周练习过的代码（例） |
| （本周练习写过的代码，如实填写，疑似抄袭会单独测试）  *代码1：*  *#include<stdio.h>*  *int main()*  *{*  *printf("HelloWorld\n");*  *return 0;*  *}*  *代码2：*  …….. |
| 本周练习过的代码 |
| import java.util.Arrays;  public class Demo912\_3 {     public static void main(String[] args) {         int[] nums = {-1, 2, -8, -10};                   int[] after = sortArray(nums);               System.out.println(Arrays.toString(after));      }      private static int[] sortArray(int[] nums) {         int len = nums.length;           quickSort(nums, 0, len - 1);          return nums;         }     private static void quickSort(int[] nums, int left, int right) {         if (left >= right)              return;         }         int pivotIndex = partition(nums, left, right);          quickSort(nums, left, pivotIndex - 1);         quickSort(nums, pivotIndex + 1, right);     }     private static int partition(int[] nums, int left, int right) {         int pivot = nums[left];          int j = left;         for (int i = left + 1; i <= right; i++) {             if (nums[i] <= pivot) {                 j++;                   swap(nums, i, j);             }         }         swap(nums, left, j);         return j;     }      private static void swap(int[] nums, int index1, int index2) {         int temp = nums[index1];         nums[index1] = nums[index2];         nums[index2] = temp;     } } import java.util.Arrays; import java.util.Random;  public class Demo912\_4 {     public static void main(String[] args) {         int[] nums = {-1, 2, -8, -10};                  int[] after = sortArray(nums);                System.out.println(Arrays.toString(after));      }      private static int[] sortArray(int[] nums) {         int len = nums.length;           quickSort(nums, 0, len - 1);          return nums;         }          private static void quickSort(int[] nums, int left, int right) {         if (left >= right) {               return;         }         int pivotIndex = partition(nums, left, right);           quickSort(nums, left, pivotIndex - 1);         quickSort(nums, pivotIndex + 1, right);     }      private static int partition(int[] nums, int left, int right) {         int randomIndex = left + random.nextInt(right - left + 1);          swap(nums,randomIndex,left);          int pivot = nums[left];             int i = left + 1;         int j = right;         while (true) {             while (i <=j && nums[i] < pivot){                 i ++;             }             while (j <= right && nums[j] > pivot) {                 j --;             }             if (i >= j){                 break;             }             swap(nums,i,j);             i++;             j--;         }                  swap(nums, left, j);         return j;     }        private  static Random random = new Random(System.currentTimeMillis());      private static void swap(int[] nums, int index1, int index2) {         int temp = nums[index1];         nums[index1] = nums[index2];         nums[index2] = temp;     }  import java.util.Arrays; import java.util.Random;  public class Demo912\_5 {     public static void main(String[] args) {         int[] nums = {-1, 2, -8, -10};                   int[] after = sortArray(nums);               System.out.println(Arrays.toString(after));      }      private static int[] sortArray(int[] nums) {         int len = nums.length;           quickSort(nums, 0, len - 1);          return nums;         }       private static void quickSort(int[] nums, int left, int right) {         if (left >= right) {             return;         }               int randomIndex = left + random.nextInt(right - left + 1);          swap(nums, randomIndex, left);          int pivot = nums[left];         int i = left;         int j = right;         int k = left + 1;         while (k <= j) {             if (nums[k] < pivot) {                 i++;                 swap(nums, i, k);                 k++;             } else if (nums[k] == pivot) {                 k++;             } else {                 swap(nums, k, j);                 j--;             }         }             swap(nums, left, i);                  quickSort(nums, left, i - 1);         quickSort(nums, j + 1, right);     }      private static Random random = new Random(System.currentTimeMillis());     private static void swap(int[] nums, int index1, int index2) {         int temp = nums[index1];         nums[index1] = nums[index2];         nums[index2] = temp;     } } |

1. 该表信息将会作为你报名申请的重要依据，请认真仔细填写。
2. 培训班有严格的制度，请认真阅读规则并结合自身情况填写该表。
3. 完成该表填写后以“xxx学习情况”命名，及时上传作业。

最后希望大家能够加入我们，一起努力，共同进步！