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. **归并思想和过程** 3. **实现归并算法** 4. **归并算法的原理剖析** 5. **跟踪归并排序法的调试** 6. **归并排序法的复杂度** 7. **归并排序的一些事例** 8. **学习收获（思考）** 9. **了解归并排序法的思想** 10. **归并排序法的代码实现** 11. **掌握了归并排序的原理** |
| 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 |
| package Sort; public class { public static void main(String[] args) { int[] arr = {1,4,7,8,3,6,9}; merge(arr); } static void merge(int[] arr) { int mid = arr.length / 2; int[] temp = new int[arr.length];  int i = 0; int j = mid+1;  int k = 0;   while (i <= mid && j < arr.length) { if (arr[i] <= arr[j]) { temp[k] = arr[i]; i++; k++; } else { temp[k] = arr[j]; j++; k++; } } while (i <= mid) { temp[k++] = arr[i++]; } while (j < arr.length) { temp[k++] = arr[j++];  } print(temp); } static void swap(int[] arr,int i,int j) { int temp = arr[i]; arr[i] = arr[j]; arr[j] = temp; } static void print(int[] arr) { for (int i = 0; i < arr.length; i++) { System.out.print(arr[i] + " "); } } }  package Sort;   public class   { public static void main(String[] args) { // int[] arr = {1,4,7,8,3,6,9}; int[] arr = {3,0,6,4,1,3,7,8,5,9}; sort(arr,0,arr.length-1); print(arr); }  public static void sort(int[] arr,int left,int right) { if (left == right) { return; } int mid = left +(right-left)/2; sort(arr, left, mid); sort(arr, mid +1, right);  merge(arr, left, mid+1, right); }  static void merge(int[] arr,int leftPtr,int rightPtr,int rightBound) { int mid = rightPtr - 1;  int[] temp = new int[rightBound - leftPtr + 1];  int i = leftPtr;  int j = rightPtr;  int k = 0;   while (i <= mid && j <= rightBound) { if (arr[i] <= arr[j]) { temp[k] = arr[i]; i++; k++; } else { temp[k] = arr[j]; j++; k++; } } while (i <= mid) { temp[k++] = arr[i++]; } while (j <= rightBound) { temp[k++] = arr[j++];  }  for(int m = 0; m < temp.length;m++) arr[leftPtr+m] = temp[m]; } static void swap(int[] arr,int i,int j) { int temp = arr[i]; arr[i] = arr[j]; arr[j] = temp; }  static void print(int[] arr) { for (int i = 0; i < arr.length; i++) { System.out.print(arr[i] + " "); } } }   public static void merge(Comparable[] arr, int lo, int mid, int hi) {                int i = lo;         int j = mid + 1;         Comparable[] aux = new Comparable[arr.length];         for (int k = lo; k <= hi; k++) {             aux[k] = arr[k];         }         for (int k = lo; k <= hi; k++) {             if (i > mid) {                 arr[k] = aux[j++];             } else if (j > hi) {                 arr[k] = aux[i++];             } else if (less(aux[j], aux[i])) {                 arr[k] = aux[j++];             } else {                 arr[k] = aux[i++];             }         }     }      private static boolean less(Comparable v, Comparable w) {         return      }  class Solution {     public int reversePairs(int[] nums) {         if (nums == null || nums.length < 2) {             return 0;         }         int[] tempArr = new int[nums.length];         return reversePairs(nums, 0, nums.length-1, tempArr);     }       public int reversePairs(int[] nums, int left, int right, int[] tempArr) {                  if (left == right) {             return 0;         }                  for (int i=left; i<=right; i++) {             tempArr[i] = nums[i];         }                int mid = left + (right-left)/2;         int leftCount = reversePairs(nums, left, mid, tempArr);         int rightCount = reversePairs(nums, mid+1, right, tempArr);                 if (nums[mid] <= nums[mid+1]) {             return leftCount + rightCount;         }                int crossCount = crossMerge(nums, left, mid, right, tempArr);         return leftCount + rightCount + crossCount;     }       public int crossMerge(int[] nums, int left, int mid, int right, int[] tempArr) {         for (int i=left; i<=right; i++) {             tempArr[i] = nums[i];         }         int i = left, j = mid+1, count = 0;         for (int k=left; k<= right; k++) {             if (i == mid+1) {                 nums[k] = tempArr[j];                 j++;             } else if (j == right+1) {                 nums[k] = tempArr[i];                 i++;             } else if (tempArr[i] <= tempArr[j]) {                 nums[k] = tempArr[i];                 i++;             } else {                 nums[k] = tempArr[j];                 j++;                 count += (mid - i + 1);             }         }         return count;     } }  import java.util.\*;  public class AntiOrder { public int count(int[] A,int n) { if (A == null || n == 0) { return 0; } return mergeSortRecursion(A,0,n-1); }  public static int mergeSortRecursion(int[] arr,int l, int r) { if (l == r) { return 0; } int mid = (l + r) / 2; return mergeSortRecursion(arr,l,mid) + mergeSortRecursion(arr,mid+1,r) + merge(arr,l,mid,r); }  public static int merge(int[] arr, int left, int mid, int right) { int[] temp = new int[right - left + 1]; int index = 0; int i = left; int j = mid + 1; int inverseNum = 0;  while (i <= mid && j <= right) { if (arr[i] <= arr[j]) { temp[index++] = arr[i++]; } else { inverseNum += (mid - i + 1); temp[index++] = arr[j++]; } } while (i <= mid) { temp[index++] = arr[i++]; } while (j <= right) { temp[index++] = arr[j++]; } for (int k = 0; k < temp.length; k++) { arr[left++] = temp[k]; } return inverseNum； } } |

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！