**华东交通大学信息工程学院**

**上机实验报告册**

**专 业 计算机科学与技术**

**班 级 计算机2020-4**

**姓 名 惠佳乐**

**学 号 2020151002000107**

**课程名称 算法设计与分析**

**教 师 李广丽**

**学 期 2022—2023学年第一学期**

**2022年9月30日**

|  |
| --- |
| 1. **实验目的**   1．加深学生对算法设计方法的基本思想、基本步骤、基本方法的理解与掌握；  2．提高学生利用课堂所学知识解决实际问题的能力；  3．提高学生综合应用所学知识解决实际问题的能力。  **二、实验任务**  1、**排序算法**  **目前已知有几十种排序算法，请查找资料，并尽可能多地实现多种排序算法（至少实现8种）并分析算法的时间复杂度。比较各种算法的优劣。**  2、**三壶谜题：**  有一个充满水的8品脱的水壶和两个空水壶（容积分别是5品脱和3品脱）。通过将水壶完全倒满水和将水壶的水完全倒空这两种方式，在其中的一个水壶中得到4品脱的水。  3、**交替放置的碟子**  我们有数量为2n的一排碟子，n黑n白交替放置：黑，白，黑，白…  现在要把黑碟子都放在右边，白碟子都放在左边，但只允许通过互换相邻碟子的位置来实现。为该谜题写个算法，并确定该算法需要执行的换位次数。  **4、带锁的门：**  在走廊上有n个带锁的门，从1到n依次编号。最初所有的门都是关着的。我们从门前经过n次，每次都从1号门开始。在第i次经过时（i = 1,2，..., n)我们改变i的整数倍号锁的状态；如果门是关的，就打开它；如果门是打开的，就关上它。在最后一次经过后，哪些门是打开的，哪些门是关上的？有多少打开的门？ |
| 1. **实验设备及编程开发工具**   **实验设备：win10电脑**  **开发工具：IDEA** |
| **四、实验过程设计**（算法思路及描述，代码设计）  一丶排序算法   1. 希尔排序   public class ceshi {  public static void main(String[] args) {  int[] arr=new int[3000];  for (int i = 0; i < 3000; i++) {  arr[i]=(int)(5000\*Math.*random*());  }  for (int d = arr.length / 2;d>0;d /= 2){  for (int i = d; i < arr.length; i++){  for (int j = i-d; j>=0; j-=d){  if (arr[j] > arr[j + d]) {  int temp = arr[j];  arr[j] = arr[j + d];  arr[j + d] = temp;  }  }  }  }  for (int i = 0; i < 3000; i++) {  System.*out*.println(arr[i]);  }  } }   1. 插入排序   package com.hjl;  */\*\*  \** ***@author*** *惠佳乐  \*/* public class charu {  public static void main(String[] args) {  int[] arr=new int[3000];  for (int i = 0; i < 3000; i++) {  arr[i]=(int)(5000\*Math.*random*());  }  for (int i = 0; i < arr.length - 1; i++) {  for (int j = 0; j < i+1; j++) {  if (arr[i+1]>arr[j]){  int temp=0;  temp=arr[i+1];  arr[i+1]=arr[j];  arr[j]=temp;  }  }  }  for (int i = 0; i < arr.length; i++) {  System.*out*.println(arr[i]);  }  } }   1. 归并排序   import java.util.Arrays;  import static java.util.Arrays.*sort*;  */\*\*  \** ***@author*** *惠佳乐  \*/* public class guibing {  public static void main(String[] args) throws Exception {  int[] arr = new int[3000];  for (int i = 0; i < 3000; i++) {  arr[i] = (int) (5000 \* Math.*random*());  }  arr=*sort*(arr);  for (int i = 0; i < 3000; i++) {  System.*out*.println(arr[i]);  }  }  public static int[] sort(int[] sourceArray) throws Exception {  // 对 arr 进行拷贝，不改变参数内容  int[] arr = Arrays.*copyOf*(sourceArray, sourceArray.length);   if (arr.length < 2) {  return arr;  }  int middle = (int) Math.*floor*(arr.length / 2);   int[] left = Arrays.*copyOfRange*(arr, 0, middle);  int[] right = Arrays.*copyOfRange*(arr, middle, arr.length);   return *merge*(*sort*(left), *sort*(right));  }  public static int[] merge(int[] left, int[] right){  int[] result = new int[left.length + right.length];  int i = 0;  while (left.length > 0 && right.length > 0) {  if (left[0] <= right[0]) {  result[i++] = left[0];  left = Arrays.*copyOfRange*(left, 1, left.length);  } else {  result[i++] = right[0];  right = Arrays.*copyOfRange*(right, 1, right.length);  }  }   while (left.length > 0) {  result[i++] = left[0];  left = Arrays.*copyOfRange*(left, 1, left.length);  }   while (right.length > 0) {  result[i++] = right[0];  right = Arrays.*copyOfRange*(right, 1, right.length);  }   return result;  }  }   1. 基数排序   package com.hjl;  import java.util.ArrayList;  */\*\*  \** ***@author*** *惠佳乐  \*/* public class jishu {  public static void main(String[] args) {  int[] arr=new int[3000];  for (int i = 0; i < 3000; i++) {  arr[i]=(int)(5000\*Math.*random*());  }  *jishusort*(arr);  *printArray*(arr);  }  public static void jishusort(int[] arr){  int max=*findmax*(arr);  for (int i = 0; i < max; i++) {  *bucketsort*(arr,i);  }  }  //每一位的排序  public static void bucketsort(int[] arr,int max){  int temp=(int)Math.*pow*(10,max);  ArrayList<ArrayList<Integer>> buckets=new ArrayList();//每一个元素还是一个arraylist  for (int i = 0; i < 10; i++) {  buckets.add(new ArrayList<Integer>());  }  for (int i :arr) {  int m=i/temp%10;  buckets.get(m).add(i);  }  int suoyin=0;  for (ArrayList<Integer> bucket :buckets) {  for (int i :bucket) {  arr[suoyin++] = i;  }  }  }  //寻找最大位有几位  public static int findmax(int[] arr){  int max=1;  int temp=10;  for (int i :arr) {  while(i>temp){  max++;  temp\*=10;  }  }  return max;  }  public static void printArray(int[] arr) {  for (int i = 0; i < arr.length; i++) {  System.*out*.print(arr[i] + " ");  }  } }   1. 冒泡排序   package com.hjl;  */\*\*  \** ***@author*** *惠佳乐  \*/* public class maopao {  public static void main(String[] args) {  double[] arr=new double[200];  for (int i = 0; i < 200; i++) {  arr[i]=5000\*(Math.*random*());  }  for (int i = 0; i < arr.length-1; i++) {  boolean temp=true;  for (int j = 0; j < arr.length-i-1; j++) {  if (arr[j+1]<arr[j]){  double h=arr[j];  arr[j]=arr[j+1];  arr[j+1]=h;  temp=false;  }  }  if (temp){  break;  }  }  for (int i = 0; i < 200; i++) {  System.*out*.println(arr[i]);  }  } }   1. 快速排序   package com.hjl;  */\*\*  \* 正确分析的  \** ***@author*** *惠佳乐  \*/* public class Quick {  public static void main(String[] args) {  int[] arr=new int[3000];  for (int i = 0; i < 3000; i++) {  arr[i]=(int)(5000\*Math.*random*());  }  arr=*quick*(arr,0, arr.length-1);  for (int i = 0; i < 3000; i++) {  System.*out*.println(arr[i]);  }  }   public static int[] quick(int[] arr,int left,int right){  if (left<right){  int p=*zhongjian*(arr,left,right);  //递归思想  *quick*(arr,left,p-1);  *quick*(arr,p+1,right);  }  return arr;  }  public static int zhongjian(int[] arr,int left,int right){  int pivot=left;//找一个值，开头左边的为这个值  //这个是向后遍历的指针  int index=left+1;  //找到比基准值小的就放到index位置  for (int i=index; i<=right ;i++) {  if (arr[i]<arr[pivot]){  *transfer*(arr,i,index);  index++;  }  }  //最后index的位置是在找到所有比基准值小的数后面一位，所以index-1为比基准值小的数的组合最后一位索引  //然后将他与基准值换位  *transfer*(arr,pivot,index-1);  return index-1;  }  //换位方法  public static void transfer(int[] arr,int i,int index){  int temp=0;  temp=arr[i];  arr[i]=arr[index];  arr[index]=arr[i];  } }   1. 随即快速排序   package com.hjl; import java.util.Random;  */\*\*  \** ***@author*** *惠佳乐  \*/* public class suijiquick {   public static void main(String[] args) {  int[] arr=new int[3000];  for (int i = 0; i < 3000; i++) {  arr[i]=(int)(5000\*Math.*random*());  }  arr=*quick*(arr,0, arr.length-1);  for (int i = 0; i < 3000; i++) {  System.*out*.println(arr[i]);  }  }   public static int[] quick(int[] arr,int left,int right){  if (left<right){  int p=*zhongjian*(arr,left,right);  //递归思想  *quick*(arr,left,p-1);  *quick*(arr,p+1,right);  }  return arr;  }  public static int[] quick2(int[] arr,int left,int right){  if (left<right){  int q=*zhongjian2*(arr,left,right);  *quick2*(arr,left,q-1);  *quick2*(arr,q+1,right);  }  return arr;  }  public static int zhongjian2(int[]arr ,int left,int right){  //int r=random.nextInt(right-left)+left;  //transfer(arr,left,r);  return *zhongjian*(arr,left,right);  }  public static int zhongjian(int[] arr,int left,int right){  int pivot=left;//找一个值，开头左边的为这个值  //这个是向后遍历的指针  int index=left+1;  //找到比基准值小的就放到index位置  for (int i=index; i<=right ;i++) {  if (arr[i]<arr[pivot]){  *transfer*(arr,i,index);  index++;  }  }  //最后index的位置是在找到所有比基准值小的数后面一位，所以index-1为比基准值小的数的组合最后一位索引  //然后将他与基准值换位  *transfer*(arr,pivot,index-1);  return index-1;  }  //换位方法  public static void transfer(int[] arr,int i,int index){  int temp=0;  temp=arr[i];  arr[i]=arr[index];  arr[index]=arr[i];  } }   1. 选择排序   package com.hjl;  */\*\*  \** ***@author*** *惠佳乐  \*/* public class xuanze {  public static void main(String[] args) {  double[] arr=new double[200];  for (int i = 0; i < 200; i++) {  arr[i]=5000\*(Math.*random*());  }  for (int i = 0; i < arr.length-1; i++) {  for (int j = 0; j < arr.length-i-1; j++) {  if (arr[i]>arr[i+j+1]){  double temp=0;  temp=arr[i];  arr[i]=arr[i+j+1];  arr[i+j+1]=temp;  }  }  }  for (int i = 0; i < 200; i++) {  System.*out*.println(arr[i]);  }  } }  二丶三壶谜题  package com.qqq;  import java.util.ArrayList; import java.util.List; import java.util.Map; import java.util.Scanner;  */\*\*  \** ***@author*** *惠佳乐  \*/* public class ThreePotProblem {  public static void main(String[] args) {   Scanner sc = new Scanner(System.*in*);  System.*out*.println("请输入三个值表示容器的容水量：");  int x1 = sc.nextInt();  int x2 = sc.nextInt();  int x3 = sc.nextInt();   String string = x1+""+x2+""+x3;  System.*out*.println(string);   ContainerPot containerPot = new ContainerPot(8, 0, 0);  List<String> list = new ArrayList<String>();  list.add(string);  containerPot.init();  String result = "";  int n = 0;//状态计数  do {  //遍历倒水  for (int i = 0; i < 3 && containerPot.flag; i++) {  for (int j = 0; j < 3 && containerPot.flag; j++) {  if (i != j) { //不能让壶相同  if (containerPot.canPour(i, j)) { //是否可以倒水  String str = containerPot.getString();//保存初始状态  //倒入水  containerPot.pour(i, j);  String strResult = containerPot.getString();  if (!list.contains(strResult)) {  list.add(strResult);  containerPot.addList(str, strResult);  }  //如果出现容量为4的壶就停止  if (containerPot.pot[0] == 4 || containerPot.pot[1] == 4 || containerPot.pot[2] == 4) {  result = result + containerPot.pot[0] + containerPot.pot[1] + containerPot.pot[2];  containerPot.flag = false;  break;  }  containerPot.intPot(str);  }  }  }  }  n++;  containerPot.intPot(list.get(n));  } while (containerPot.flag);  System.*out*.println(list);  System.*out*.println(containerPot.listMap);  String key = result;  System.*out*.print(key + "<--");  do {  for (Map.Entry<String, List<String>> entry : containerPot.listMap.entrySet()) {  for (int i = 0; i < entry.getValue().size(); i++) {  if (entry.getValue().get(i).equals(key)) {  System.*out*.print(entry.getKey() + "<--");  key = entry.getKey();  break;  }  }   }  } while (containerPot.listMap.containsKey(key) && !key.equals(string));   System.*out*.println();  } }  boolean flag = true;   public ContainerPot(int a, int b, int c) {  pot[0] = a;  pot[1] = b;  pot[2] = c;  }   public void init() {  maxP[0] = 8;  maxP[1] = 5;  maxP[2] = 3;  }   //将key值作为它的上一步  public void addList(String key, String str) {  if (listMap.isEmpty() || !listMap.containsKey(key)) {  List<String> temp = new ArrayList<String>();  temp.add(str);  listMap.put(key, temp);  } else { // System.out.println(listMap.containsKey(key)); // System.out.println(listMap.get(key));  listMap.get(key).add(str);  }  }   //判读是否已经有这个key值在里面存放了  public boolean isAdd(String str) {  return listMap.containsKey(str);  }   //判读是否可以从from壶倒水到to壶  public boolean canPour(int from, int to) {  //如果没用水就不能倒了  if (pot[from] == 0) {  return false;  }   //如果里面已经有水了 就不能倒了  if (pot[to] == maxP[to]) {  return false;  } else {  return true;  }  }   //倒水的过程  public void pour(int from, int to) {  //做一个判读看看是否能剩水  if (pot[from] + pot[to] > maxP[to]) {  pot[from] = pot[from] - (maxP[to] - pot[to]);  pot[to] = maxP[to];  } else {  pot[to] = pot[to] + pot[from];  pot[from] = 0;  }  }   public String getString() {  String str = "";  for (int i = 0; i < pot.length; i++) {  str = str + pot[i];  }  return str;  }   public void intPot(String str) {  for (int i = 0; i < str.length(); i++) {  pot[i] = Integer.*parseInt*(String.*valueOf*(str.charAt(i)));  }  }   public void printDD() {  for (List<String> value : listMap.values()) {  System.*out*.println(value.toString());  }  } }  三丶交换放置的碟子  package com.hjl;  import java.util.Scanner;  */\*\*  \** ***@author*** *惠佳乐  \*/* public class swapdishes {  public static void main(String[] args) {  System.*out*.println("请输入碟子的数量：");  Scanner input=new Scanner(System.*in*);  int n= input.nextInt();  if (n%2==0) {  int[] arr = new int[n];  for (int i = 0; i < n; i++) {  if (i % 2 == 0) {  arr[i] = 1;  } else {  arr[i] = 0;  }  }  int h=0;  h=*sort*(arr);  System.*out*.println("n的值为："+h);  }else{  System.*out*.println("n的值不符合题意");  }  }  public static int sort(int[] arr){  int m=0;  int n=arr.length;  while (n!=0){  m+=n/2;  *mmm*(arr);  n-=2;  }  return m;  }  public static void mmm(int[] arr){  for (int i = 0; i <arr.length-1; i++) {  if (arr[i]!=arr[i+1]){  int temp=arr[i];  arr[i]=arr[i+1];  arr[i+1]=temp;  }  }  }  }  四丶带锁的门  package com.hjl;  import java.util.ArrayList; import java.util.Scanner;  */\*\*  \** ***@author*** *惠佳乐  \*/* public class qqq {  public static void main(String[] args) {   System.*out*.println("请输入带锁的门的个数：");  int n = new Scanner(System.*in*).nextInt();   //sum统计开门的个数  int sum = 0;  //open存储开着的门的编号  ArrayList<Integer> open = new ArrayList<>();   for (int i = 1; i \* i <= n; i++) {  open.add(i \* i);  sum++;  }   //close存储关闭的门的编号  ArrayList<Integer> close = new ArrayList<>();  for (int i = 1; i <= n; i++) {  if (!open.contains(i))  close.add(i);  }   System.*out*.println("打开的门的编号为：");  for (Integer integer : open) {  System.*out*.print(integer + " ");  }  System.*out*.println("\r\n");   System.*out*.println("关着的门的编号为：");  for (Integer integer : close) {  System.*out*.print(integer + " ");  }  System.*out*.println("\r\n");   System.*out*.println("打开的门的个数为：" + sum);  }  }     1. **实验结果及算法复杂度分析**   **一丶排序**  **希尔排序**  **时间复杂度：平均：o（nlogn）最好：（nlog²n）最坏：（nlog²n）**  IMG_256  **插入排序**  **时间复杂度：平均o（n²）最好o（n）最坏o（n²）**  IMG_256  **基数排序**  **时间复杂度:平均最好最坏都是o（n×k）**  IMG_256  **冒泡排序**  **时间复杂度：平均o（n²）最好o（n）最坏o（n²）**  IMG_256  **快速排序**  **时间复杂度：平均o（nlogn）最好o（nlogn）最坏o（n²）**  IMG_256  **选择排序**  **时间复杂度平均最好最坏都是o（n²）**  IMG_256  **归并排序**  **时间复杂度平均最好最坏都是o（nlogn）**  IMG_256  **二丶三壶谜题**  IMG_256  **三丶交换放置的碟子**  IMG_256  四丶带锁的门  IMG_256 |
| **六、实验小结**（包括问题和解决方法、心得体会等）  学会了使用递归，在交换碟子中学到了利用排序中的方法，还查了相关的Arraylist集合的使用方法，并且掌握，在编写代码的过程中扩展了自家的代码思维 |