|  |  |
| --- | --- |
| haut | 河南工业大学信息科学与工程学院 |

**《算法设计与分析》实验报告**

|  |  |
| --- | --- |
| 课程名称： | 算法设计与分析 |
| 学生姓名： | 刘文博 |
| 学生学号： | 201716040224 |
| 学生专业： | 软件工程1702 |

**实验一： 递归设计：排序**

苗建雨

2019 年 5 月 9 日

实验日期： 2019 年 5 月 9 日 班级： 软件工程1702

学号（后四位）：\_\_\_0224\_\_\_\_\_\_ 姓名：\_\_\_\_刘文博\_\_\_\_\_ 成绩：

**一、 实验目的：**

1. 掌握递归排序的思想和方法。

2. 掌握排序的算法实现。

**二、实验要求：**

1. 熟悉java语言的集成开发环境；

2. 通过两种利用分治算法求解的排序算法来加深对递归设计和分治算法的理解；

掌握递归算法的概念和基本思想，分析并掌握排列问题的递归算法。对于一个序列， 使用快速排序算法和归并排序算法对其实现排序。

**三、实验内容：**

**1.快速排序**

快速排序的基本思想是，对于输入的子数组a[p:r]按以下三个步骤进行排序： • 分解(divide)：以a[p]为基准元素将a[p:r]划分成2部分a[p:q-1],a[q+1:r]，其中前半部分的 所有元素均小于a[p]，而后半部分的所有元素均大于a[p]； • 递归求解(conquer)：通过递归调用分别对a[p:q-1]和a[q+1:r]进行排序； • 合并(merge)：由于前半部分和后半部分已经排好顺序了，因而不需要执行任何操作。

**代码实现：**

import java.util.ArrayList;

import java.util.Random;

public class QuickSort {

public static void main(String[] args) {

ArrayList<Integer> arr = new ArrayList<Integer>();

Random ran = new Random();

for(int i = 0 ; i < 20 ; ++i) {

arr.add(ran.nextInt(100));

}

quickSort(arr, 0, 19);

System.out.println("排序后：");

for (int i = 0; i < arr.size(); i++) {

System.out.print(+arr.get(i) + " " );

}

}

public static void quickSort(ArrayList<Integer> arr,int low,int high) {

int begin = low;

int end = high;

int temp = arr.get(low);

if(low <0 || high > arr.size()) {

return;

}

if(begin > end) {

return;

}

while(begin < end) {

while(end > begin && arr.get(end) >= temp) {

end--;

}

while(begin < end && arr.get(begin) <= temp) {

begin++;

}

if(begin < end) {

int num = arr.get(end);

arr.set(end, arr.get(begin));

arr.set(begin, num);

}

}

arr.set(low, arr.get(begin));

arr.set(begin, temp);

temp = end;

try {

quickSort(arr, low, temp - 1);

quickSort(arr, temp + 1, high);

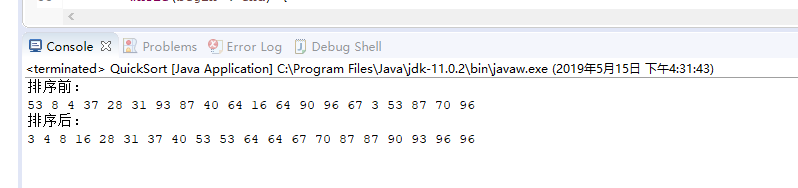
}catch (Exception e) {

}

}

}

**实验截图：**



**2 归并排序**

归并排序非常接近递归分治算法框架，有如下三个步骤： • 分解(divide)：将n个元素的序列划分成两个等长的序列； • 递归求解(conquer)：递归调用归并排序程序对两个子序列分别进行排序； • 合并(merge)：合并两个有序的子序列产生一个最终的有序序列。

**代码实现：**

import java.util.Random;

public class MergeSort {

public static void main(String[] args) {

mergeSort.test();

}

public static class mergeSort {

public static void merge(int []a,int left,int mid,int right){

int []tmp=new int[a.length];

int p1=left,p2=mid+1,k=left;

while(p1<=mid && p2<=right){

if(a[p1]<=a[p2])

tmp[k++]=a[p1++];

else

tmp[k++]=a[p2++];

}

while(p1<=mid) tmp[k++]=a[p1++];

while(p2<=right) tmp[k++]=a[p2++];

for (int i = left; i <=right; i++)

a[i]=tmp[i];

}

public static void mergeSort(int [] a,int start,int end){

if(start<end){

int mid=(start+end)/2;

mergeSort(a, start, mid);

mergeSort(a, mid+1, end);

merge(a, start, mid, end);

}

}

public static void test(){

int[] a = new int[20];

for (int i = 0; i < 20; i++) {

a[i] = new Random().nextInt(100);

}

mergeSort(a, 0, a.length-1);

System.out.println("排好序的数组：");

for (int e : a)

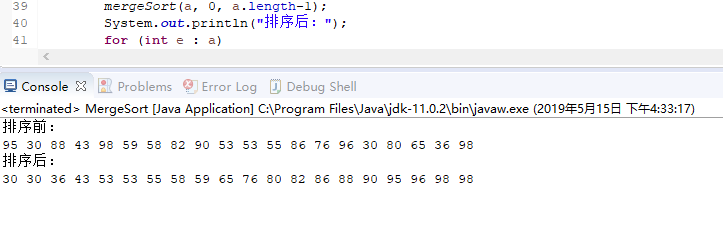
System.out.print(e+" ");

}

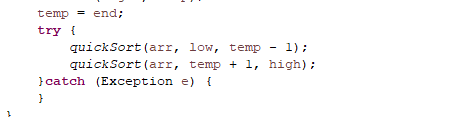
}

}

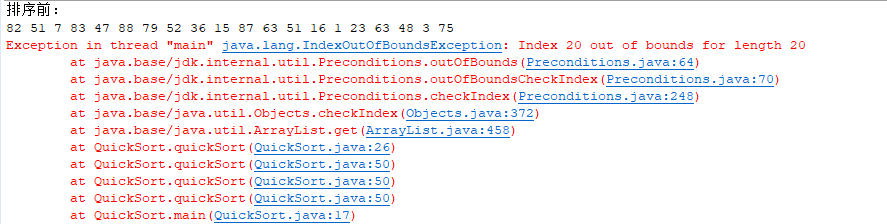
**实验截图：**

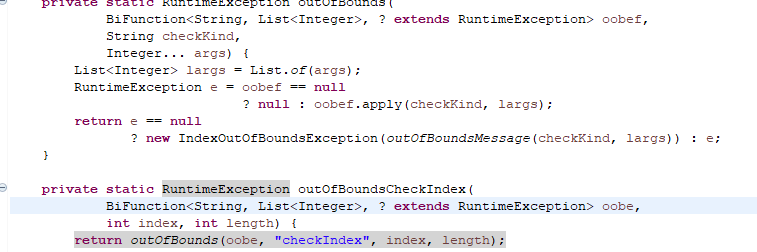


**四、实验结果&总结：**

本实验中的快速排序，使用的是动态数组，在递归调用的时候，出现了运行报错的情况

这段代码，之前没有进行try catch异常捕获，无法运行，





检查发现动态数组在检查下表的时候，最后一个元素访问会抛出异常，于是，进行了异常捕获，问题解决。