**新疆政法学院**

**数据结构与算法**

**实验报告**

班 级： 计算机科学与技术4班

学 号： 2124030160

姓 名： 孙 久 猛

指导老师： 张 家 琦

学 期： 2022-2023第一学期

|  |  |  |  |
| --- | --- | --- | --- |
| **实验题目** | 实验八：排序算法的理解与应用 | | |
| **实验时间** | **10:00-11:40** | **实验地点** | **东训0114** |
| **实验成绩** |  | **实验性质** | **□验证性 √设计性 □综合性** |
| 教师评价：  □算法/实验过程正确； □源程序/实验内容提交 □程序结构/实验步骤合理；  □实验结果正确； □语法、语义正确； □报告规范；  其他：  评价教师签名： | | | |
| 一、实验目的  1.理解排序在计算机领域及现实生活中的实际应用意义。  2.掌握冒泡排序的算法思想、程序执行过程。  3.掌握快速排序的算法思想、理解程序执行过程。 | | | |
| **二、实验项目内容（实验题目）**  所谓排序算法，即通过特定的算法因式将一组或多组数据按照既定模式进行重新排序。这种新序列遵循着一定的规则，体现出一定的规律，因此，经处理后的数据便于筛选和计算，大大提高了计算效率。现针对排序算法，做出以下两个实验内容：  （一）冒泡排序  冒泡排序是我们接触到的第一种排序方法，也是计算机学科比较重要的一类排序算法；针对该算法，要求大家理解并掌握冒泡排序的算法思想、程序执行过程。现有49，38，65，97，76，13，27，49共计8个数，请用冒泡排序的思想编写代码，实现冒泡排序。  （二）快速排序  快速排序作为计算机专业内比较优秀的排序算法，在排序领域内有着一定的比重。现有49，38，65，97，76，13，27，49共计8个数，请用快速排序的思想编写代码，实现冒泡排序。  **三、源程序（实验步骤/实验过程/算法）**  （一）冒泡排序  //Deom\_08\_1(bubble\_sort).c  #include <stdio.h>  void bubble\_sort(int arr[], int len) {  int i, j, temp;  for (i = 0; i < len - 1; i++) //change's times  for (j = 0; j < len - 1 - i; j++)//change's numbers  if (arr[j] > arr[j + 1]) {  temp = arr[j];  arr[j] = arr[j + 1];  arr[j + 1] = temp;  }  } // if (arr[j] > arr[j + 1]) change  int main() {  int arr[] = {49,38,65,97,76,13,27,49};  int len = (int) sizeof(arr) / sizeof(\*arr); //Get follow\_space of arr[].  bubble\_sort(arr, len); //into funtion(bubble\_sort()) to sort  int i;  for (i = 0; i < len; i++)  printf("%d ", arr[i]); //printf answer  return 0;  }  （二）快速排序  //Deom\_08\_2(Kuai\_sort).c  #include<stdio.h>  #define N 8  #define MAXSIZE 20 /\* 一个用作示例的小顺序表的最大长度 \*/  typedef int KeyType; /\* 定义关键字类型为整型 \*/  typedef int InfoType;  typedef struct  {  KeyType key; /\* 关键字项 \*/  InfoType otherinfo; /\* 其它数据项，具体类型在主程中定义 \*/  }RedType; /\* 记录类型 \*/  typedef struct  {  RedType r[MAXSIZE+1]; /\* r[0]闲置或用作哨兵单元 \*/  int length; /\* 顺序表长度 \*/  }SqList; /\* 顺序表类型 \*/  int Partition(SqList \*L,int low,int high)  { /\* 交换顺序表L中子表L.r[low..high]的记录，使枢轴记录到位， \*/  /\* 并返回其所在位置，此时在它之前(后)的记录均不大(小)于它。\*/  RedType t;  KeyType pivotkey;  pivotkey=(\*L).r[low].key; /\* 用子表的第一个记录作枢轴记录 \*/  while(low<high)  { /\* 从表的两端交替地向中间扫描 \*/  while(low<high&&(\*L).r[high].key>=pivotkey)  --high;  t=(\*L).r[low]; /\* 将比枢轴记录小的记录交换到低端 \*/  (\*L).r[low]=(\*L).r[high];  (\*L).r[high]=t;  while(low<high&&(\*L).r[low].key<=pivotkey)  ++low;  t=(\*L).r[low]; /\* 将比枢轴记录大的记录交换到高端 \*/  (\*L).r[low]=(\*L).r[high];  (\*L).r[high]=t;  }  return low; /\* 返回枢轴所在位置 \*/  }  void QSort(SqList \*L,int low,int high)  { /\* 对顺序表L中的子序列L.r[low..high]作快速排序。\*/  int pivotloc;  if(low<high)  { /\* 长度大于1 \*/  pivotloc=Partition(L,low,high); /\* 将L.r[low..high]一分为二 \*/  QSort(L,low,pivotloc-1); /\* 对低子表递归排序，pivotloc是枢轴位置 \*/  QSort(L,pivotloc+1,high); /\* 对高子表递归排序 \*/  }  }  void QuickSort(SqList \*L)  { /\* 对顺序表L作快速排序。\*/  QSort(L,1,(\*L).length);  }  void print(SqList L)  {  int i;  for(i=1;i<=L.length;i++)  printf("(%d,%d)",L.r[i].key,L.r[i].otherinfo);  printf("\n");  }  void main()  {  RedType d[N]={{49,1},{38,2},{65,3},{97,4},{76,5},{13,6},{27,7},{49,8}};  SqList l;  int i;  for(i=0;i<N;i++)  l.r[i+1]=d[i];  l.length=N;  printf("排序前:\n");  print(l);  QuickSort(&l);  printf("排序后:\n");  print(l);  }  **四、运行结果**  //Deom\_08\_1(bubble\_sort).exe      //Deom\_08\_2(Kuai\_sort).c | | | |

日 期： 2022年11月9日