**《数据结构综合设计》实验报告**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 学院 | 人工智能与大数据学院 | 专业 | 虚拟现实技术 | 班级 | 21级2班 | 学生姓名 | 刘欢 |
| 实验  周次 | 13-14 | 实验  日期 | 2023.5.26 | 学时 | 6 | 教师姓名 | 李昊康 |
| 项目名称 | | 排序的应用 | | | | | |
| 实验  类别 | 🗹验证型实验 🞎设计型实验 🞎综合型实验 🞎其它 | | | | | 成绩：92 | |
| 实验目的及具体要求  实验目的：  1.实现多种类型的排序算法（插入排序、交换排序、选择排序、归并排序等）；  2.理解排序过程；  3.计算比较次数和移动次数，对比分析算法性能的优劣与适用场景；  具体要求：  编写程序实现插入排序、希尔排序、冒泡排序、快速排序、简单选择排序、堆排序。中任意3种。  实验仪器、设备和材料  硬设备：PC机  软件环境：Windows VS2019  实验内容、步骤及实验数据记录  1.插入排序  #include<stdio.h>  #define MaxSize 100//可输入100个字符  void InsertionSort(int a[], int N) {  int tmp = 0, p, j = 0;  /\*p控制趟数，j控制移动的次数\*/  for (p = 1; p < N; p++) {  /\*需要排N-1趟\*/  tmp = a[p];  for (j = p; j > 0 && a[j - 1] > tmp; j--) {  /\*控制移动的次数，只要前面的数大于位置p的数，就往右移\*/  a[j] = a[j - 1];  }  a[j] = tmp;/\*跳出循环，合适的位置已找到，将位置p的值插入\*/  }  }  int main() {  int a[MaxSize];  int num, i;  printf("请输入要排序的个数:");  scanf\_s("%d", &num);  printf("请输入要排序的数据\n");  for (i = 0; i < num; i++) {  scanf\_s("%d", &a[i]);  }  InsertionSort(a, num);  printf("排序后的序列\n");  for (i = 0; i < num; i++) {  printf("%d\t", a[i]);  }  return 0;  001}  2.冒泡排序  #include <iostream>  #include <string>  using namespace std;  int main() {  int n[9];  int i, j;  int temp; //用于缓存需要交换的数字  cout << "请输入九个数字！" << endl;  for (i = 0; i < 9; i++) {  cin >> n[i];  }  for (i = 0; i <8; i++) { //共进行9步  for (j = 0; j < 8- i; j++) { //在每一步进行10-i次两两比较  if (n[j] > n[j + 1]) {  temp = n[j];  n[j] = n[j + 1];  n[j + 1] = temp;  }  }  }  cout << "排序后的数据是：" << endl;  for (i = 0; i < 9; i++)  {  cout << n[i] << ' ';  }  cout << endl;  system("pause");  return 0;  }  002  3.快速排序  #include <stdio.h>  #include <iostream>  #include <math.h>  #include <algorithm>  using namespace std;  int part(int\* r, int low, int hight) //划分函数  {  int i = low, j = hight, pivot = r[low]; //基准元素  while (i < j)  {  while (i<j && r[j]>pivot) //从右向左开始找一个 小于等于 pivot的数值  {  j--;  }  if (i < j)  {  swap(r[i++], r[j]); //r[i]和r[j]交换后 i 向右移动一位  }  while (i < j && r[i] <= pivot) //从左向右开始找一个 大于 pivot的数值  {  i++;  }  if (i < j)  {  swap(r[i], r[j--]); //r[i]和r[j]交换后 i 向左移动一位  }  }  return i; //返回最终划分完成后基准元素所在的位置  }  void Quicksort(int\* r, int low, int hight)  {  int mid;  if (low < hight)  {  mid = part(r, low, hight); // 返回基准元素位置  Quicksort(r, low, mid - 1); // 左区间递归快速排序  Quicksort(r, mid + 1, hight); // 右区间递归快速排序  }  }  int main()  {  int a[10001];  int N;  cout << "请输入要排序的数据的个数： " << endl;  cin >> N;  cout << "请输入要排序的数据： " << endl;  for (int i = 0; i < N; i++)  {  cin >> a[i];  }  cout << endl;  Quicksort(a, 0, N - 1);  cout << "排序后的序列为： " << endl;  for (int i = 0; i < N; i++)  {  cout << a[i] << " ";  }  cout << endl;  return 0;  }  003  分析：  1.**插入排序**：  时间复杂度：最好情况就是全部有序，此时只需遍历一次，最好的时间复杂度为O(n)  最坏情况全部反序，内层每次遍历已排序部分，最坏时间复杂度为O(n²)  综上，因此直接插入排序的平均时间复杂度为O(n²)。  空间复杂度：该算法只使用了常数量级的辅助空间，故空间复杂度为O(1)  优点：稳定，相对于冒泡排序与选择排序更快，在数据相对有序的情况下有较高的效率(只需遍历一次O(n))。  缺点：比较次数不一定，比较次数越少，插入点后的数据移动越多，特别是当数据总量大的时候，时间复杂度可能很高(全部反序O(n²)) 。  2.**冒泡排序**：  通过数去找位置，选择排序是给定位置去找数； 冒泡排序优缺点：优点:比较简单，空间复杂度较低，是稳定的； 缺点:时间复杂度太高，效率慢；  3.**快速排序**：  稳定性：快排是一种不稳定排序，比如基准值的前后都存在与基准值相同的元素，那么相同值就会被放在一边，这样就打乱了之前的相对顺序；比较性：因为排序时元素之间需要比较，所以是比较排序；时间复杂度：快排的时间复杂度为O (nlogn)；空间复杂度：排序时需要另外申请空间，并且随着数列规模增大而增大，其复杂度为：O (nlogn)；  缺点：对于小规模的数据集性能不是很好。  通过这节课的学习，我了解了许多不足，从实践中发现问题并解决问题，这节课，我收获颇多。 | | | | | | | |
|  | | | | | | | |

说明：1. 实验周次：填写实际上课周，如第5-8周上课填“5-8”或第10周上课填“10”。

1. 实验报告各部分内容需详实填写，按实验指导书上的评分标准给出分数。
2. 实验类型参考实验类型说明文件。