浙江水利水电学院

实验报告

（18-19学年 第2学期）

|  |  |
| --- | --- |
| 课程名称： | 数据结构与算法 |
| 班 级： | 软工S18-1 |
| 学 号： | 2018b31007 |
| 姓 名： | 曹愉 |
| 实验室（中心）名 称： | 软件实施分室(一) |
| 教学单位： | 信息工程与艺术设计学院 |

2019 年 6 月 12 日

实 验 名 称：排序 指导教师：寿焕君

实 验 日 期：2019.6.12 地 点：现北306

同组学生姓名：无

|  |
| --- |
| 实验内容及要求：  实验内容：   1. 对N个数进行排序并测试 2. 用直接插入排序、折半插入排序、希尔排序、冒泡排序、快速排序、简单选择排序、堆排序这7种算法进行排序 3. 比较各种排序算法的时间复杂度和空间复杂度 4. 尝试算法的改进   实验要求：   1. 掌握线性表的存储方式，并在此基础上实现典型的排序算法 2. 理解并掌握内部排序中几种常用算法的性能和适用场合 3. 理解并比较各种排序算法的时间复杂度和空间复杂度 |
| 主要仪器名称及型号：   1. 计算机：I5-2400/IntelH61/4G DDR3/500G SATA/HD7350 1GB显存独立显卡 2. 开发工具：Cfree |
| 实验过程：（可附页）  **任务1：排序比较（保存文件为：学号.c）**  对N个数（N大于2000），从直接插入排序、折半插入排序、希尔排序、冒泡排序、快速排序、简单选择排序、堆排序这7种算法中选取4种排序算法进行排序。  1．设计合适的数据结构存储这N个数。  2．对这N个数分别用下列三种情况进行测试：  （1）N个从小到大的有序整数  （2）N个从大到小的有序整数  （3）N个随机产生的无序数  3．比较各类排序算法的时间复杂度和空间复杂度。  4．对上述算法提出改进之处（创新思维）。  提示：rand()随机函数在stdlib.h文件中，在调用之前，先调用srand(time(0或NULL))函数生成一个随机种子（time()函数在time.h文件中）。  要求：源程序文件中要求对每个函数添加注释，说明函数的功能。  **任务2：成果提交**  实验报告保存文件为：**学号实验报告-排序.docx**，连同1个C源程序文件上传到<ftp://192.168.106.99>上。（不要打包） |
| 实验数据记录或图片：（可附页）   1. 选用的排序算法：  |  | | --- | | 4种：  直接插入排序  折半插入排序  冒泡排序  简单选择排序 |  1. 选用的数据结构：  |  | | --- | | typedef int ElemType;  typedef struct  {  ElemType key;  } node; |  1. 实验步骤或算法描述：  |  | | --- | | 用适当的形式表达实现步骤，如文字描述、程序流程图、类图、时序图等（选择一种排序算法进行描述）：  //直接插入排序  void stinsort(node r[MAXSIZE], int n)   //传入关键字列表  {      clock\_t start, stop;    //定义开始结束系统时钟      start = clock();        //开始解释      compare\_times[0] = 0;   //比较次数      move\_times[0] = 0;      //移动次数      runtime[0] = 0;         //运行时间      int i, j;                   //i为未排序 j为已排序列表      for (i = 2; i <= n; i++)    //共n-1趟排序      {          r[0] = r[i];            //r[0]为监视哨，也可以作为循环结束标志          j = i - 1;          compare\_times[0]++;     //增加比较次数          while (r[j].key > r[0].key)     //比较r[0]，即r[i]和r[j]          {              r[j + 1] = r[j];              j--;              move\_times[0]++;            //增加移动次数          }          r[j + 1] = r[0];                //将r[i]内容插入到r[j]后一位置      }      stop = clock();             //记录时间结束      //显示结果      /\*      for (int i = 1; i <= n; i++)      {          printf("%d,", r[i].key);      }      \*/      runtime[0] = (double)(stop - start) / CLOCKS\_PER\_SEC;  } |   4．实验结果：   |  | | --- | | 4种算法在三种情况测试下的实验结果（比较次数、移动次数和运行耗时）：  从小到大：    从大到小：    随机产生： |   5．算法性能分析：   |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | 根据实验结果，对4种算法进行比较分析，包括它们的时间复杂度和空间复杂度：   |  |  |  | | --- | --- | --- | | 排序方法 | 平均时间复杂度 | 空间复杂度 | | 直接插入排序 | O(n2) | O(1) | | 折半插入排序 | O(n2) | O(1) | | 冒泡排序 | O(n2) | O(1) | | 简单选择排序 | O(n2) | O(1) |   对于正序有序数据来说，冒泡排序、直接插入和折半插入比较好  冒泡排序在反序和随机数据情况下效率比较低  简单选择排序的比较次数较多，移动次数较少，在随机以及反序的数据排序来说效率占优。 |   6．总结   |  | | --- | | 调试过程中出现的问题及解决办法：  （1）监视哨导致的排序关键字定义数量不对，传参的n出问题  解决办法：改变定义关键字列表时的数据 循环从下标1而非0开始  node list[n + 1]; 定义的时候n+1  （2）循环排序，下一个排序函数调用已经排好的list  解决办法：每次调用排序算法的时候，重新创建关键字列表 | | 实验心得体会：  这次实验了解了各种排序算法的共同点，以及他们在应对不同关键字列表时效率的差异。所以在排序算法的选择上，我们要根据关键字类型以及顺序的不同，选择更为合适的排序算法才能提高算法的排序效率，尤其是大量数据的排序在选用算法不同的时候，他们排序所需的时间差别还是非常大的。  希望这种实验里的探究精神能更多的用到学习和以后的工作中，提升自己的综合能力。 | | 算法的改进意见：  减少全局变量的使用，局部变量的话就不用考虑共用性的问题，提高耦合性，提升算法效率 | |
| 教师评语：  该同学在本次实验过程中学习态度认真，具有较强的观察、分析和解决问题的实验能力，能按时完成本次实验的全部任务。经程序运行测试，实验结果正确。  实验完成情况：优；代码规范：优；实验报告表达能力：好。  成绩：优 批阅教师：寿焕君  2019年6月12日 |