**实 验9：内部排序算法比较 （4学时）**

**[问题描述]**

排序是计算机程序设计中一种重要操作，它的功能是将一个数据元素（或记录）的任意序列重新排列成一个按关键字有序的序列。本实验熟悉几种典型的排序方法，并对各种算法的特点、使用范围和效率进行进一步的了解。

**[实验目的]**

1. 深刻理解排序的定义和各类排序的算法思想，并能灵活应用。
2. 掌握各类排序的时间复杂度的分析方法，能从“关键字间的比较次数”分析算法的平均情况、最好情况和最坏情况。
3. 理解排序方法“稳定”和“不稳定”的含义。

**[实验内容及要求]**

1. 数据由键盘输入或随机函数产生。
2. 实现折半插入排序、shell排序、快速排序、归并排序或基数排序算法（任选三）。

**[测试数据]**

**#include** <stdio.h>

**#define** MAX 10                            //数组长度

***int*** t[] **=** {5, 2, 6, 0, 3, 9, 1, 7, 4, 8}, // 原始数组

    a[MAX];                               // 用于排序的数组

// 初始化

***void*** init()

{

**for** (***int*** i **=** 0; i **<** MAX; i**++**)

    a[i] **=** t[i];

}

// 输出

***void*** show(***int*** *t*[])

{

  printf("\n");

**for** (***int*** i **=** 0; i **<** MAX; i**++**)

    printf("%d ", *t*[i]);

  printf("\n\n");

  init(); // show之后再重置需要排序的数组

} //show

//简单选择排序

***void*** selectsort(***int*** *a*[], ***int*** *n*)

{

***int*** i, j, min, temp, c1 **=** 0, c2 **=** 0;

**for** (i **=** 0; i **<** *n*; i**++**)

  {

    min **=** i;

**for** (j **=** i **+** 1; j **<** *n*; j**++**) //寻找最小的值

    {

      c1**++**;

**if** (*a*[j] **<** *a*[min])

        min **=** j;

    }

**if** (min **!=** i) //如果最小的值不是i,则交换

    {

      temp **=** *a*[i];

*a*[i] **=** *a*[min];

*a*[min] **=** temp;

      c2**++**;

    }

  }

  printf("数据交换%d次, 访问数组%d次.", c2, c1);

}

// 折半插入，len为数组长度

***void*** BinsertSort(***int*** *t*[], ***int*** *len*)

{

***int*** i, j, low, hight, mid;

***int*** t1;

**for** (i **=** 1; i **<** *len*; i**++**)

  {

    t1 **=** *t*[i]; /\*i 为要排序的点\*/

    low **=** 0;

    hight **=** i **-** 1;

**while** (low **<=** hight)

    {

      mid **=** (low **+** hight) **/** 2;

**if** (t1 **<** *t*[mid])

        hight **=** (mid **-** 1);

**else**

        low **=** mid **+** 1;

    }

**for** (j **=** i **-** 1; j **>=** low; j**--**)

*t*[j **+** 1] **=** *t*[j];

*t*[low] **=** t1;

  }

}

// 快速排序

***void*** quickSort(***int*** *t*[], ***int*** *low*, ***int*** *hight*)

{

***int*** i, j;

  i **=** *low*;

  j **=** *hight*;

**if** (i **>=** j)

**return**;

***int*** t1;

  t1 **=** *t*[i];

  /\*将小于t1的都放在前面，大于t1的都在后面\*/

**while** (i **!=** j)

  {

**while** ((t1 **<=** *t*[j]) **&&** (j **>** i)) /\*从高处向低处扫\*/

      j**--**;

**if** (i **<** j)

*t*[i] **=** *t*[j];

**while** ((*t*[i] **<=** t1) **&&** (j **>** i)) /\*从低处向高处扫\*/

      i**++**;

**if** (i **<** j)

*t*[j] **=** *t*[i];

  }

*t*[i] **=** t1;

  quickSort(*t*, *low*, i **-** 1);

  quickSort(*t*, i **+** 1, *hight*);

}

// 两组归并算法，将t1 并归到 t2数组中

***void*** Merge(***int*** *t1*[], ***int*** *t2*[], ***int*** *low*, ***int*** *mid*, ***int*** *hight*)

{

***int*** i **=** *low*,

      j **=** *mid* **+** 1,

      k **=** *low*;

**while** ((i **<=** *mid*) **&&** (j **<=** *hight*))

  {

**if** (*t1*[i] **<** *t1*[j])

    {

*t2*[k**++**] **=** *t1*[i**++**];

    }

**else**

*t2*[k**++**] **=** *t1*[j**++**];

  }

  /\*将剩余的数组复制到新的数组中\*/

**while** (i **<=** *mid*)

*t2*[k**++**] **=** *t1*[i**++**];

**while** (j **<=** *hight*)

*t2*[k**++**] **=** *t1*[j**++**];

}

// 一趟并归，n 为数组长度，len为子文件长度,将t1 并归到 t2数组中

***void*** MergePass(***int*** *t1*[], ***int*** *t2*[], ***int*** *n*, ***int*** *len*)

{

***int*** i, j;

  i **=** 0;

**while** ((i **+** 2 **\*** *len* **-** 1) **<** *n*)

  {

    Merge(*t1*, *t2*, i, (i **+** *len* **-** 1), (i **+** 2 **\*** *len* **-** 1)); /\*并归两个长度为len的子文件到t2中\*/

    i **+=** 2 **\*** *len*;

  }

**if** ((i **+** *len* **-** 1) **<** (*n* **-** 1)) /\*剩下两个子文件，其中一个长度小于len\*/

    Merge(*t1*, *t2*, i, i **+** *len* **-** 1, *n* **-** 1);

**else**

**for** (j **=** i; j **<** *n*; j**++**) /\*将最后一个子文件复制到t2数组中\*/

*t2*[j] **=** *t1*[j];

}

***int*** main()

{

  // 原始

  init(); //最开始初始化,之后的重置结合在show中.

  printf("原始数组:");

  show(a);

  // 折半排序

  printf("折半插入排序:");

  BinsertSort(a, MAX);

  show(a);

  // 快排

  printf("快速排序:");

  quickSort(a, 0, (MAX **-** 1));

  show(a);

  // 简单选择排序

  printf("简单选择排序:");

  selectsort(a, MAX);

  show(a);

}

// 原始数组

***int*** t[] **=** {5, 2, 6, 0, 3, 9, 1, 7, 4, 8}

