一、【实验目的】

 (1)  理解插入排序算法的实现过程；

（2）理解不同排序算法的时间复杂度及适用环境；

（3）了解算法性能测试的基本方法。

二、【实验内容】

  （1）以下是一个通过随机数来测试排序算法运行时间的程序，中间留出了加入排序算法的部分。其中可以通过修改RANDNUM的值来更改测试的数据量:

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

#define RANDNUM 5 //随机数的个数

//直接插入排序算法

void insertSort(int a[])

{

int i,j;

int temp;

for(i=1;i<RANDNUM;i++)

{

temp=a[i];

for(j=i-1;j>=0&&temp<=a[j];j--)

{

a[j+1]=a[j];

}

a[j+1]=temp;

}

}

//希尔排序

void shellSort(int a[])

{

int gap = RANDNUM;

while (gap > 1)

{

gap = gap/2; //调整步长因子t(n+1)=t(n)/2

int i = 0;

for (i = 0; i < RANDNUM - gap; i++) //从0遍历到RANDNUM-gap-1

{

int end = i;

int temp = a[end + gap];

while (end >= 0)

{

if (a[end] > temp)

{

a[end + gap] = a[end];

end -= gap;

}

else

{

break;

}

}

a[end + gap] = temp; //以 end+gap 作为插入位置

}

}

}

//测量运行时间

void Time\_(int a\_1[],int a\_2[])

{

clock\_t first,second; //记录开始和结束时间（以毫秒为单位）

first=clock(); //开始时间

//直接插入排序

insertSort(a\_1);

second=clock();//结束时间

//显示排序算法所用的时间

printf("直接插入排序算法运行时间：%ld\n",second-first);

first=clock();

//希尔排序

shellSort(a\_2);

second=clock();

printf("希尔排序算法运行时间：%ld\n",second-first);

printf("\n");

}

//主函数

void main()

{

int iRandNum\_1[RANDNUM];

int iRandNum\_2[RANDNUM];//存放随机数,iRandNum[RANDNUM]\_1用于测量直接插入排序，iRandNum\_2[RANDNUM]用于希尔排序

int i,temp;

//产生2万个随机数

for(i=0;i<RANDNUM;i++)

{

iRandNum\_1[i]=rand()%RANDNUM;

iRandNum\_2[i]=iRandNum\_1[i];

//使两个排序算法操作对象数值相同

}

//无序

printf("在序列无序的情况下：\n");

Time\_(iRandNum\_1,iRandNum\_2);

//顺序 经过排序后，两个序列都处于顺序，此时可以直接测量两排列操作顺序的运行时间

printf("在序列顺序的情况下：\n");

Time\_(iRandNum\_1,iRandNum\_2);

//逆序 需先将顺序序列转为逆序

for(i=0;i<RANDNUM/2;i++)

{

temp=iRandNum\_1[i];

iRandNum\_1[i]=iRandNum\_1[RANDNUM-i-1];

iRandNum\_1[RANDNUM-i-1]=temp;

//

temp=iRandNum\_2[i];

iRandNum\_2[i]=iRandNum\_2[RANDNUM-i-1];

iRandNum\_2[RANDNUM-i-1]=temp;

}//完成逆序操作

printf("在序列逆序的情况下：\n");

Time\_(iRandNum\_1,iRandNum\_2);

// for(i=0;i<RANDNUM;i++)

// {

// printf(" %d",iRandNum\_1[i]);

// }

}

（2） 若待排序的数据量为20000，编写直接插入排序、希尔排序算法进行测试，记录顺序、逆序、无序三种状态下的运行时间，比较测试结果与理论分析是否一致。

 提示:在程序的实现过程中要用到以下函数，请大家在实验之前自学这几个函数的用法:

1）随机函数rand（）

2)时间函数clock（）

**三、实验源代码**

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

#define RANDNUM 10000 //随机数的个数

//直接插入排序算法

void insertSort(int a[])

{

int i,j;

int temp;

for(i=1;i<RANDNUM;i++)

{

temp=a[i];

for(j=i-1;j>=0&&temp<=a[j];j--)

{

a[j+1]=a[j];

}

a[j+1]=temp;

}

}

//希尔排序

void shellSort(int a[])

{

int gap = RANDNUM;

while (gap > 1)

{

gap = gap/2; //调整步长因子t(n+1)=t(n)/2

int i = 0;

for (i = 0; i < RANDNUM - gap; i++) //从0遍历到RANDNUM-gap-1

{

int end = i;

int temp = a[end + gap];

while (end >= 0)

{

if (a[end] > temp)

{

a[end + gap] = a[end];

end -= gap;

}

else

{

break;

}

}

a[end + gap] = temp; //以 end+gap 作为插入位置

}

}

}

//测量运行时间

void Time\_(int a\_1[],int a\_2[])

{

clock\_t first,second; //记录开始和结束时间（以毫秒为单位）

first=clock(); //开始时间

//直接插入排序

insertSort(a\_1);

second=clock();//结束时间

//显示排序算法所用的时间

printf("直接插入排序算法运行时间：%ld\n",second-first);

first=clock();

//希尔排序

shellSort(a\_2);

second=clock();

printf("希尔排序算法运行时间：%ld\n",second-first);

printf("\n");

}

//主函数

void main()

{

int iRandNum\_1[RANDNUM];

int iRandNum\_2[RANDNUM];//存放随机数,iRandNum[RANDNUM]\_1用于测量直接插入排序，iRandNum\_2[RANDNUM]用于希尔排序

int i,temp;

//产生2万个随机数

for(i=0;i<RANDNUM;i++)

{

iRandNum\_1[i]=rand()%RANDNUM;

iRandNum\_2[i]=iRandNum\_1[i];

//使两个排序算法操作对象数值相同

}

//无序

printf("在序列无序的情况下：\n");

Time\_(iRandNum\_1,iRandNum\_2);

//顺序 经过排序后，两个序列都处于顺序，此时可以直接测量两排列操作顺序的运行时间

printf("在序列顺序的情况下：\n");

Time\_(iRandNum\_1,iRandNum\_2);

//逆序 需先将顺序序列转为逆序

for(i=0;i<RANDNUM/2;i++)

{

temp=iRandNum\_1[i];

iRandNum\_1[i]=iRandNum\_1[RANDNUM-i-1];

iRandNum\_1[RANDNUM-i-1]=temp;

//

temp=iRandNum\_2[i];

iRandNum\_2[i]=iRandNum\_2[RANDNUM-i-1];

iRandNum\_2[RANDNUM-i-1]=temp;

}//完成逆序操作

printf("在序列逆序的情况下：\n");

Time\_(iRandNum\_1,iRandNum\_2);

// for(i=0;i<RANDNUM;i++)

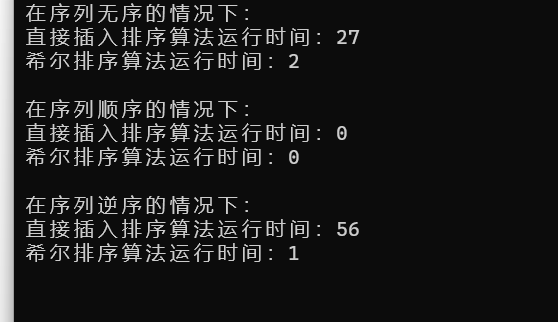
// {

// printf(" %d",iRandNum\_1[i]);

// }

}

**四、实验结果**



**五、实验总结**

测试用例不同不同算法各有优势，不同条件下各有优势