**Project 1 算法性能测量**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 学号 | 2021111011004 | 姓名 | 王齐 | 上交时间 | 2021-11-04 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 题号 | 程序逻辑(40) | 算法新颖性(20) | 代码规范 (20) | 实验报告(20) | 总分 |
| 得分 |  |  |  |  |  |

**1. 问题描述**

随机生成N个整数，请分别用“选择排序”算法和“希尔排序”算法对其进行从小到大的排序，并测试不同规模N下两算法的运行时间。

**2. 算法描述**

SelectionSort（选择排序）：

void SelecttionSort(Record Array[], int n) {

// 依次选出第i小的记录，即剩余记录中最小的那个

for (int i=0;i<n-1;i++){

//首先记录i为当前最小值

int Samllest = i;

//开始向后扫描剩余记录

for (int j=i+1; j< n;j++)

//如果发现更小的记录，则记录其位置

if (Array[j] < Array[Smallest])

Smallest = j;

// 将第i小的记录放在数组中第i个位置

swap(Array, i, Smallest);

}

}

ShellSort（“增量每次除以2递减”的希尔排序）：

void ShellSort(Record Array[], int n){

//Shell排序，Array[]为待排序数组，n为数组长度

int i,j,delta,tmp;

//增量delta每次除以2递减

for (delta = n/2; delta>0;delta/=2)

for (i=0; i<delta; i++){

//分别对delta个子序列进行插入排序

tmp = A[i];

for (j=i; j<n; j += delta){

if (A[j] > A[j+delta])

A[j] = A[j+delta];

else

break;

}

A[j] = tmp;

}

}

**3. 测试结果**

**表1 二个函数的运行时间统计（秒）**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| --- | N | 1000 | 2000 | 4000 | 8000 | 16000 | 32000 | 64000 |
| SelectionSort | Iterations (K) | 1000 | 100 | 100 | 10 | 10 | 1 | 1 |
| Total Time (sec) | 0.871000 | 0.352000 | 1.392000 | 0.559000 | 2.210000 | 0.904000 | 3.524000 |
| Duration (sec) | 0.000871 | 0.003520 | 0.013920 | 0.055900 | 0.221000 | 0.904000 | 3.524000 |
| ShellSort | Iterations (K) | 10000 | 10000 | 10000 | 1000 | 1000 | 1000 | 1000 |
| Total Time (sec) | 0.204000 | 0.46000 | 1.042000 | 0.230000 | 0.505000 | 1.101000 | 2.337000 |
| Duration (sec) | 0.000020 | 0.000047 | 0.000104 | 0.000230 | 0.000505 | 0.001101 | 0.002337 |

**4. 分析与评论**

**图1 SelectionSort函数的运行时间（秒）**

**图2 ShellSort函数的运行时间（秒）**

**图3 SelectionSort与ShellSort函数的运行时间统计（秒）比较**

通过以上的几幅图可以看出，在数量较大的问题求解中希尔排序的求解速度要比简单选择排序快很多。