河南工业大学 数据结构 实验报告

课程名称 数据结构 实验项目 实验六　排序相关典型算法实现

院　　系 信息学院计科系 专业班级 计科 xxxx 班

姓　　名 xxxxx 学　　号 xxxxxxxxxxxxx

指导老师 xxxxxxx 日　　期 2020.12.1

批改日期 成　　绩

一 实验目的

1. 熟悉并掌握各种排序方法的设计思路。
2. 掌握各种具体排序算法在计算机上的实现。
3. 掌握各种排序方法的性能比较。

二 实验内容及要求

实验内容：

1. 编程实现如下功能：

输入同样一组整型数据，作为待排序记录的关键字序列。

（1）实现快速排序算法

（2）实现堆排序算法

希尔排序选做。

实验要求：

1.键盘输入数据；

2.屏幕输出运行结果。

3.要求记录实验源代码及运行结果。

4.运行环境：CodeBlocks/Dev c++/VC6.0 等 C 编译环境

三 实验过程及运行结果

(1) 算法设计思路

1.堆排序：这里以排成升序为例，我们需要将其原始数组调整成下标从 0 开始的最大堆，再每次将最大堆顶与当前最后的元素交换（相当于删除最大堆顶）后重新调整为最大堆，最大堆的调整方法是：在线性时间复杂度下建立最大堆，将 N 个元素按输入顺序存入，使其先满足完全二叉树的结构特性，调整各结点位置，使其满足最大堆的有序特性（每个子树都为最大堆）

2.快速排序：

①设 k = a[0],将 k 挪到适当位置，使得比 k 小的元素都在 k 左边，比 k 大的元素都在 k 右边(在 O(n)时间完成)

②把 k 左边的部分快速排序

③把 k 右边的部分快速排序

快速排序是所有内部排序的最快的算法，但其不稳定，在最坏情况下效率较慢。

3. 希尔排序：利用了插入排序的简单，并克服了插入排序只能交换相邻两元素的缺点。

希尔排序是把记录按下标的一定增量分组，对每组使用直接插入排序算法排序；随着增量逐渐减少，每

组包含的关键词越来越多，当增量减至 1 时，整个文件恰被分成一组，算法便终止。
定义增量序列 Dm >Dm-1>…>D1 = 1
对每个 Dk 进行“Dk 间隔”排序（k=M，M-1，……，1）
“Dk 间隔”有序的序列，在执行“Dk-1 间隔”排序后，仍然是“Dk 间隔”有序的。

(2) 源程序代码

1.堆排序：

①将 N 个元素的数组中以 a[rt]为根的子堆调整为最大堆，由上往下，先暂存根节点 a[rt]的值，若孩子节点比父节点值大，则将孩子结点赋给父节点，继续看该孩子的孩子结点，以此类推，保证每个子堆都为最大堆。

```
void PercDown(Datatype a[], int N, int rt) {
    //将N个元素的数组中以a[rt]为根的子堆调整为最大堆
    int father, son;
    Datatype tmp = a[rt];
    for (father = rt; (father * 2 + 1) < N; father = son) {    //从上往下
        son = father * 2 + 1;                                  //左儿子
        if (son != N - 1 && a[son] < a[son + 1])  //右儿子存在且比左儿子大
            son++;
        if (tmp >= a[son])
            break;   //找到该放的地方
        else
            a[father] = a[son];    //下滤
    }
    a[father] = tmp;
}
```

②将已有N个元素的数组调整为最大堆 a[0]为最大元素，即从最下边的子树开始向上。

```
inline void BuildHeap(Datatype a[], int N) {
    for (int i = N / 2 - 1; i >= 0; --i) {
        PercDown(a, N, i);
    }
}
```

③堆排序本体，将最大堆顶a[0]与a[i]交换后进行调整。

```
void Heap_Sort(Datatype a[], int N) {
    printf("1.调整为最大堆结果\n");
    BuildHeap(a, N);
    PrintArr(a, N);
    cout << endl;
    printf("2.交换调整过程\n");
    for (int i = N - 1; i > 0; --i) {
        cout << "最大堆顶" << a[0] << "与" << a[i] << "交换:";
        swap(a[0], a[i]);  //最大堆顶a[0]与a[i]交换
        PrintArr(a, N);
        cout << endl;
        PercDown(a, i, 0);    //删除后进行调整
        cout << " 调整为-> ";
        PrintArr(a, N);
        cout << endl;
```

```
    }
}
```

2.快速排序：

①设 k = a[0],将 k 挪到适当位置，使得比 k 小的元素都在 k 左边，比 k 大的元素都在 k 右边(在 O(n)时间完成)

②把 k 左边的部分快速排序

③把 k 右边的部分快速排序

```
void QuickSort(Datatype a[], int s, int e) {  //将a[s,e]快排
    if (s >= e)
        return;
    printf("将%d放到合适位置使左边元素都比它小，右边元素都比他大\n",a[s]);
    Datatype k = a[s];
    int i = s, j = e;
    while (i != j) {
        while (j > i && a[j] >= k)
            --j;
        if (i == j) break;
        swap(a[i], a[j]);
        PrintArr(a, N);
        cout << endl;
        while (i < j && a[i] <= k)
            ++i;
        if (i == j) break;
        swap(a[i], a[j]);
        PrintArr(a, N);
        cout << endl;
    }                           //处理完后a[i] = k;
    QuickSort(a, s, i - 1);     //快排左边部分
    QuickSort(a, i + 1, e);     //快排右边部分
}
void Quick_Sort(Datatype a[], int N) {
    QuickSort(a, 0, N-1);
}
```

3.希尔排序

①选取增量序列 Dk、Dk-1……

②进行“Dk 间隔”的插入排序

```
void Shell_Sort(Datatype a[], int N) {
    for (int D = N / 2; D > 0; D /= 2) {  //希尔增量序列
        cout << "增量为" << D << endl;
        for (int P = D; P < N; ++P) {    //插入排序
            Datatype t = a[P];
            int i;
            for (i = P; i >= D && a[i - D] > t; i -= D)
                a[i] = a[i - D];
            a[i] = t;
        }
```

```
        PrintArr(a, N);
        cout << endl;
    }
}
```

(3) 运行结果截图

```
E:\Programming\数据结构\class\实验\计科201916010728余思婉实验6实验报告\Sort\Debug\Sort.exe
请输入待排序数数量N：10
请输入N个数：3 4 8 2 13 1 5 7 9 11
堆排序如下：1.调整为最大堆结果
13 11 8 9 4 1 5 7 2 3
2.交换调整过程
最大堆顶13与3交换:3 11 8 9 4 1 5 7 2 13
 调整为-> 11 9 8 7 4 1 5 3 2 13
最大堆顶11与2交换:2 9 8 7 4 1 5 3 11 13
 调整为-> 9 7 8 3 4 1 5 2 11 13
最大堆顶9与2交换:2 7 8 3 4 1 5 9 11 13
 调整为-> 8 7 5 3 4 1 2 9 11 13
最大堆顶8与2交换:2 7 5 3 4 1 8 9 11 13
 调整为-> 7 4 5 3 2 1 8 9 11 13
最大堆顶7与1交换:1 4 5 3 2 7 8 9 11 13
 调整为-> 5 4 1 3 2 7 8 9 11 13
最大堆顶5与2交换:2 4 1 3 5 7 8 9 11 13
 调整为-> 4 3 1 2 5 7 8 9 11 13
最大堆顶4与2交换:2 3 1 4 5 7 8 9 11 13
 调整为-> 3 2 1 4 5 7 8 9 11 13
最大堆顶3与1交换:1 2 3 4 5 7 8 9 11 13
 调整为-> 2 1 3 4 5 7 8 9 11 13
最大堆顶2与1交换:1 2 3 4 5 7 8 9 11 13
 调整为-> 1 2 3 4 5 7 8 9 11 13
```

```
快速排序如下：
将3放到合适位置使左边元素都比它小，右边元素都比他大
1 4 8 2 13 3 5 7 9 11
1 3 8 2 13 4 5 7 9 11
1 2 8 3 13 4 5 7 9 11
1 2 3 8 13 4 5 7 9 11
将1放到合适位置使左边元素都比它小，右边元素都比他大
将8放到合适位置使左边元素都比它小，右边元素都比他大
1 2 3 7 13 4 5 8 9 11
1 2 3 7 8 4 5 13 9 11
1 2 3 7 5 4 8 13 9 11
将7放到合适位置使左边元素都比它小，右边元素都比他大
1 2 3 4 5 7 8 13 9 11
将4放到合适位置使左边元素都比它小，右边元素都比他大
将13放到合适位置使左边元素都比它小，右边元素都比他大
1 2 3 4 5 7 8 11 9 13
将11放到合适位置使左边元素都比它小，右边元素都比他大
1 2 3 4 5 7 8 9 11 13

希尔排序如下：
增量为5
1 4 7 2 11 3 5 8 9 13
增量为2
1 2 5 3 7 4 9 8 11 13
增量为1
1 2 3 4 5 7 8 9 11 13
```

四 调试情况、设计技巧及体会

暑假时就已在 PTA 上进行过各种排序算法的测试，这是当时测试的结果

堆排序

题目	编译器	耗时
09-排序1	C++ (g++)	53 ms

耗时	内存
3 ms	384 KB
4 ms	424 KB
4 ms	384 KB
7 ms	552 KB
53 ms	1664 KB
47 ms	1832 KB
48 ms	1832 KB
47 ms	1764 KB
50 ms	1508 KB

https://blog.csdn.net/qq_

快速排序

题目	编译器	耗时
09-排序1	C++ (g++)	4156 ms

耗时	内存
6 ms	360 KB
6 ms	384 KB
6 ms	384 KB
18 ms	624 KB
58 ms	1836 KB
4156 ms	1628 KB
4134 ms	3248 KB
2913 ms	1972 KB
67 ms	1548 KB

https://blog.csdn.net/qq_

希尔排序

题目	编译器	耗时
09-排序1	C++ (g++)	64 ms

耗时	内存
5 ms	424 KB
5 ms	352 KB
5 ms	388 KB
11 ms	384 KB
64 ms	1792 KB
47 ms	1624 KB
55 ms	1804 KB
45 ms	1664 KB
54 ms	1600 KB

https://blog.csdn.net/

在这些测试数据里堆排序速度最快，
不过快排是最快的，只是在某些最坏情况下快排耗时挺大的，另外这三个排序都不稳定。
右图是总结的各排序时间空间复杂度及稳定性

排序方法	平均时间复杂度	最坏情况下时间复杂度	额外空间复杂度	稳定性
简单选择排序	$O(N^2)$	$O(N^2)$	O(1)	不稳定
冒泡排序	$O(N^2)$	$O(N^2)$	O(1)	稳定
直接插入排序	$O(N^2)$	$O(N^2)$	O(1)	稳定
希尔排序	$O(N^d)$	$O(N^2)$	O(1)	不稳定
堆排序	O(NlogN)	O(NlogN)	O(1)	不稳定
快速排序	O(NlogN)	$O(N^2)$	O(logN)	不稳定
归并排序	O(NlogN)	O(NlogN)	O(N)	稳定
基数排序	O(P(N+B))	O(P(N+B))	O(N+B)	稳定