Java数据结构和算法（三）——冒泡、选择、插入排序算法

目录

上一篇博客我们实现的数组结构是无序的，也就是纯粹按照插入顺序进行排列，那么如何进行元素排序，本篇博客我们介绍几种简单的排序算法。

回到顶部

1、冒泡排序

这个名词的由来很好理解，一般河水中的冒泡，水底刚冒出来的时候是比较小的，随着慢慢向水面浮起会逐渐增大，这物理规律我不作过多解释，大家只需要了解即可。

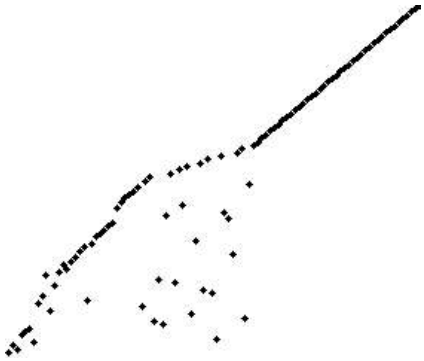
冒泡算法的运作规律如下：

①、比较相邻的元素。如果第一个比第二个大，就交换他们两个。

②、对每一对相邻元素作同样的工作，从开始第一对到结尾的最后一对。这步做完后，最后的元素会是最大的数（也就是第一波冒泡完成）。

③、针对所有的元素重复以上的步骤，除了最后一个。

④、持续每次对越来越少的元素重复上面的步骤，直到没有任何一对数字需要比较。



昵称：YSOcean
园龄：2年
粉丝：2067
关注：13
+加关注

< 2019年3月 >

日	一	二	三	四	五	六
24	25	26	27	28	1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30
31	1	2	3	4	5	6

我的标签

Linux系列教程(25)
深入理解计算机系统(24)
Java数据结构和算法(16)
MyBatis详解系列(11)
JDK源码解析(11)
Maven系列教程(8)
Redis详解(8)
Spring入门系列(8)
Java IO详解系列(7)
Java高并发设计(7)
更多

随笔分类

Java SE(22)
JavaWeb(34)
Java高并发
Java关键字(6)
Java数据结构和算法(15)

代码如下：

```
package com.ys.sort;

public class BubbleSort {
    public static int[] sort(int[] array){
        //这里for循环表示总共需要比较多少轮
        for(int i = 1 ; i < array.length; i++){
            //设定一个标记，若为true，则表示此次循环没有进行交换，也就是待排序列已经有序，排序已经
            boolean flag = true;
            //这里for循环表示每轮比较参与的元素下标
            //对当前无序区间array[0......length-i]进行排序
            //j的范围很关键，这个范围是在逐步缩小的,因为每轮比较都会将最大的放在右边
            for(int j = 0 ; j < array.length-i ; j++){
                if(array[j]>array[j+1]){
                    int temp = array[j];
                    array[j] = array[j+1];
                    array[j+1] = temp;
                    flag = false;
                }
            }
            if(flag){
                break;
            }
            //第 i轮排序的结果为
            System.out.print("第"+i+"轮排序后的结果为:");
            display(array);

        }
        return array;

    }

    //遍历显示数组
    public static void display(int[] array){
        for(int i = 0 ; i < array.length ; i++){
            System.out.print(array[i]+" ");
        }
```

Java虚拟机
JDK源码解析(11)
Linux(9)
Linux详解(24)
Nginx详解(4)
Redis详解(8)
TCP/IP协议
编程小技巧(1)
查找算法(1)
大数据(3)
工具使用(15)
计算机系统与结构(24)
计算机组成与系统结构
浪潮之巅(1)
排序算法
前端(5)
日常工作问题(7)
设计模式(1)
算法分析(1)
消息中间件(6)
邮件服务(4)

阅读排行榜
1. Tomcat 部署项目的三种方法(152999)
2. Java 集合详解(74904)
3. Java数据结构和算法（一）——简介(62805)
4. MyBatis 详解（一对一，一对多，多对多）(62100)

```
37          System.out.println();
38      }
39
40      public static void main(String[] args) {
41          int[] array = {4,2,8,9,5,7,6,1,3};
42          //未排序数组顺序为
43          System.out.println("未排序数组顺序为：");
44          display(array);
45          System.out.println("------------------------");
46          array = sort(array);
47          System.out.println("------------------------");
48          System.out.println("经过冒泡排序后的数组顺序为：");
49          display(array);
50      }
51
52  }
```

结果如下：

```
未排序数组顺序为：
4 2 8 9 5 7 6 1 3
------------------------
第1轮排序后的结果为:2 4 8 5 7 6 1 3 9
第2轮排序后的结果为:2 4 5 7 6 1 3 8 9
第3轮排序后的结果为:2 4 5 6 1 3 7 8 9
第4轮排序后的结果为:2 4 5 1 3 6 7 8 9
第5轮排序后的结果为:2 4 1 3 5 6 7 8 9
第6轮排序后的结果为:2 1 3 4 5 6 7 8 9
第7轮排序后的结果为:1 2 3 4 5 6 7 8 9
------------------------
经过冒泡排序后的数组顺序为：
1 2 3 4 5 6 7 8 9
```

本来应该是 8 轮排序的，这里我们只进行了 7 轮排序，因为第 7 轮排序之后已经是有序数组了。

冒泡排序解释：

冒泡排序是由两个for循环构成，第一个for循环的变量 i 表示总共需要多少轮比较，第二个for循环的变量 j 表示每轮参与比较的元素下标【0,1，……，length-i】，因为每轮比较都会出现一个最大值放在最右边，所以每轮比较后的元素个数都会少一个，这也是为什么 j 的范围是逐渐减小的。相信大家理解之后快速写出一个冒泡排序并不难。

冒泡排序性能分析：

假设参与比较的数组元素个数为 N，则第一轮排序有 N-1 次比较，第二轮有 N-2 次，如此类推，这种序列的求和公式为：

（N-1）+（N-2）+...+1 = N*（N-1）/2

当 N 的值很大时，算法比较次数约为 $N^2/2$次比较，忽略减1。

假设数据是随机的，那么每次比较可能要交换位置，可能不会交换，假设概率为50%，那么交换次数为 $N^2/4$。不过如果是最坏的情况，初始数据是逆序的，那么每次比较都要交换位置。

交换和比较次数都和N^2 成正比。由于常数不算大 O 表示法中，忽略 2 和 4，那么冒泡排序运行都需要 $O(N^2)$ 时间级别。

其实无论何时，只要看见一个循环嵌套在另一个循环中，我们都可以怀疑这个算法的运行时间为 $O(N^2)$级，外层循环执行 N 次，内层循环对每一次外层循环都执行N次（或者几分之N次）。这就意味着大约需要执行N^2次某个基本操作。

回到顶部

2、选择排序

5. Java数据结构和算法（七）——链表(46217)

评论排行榜

1. 深入理解计算机系统（1.1）------Hello World 是如何运行的(27)
2. SpringMVC详解（四）------SSM三大框架整合之登录功能实现(23)
3. 深入理解计算机系统（序章）------谈程序员为什么要懂底层计算机结构(20)
4. Tomcat 部署项目的三种方法(18)
5. Java数据结构和算法（十）——二叉树(18)

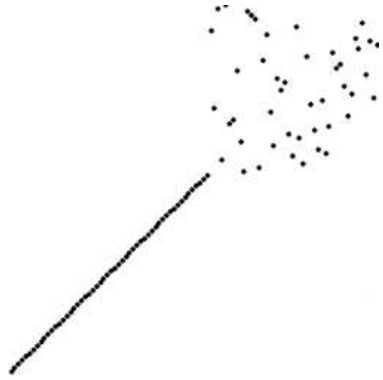
选择排序是每一次从待排序的数据元素中选出最小的一个元素，存放在序列的起始位置，直到全部待排序的数据元素排完。

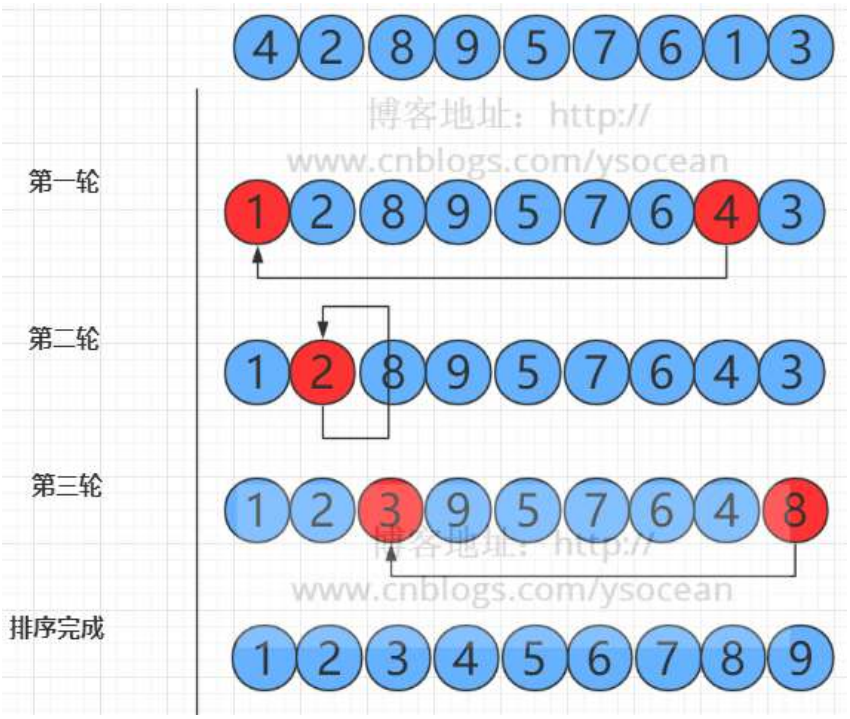
分为三步：

①、从待排序序列中，找到关键字最小的元素

②、如果最小元素不是待排序序列的第一个元素，将其和第一个元素互换

③、从余下的 N - 1 个元素中，找出关键字最小的元素，重复(1)、(2)步，直到排序结束





代码如下：

```
package com.ys.sort;

public class ChoiceSort {
    public static int[] sort(int[] array){
        //总共要经过N-1轮比较
        for(int i = 0 ; i < array.length-1 ; i++){
            int min = i;
            //每轮需要比较的次数
            for(int j = i+1 ; j < array.length ; j++){
                if(array[j]<array[min]){
                    min = j;//记录目前能找到的最小值元素的下标
                }
            }
            //将找到的最小值和i位置所在的值进行交换
            if(i != min){
                int temp = array[i];
                array[i] = array[min];
                array[min] = temp;
            }
```

```
            //第 i轮排序的结果为
            System.out.print("第"+(i+1)+"轮排序后的结果为:");
            display(array);
        }
        return array;
    }

    //遍历显示数组
    public static void display(int[] array){
        for(int i = 0 ; i < array.length ; i++){
            System.out.print(array[i]+" ");
        }
        System.out.println();
    }

    public static void main(String[] args){
        int[] array = {4,2,8,9,5,7,6,1,3};
        //未排序数组顺序为
        System.out.println("未排序数组顺序为：");
        display(array);
        System.out.println("------------------------");
        array = sort(array);
        System.out.println("------------------------");
        System.out.println("经过选择排序后的数组顺序为：");
        display(array);
    }
}
```

运行结果：

```
未排序数组顺序为：
4 2 8 9 5 7 6 1 3
------------------------
第1轮排序后的结果为:1 2 8 9 5 7 6 4 3
第2轮排序后的结果为:1 2 8 9 5 7 6 4 3
第3轮排序后的结果为:1 2 3 9 5 7 6 4 8
第4轮排序后的结果为:1 2 3 4 5 7 6 9 8
第5轮排序后的结果为:1 2 3 4 5 7 6 9 8
第6轮排序后的结果为:1 2 3 4 5 6 7 9 8
第7轮排序后的结果为:1 2 3 4 5 6 7 9 8
第8轮排序后的结果为:1 2 3 4 5 6 7 8 9
------------------------
经过选择排序后的数组顺序为：
1 2 3 4 5 6 7 8 9
```

选择排序性能分析：

选择排序和冒泡排序执行了相同次数的比较：N*（N-1）/2，但是至多只进行了N次交换。

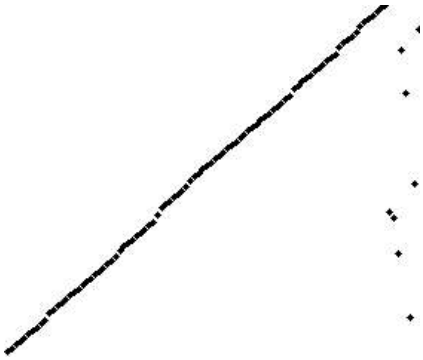
当 N 值很大时，比较次数是主要的，所以和冒泡排序一样，用大O表示是O(N^2) 时间级别。但是由于选择排序交换的次数少，所以选择排序无疑是比冒泡排序快的。当 N 值较小时，如果交换时间比选择时间大的多，那么选择排序是相当快的。

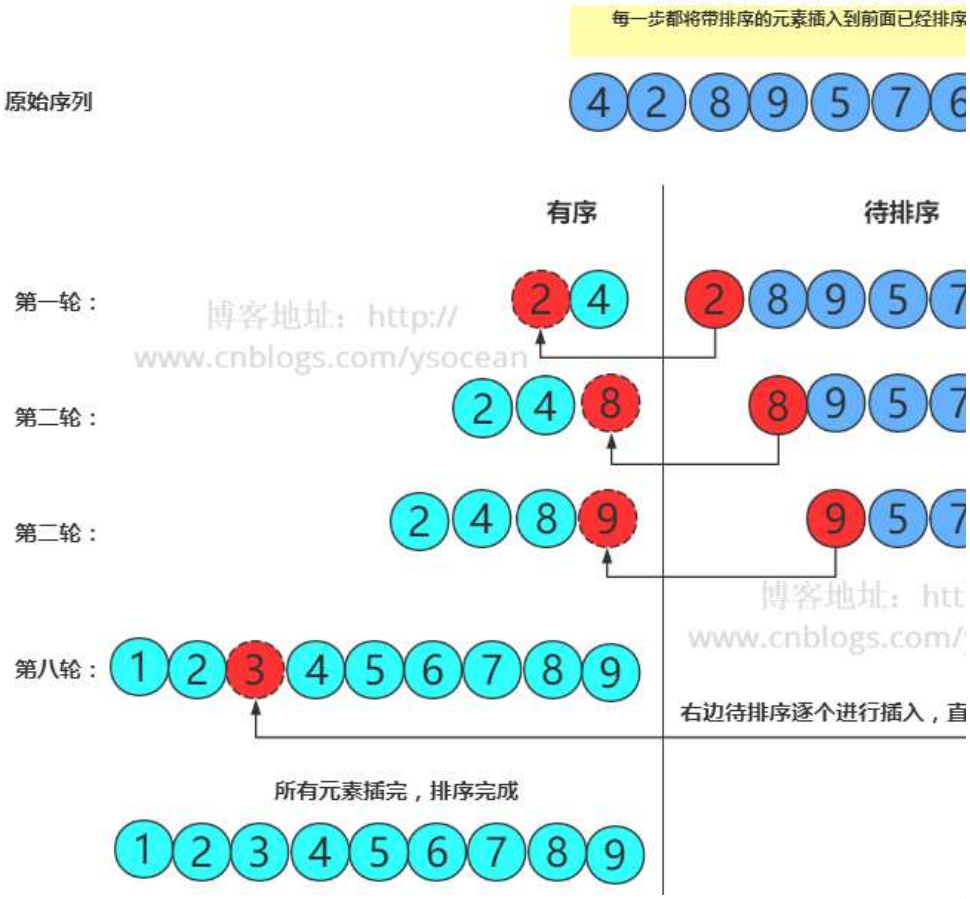
回到顶部

3、插入排序

直接插入排序基本思想是每一步将一个待排序的记录，插入到前面已经排好序的有序序列中去，直到插完所有元素为止。

插入排序还分为直接插入排序、二分插入排序、链表插入排序、希尔排序等等，这里我们只是以直接插入排序讲解，后面讲高级排序的时候会将其他的。





代码如下：

```
package com.ys.sort;

public class InsertSort {
    public static int[] sort(int[] array){
        int j;
        //从下标为1的元素开始选择合适的位置插入，因为下标为0的只有一个元素，默认是有序的
        for(int i = 1 ; i < array.length ; i++){
            int tmp = array[i];//记录要插入的数据
            j = i;
            while(j > 0 && tmp < array[j-1]){//从已经排序的序列最右边的开始比较，找到比其小的数
                array[j] = array[j-1];//向后挪动
                j--;
            }
            array[j] = tmp;//存在比其小的数，插入
        }
        return array;
    }
```

```
18
19      //遍历显示数组
20      public static void display(int[] array){
21          for(int i = 0 ; i < array.length ; i++){
22              System.out.print(array[i]+" ");
23          }
24          System.out.println();
25      }
26
27      public static void main(String[] args){
28          int[] array = {4,2,8,9,5,7,6,1,3};
29          //未排序数组顺序为
30          System.out.println("未排序数组顺序为：");
31          display(array);
32          System.out.println("-----------------------");
33          array = sort(array);
34          System.out.println("-----------------------");
35          System.out.println("经过插入排序后的数组顺序为：");
36          display(array);
37      }
38
39  }
```

运行结果：

```
未排序数组顺序为：
4 2 8 9 5 7 6 1 3
-----------------------
-----------------------
经过插入排序后的数组顺序为：
1 2 3 4 5 6 7 8 9
```

插入排序性能分析：

在第一轮排序中，它最多比较一次，第二轮最多比较两次，一次类推，第N轮，最多比较N-1次。因此有 1+2+3+...+N-1 = N*（N-1）/2。

假设在每一轮排序发现插入点时，平均只有全体数据项的一半真的进行了比较，我们除以2得到：N*（N-1）/4。用大O表示法大致需要需要 $O(N^2)$ 时间级别。

复制的次数大致等于比较的次数，但是一次复制与一次交换的时间耗时不同，所以相对于随机数据，插入排序比冒泡快一倍，比选择排序略快。

这里需要注意的是，如果要进行逆序排列，那么每次比较和移动都会进行，这时候并不会比冒泡排序快。

回到顶部

4、总结

上面讲的三种排序，冒泡、选择、插入用大 O 表示法都需要 $O(N^2)$ 时间级别。一般不会选择冒泡排序，虽然冒泡排序书写是最简单的，但是平均性能是没有选择排序和插入排序好的。

选择排序把交换次数降低到最低，但是比较次数还是挺大的。当数据量小，并且交换数据相对于比较数据更加耗时的情况下，可以应用选择排序。

在大多数情况下，假设数据量比较小或基本有序时，插入排序是三种算法中最好的选择。

后面我们会讲解高级排序，大O表示法的时间级别将比$O(N^2)$小。

分类: Java数据结构和算法

标签: Java数据结构和算法

好文要顶 关注我 收藏该文

YSOcean
关注 - 13
粉丝 - 2067
+加关注

10 1

« 上一篇：Java数据结构和算法（二）——数组
» 下一篇：Java数据结构和算法（四）——栈

posted @ 2017-12-01 10:02 YSOcean 阅读(15034) 评论(6) 编辑 收藏

评论列表

#1楼 2017-12-01 11:23 东风冷雪

这几个排序是最简单的，我也只会这几个（捂嘴笑）。。

支持(0) 反对(0)

#2楼 2018-01-03 07:45 yxlaisj

冒泡排序if(flag){
break;
}
这里为什么不是continue?

支持(0) 反对(0)

#3楼[楼主] 2018-04-02 09:19 YSOcean

@ yxlaisj
当 flag 等于true时，表示当前数组都已经有序了，所以不用在进行冒泡比较了，这里就直接通过break退出循环了，如果是continue，那意思是还得进行for循环比较

支持(2) 反对(0)

#4楼 2018-10-06 02:02 正在修炼的标准程序猴

非常感谢博主，您的文章给了我很大的帮助。

支持(0) 反对(0)

#5楼 2018-10-23 19:04 缃缪

感谢+1，干货满满，很有收获。

支持(0) 反对(0)

#6楼 2018-12-24 10:53 我爱coding1

sigaoai

支持(0) 反对(0)

刷新评论 刷新页面 返回顶部

注册用户登录后才能发表评论，请 登录 或 注册，访问网站首页。

相关博文：
- Java数据结构和算法（三）——冒泡、选择、插入排序算法
- Java数据结构和算法（三）——冒泡、选择、插入排序算法
- Java数据结构和算法（三）——冒泡、选择、插入排序算法
- Java数据结构和算法总结-冒泡排序、选择排序、插入排序算法分析
- （三）Java数据结构和算法——冒泡、选择、插入排序算法

最新新闻：
- 采访Facebook产品设计师：我是如何从零开始转行成功的？
- 游戏陪练，电竞行业造血者？
- 露露事件背后是腾讯资产的流失
- 电池和续航，可能是苹果AirPods便利性的「最大受害者」
- 收购爱康国宾，阿里巴巴图什么？

» 更多新闻...