

dreamcatcher-cx

why is more important than what.

博客园 首页 新随笔 联系 订阅 管理

图解排序算法(一)之3种简单排序(选择, 冒泡, 直接插入)

排序是数据处理中十分常见且核心的操作, 虽说实际项目开发中很小几率会需要我们手动实现, 毕竟每种语言的类库中都有n多种关于排序算法的实现。但是了解这些精妙的思想对我们还是大有裨益的。本文简单温习下最基础的三类算法: 选择, 冒泡, 插入。

先定义个交换数组元素的函数, 供排序时调用



```
/**
 * 交换数组元素
 * @param arr
 * @param a
 * @param b
 */
public static void swap(int []arr,int a,int b){
    arr[a] = arr[a]+arr[b];
    arr[b] = arr[a]-arr[b];
    arr[a] = arr[a]-arr[b];
}
```



简单选择排序

公告

昵称: dreamcatcher-cx
园龄: 4年7个月
粉丝: 1066
关注: 33
[+加关注](#)

< 2021年4月 >						
日	一	二	三	四	五	六
28	29	30	31	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	1
2	3	4	5	6	7	8

积分与排名

积分 - 63297


排名 - 17684

随笔分类 (20)


简单选择排序是最简单直观的一种算法，基本思想为**每一趟从待排序的数据元素中选择最小（或最大）的一个元素作为首元素**，直到所有元素排完为止，简单选择排序是**不稳定排序**。

在算法实现时，每一趟确定最小元素的时候会通过不断地比较交换来使得首位置为当前最小，交换是个比较耗时的操作。其实我们很容易发现，在还未完全确定当前最小元素之前，这些交换都是无意义的。我们可以通过设置一个变量min，每一次比较仅存储较小元素的数组下标，当轮循环结束之后，那这个变量存储的就是当前最小元素的下标，此时再执行交换操作即可。代码实现很简单，一起来看下。

代码实现



```
/**
 * 简单选择排序
 *
 * @param arr
 */
public static void selectSort(int[] arr) {
    for (int i = 0; i < arr.length - 1; i++) {
        int min = i; //每一趟循环比较时, min用于存放较小元素的数组下标, 这样当前批次比较完毕最终存放的就是此趟内最小的元素的下标, 避免每次遇到较小元素都要进行交换。
        for (int j = i + 1; j < arr.length; j++) {
            if (arr[j] < arr[min]) {
                min = j;
            }
        }
        //进行交换, 如果min发生变化, 则进行交换
        if (min != i) {
            swap(arr,min,i);
        }
    }
}
```



简单选择排序通过上面优化之后，无论数组原始排列如何，比较次数是不变的；对于交换操作，在最好情况下也就是数组完全有序的时候，无需任何交换移动，在最差情况下，也就是数组倒序的时候，交换次数为n-1次。综合下来，时间复杂度为**O(n²)**

冒泡排序

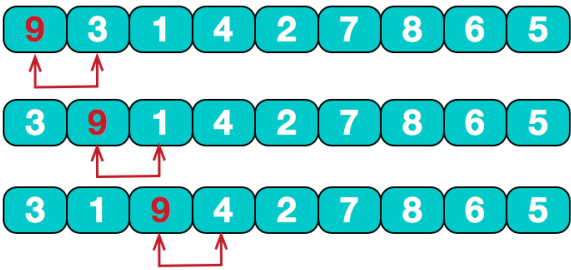
java集合框架(1)
Oracle(4)
并发编程(8)
数据结构(2)
算法(5)

随笔档案 (20)
2017年7月(2)
2017年6月(1)
2017年5月(2)
2017年4月(1)
2017年3月(1)
2017年2月(1)
2017年1月(1)
2016年12月(3)
2016年11月(4)
2016年10月(2)
2016年9月(2)

最新评论

冒泡排序的基本思想是，对相邻的元素进行两两比较，顺序相反则进行交换，这样，每一趟会将最小或最大的元素“浮”到顶端，最终达到完全有序

相邻元素两两比较，反序则交换



第一轮完毕，将最大元素9浮到数组顶端



同理,第二轮将第二大元素8浮到数组顶端



排序完成



代码实现

在冒泡排序的过程中，如果某一趟执行完毕，没有做任何一次交换操作，比如数组[5,4,1,2,3]，执行了两次冒泡，也就是两次外循环之后，分别将5和4调整到最终位置[1,2,3,4,5]。此时，再执行第三次循环后，一次交换都没有做，这就说明剩下的序列已经是有序的，排序操作也就可以完成了，来看下代码

```
/**
 * 冒泡排序
 *
 * @param arr
 */
public static void bubbleSort(int[] arr) {
    for (int i = 0; i < arr.length - 1; i++) {
        boolean flag = true; // 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，排序已然完成。
        for (int j = 0; j < arr.length - 1 - i; j++) {
            if (arr[j] > arr[j + 1]) {
                swap(arr, j, j + 1);
                flag = false;
            }
        }
    }
}
```

1. Re:图解排序算法(四)之归并排序

算法小白，其实有一个比较疑惑的问题，相比插入/冒泡的排序算法，使用分治思想的排序算法在开始排序之前还要经过一个拆分数组的过程，总共的时间加起来，真的会比冒泡快吗，还是说随着数组长度变化，长度跟时间的曲...

--去骨鸡腿排

2. Re:图解排序算法(四)之归并排序

初始化 t=0有问题吧

t=left

--java渣渣

3. Re:图解排序算法(二)之希尔排序

@Boblim 博主理解是对的，有写到实际实现时不需要严格按照分组进行，这样能减少一层你写的for(int k=0;k<div;++k)分组循环，你可以模拟运行再体会下。你的代码虽然是严格按照算法理解...

--stagelovepig

4. Re:ConcurrentHashMap实现原理及源码分析

很棒

--收纸箱易拉罐

5. Re:图解排序算法(二)之希尔排序

不停的跳组,分组插入排序,相同的分组之间就是排序两个有序数组,感觉和归并差不多啊(狗头).

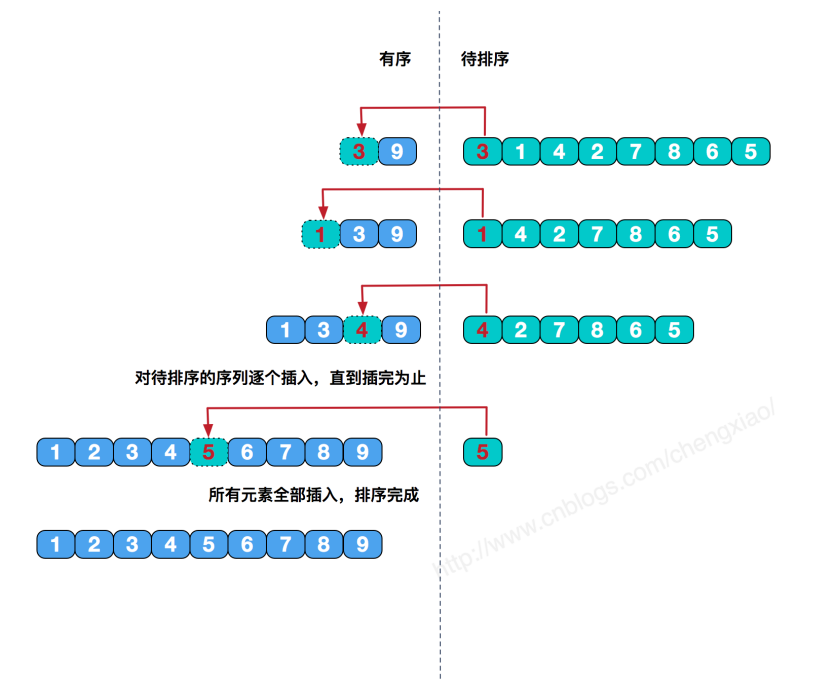
--吕思豪

```
        }
    }
    if (flag) {
        break;
    }
}
}
```

根据上面这种冒泡实现，若原数组本身就是有序的（这是最好情况），仅需 $n-1$ 次比较就可完成；若是倒序，比较次数为 $n-1+n-2+\cdots+1=n(n-1)/2$ ，交换次数和比较次数等值。所以，其时间复杂度依然为 $O(n^2)$ 。综合来看，冒泡排序性能还还是稍差于上面那种选择排序的。

直接插入排序

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。**



代码实现

```
/**
 * 插入排序
 *
 * @param arr
```

阅读排行榜

1. 图解排序算法(三)之堆排序(563527)
2. 图解排序算法(四)之归并排序(355483)
3. HashMap实现原理及源码分析(332762)
4. 图解排序算法(一)之3种简单排序(选择, 冒泡, 直接插入)(240501)
5. 图解排序算法(二)之希尔排序(222641)

评论排行榜

1. HashMap实现原理及源码分析(70)
2. 图解排序算法(三)之堆排序(67)
3. 图解排序算法(四)之归并排序(47)
4. 图解排序算法(二)之希尔排序(27)
5. 图解排序算法(一)之3种简单排序(选择, 冒泡, 直接插入)(18)

推荐排行榜

1. HashMap实现原理及源码分析(167)
2. 图解排序算法(三)之堆排序(166)
3. 图解排序算法(四)之归并排序(132)
4. ConcurrentHashMap实现原理及源码分析(49)

5. 图解排序算法(二)之希尔排序(48)

```
*/  
  
public static void insertionSort(int[] arr) {  
    for (int i = 1; i < arr.length; i++) {  
        int j = i;  
        while (j > 0 && arr[j] < arr[j - 1]) {  
            swap(arr, j, j-1);  
            j--;  
        }  
    }  
}
```



简单插入排序在最好情况下, 需要比较 $n-1$ 次, 无需交换元素, 时间复杂度为 $O(n)$;在最坏情况下, 时间复杂度依然为 $O(n^2)$ 。但是在数组元素随机排列的情况下, 插入排序还是要优于上面两种排序的。

总结

本文列举了排序算法中最基本的三种算法(简单选择, 冒泡, 插入), 这三种排序算法的时间复杂度均为 $O(n^2)$, 后续会陆续更新其他更高阶一些的排序算法, 时间复杂度也会逐步突破 $O(n^2)$, 谢谢支持。

作者: dreamcatcher-cx

出处: <<http://www.cnblogs.com/chengxiao/>>

本文版权归作者和博客园共有, 欢迎转载, 但未经作者同意必须保留此段声明, 且在页面明显位置给出原文链接。

分类: 算法

好文要顶

关注我

收藏该文



dreamcatcher-cx

关注 - 33

粉丝 - 1066

+加关注

37

1

« 上一篇: HashMap实现原理及源码分析

» 下一篇: 图解排序算法(二)之希尔排序

posted @ 2016-11-26 12:46 dreamcatcher-cx 阅读(240508) 评论(18)

) 编辑 收藏

[刷新评论](#) [刷新页面](#) [返回顶部](#)

登录后才能查看或发表评论, 立即 [登录](#) 或者 [逛逛](#) [博客园首页](#)

【推荐】阿里云云小站限量代金券, 新老用户同享, 上云优惠聚集地
【推荐】大型组态、工控、仿真、CAD\GIS 50万行VC++源码免费下载!
【推荐】#悄悄变强大# 五一假期提升指南, 你若学习, 机会自来
【推荐】限时秒杀! 国云大数据魔镜, 企业级云分析平台

园子动态:

- 致园友们的一封检讨书: 都是我们的错
- 数据库实例 CPU 100% 引发全站故障
- 发起一个开源项目: 博客引擎 fluss

最新新闻:

- 法官力挺亚马逊 拒绝驳回干预AWS采购100亿美元国防合同的投诉
 - Facebook向广告商详细阐述苹果ATT对他们的影响
 - 特斯拉全新MPV渲染图曝光 造型汽车史上绝无仅有
 - 大部分欧美游戏开发者不认为Steam应获得30%的收入
 - 首战告捷! 中国空间站出征太空 3年后或全球唯一
- » 更多新闻...

Copyright © 2021 dreamcatcher-cx
Powered by .NET 5.0 on Kubernetes