飞翔的猫咪

http://flyingcat2013.blog.51cto.com 【复制】 【订阅】

原创 用Java写算法之五：快速排序

2013-08-23 16:54:45

标签：用Java写算法 算法 快速排序

快速排序是一个知名度极高的排序算法，其对于大数据的优秀排序性能和相同复杂度算法中相对简单的实现使它注定得到比其他算法更多的宠爱。

算法概述/思路

快速排序一般基于递归实现。其思路是这样的：

1.选定一个合适的值（理想情况中值最好，但实现中一般使用数组第一个值）,称为“枢轴”(pivot)。

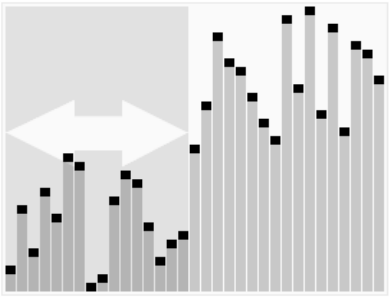
2.基于这个值，将数组分为两部分，较小的分在左边，较大的分在右边。

3.可以肯定，如此一轮下来，这个枢轴的位置一定在最终位置上。

4.对两个子数组分别重复上述过程，直到每个数组只有一个元素。

5.排序完成。

下面是快速排序的示意图（图片来自维基百科）：



代码实现

```
1  public static void quickSort(int[] arr){
2      qsort(arr, 0, arr.length-1);
3  }
4  private static void qsort(int[] arr, int low, int high){
5      if (low < high){
6          int pivot=partition(arr, low, high);        //将数组分为两部分
7          qsort(arr, low, pivot-1);                   //递归排序左子数组
8          qsort(arr, pivot+1, high);                  //递归排序右子数组
9      }
10 }
11 private static int partition(int[] arr, int low, int high){
12     int pivot = arr[low];     //枢轴记录
13     while (low<high){
14         while (low<high && arr[high]>=pivot) --high;
15         arr[low]=arr[high];             //交换比枢轴小的记录到左端
16         while (low<high && arr[low]<=pivot) ++low;
17         arr[high] = arr[low];           //交换比枢轴小的记录到右端
18     }
19     //扫描完成，枢轴到位
20     arr[low] = pivot;
21     //返回的是枢轴的位置
22     return low;
23 }
```

加友情链接

用户名：飞翔的猫咪
文章数：44
评论数：4
访问量：149143
无忧币：5023
博客积分：528
博客等级：3
注册日期：2013-04-25

热门专题 更多>>

Oracle零基础成长之路
阅读量：1297

原来你也在这里（征文）
阅读量：3317

从菜鸟到老鸟-教你玩转Mac操作系统
阅读量：450446

QT学习之路：从入门到精通
阅读量：1134998

热门文章

SVN cleanup操作反复失败..
用Java写算法之五：快速排序
Android使用ViewPager实..
Android 查看/data/data..
用Java写算法之四：归并排序
Android 数据库升级完整..
Android 开机自启动程序..
Android获得bitmap的大小

意见反馈

算法性能/复杂度

可以看出，每一次调用partition()方法都需要扫描一遍数组长度（注意，在递归的时候这个长度**并不是**原数组的长度n，而是被分隔出来的小数组，即n*(2^(-i))），其中i为调用深度。而在这一层同样长度的数组有2^i个。那么，每层排序大约需要O(n)复杂度。而一个长度为n的数组，调用深度最多为log(n)层。二者相乘，得到快速排序的平均复杂度为O(n logn)。

通常，快速排序被认为是在所有同数量级的排序方法中，平均性能最好。

从代码中可以很容易地看出，快速排序单个栈的空间复杂度不高，每次调用partition方法时，其额外开销只有O(1)。所以，最好情形下快速排序空间复杂度大约为O(logn)。

算法优化

上面这个快速排序算法可以说是最基本的快速排序，因为它并没有考虑任何输入数据。但是，我们很容易发现这个算法的缺陷：这就是在我们输入数据基本有序甚至完全有序的时候，这算法退化为冒泡排序，不再是O(nlogn)，而是O(n^2)了。

究其根源，在于我们的代码实现中，每次只从数组第一个开始取。如果我们采用“三者取中”，即arr[low],arr[high],arr[（low+high）/2]三者的中值作为枢轴记录，则可以大大提高快速排序在最坏情况下的性能。但是，我们仍然无法将它在数组有序情形下的性能提高到O(n)。还有一些方法可以不同程度地提高快速排序在最坏情况下的时间性能。

此外，快速排序需要一个递归栈，通常情况下这个栈不会很深，为log(n)级别。但是，如果每次划分的两个数组长度严重失衡，则为最坏情况，栈的深度将增加到O(n)。此时，由栈空间带来的空间复杂度不可忽略。如果加上额外变量的开销，这里甚至可能达到恐怖的O(n^2)空间复杂度。所以，快速排序的最差空间复杂度不是一个定值，甚至可能不在一个级别。

为了解决这个问题，我们可以在每次划分后比较两端的长度，并先对**短**的序列进行排序（目的是先结束这些栈以释放空间），可以将最大深度降回到O(logn)级别。

算法稳定性

快速排序并不是稳定的。这是因为我们无法保证相等的数据按顺序被扫描到和按顺序存放。

算法适用场景

快速排序在大多数情况下都是适用的，尤其在数据量大的时候性能优越性更加明显。但是在必要的时候，需要考虑下优化以提高其在最坏情况下的性能。

快排的非递归实现

按照通常的理论，我们知道递归算法一般比较直观自然，容易理解和书写；而非递归算法一般更为晦涩，但是性能比递归算法更优良，因为其省去了大量的函数调用开销。快速排序肯定有非递归实现的版本，例如这篇博客。有趣的是，这位作者认为快速排序的非递归实现比递归还要慢，并做出了分析。而下面的这位博主则写了另一篇博文，证明“非递归算法总要比响应(应为"相应"--本博作者注)的递归算法速度快”，并认为前面的现象是由于Windows 下的STL效率比较低。

快速排序的Java非递归实现当然有，通常都是用自己实现的栈来模拟递归操作（实际上，前面两位使用C++的同学也是如此做的）。但是我并不认为它们比递归的方式有极大的性能提升，反而丢失了可读性，晦涩难懂。因此个人不提倡使用非递归方式。

参考资料

1. 维基百科 http://zh.wikipedia.org/zh-cn/%E5%BF%AB%E9%80%9F%E6%8E%92%E5%BA%8F

上一篇 用Java写算法之四：归并排序 下一篇 用Java写算法之六：希尔排序

相关文章

c++快速排序算法
快速排序算法
用Java写算法之三：插入排序
快速排序算法
子网掩码快速算法
用Java写算法之四：归并排序
用Java写算法之二：选择排序
分治法——桶排序
用Java写算法之一：冒泡排序
无限式查找-----2013年2月28日

文章评论

发表评论

昵　　称 登录 快速注册

验证码： 请点击后输入验证码 博客过2级，无需填写验证码

内　　容

意见反馈