Java中Sort排序原理

排序算法	平均时间复杂度	最好情况	最坏情况	空间复杂度	排序方式	稳定性
冒泡排序	$O(n^2)$	$O(n)$	$O(n^2)$	$O(1)$	In-place	稳定
选择排序	$O(n^2)$	$O(n^2)$	$O(n^2)$	$O(1)$	In-place	不稳定
插入排序	$O(n^2)$	$O(n)$	$O(n^2)$	$O(1)$	In-place	稳定
希尔排序	$O(n \log n)$	$O(n \log^2 n)$	$O(n \log^2 n)$	$O(1)$	In-place	不稳定
归并排序	$O(n \log n)$	$O(n \log n)$	$O(n \log n)$	$O(n)$	Out-place	稳定
快速排序	$O(n \log n)$	$O(n \log n)$	$O(n^2)$	$O(\log n)$	In-place	不稳定
堆排序	$O(n \log n)$	$O(n \log n)$	$O(n \log n)$	$O(1)$	In-place	不稳定
计数排序	$O(n + k)$	$O(n + k)$	$O(n + k)$	$O(k)$	Out-place	稳定
桶排序	$O(n + k)$	$O(n + k)$	$O(n^2)$	$O(n + k)$	Out-place	稳定
基数排序	$O(n \times k)$	$O(n \times k)$	$O(n \times k)$	$O(n + k)$	Out-place	稳定

java中Arrays.sort使用了两种排序方法，快速排序和优化的合并排序。Collections.sort方法底层就是调用的Arrays.sort方法。

快速排序主要是对那些基本类型数据（int,short,long等）排序，而归并排序用于对Object类型进行排序。

使用不同类型的排序算法主要是由于快速排序是不稳定的，而归并排序是稳定的。这里的稳定是指比较相等的数据在排序之后仍然按照排序之前的前后顺序排列。对于基本数据类型，稳定性没有意义，而对于Object类型，稳定性是比较重要的，因为对象相等的判断可能只是判断关键属性，最好保持相等对象的非关键属性的顺序与排序前一致；另外一个原因是由于归并排序相对而言比较次数比快速排序少，移动（对象引用的移动）次数比快速排序多，而对于对象来说，比较一般比移动耗时。

此外，对大数组排序。快速排序的sort()采用递归实现，数组规模太大时会发生堆栈溢出，而归并排序sort()采用非递归实现，不存在此问题。

Sort执行流程

- 数组长度大于286，看数组具不具备结构。

  方法：每降序为一个组，像1,9,8,7,6,8。9到6是降序，为一个组，然后把降序的一组排成升序：1,6,7,8,9,8。然后最后的8后面继续往后面找。。。

  每遇到这样一个降序组，++count，当count大于MAX_RUN_COUNT（67），被判断为这个数组不具备结构（也就是这数据时而升时而降），然后送给之前的sort(里面的快速排序)的方法

  如果count少于MAX_RUN_COUNT（67）的，说明这个数组还有点结构，就继续往下走下面的归并排序：
- 首先先判断需要排序的数据量是否大于47。
  - 小于47：使用插入排序，插入排序是稳定的
  - 大于47的数据量会根据数据类型选择排序方式：
    - 基本类型：使用调优的快速排序（双轴快速排序）。
    - Object类型：使用改进的归并排序。因为归并排序具有稳定性。

不管是快速排序还是归并排序。在二分的时候小于47的数据量依旧会使用插入排序。