排序

高级知识点

- 稳定排序
  - 大小相同的值再排序完后相对位置仍然保持不变。
  - 归并排序
    - 排序10G大数据，先再各个主机内部排序，再用归并排序将各个主机排好的数据汇总，在归并排序时使用缓冲区（使用Iterable接口）缓冲归并和读取数据之间的速度差异。
  - 冒泡排序
  - 插入排序
  - 基数排序（最稳定）
- 不稳定排序
  - 选择排序
  - 快速排序
  - 希尔排序
  - 堆排序
-

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

- 堆
  - 根是最小的值（包括子树的根）
  - 是完全二叉树（从左到右铺子节点）
  - 优先队列
- 死锁
  - 必须满足
    - 互斥等待
    - hold and wait -->破除方法，一次性获取所有资源
    - 循环等待 -->按顺序获取资源
    - 等待无法被停止 --->加入超时
- nio的缺点
  - 由于是单线程，不适用于运算密集型
  - select系统需要轮询所有fd，最多1024个
  - 不同操作系统select底层实现不同
  - Java Nio会自动选择底层
- 引用分类
  - 强引用（ Final Reference）
    - 就是指在程序代码中普遍存在的，类似 Object obj = new Object() 这类的引用，只要强引用还存在，垃圾收集器永远不会回收掉被引用的对象。
    - 1. 强引用可以直接访问目标对象；
    - 2. 强引用锁指向的对象在任何时候都不会被系统回收。JVM宁愿抛出OOM异常也不回收强引用所指向的对象；
    - 3. 强应用可能导致内存泄露；
  - 软引用（Soft Reference）
    - 对于软引用关联着的对象，在系统将要发生内存溢出异常之前，将会把这些对象列进

回收范围之中进行第二次回收。如果这次回收还没有足够的内存，才会抛出内存溢出异常。软引用可用来实现内存敏感的高速缓存

- 弱引用（Weak Reference）
  - 被弱引用关联的对象只能生存到下一次垃圾收集发送之前。当垃圾收集器工作时，无论当前内存是否足够，都会回收掉只被弱引用关联的对象。弱引用可以用来在回调函数中防止内存泄露
- 虚引用（Phantom Reference）
  - 虚引用也称为幽灵引用或者幻影引用，它是最弱的一种引用关系。一个持有虚引用的对象，和没有引用几乎是一样的，随时都有可能被垃圾回收器回收。它的作用在于跟踪垃圾回收过程。虚引用能在这个对象被收集器回收时收到一个系统通知
-