10.排序优化

1.各个排序算法

	时间复杂度	是稳定排序?	是原地排序?
冒泡排序	$O(n^2)$	✓	✓
插入排序	$O(n^2)$	✓	✓
选择排序	$O(n^2)$	×	✓
快速排序	$O(n\log n)$	×	✓
归并排序	$O(n\log n)$	✓	×
计数排序	$O(n+K)$ K是数据范围	✓	×
桶排序	$O(n)$	✓	×
基数排序	$O(dn)$ d是维度	✓	×

2.快速排序的优化

问题：
如果数据原来就是有序的或者接近有序的，每次分区点都选择最后一个数据，那快速排序算法就会变得非常糟糕，时间复杂度就会退化为 O(n2)，出现这种情况的原因还是因为我们分区点选的不够合理

解决方法：
主要就是对递归过程中选择分区位的方式不同
1.三位取中：从区间的首、尾、中间，分别取出一个数，对比大小，取这 3 个数的中间值作为分区点。将中间值作为分区点的分区算法，肯定要比单纯取某一个数据更好。但是，如果要排序的数组比较大，那“三数取中”可能就不够了，可能要“五数取中”或者“十数取中”。
2. 随机法：随机法就是每次从要排序的区间中，随机选择一个元素作为分区点。这种方法并不能保证每次分区点都选的比较好，但是从概率的角度来看，也不大可能会出现每次分区点都选的很差的情况，所以平均情况下，这样选的分区点是比较好的。时间复杂度退化为最糟糕的 O(n2) 的情况，出现的可能性不大

问题2：为了避免快速排序里，递归过深而堆栈过小，导致堆栈溢出
第一种是限制递归深度。一旦递归过深，超过了我们事先设定的阈值，就停止递归。
第二种是通过在堆上模拟实现一个函数调用栈，手动模拟递归压栈、出栈的过程，这样就没有了系统栈大小的限制

3. 通用排序函数实现思路

通用排序函数实现技巧
1.数据量不大时，可以采取用时间换空间的思路
2.数据量大时，优化快排分区点的选择
3.防止堆栈溢出，可以选择在堆上手动模拟调用栈解决
4.在排序区间中，当元素个数小于某个常数是，可以考虑使用O(n^2)级别的插入排序
5.用哨兵简化代码，每次排序都减少一次判断，尽可能把性能优化到极致

XMind: ZEN | 试用模式