P405 2

a

快速排序的基本思想是选择一个元素作为基准，然后讲数组分成两部分，一部分包含小于基准的元素，另一部分包含大于基准的元素，然后再对这两个部分分别进行快速排序。

当输入列表已经是排好的时候，因为每次选择的基准是列表的第一个或者最后一个元素，那么分区操作就会产生最不平衡的结果，获得的子数组一个为空，一个包含其余所有元素，这样的话，每次递归只减少一个元素，递归深度变为n。在每一次递归上，需要遍历所有元素来进行分割，这需要时间n，因此总的时间复杂度变为*O*(*n*2)。

b

对于一个n个元素的数组来说，每次选取一个元素作为基准将一个数组分成两部分，最坏的情况就是选择的基准是最小的或者最大的元素，而如果每次都发生这种情况，那么递归的深度就为n，每次分区操作遍历所有元素，需要线性时间n，因此最坏的情况下，时间复杂度为*O*(*n*2)。

c

快速排序的基本思想就是选择一个基准，然后通过一系列的交换操作，将小于基准值的元素移动到基准左边，大于基准值的元素移动到基准的右边。当完成一遍操作之后，基准值就处于最终排序的位置上。

而在实际操作的过程中，使用left和right指针来进行操作，left指针一开始在数组的最左端，right在指针的最右端，将这两个指针从两端向中间移动，指导找到需要交换的元素，当left的位置超过right的位置时，说明所有的元素已经被检查过一遍，这时候分区已经完成，这个语句是判断一次递归内分区操作是否完成的条件。

P412 1

初始数组：12 2 16 30 8 28 4 10 20 6 18

归并一次：2 12 16 30 8 28 4 10 6 20 18

归并两次：2 12 16 30 4 8 10 28 6 18 20

归并三次：2 4 8 10 12 16 28 30 6 18 20

归并四次：2 4 6 8 10 12 16 18 20 28 30

P416 2

堆排序（Heap Sort）是一种不稳定的排序算法。所谓“不稳定”是指在排序过程中，具有相同键值（key）的记录可能会改变它们之间的相对顺序。这种特性在某些应用场景中可能是不希望出现的。下面用中文给出一个堆排序中不稳定性的例子。

假设我们有一个待排序的列表，其中包含有相等键值的元素，例如：`[(2, 'A'), (1, 'B'), (2, 'C'), (1, 'D')]`。这里，每个元组的第一个元素是键值，第二个元素是用来区分不同记录的附加信息。

在进行堆排序之前，列表中键值为 `2` 的两个记录 `(2, 'A')` 和 `(2, 'C')` 的顺序是 `'A'` 在 `'C'` 前面。但在堆排序过程中，由于堆排序只考虑键值而不考虑附加信息，所以当这两个记录被提取出来并放入排序后的位置时，它们之间的相对顺序可能会改变。

例如，在排序过程中，记录 `(2, 'C')` 可能会先于 `(2, 'A')` 被放置到排序后的列表中，导致最终排序结果变为 `[(1, 'B'), (1, 'D'), (2, 'C'), (2, 'A')]`。这里，尽管 `(2, 'A')` 和 `(2, 'C')` 的键值相同，但它们的相对顺序在排序后发生了改变。

这个例子展示了堆排序作为一种不稳定排序算法，在处理有相等键值的记录时可能不会保持它们原始的相对顺序。这就是为什么在需要保持记录之间相对顺序的应用中，稳定排序算法（如归并排序或冒泡排序）通常更为适用。