

1. (**T**, **F**) 递归式 $T(n) = 4T(n/4) + n \lg n$ 的解满足 $T(n) = \Theta(n^2 \lg n)$ 。
F 错误。

由主定理可得, $f(n) = \Theta(n^{\log_4 4} \lg n)$, 则 $T(n) = \Theta(n \lg^2 n)$ 。

2. (**T**, **F**) 堆排序和快速排序都是渐近最优的比较排序算法，他们在最坏情况下都需要做 $O(n \lg n)$ 次比较。

F 错误。

快速排序在最坏情况下都需要做 $O(n^2)$ 次比较。

3. (T, F) 选择排序的时间复杂度与输入数据的初始排列有关。

F 错误。

选择排序的时间复杂度是 $O(n^2)$ ，与输入数据的初始排列无关。无论输入数据是什么样的排列，选择排序都需要进行相同数量的比较和交换操作。

4. (T, F) 在使用 RANDOMIZED-SELECT 算法选择第 k 大的元素时, 算法的期望运行时间是线性的。

T 正确。

RANDOMIZED-SELECT 算法是随机化的选择算法, 其期望运行时间为线性。在每次划分中, 算法通过随机选择一个主元素, 然后将数组划分为两部分, 从而实现了平均情况下的线性运行时间。通过随机性的引入, RANDOMIZED-SELECT 能够避免最坏情况下的概率, 从而保证期望运行时间为 $O(n)$ 。

5. (T, F) 可以使用基于比较的算法, 以 $O(n)$ 的最坏时间复杂度将二叉堆转换为二叉搜索树。

F

如果这样的算法存在, 那么对于一个无序数组, 可以先用 $O(n)$ 时间建堆, 再调用该算法用 $O(n)$ 时间转换为二叉搜索树, 再用 $O(n)$ 时间进行中序遍历, 即得到排序后的数组, 总用时为 $O(n)$, 这与基于比较的排序算法的最坏时间复杂度下界是 $\Omega(n \lg n)$ 矛盾。

6. (T, F) 如果把一棵红黑树除根节点和叶子节点外所有节点的颜色反转后，所得到的仍是合法的红黑树，那么任意一个节点的左右子树节点数相等。

T

由红黑树性质以及颜色反转后的性质，任意一条从根节点到叶子节点的路径上红色节点和黑色节点的数量均相等，因此所有叶子节点的高度都相同，这说明这棵树的形状是完美对称的，因此任意一个节点的左右子树节点数相等。

1. (**T**, **F**) 递归式 $T(n) = 2T(n/5) + \sqrt{n}$ 的解满足 $T(n) = \Theta(n^{\log_5 2})$ 。
F 错误。

由主定理可得, 对于 $\varepsilon = 0.5 - \log_5 2$ 有 $f(n) = \Omega(n^{\log_5 2 + \varepsilon})$, 则 $T(n) = \Theta(\sqrt{n})$ 。

2. (T, F) 堆排序是一种不稳定的排序算法，给长度为 n 的数组 A 建立最大堆的时间复杂度为 $\Theta(n \log n)$ 。

F 错误。

给长度为 n 的数组 A 建立最大堆的时间复杂度为 $\Theta(n)$ 。

3. (T, F) 插入排序在最后一趟开始之前, 所有的元素可能都不在其最终的位置上。

T 正确。

在插入排序中, 如果待排序列中的最后一个元素其关键字值为最小, 则在最后一趟开始之前, 前 $n-1$ 个排好序的元素都不在其最终位置上, 与排好序后的位置相差一个位置。

4. (T, F) 在处理链表而不是数组的数据结构时，快速排序比归并排序更具有优势，因为它不需要额外的空间来进行合并。

F 错误。

在链表中使用归并排序不需要额外空间进行合并。同时链表不支持随机访问，因此快速排序无法高效地在链表中选择基准和进行分区，归并排序在链表上更具有优势。

5. (T, F) 某二叉树的后序和中序遍历序列正好一样, 则该二叉树中的任何结点一定都无右孩子。
T 正确。

后序: L, R, root, 中序: L, root, R, 要想两个序列一样, 一定无 R。

6. (T, F) 对于红黑树中的每一个结点 x , x 的左子树和右子树的高度最多相差 1。

F 错误。

题述是二叉平衡树，红黑树每个结点左子树高度不会超过右子树高度的 2 倍，右子树高度也不会超过左子树高度的 2 倍。

1. (T, F) 递归式 $T(n) = 2T(\frac{2n}{3}) + n \log n$ 的解满足 $T(n) = \Theta(n^2)$ 。
F 错误。

由主定理可得, 对于 $\varepsilon = \frac{1}{2} \log_{\frac{3}{2}} \frac{4}{3}$ 有 $f(n) = O(n^{\log_{\frac{3}{2}} 2 - \varepsilon}) = O(n^{1 + \log_{\frac{3}{2}} \frac{4}{3} - \varepsilon})$, 则 $T(n) = \Theta(n^{\log_{\frac{3}{2}} 2})$ 。

2. (T, F) 记 $\sigma(n)$ 表示数字 n 的因子个数, 例如 $\sigma(6) = 4$, 因为 $1, 2, 3, 6$ 都是 6 的因子。那么 $\sum_{i=1}^n \sigma(i)$ 是 $\Theta(n\sqrt{n})$ 的。

F 错误。

对于每个数字 i , 包含 i 这个因子的数字有 $i, 2i, 3i, \dots, \lfloor \frac{n}{i} \rfloor i$, 一共 $\lfloor \frac{n}{i} \rfloor$ 个, 求和就是:

$$\sum_{i=1}^n \frac{n}{i} = \Theta(n \log n)$$

等价于调和级数。

3. (**T**, **F**) 归并排序是稳定的, 其时间复杂度在任何情况下都一定是 $O(n\log n)$ 。
T 正确。
均是归并排序的基本结论。

4. (**T**, **F**) 计数排序 (Counting Sort) 的空间复杂度是 $O(1)$ 。
F 错误。计数排序需要 $O(n+k)$ 的空间。

5. (T, F) 某二叉树的前序、中序、后序遍历中，任意给定两项，都可以唯一确定第三项。

F 错误。

只有前序和后序时无法确定中序遍历，例如前序是 1,2，后序是 2,1，此时无法确定 2 是左儿子还是右儿子。

6. (T F) 对于一棵黑高为 h 的红黑树，其内部节点（也就是有键值的点）的数量至多是 $2^{h+1} - 1$ 。
F 错误。
至多为 $2^{2h} - 1$ ，构造从根开始黑红交替的满二叉树即可。