一、概念题：

（1）排序算法时间复杂度：

排序算法	最好	最坏	平均
插入	O(n)	$O(n^2)$	$O(n^2)$
归并	O(nlogn)	O(nlogn)	O(nlogn)
快排	O(nlogn)	$O(n^2)$	O(nlogn)

排序算法空间复杂度：

1、所有简单排序和堆排序都是 0(1)

2、快速排序为 0(logn)，要为递归程序执行过程栈所需的辅助空间

3、归并排序和基数排序所需辅助空间最多，为 O(n)

（2）渐近记号

1、渐近确界：Θ(g(n)) = { f(n)： 存在正常数 c1 和 c2 和 n0 ，使对所有的 n >= n_0，都有 0 <= c_1g(n) <= f(n) <= c_2g(n) }。大Θ记号给出函数的渐进确界。

2、渐近下界：Ω(g(n)) = { f(n)： 存在正常数 c 和 n0 ，使对所有的 n >= n0，都有 0 <= cg(n) <= f(n) }。大Ω记号给出函数的渐进下界。

3、渐近上界：O(g(n)) = { f(n)： 存在正常数 c 和 n0 ，使对所有的 n >= n0，都有 0 <= f(n) <= cg(n) }。大 O 记号给出函数的渐进上界。

（3）二叉查找树：

执行基本操作的时间与树的高度成正比。搜索、插入、删除的复杂度等于树高，期望 O(lgn)，最坏 O(n)（数列有序，树退化成线性表）

（4）红黑树：

1、时间复杂度：

基本动态集合操作：O(log n)，n 是树中元素的数目。

2、性质：

1) 节点是红色或黑色。
2) 根节点是黑色。
3) 每个叶节点（NIL 节点）是黑色的。
4) 如果一个结点是红的，则它的两个儿子都是黑的（不能有两个连续红结点）
5) 从任一节点到其子孙结点的所有路径都包含相同数目的黑色节点。

3、相关概念，定理：

1）黑高度：从某个结点出发（不包括该结点）到达一个叶结点的任意一条路径上，黑色结点的个数称为该结点 x 的黑高度，bh(x)。红黑树的黑高度定义为其根节点的黑高度。

2）一颗有 n 个内结点的红黑树的高度至多为 2lg(n+1)。(用 2-3-4 树理解)

3）在一颗黑高度为 K 的红黑树中，总结点数最多有 2^{2k+1}-1，此时内结点

最多为 2^{2k}-1（满二叉树，红黑交替），内结点最少有 2^k-1

4）RB-INSERT-FIXUP 操作所作的旋转不超过两次，RB-DELETE-FIXUP 所作的操作至多三次旋转

（5）动态规划：

1、装配线调度：FASTEST-WAY 时间复杂度 O（n）

2、矩阵链乘法：MATRIX-CHAIN-ORDER 时间复杂度 O（n3）

3、最长公共子序列：LCS-LENGTH 时间复杂度为 O（mn），m、n 为序列的长度

4、最优二叉查找树：OPTIMAL-BST 时间复杂度为 O（n3）

（6）贪心算法：

1、活动选择问题：初试时活动已按结束时间排序，O（n），否则可在 O（nlgn）内排序

2、哈夫曼编码：Q 用最小二叉堆实现，运行时间在 O（nlgn）

3、任务调度问题：时间复杂度为 O（n2），因为算法中 O（n）次独立性检查中每一次都有花 O（n）的时间

（7）二项堆：

1、可合并堆时间复杂度

过程	二叉堆（最坏）	二项堆（最坏）	Fibonacci（平摊）
MAKE-HEAP	Θ(1)	Θ(1)	Θ(1)
INSERT	Θ(lgn)	Ω(lgn)	Θ(1)
MINIMUM	Θ(1)	Ω(lgn)	Θ(1)
EXTRACT-MIN	Θ(lgn)	Θ(lgn)	O(lgn)
UNION	Θ(n)	Θ(lgn)	Θ(1)
DECREASE-KEY	Θ(lgn)	Θ(lgn)	Θ(1)
DELETE	Θ(lgn)	Θ(lgn)	O(lgn)

2、二项树 B_k 是一种递归定义的树，由两颗 B_{k-1} 连接而成，其中一颗树的根是另一颗树的根的最左孩子

性质：

1） 共有 2^k 个结点

2） 树的高度为 k

3） 在深度 i 处恰有(上 k,下 i)（因此叫二项树）个结点，其中 i=0,...,k；

4） 根的度数为 k，它大于任何其他结点的度数，并且，如果对根的子女从左到右编号为 k-1，k-2,...，0，子女 i 是子树 Bi 的根。

5） 在一颗包含 n 个结点的二项树中，任意结点的最大度数为 lgn

3、二项堆 H 由一组二项树构成，但需要满足下面两个性质：

1）H 中的每个二项树遵循最小堆的性质：结点的关键字大于等于其父结点

的关键字。（最小堆性质、度的唯一性）

2）对于任意非负整数 k，在 H 中至多有一棵二项树的根具有度数 k。

二、综合：

（1）分治法（自顶向下）

1、分解：将原问题分解成一系列子问题

2、解决：递归的解各子问题，若子问题足够小，则直接求解

3、合并：将子问题的结果合并成原问题的解

适用条件：

1、原问题可以分解为若干与原问题相似的子问题

2、子问题的解可以求出

3、子问题的解可以合并成原问题的解

4、分解出的子问题应相互独立，即没有重叠子问题

（3）合并排序（merge sort）：

1、分解：将 n 个元素分成各含 n/2 个元素的子序列；

2、解决：用合并排序法对两个子序列递归地排序；

3、合并：合并两个已排序的子序列以得到排序结果。

（4）动态规划（自底向上）：

1、描述问题的最优解结构特征

2、递归定义最优解值

3、自底向上计算最优解值

4、从已计算最优解值的信息中构造最优解结构

两个要素：最优子结构和重叠子问题

（5）贪心算法

1、确定问题的最优子结构性质

2、将优化的问题转化为一种选择，即贪心选择

3、贪心选择只能有一个子问题非空

4、证明贪心选择是正确的

两个要素：贪心选择性质和最优子结构

（6）主方法

$T(n) = a*T(n/b) + f(n)$　其中 $a\geq1$ 和 $b>1$ 是常数，$f(n)$是一个渐近正的函数。n 为非负整数，n/b 指 floor(n/b)或 ceiling(n/b)。那么 T(n)可能有如下的渐近界：

1、若对于某常数$\varepsilon>0$，有 $f(n) = O(n^{\log_b a-\varepsilon})$，则 $T(n)=\Theta(n^{\log_b a})$；

2、若 $f(n)=\Theta(n^{\log_b a})$，则 $T(n) = \Theta(n^{\log_b a} * \lg n)$；

3、若对某常数$\varepsilon>0$，有 $f(n)=\Omega(n^{\log_b a + \varepsilon})$，且对常数 $c<1$ 与足够大的 n，有 $a*f(n/b) \leq c*f(n)$，则 $T(n) = \Theta(f(n))$。

（7）将 41，38，31，12，19，8 插入到初试为空的红黑树