《算法设计与分析》复习

考试题型

填空 10x1’

选择 15x2’

简答 2x5’

大题 5x10’

1. 5 个重要特性： 输入 、 输出 、 有穷性 、 确定性 、 可行性
2. 5 个特性： 正确性 、 健壮性 、 可理解性 、 抽象分级 、 高效性
3. 描述方法： 自然语言 、 流程图 、 程序设计语言 、 伪代码
4. 问题类型： 查找问题 、 排序问题 、 图问题 、 组合问题 、 几何问题
5. 输入规模：输入量的多少
6. 算法就是一组有穷的规则，它们规定了解决某一特定问题的一系列运算
7. 算法是由若干条指令组成的有序序列，解决问题的一种方法或一个过程
8. 基本语句：执行次数与整个算法的执行次数成正比的语句，它对算法运行时间的贡献最大，是算法最重要的操作
9. 算法的时间复杂性分析 是一种事前分析估算方法，它是对算法所消耗资源的一种渐进分析方法。
10. 算法的空间复杂性 是指在算法的执行过程中需要的**辅助空间**数量，也就是除**算法本身**和**输入输出**数据所占用的空间外，算法**临时开辟**的存储空间，这个辅助空间数量也应该是输入规模的函数，通常记作：$S(n) = O(f(n))$。其中 n 为输入规模，分析方法与算法的时间复杂性类似。
11. 渐进分析 是指忽略具体机器、编程语言和编译器的影响，只关注在输入规模增大时算法运行时间的增长趋势。
12. 大 O 符号：当输入规模充分大时，算法中基本语句的执行次数在渐近意义下的阶，上界
13. 大Ω符号：问题的计算复杂性下界是求解这个问题所需的最少工作量，求解该问题的任何算法的时间复杂性都不会低于这个下界
14. 计算复杂性下界：对于任何待求解的问题，如果能找到一个尽可能大的函数

g(n)(n 为输入规模)，使得求解该问题的所有算法都可以在**Ω(g(n))**的时间内完成，则函数 **g(n)** 即是下界

15. 下界紧密：如果已经知道一个和下界的效率类型相同的算法，则该下界紧密
16. 扩展递归：一种常用的求解递推关系式的基本技术，扩展就是将递推关系式中等式右边的项根据递推式进行替换，扩展后的项被再次扩展，依此下去，会得到一个求和表达式，然后就可以借助于求和技术了
17. 最优算法：一般情况下，如果能够证明某问题的时间下界是**Ω(g(n))**，那么对以时间**O(g(n))**来求解该问题的任何算法，都认为是求解该问题的最优算法
18. 平凡下界：对问题的输入中必须要处理的元素进行计数，同时对必须要输出的元素进行计数。因为任何算法至少要“读取”所有要处理的元素，并“写出”它的全部输出
19. 蛮力法，即穷举法，枚举法。
20. 分治法求解过程的三个阶段是 划分 、 求解子问题 、 合并
21. 平衡子问题：将一个问题划分成规模相等的 k 个子问题，这种使子问题规模大致相等的做法称为**平衡子问题**（分治法）
22. 出自“平衡子问题”的思想，通常分治法在分割原问题，形成若干子问题时，这些子问题的规模都大致相同
23. 约束条件：该问题有 n 个输入，问题的解由这 n 个输入的一个子集组成，这个子集必须满足某些事先给定的条件，即约束条件
24. 可行解：满足约束条件的解
25. 目标函数：为了衡量可行解的优劣，通常以函数的形式给出一定的标准，这些标准函数称为目标函数
26. 最优解：使目标函数取得极值（极大或极小）的可行解称为最优解。这类问题称为**最优化问题**
27. 最优性原理：多阶段决策过程满足 xx：无论决策过程的初始状态和初始决策是什么，其余的决策都必须相对于初始决策所产生的当前状态，构成一个最优决策序列
28. 最优子结构性质的含义是问题的最优解包含其子问题的最优解
29. 由于**贪心法**并不是从整体最优考虑，它所做出的选择只是在某种意义上的局

部最优，这种局部最优选择并不总能获得**整体最优解**，但通常能获得**近似最优解**

30. 解空间：复杂问题常常有很多的可能解，这些可能解构成了问题的解空间，并且可能解的表示方式隐含了解空间及其大小。
31. 解空间树：即状态空间树，问题的解空间一般用解空间树表示。树的根结点位于第 1 层，表示搜索的初始状态，第 2 层的结点表示对解向量的第一个分量做出选择后到达的状态，1 和 2 层之间的边表示对第一个分量选择的结果。
32. 分支限界法常以 广度优先 或 以最小耗费（最大效益）优先 的方式搜索问题的解空间树
33. 在对问题的解空间树进行搜索的方法中，一个活结点有多次机会成为活结点的是（回溯法），只有一次的是（分支限界法）
34. 可/不可计算问题：Turing 论题，一个问题是可计算的当且仅当它在图灵机上经过有限步骤后得到正确的结果。
35. 易解问题：可以在多项式时间内求解的问题，如排序问题、串匹配问题
36. 难解问题：需要指数时间求解的问题，如汉诺塔问题、TSP 问题
37. 确定性算法：设 A 是求解问题Ⅱ的一个算法，如果在算法的整个执行过程中，每一步只有一个确定的选择，并且对于同一输入实例运行算法，所得的结果严格一致，则称算法 A 是确定性算法
38. P 类问题（多项式复杂程度问题）：如果对于某个判定问题Ⅱ，存在一个非负整数 k，对于输入规模为 n 的实例，能够以 $O(n^k)$ 的时间运行一个**确定性算法**，得到 yes 或 no 的答案
39. NP 类问题（多项式复杂程度的非确定性问题）：如果对于某个判定问题Ⅱ，存在一个非负整数 k，对于输入规模为 n 的实例，能够以 $O(n^k)$ 的时间运行一个**非确定性算法**，得到 yes 或 no 的答案
40. $P \subseteq NP$，解释：如果一个问题是 P 类问题，那么也可以构造一个非确定性算法，因此也是 NP 类问题。即 P 类问题包含在 NP 类问题中
41. NPC 问题：图着色、哈密顿回路问题、TSP 问题、顶点覆盖问题、子集和问题
42. NPH 问题：排列、组合、最优化

43. 问题的 最优子结构性质 是该问题可用动态规划算法或贪心算法求解的关键特征
44. 最优子结构是指问题的最优解包含了其子问题的最优解
45. 贪心选择性质 是贪心算法可行的第一个基本要素，也是贪心算法与动态规划算法的主要区别
46. 所谓贪心选择性质是指所求问题的整体最优解可以通过一系列局部最优解的选择，即贪心选择来达到
47. 贪心算法的基本要素是 贪心选择 性质和 最优子结构 性质
48. 动态规划算法的两个基本要素是 最优子结构性质和 重叠子问题 性质
49. 计算一个算法时间复杂度通常可以计算 循环次数 、 基本操作的频率 或计算步。
50. 利用概率的性质计算近似值的随机算法是数值概率算法，运行时以一定的概率得到正确解的随机算法是蒙特卡罗算法
51. 回溯法是一种既带有 系统性 又带有 跳跃性 的搜索算法
52. 回溯法搜索解空间树时，常用的两种剪枝函数为 约束函数 和 限界函数

查找问题

蛮力法：顺序查找、串匹配问题

减治法：折半查找、二叉查找树、第 k 小的元素（选择问题）

排序问题

蛮力法：选择排序、起泡排序

分治法：归并排序、快速排序

减治法：插入排序、堆排序

（简答题围绕 4-6 章）

分治法与减治法的区别

分治法将一个难以解决的大问题划分成一些规模较小的子问题，分别求解各个子问题，再合并子问题的解得到原问题的解

减治法同样是把一个大问题划分为若干个子问题，但是这些子问题不需要分别求解，只需求解其中的一个子问题，因而也无需对子问题的解进行合并。

分治法的设计思想

分治法将一个难以解决的大问题划分成一些规模较小的子问题，分别求解各个子问题，再合并子问题的解得到原问题的解

分治法与动态规划法的相同点

将待求解的问题分解成若干个子问题，先求解子问题，然后从这些子问题的解得到原问题的解。

两者的不同点是：适合于用动态规划法求解的问题，经分解得到的子问题往往不是互相独立的，是重叠的。而用分治法求解的问题，经分解得到的子问题往往是互相独立的。

分治法所能解决的问题一般具有的特征

(1) 该问题的规模缩小到一定的程度就可以容易地解决

(2) 该问题可以分解为若干个规模较小的相同问题，即该问题具有最优子结构性质

(3) 利用该问题分解出的子问题的解可以合并为该问题的解

(4) 原问题所分解出的各个子问题是相互独立的，即子问题之间不重叠

二路归并排序的分治策略

划分：将待排序序列 $r_1, r_2, \ldots, r_n$ 划分成两个长度相等的序列 $r_1, \ldots, r_{n/2}$ 和 $r_{n/2+1}, \ldots, r_n$；

求解子问题：分别对这两个子序列进行排序，得到两个有序子序列

合并：将这两个有序子序列合并成一个有序序列

快速排序的基本思想

在待排序的 N 个记录中任取一个记录，把该记录放在最终位置后，数据序列被此记录分成两部分。所有关键字小于等于该记录的放在该记录的左侧，大于该记录的在右侧，这个过程称为一次快速排序。之后重复上述过程，直到每一部分只有一个元素为止。

归并排序和快速排序的区别

归并排序按照记录在序列中的位置对序列进行划分，对子问题的解需要合并

快速排序按照记录的值对序列进行划分，各子问题的解无需合并。

动态规划的求解阶段

划分子问题：将原问题划分解成若干子问题，每个子问题对应一个决策阶段，并且子问题之间具有重叠关系

确定动态规划函数：根据子问题之间的重叠关系找到子问题满足的递推关系式（即动态规划函数），这是动态规划法的关键

填写表格：设计表格，以自底向上的方式计算各个子问题的解并填表，实现动态规划的过程

请说明动态规划方法为什么需要最优子结构性质

最优子结构性质是指大问题的最优解包含子问题的最优解。

动态规划方法是自底向上计算各个子问题的最优解，即先计算子问题的最优解，然后再利用子问题的最优解构造大问题的最优解，因此需要最优子结构.

算法 5 个重要特性：

输入：一个算法有零个或多个输入（即算法可以没有输入），这些输入通常取自于某个特定的对象集合

输出：一个算法有一个或多个输出（即算法必须要有输出），通常输出与输入之

间有着某种特定的关系

有穷性：一个算法必须总是（对任何合法的输入）在执行有穷步之后结束，且每一步都在有穷时间内完成

确定性：算法的每一条指令必须有确切的含义，不存在二义性。并且，在任何条件下，对于相同的输入只能得到相同的输出。

可行性：算法描述的操作可以通过已经实现的基本操作执行有限次来实现。

蛮力法的设计思想

蛮力法所依赖的基本技术是**遍历**（扫描），即采用一定的策略依次处理待求解问题的所有元素，从而找出问题的解。依次处理所有元素是蛮力法的关键，为了避免陷入重复试探，应保证处理过的元素不再被处理。典型的**指数时间算法**一般采用蛮力穷举。

回溯法的设计思想

回溯法在包含问题的所有可能解的解空间树中，从根结点出发，按照**深度优先**的策略进行搜索，对于解空间树的某个结点，如果该结点不满足问题的约束条件，则将以该结点为根结点的子树进行剪枝；否则进入该子树继续进行搜索，当搜索到一个满足条件的叶子结点时，就找到了一个可行解，对整个解空间树的搜索结束后，所有的可行解中的最优解就是该问题的最优解；适用于求解**组合数较大**的问题。

回溯法中常见的两类典型的解空间树是子集树和排列树。

当所给的问题是从 n 个元素的集合 S 中找出满足某种性质的子集时，相应的解空间树称为子集树。这类子集树通常有 2^n 个叶结点，遍历子集树需 $O(2^n)$计算时间 。

当所给的问题是确定 n 个元素满足某种性质的排列时，相应的解空间树称为排列树。这类排列树通常有 n!个叶结点。遍历排列树需要 O(n!)计算时间

分支限界法的设计思想

首先确定一个合理的**限界函数**，并根据限界函数确定目标函数的界[down, up]。然后，按照**广度优先策略**搜索问题的解空间树，在分支结点上，依次扩展该结点的所有孩子结点，分别估算这些孩子结点的目标函数的可能取值，如果取值超出目标函数的界，则将其丢弃；否则，将其加入待处理结点表（表 PT）中。依次从表 PT 中选取使目标函数取得极值的结点成为当前扩展结点，重复上述过程，直至找到最优解。适用于求解**最优化问题**。

分支限界法与回溯法的区别

方法	回溯法	分支限界法
对解空间树的搜索方式	深度优先	广度优先 或最小消耗（最大效益）
存储结点的数据结构	堆栈	队列、优先队列
结点存储特征	可重复成为活结点	一个结点只能成为活结点一次
常用应用	找出满足约束条件的所有解	找出满足约束条件的一个解或特定意义下的最优解

在算法复杂性分析中，O、Ω、Θ这三个记号的意义是什么？在忽略常数因子的情况

如果存在两个正常数 c 和 N_0,对于任意 $N \geq N_0$，有$|f(N)| \leq C|g(N)|$，则记作：$f(N)=O(g(N))$。这时我们说 $f(N)$的阶**不高于** $g(N)$的阶。

若存在两个正常数 C 和自然数 N0，使得当 $N \geq N0$ 时有$|f(N)| \geq C|g(N)|$，记为 $f(N)=\Omega(g(N))$。这时我们说 $f(N)$的阶**不低于** $g(N)$的阶。

如果存在正常数 c1，c2 和 n0，对于所有的 n≥n0，有 c1|g(N)| ≤|f(N)| ≤ c2|g(N)|则记作 $f(N)=\Theta(g(N))$

O、Ω、Θ分别提供了算法运行时间的上界、下界、平均

近似算法的基本思想

用近似最优解代替最优解，以换取算法设计上的简化和时间复杂性的降低。为了

具有实用性，近似算法必须能够给出算法所产生的解与最优解之间的差别，以保证任意一个实例的近似最优解与最优解之间相差的程度。

概率算法的设计思想

概率算法把“对于所有合理的输入都必须给出正确的输出”这一求解问题的条件放宽，允许算法在执行过程中随机选择下一步该如何进行，同时允许结果以较小的概率出现错误，并以此为代价，获得算法运行时间的大幅度减少。

舍伍德型概率算法

很多算法对于不同的输入实例，其运行时间差别很大。此时，可以采用舍伍德型概率算法来消除算法的时间复杂性与输入实例间的这种联系。舍伍德算法总能求得问题的一个解，且所求得的解总是正确的

拉斯维加斯概率算法

对同一个输入实例反复运行多次，直到运行成功，获得问题的解，如果运行失败，则在相同的输入实例上再次运行算法。它所做的随机性决策有可能导致算法找不到所需要的解。但如果给出解则必定正确。

蒙特卡罗型概率算法

求解正确解的概率依赖于算法的执行次数，算法的执行次数越多，得到正确解的概率就越高。蒙特卡罗算法能求得问题的一个解，但这个解未必是正确的。

分支限界法-TSP 问题 P175

U = (r1, r2, …, rk)

限界函数：$\boldsymbol{lb} = (\mathbf{2}\sum_{i=1}^{k-1} \boldsymbol{c}[\boldsymbol{r_i}][\boldsymbol{r_{i+1}}] + \sum_{i=1,k} \boldsymbol{r_i}$*行不在路径上的最小元素*

$+ \sum_{r_j \notin U} \boldsymbol{r_j}$*行最小的两个元素*$)/\mathbf{2}$

[down, up]，up 使用贪心法求，down 为每一行最小的两数之和全部相加，再除了 2

函数的含义是路径长度的两倍，加上首尾两行排除掉路径上的那个元素的最小元素，再加不在 U 中的点所在行的最小两个元素，最后除以 2

取极小值，每一轮计算结点，并加入 PT 表，一层搜索完后，从 PT 表中选取最小值的点，继续搜索。每轮要选取表 PT 中目标函数值最小的结点，如果相等，则取最上层的。

当目标函数值都相等，且出现叶子结点时，叶子结点所对应的解即是 TSP 问题的最优解（不必画出解空间树的所有结点，会来不及）

分支限界法-01 背包问题 P180

单位重量价值从大到小排序

限界函数：$\boldsymbol{ub} = \boldsymbol{V} + (\boldsymbol{W} - \boldsymbol{w}) \times (\boldsymbol{v_{i+1}}/\boldsymbol{w_{i+1}})$

求极大值，down 使用贪心法，up 假设全装第一个物品

动态规划法-多段路的最短路径 P102

首先求解初始子问题

d(0, 1) = c_{01} = 4 (0 -> 1)

d(0, 2) = c_{02} = 3 (0 -> 2)

…

求解第一阶段子问题

d(0, 3) = min{d(0, 1) + c_{13}, d(0, 2) + c_{23}}(2 -> 3)

…

最后将状态回溯，可得最短路径 0->3->5->8>9，长度为 16

注意，如果出现长度相等的情况，需要在每一括号内全部列出来

动态规划法-01 背包问题 P114

具体计算过程（填表过程）：

$$V(i,j)=\begin{cases}V(i-1,j) & j<w_i\\ \max\{V(i-1,j),\ V(i-1,j-w_i)+v_i\} & j\geq w_i\end{cases}$$

具体计算过程（回溯过程）：

$$x_i=\begin{cases}0 & V(i,j)=V(i-1,j)\\ 1,\ j=j-w_i & V(i,j)>V(i-1,j)\end{cases}$$

动态规划法-最长公共子序列 P111

长度矩阵 L：

$$L(i,j)=\begin{cases}0 & i=0\text{或}j=0\\ L(i-1,j-1)+1 & x_i=y_j,\ \text{且}i\geq 1,j\geq 1\\ \max(L(i,j-1),L(i-1,j)) & x_i\neq y_j,\ \text{且}i\geq 1,j\geq 1\end{cases}$$

相等为对角值+1，不相等取左上中的最大值

状态矩阵 S：

$$S(i,j)=\begin{cases}1 & x_i=y_j\\ 2 & x_i\neq y_j\text{且}L(i,j-1)\geq L(i-1,j)\\ 3 & x_i\neq y_j\text{且}L(i,j-1)<L(i-1,j)\end{cases}$$

长度矩阵中左边的值大于等于上面，则填 2

状态矩阵中取 1 即为最长公共子序列，从右下角开始，遇 1 往斜上方走

动态规划-近似串匹配（最短编辑距离）

首先要进行初始化：

$D(0,j)=j$，因为空样本与文本 t 有 j 处差别

$D(i,0)=i$，因为样本 p 与空文本有 i 处差别

状态转移方程如下：

$$D(i,j)=\begin{cases}\min\{D(i-1,j-1),D(i-1,j)+1,D(i,j-1)+1\} & i>0,j>0,p_i=t_j\\ \min\{D(i-1,j-1)+1,D(i-1,j)+1,D(i,j-1)+1\} & i>0,j>0,p_i\neq t_j\end{cases}$$

两个字符相等的情况下，斜对角不用+1

贪心法-背包问题 P138

（不是 01 背包）

按单位重量价值v[i]/w[i]降序排列，最后一个物品可能要部分装入

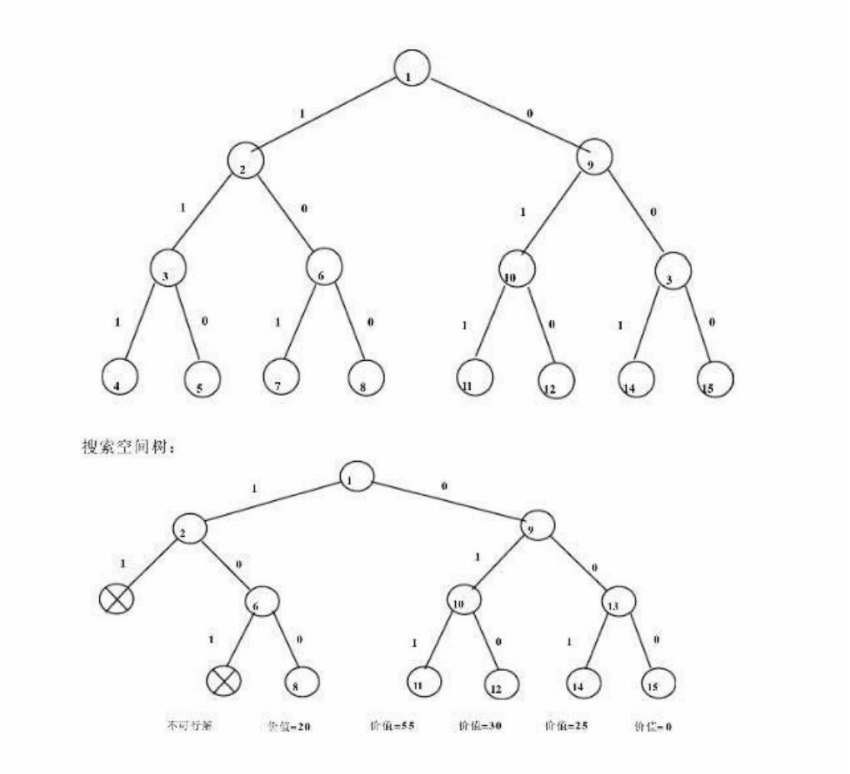
回溯法-图的三着色问题 P155

画解空间树时，顶点无特殊含义，主要在树边。树边代表着顶点的着色情况，如A=1。记住是深度优先遍历，若不发生冲突，则在该结点上往下搜索；冲突了才进行回溯。

回溯法-背包问题 P152

（注意区分解空间树和搜索空间树）

第 1 个根结点没有含义，边为 0 或 1，代表一个物品选或不选



最小生成树-贪心 P134

Prim

（类似 dijkstra 式寻找最近权值最低的边）

最近顶点策略：任选一个顶点，并以此为根结点，每一步的贪心选择是把不在生成树中的最近顶点添加到生成树中。

Kruskal

（边权升序排序+并查集）

设 G=(V, E)是一个无向连通网，令 T=(U, TE)是 G 的最小生成树。

最短边策略：从 TE={}开始，每一次贪心选择都是在边集 E 中选取最短边(u,v)，如果边(u,v)加入集合 TE 中不产生回路，则将边(u,v)加入边集 TE 中，并将它在集合 E 中删去。