算法设计与分析第四章学习指南

视频

https://www.icourse163.org/course/HIT-356006

(算法设计与分析(基础篇) 第四讲)

https://mooc.study.163.com/learn/1000005000?tid=1000005001#/learn/content

(算法设计与分析(进阶篇)第二讲，第三讲)

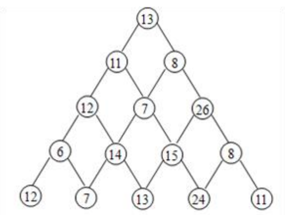
阅读

算法导论(第三版) 第 15 章，第 35.5 节，第 25 章

练习题

1. 满足递归式 $F(n)=F(n-1)+F(n-2)$和初始值 $F(0)=F(1)=1$ 的数列称为斐波那契数列。考虑如何计算该数列的第 n 项 $F(n)$。（1）说明根据递归式直接完成计算，将有子问题重复求解；（2）说明该问题具有优化子结构；（3）写出求解 $F(n)$的动态规划算法，并分析算法的时间复杂性。

2. 数字三角形问题：设有一个三角形的数塔，顶点为根结点，每个结点有一个整数值。从顶点出发，可以向左走或向右走，如图所示：



要求从根结点开始，请找出一条路径，使路径之和最大，只要输出路径的和。
要求：(1) 写出递推方程；(2)写出算法伪代码；(3) 分析算法的时间复杂度。

3. 给定一个 n×n 的矩阵 A，矩阵中的元素只取 0 或者 1。设计一个动态规划算法，求解得到 A 中元素全是 1 的子方阵使其阶数达到最大值。要求写出伪代码、递归方程并分析算法的时间复杂度。

4. 石子归并问题：在一个圆形操场的四周摆放着 n 堆石子，现要将石子有次序地合并成一堆。规定每次只能选择相邻的两堆石子合并成新的一堆，并将新一堆石子数记为该次合并的得分。试设计一个动态规划算法，计算出将 n 堆石子合并成一堆的最小得分和最大得分，要求列出递归方程，写出算法的伪代码并分析算法的计算复杂性。

5. 我们考虑将数轴上的 n 个点聚成 k 类的问题。

输入：n 个从小到大的不同实数 $x_1, x_2, \ldots, x_n$ 表示 n 个不同点，一个参数 $k \le n$.

任务：将 n 个点划分成 k 个不相交的非空集合 $S_1, ..., S_k$ 满足 $\bigcup_{i=1}^{k} S_i=\{x_1, x_2, ..., x_n\}$，$S_i$ 中所有点在 S_{i+1} 中所有点左边，$1\le i<k$，也就是说对于任意 $x\in S_i, z\in S_{i+1}, y<z$.
目标：最小化 $\sum_{i=1}^{k} cost(S_i)$,其中 $cost(S_i)=(max(S_i)-min(S_i))^2$. $max(S_i)$ 是 S_i 中的最小元素，$min(S_i)$ 是 S_i 中的最大元素。
例如，如果 $S_i=\{xj\}$，$cost(S_i)=0$，如果 $S_i=\{x_j, x_{j+1}, ..., x_{j+t}\}, x_j,<x_{j+1}< ...< x_{j+t}$，那么 $cost(S_i)=(x_{j+t}-x_j)^2$.

6. 假设书架上一共有 9 本书，每本书各有一定的页数，分配 3 个人来进行阅读。为了便于管理，分配时，各书要求保持连续，比如第 1、2、3 本书分配给第 1 人，4、5 分配给第二人，6，7，8，9 分配给第 3 人，但不能 1，4，2 分配给第 1 人，3，5，6 分配给第 2 人。即用两个隔板插入 8 个空隙中将 9 本书分成 3 部分，书不能换位。同时，分配时必须整本分配，同一本书不能拆成两部分分给两个人。为了公平起见，需要将工作量最大的那一部分最小化，请设计一个动态规划算法。用 s1,...,sn 表示各本书的页数。

（1）简明的写出问题的递推方程；
（2）描述算法伪代码；
（3）分析算法的时间复杂度。

7. 将一根木棒折成若干份，每折一次的代价是当前这段木棒的长度, 总代价是折这根木棒直到满足要求所需要的所有操作的代价。例如，将一根长度为 10 的木棒折成四段，长度分别为 2, 2, 3, 3，如果先折成长度为 2 和 8 的两段，再将长度为 8 的折成长度为 2 和 6 的两段，最后将长度为 6 的折成长度为 3 的两段，这些操作的代价是 10+8+6=24；如果先折成长度为 4 和 6 的两段，在分别将长度为 4 的折成长度为 2 的两段、长度为 6 的折成长度为 3 的两段，则这些操作的代价是 10+4+6=20，比上一种方案更好一些。该问题的输入是木棒的长度 L 和一些整数 c1,...,cn, 要求将木棒折成长度为 c1, ..., cn 的 n 段且操作代价最小，请设计动态规划算法解决该问题。

8. 给定一个浮点数序列（可能有正数、0 和负数），设计一个动态规划算法，求出一个最大的连续子序列中浮点数乘积。

（1）简明的写出问题的递推方程；
（2）描述算法伪代码；
（3）分析算法的时间复杂度。

9. 已知一个矩形区域被划分为 N*M 个小矩形格子，在格子(i,j)中有 A[i][j]个苹果。现在从左上角的格子(1,1)出发，要求每次只能向右走一步或向下走一步，最后到达(N,M)，每经过一个格子就把其中的苹果全部拿走。设计一个动态规划算法，找出能拿到最多苹果数的路线。

（1）简明的写出问题的递推方程；
（2）描述算法伪代码；
（3）分析算法的时间复杂度。
（4）将原问题变为“从起点出发，用相同的方法走两次，但是两条路线中除了起点和终点外不经过任何同一节点”，简明写出动态规划解决这个新问题的思路和递推方程。

10. 考虑三个字符串 X,Y,Z 的最长公共子序列 LCS(X,Y,Z)。

(1)寻找反例X,Y,Z 使得LCS(X,Y,Z)≠LCS(X, LCS(Y,Z));
(2)设计动态规划算法计算X,Y,Z 的最长公共子序列,分析算法的时间复杂度。

11. 令 I1, …, In 是 n 个区间，其中任一区间 Ii=(ai,bi)，假设这些区间按照 bi 从小到大排序，每一个区间有一个权重 vi,考虑如下两个问题：

区间安排问题 P1: 找到最大数量互不相交的区间，例如，对于四个区间 $I_1 = (1,2); I_2 = (2,3); I_3 = (1,4); I_4 = (4,5)$，一个解是$\{I_1, I_2, I_4\}$

加权任务安排问题 P2: 找一个互不相交区间的集合，使得这些区间的权重之和最大，例如 $I_1 = (1,2), v_1=0.9; I_2 = (2,3), v_2=0.5; I_3 = (1,4), v_3=4; I_4 = (4,5), v_4=2$，解是$\{I_3, I_4\}$。

(1) 给出解决问题 P1 的线性时间算法。

(2) 给出解决问题 P2 的动态规划算法，要求写出递归方程和伪代码，并分析算法时间复杂性。

12. 编辑距离问题

(a) 设计动态规划算法计算字符串 s1 和 s2 的扩展编辑距离。字符串 s1 和 s2 的扩展编辑距离是 s1 变换到 s2 的最少操作个数，其中操作包括 1)插入一个字符；2)删除一个字符；3)修改一个字符；4)交换两个相邻的字符。要求写出解的递推方程、算法的伪代码并分析算法的时间复杂度。

(b) 考虑字符串变换操作，增加一个字符，删除一个字符以及修改一个字符，设增加字符操作的代价为 i, 删除字符操作代价为 d, 修改字符的代价为 m，给定两个字符串 S_1 和 S_2，设计一个动态规划算法，求得从 S_1 变换到 S_2 代价最小的变换序列，要求写出递推方程，程序伪代码并分析算法复杂性。

(c) 考虑编辑距离的一种变形，其允许在字符串后无代价地插入无限多个字符，该编辑距离描述为:

ed'(A, B)=min{ed(A, C)|C是B的前缀}, 其中函数ed()是普通的编辑距离函数。根据要求设计算法，要求算法的时间复杂性都是O(|A||B|)

(1) 设计算法，对于给定的字符串A和B，计算ed'(A, B)；

(2) 设计算法，对于给定的字符串A,B和整数k，判定是否B存在某个后缀B’，满足ed’(A, B’)≤k。

13. T1 和 T2 是两棵有序树，其中每个结点都有一个标签，考虑树上的三种操作，删除一个子树、插入一个子树和更改一个结点的标签，请设计一个算法，求得从 T1 变化到 T2 所需要的最少操作数，要求写出递推方程，程序伪代码并分析时间复杂性。

14. 设计动态规划算法输出数组 A[0:n]中的最长单调递增子序列。

15. 给定 n 种物品和一背包，物品 i 的重量是 wi,其价值是 vi, 背包的容量是 c, 设计一个完全多项式时间近似算法，求装入背包的物品集合，使得装入背包中物品的总价值最大。要求设计出算法并对其近似比进行分析，说明为何此算法是完全多项式时间近似模式的算法。

16. 考虑特殊的 0-1 背包问题：有 n 个物品，每个物品 i 价值和重量都是 wi，背包能容纳物品的最大重量是 C, 选择背包能容纳的物品集合，使得这些物品价值之和最大。回答下列问题:

(1) 若物品的重量(价值)分别是1, 2, …,2^{n-1}, 证明该0-1背包问题可以用贪心法求解并写出该贪心算法的伪代码。

(2) 请写出一个物品重量(价值)序列，使得上述贪心法无法得到最优解。

17. 正整数 n 可以拆分成若干个正整数之和，考虑拆分方案的个数。

(1) 令g(i,j)表示拆分整数i时最大加数不超过j的方案个数， 证明：g(i,j)=g(i,j-1)+g(i-j,j)。

(2)根据(1)设计动态规划算法计算整数n的拆分方案个数，要求算法的时间复杂度为$O(n^2)$

18. 设计时间复杂度为 O(n2k)的动态规划算法，找到最优聚类。要求写出伪代码、递归方程并分析算法的时间复杂度。

例如，考虑将 4 个元素的集合{1,5,8,10}聚为两个类，有三种可能:

1. S_1={1}, S_2={5,8,10}，总代价是 $0^2+5^2=25$
2. S_1={1,5}, S_2={8,10}，总代价是 $4^2+2^2=20$
3. S_1={1,5,8}, S_2={10}，总代价是 $7^2+0^2=49$

所以，算法的解是最优解 S_1={1,5}, S_2={8,10}。

19. 我们考虑将数轴上的 n 个点聚成 k 类的问题。

输入：n 个从小到大的不同实数 $x_1, x_2, \ldots, x_n$ 表示 n 个不同点，一个参数 $k \le n$.

任务：将 n 个点划分成 k 个不相交的非空集合 $S_1, \ldots., S_k$ 满足$\bigcup_{i=1}^{k} S_i=\{x_1, x_2, \ldots, x_n\}$，$S_i$ 中所有点在 S_{i+1} 中所有点左边，$1 \le i < k$，也就是说对于任意 $x \in S_i$, $z \in S_{i+1}$, $y<z$.

目标：最小化$\sum_{i=1}^{k} cost(S_i)$,其中 $cost(S_i)=(\max(S_i)-\min(S_i))^2$. $\max(S_i)$是 S_i 中的最小元素，$\min(S_i)$是 S_i 中的最大元素。

例如，如果 S_i={xj}，$cost(S_i)=0$，如果 $S_i=\{x_j, x_{j+1}, \ldots, x_{j+t}\}$, $x_j,<x_{j+1}< \ldots< x_{j+t}$，那么 $cost(S_i)=(x_{j+t}-x_j)^2$.

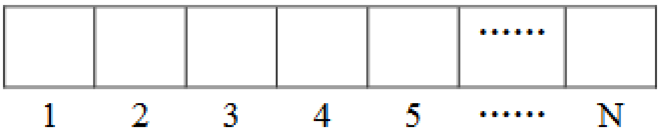
设计时间复杂度为 $O(n^2k)$的动态规划算法，找到最优聚类。要求写出伪代码、递归方程并分析算法的时间复杂度。

例如，考虑将 4 个元素的集合{1,5,8,10}聚为两个类，有三种可能:

1. S_1={1}, S_2={5,8,10}，总代价是 $0^2+5^2=25$
2. S_1={1,5}, S_2={8,10}，总代价是 $4^2+2^2=20$
3. S_1={1,5,8}, S_2={10}，总代价是 $7^2+0^2=49$

所以，算法的解是最优解 S_1={1,5}, S_2={8,10}。

20. 乌龟棋：乌龟棋的棋盘是一行 N 个格子，每个格子上一个分数（非负整数）。棋盘第 1 格是唯一的起点，第 N 格是终点，游戏要求玩家控制一个乌龟棋子从起点出发走到终点。



乌龟棋中M张爬行卡片，分成4种不同的类型（M张卡片中不一定包含所有4种类型 的卡片，见样例），每种类型的卡片上分别标有1、2、3、4四个数字之一，表示使用这种卡 片后，乌龟棋子将向前爬行相应的格子数。游戏中，玩家每次需要从所有的爬行卡片中选择 一张之前没有使用过的爬行卡片，控制乌龟棋子前进相应的格子数，每张卡片只能使用一次。 游戏中，乌龟棋子自动获得起点格子的分数，并且在后续的爬行中每到达一个格子，就得到 该格子相应的分数。玩家最终游戏得分就是乌龟棋子从起点到终点过程中到过的所有格子的分数总和。很明显，用不同的爬行卡片使用顺序会使得最终游戏的得分不同。现已知棋盘上每个格子的分数和所有的爬行卡片，求得一种卡片使用顺序使得最终游戏得分最多。

例：有9个格子上的分数分别为6 10 14 2 8 8 18 5 17，有卡片1 1 1 2 3，则使用爬行卡片顺序

为1,1,3,1,2,得到的分数为6+10+14+8+18+17=73。注意，由于起点是11，所以自动获得第11格的分数6。

21. 天灾狼群：来自东方王国的冒险家马特（Matt）遇到了了天灾狼群。有连续的 N 匹狼（从左到右从 1 到 N 编号）排成一排。马特必须击杀所有狼才能生存。每匹狼（第 i 匹狼）拥有两种属性：自身的攻击力（记作 Ai），和光环攻击力（记作 Bi）。一匹狼的实际攻击力是其自身的攻击力 Ai 加上与其相邻的两匹狼为其提供的攻击力加成（即 Ai+Bi-1+Bi+1）。马特每次只能击杀一匹狼，并且在击杀时会受到等同于这匹狼实际攻击力的伤害。击杀一匹狼后，狼之间的相邻关系会发生变化，被击杀的狼相邻的两匹狼成为新的相邻关系。求得一种击杀顺序使得马特受到的总伤害最低。

例如，假设连续有3匹狼，它们的基本攻击力Ai分别为（3、5、7）。他们的光环攻击力分别是（8，2，0）。它们当前的攻击是（5，13，9）。如果马特首先击败第二只狼，他将受到13点伤害，而活着的两匹狼变得相邻，其当前的攻击力变为（3，15）。

22. Flappy Bird：Flappy Bird 是一款风靡一时的休闲手机游戏。玩家需要不断控制点击手机屏幕的频率来调节小鸟的飞行高度，让小鸟顺利通过画面右方的管道缝隙。如果小鸟一不小心撞到了水管或者掉在地上的话，便宣告失败。为了简化问题，我们对游戏规则进行了简化和改编：

(1) 游戏界面是一个长为n，高为m的二维平面，其中有k个管道（忽略管道的宽度）。

(2) 小鸟始终在游戏界面内移动。小鸟从游戏界面最左边任意整数高度位置出发，到达游戏界面最右边时，游戏完成。

(3) 小鸟每个单位时间沿横坐标方向右移的距离为1，竖直移动的距离由玩家控制。如果点击屏幕，小鸟就会上升一定高度X，每个单位时间可以点击多次，效果叠加；如果不点击屏幕，小鸟就会下降一定高度Y。小鸟位于横坐标方向不同位置时，上升的高度X和下降的高度Y可能互不相同。

(4) 小鸟高度等于0或者小鸟碰到管道时，游 戏 失 败 。小 鸟 高 度 为m时，无法再上升。

现在，请你判断是否可以完成游戏。如果可以，求解能够完成游戏的最少点击屏幕数；否则，求解小鸟最多可以通过多少个管道缝隙。

23. 阿鲁高的阴谋：银溪镇的追随者们为阿鲁高制造了一个船，用于在海岸上扩张。在漫长的旅途中，阿鲁高必须消耗法力创造结界以存储燃料。在途中每一天，阿鲁高都可以驶到一个联盟港口骗取燃料，代价是消耗法力维持幻象。阿鲁高的目标要航行 T 天才能到达。由于天气和海域的不同，每天航行所需的燃料消耗也不同。每天都可以到达一个联盟港口，骗取一些燃料（也可以不去，从结界中取得燃料补充所需；注意，每天的消耗必须被满足），骗取每个单位燃料需要一定的法力消耗。如果骗取的燃料除供这一天消耗外还有剩余，则必须把它存到法力结界中，但是限于结界规模，只能把不超过 V 的燃料存储到结界中。在结界中存储的每个单位的燃料每天会消耗掉阿鲁高的 W 点法力值。为了留有足够多的法力，阿鲁高必须尽量地减少法力消耗。请你算出阿鲁高到达目标最少的法力消耗是多少。

例：总天数T==2，结界的最大存储量V==5，每个港口的库存A==10，结界中每存储一单位燃料一天的法力消耗==5。第一天消耗燃料5，骗取每单位消耗20法力；第二天消耗燃料3，骗取每单位消耗30法力。则最优解为在第一天骗取8单位燃料，5点消耗于当天的行程，3点存储到结界中。总法力消耗为160+15==175。

24. 正整数 n 可以拆分成若干个正整数之和，考虑拆分方案的个数。

(1) 令g(i,j)表示拆分整数i时最大加数不超过j的方案个数， 证明：g(i,j)=g(i,j-1)+g(i-j,j)。

(2)根据(1)设计动态规划算法计算整数n的拆分方案个数，要求算法的时间复杂度为$O(n^2)$

25. 设 R(X)表示将整数 X 的各个数位取逆序后得到的整数，如 R(123)=321, R(120)=21。现输入正整数 N，试设计一个动态规划算法计算方程 R(X)+X=N 的解的个数，分析算法的时间复杂度。

26. 输入数组 A[0:n]和正实数 d,试设计一个动态规划算法输出 A[0:n]的一个最长子序列，使得子序列中相继元素之差的绝对值不超过 d。分析算法的时间复杂度。

27. 给定一个整数序列 a1,…,an。相邻两个整数可以合并，合并两个整数的代价是这两个整数之和。通过不断合并最终可以将整个序列合并成一个整数，整个过程的总代价是每次合并操作代价之和。试设计一个动态规划算法给出 a1,…,an 的一个合并方案使得该方案的总代价最大。你的答案应包括：（1）用简明的语言表述这个问题的优化子结构；（2）根据优化子结构写出代价方程；（3）根据代价方程写出动态规划算法（伪代码）并分析算法的时间复杂性。

28. 输入表达式 a1O1 a2 O2……On-1 an，其中 ai 是整数(1<=i<=n),Oj∈{+,-,×}(1<=i<=n-1)。

(1)试设计一个动态规划算法，输出一个带括号的表达式（不改变操作数和操作符的次序），使得表达式的值达到最大，分析算法的时间复杂性。

(2)令Oj∈{+,-,×,÷},重新完成(1)规定各项任务。

29. 输入平面上 n 个点，点与点之间的距离定义为欧几里得距离。试设计一个动态规划算法输出一条先从左到右再从右到左的一条最短路径，使得每个输入点恰好被访问一次。

30. 输入凸 n 边形 p1,p2,…pn，其中顶点按凸多边形边界的逆时针序给出，多边形中不相邻顶点间的连线称为弦。试设计一个动态规划算法，将凸边形 p1,p2,…pn 剖分成一些无公共区域的三角形，使得所有三角形的周长之和最小。

31. 输入一棵加权树 T,其中每条边的权值均是实数，试设计一个动态规划算法输出权值最大的子树。