算法设计与分析

CS2008 班 U202015533 徐瑞达

2022.04.20

目录

1　动态规划作业

1.1　计算题

1.1.1　15.2-1

Question:

对矩阵规模序列 $(5, 10, 3, 12, 5, 50, 6)$**，求矩阵链最优括号化方案**。

Answer:

$((5 \times 10)(10 \times 3))(((3 \times 12)(12 \times 5))((5 \times 50)(50 \times 6)))$.

1.1.2　15.4-1

Question:

求 $\langle 1, 0, 0, 1, 0, 1, 0, 1 \rangle$ **和** $\langle 0, 1, 0, 1, 1, 0, 1, 1, 0 \rangle$ **的一个** LCS。

Answer:

$\langle 1, 0, 0, 1, 1, 0 \rangle$ **或者** $\langle 1, 0, 1, 0, 1, 0 \rangle$.

1.1.3　15.5-2

Question:

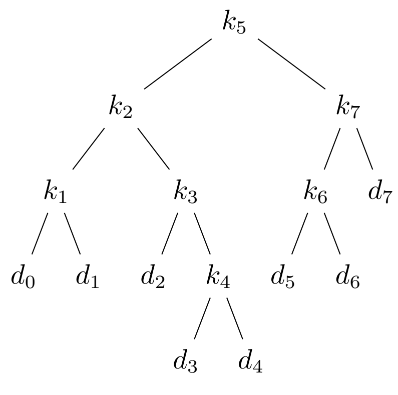
若 7 个关键字的概率如下表所示，求其最优二叉搜索树的结构和代价。

i	0	1	2	3	4	5	6	7
p_i		0.04	0.06	0.08	0.02	0.10	0.12	0.14
q_i	0.06	0.06	0.06	0.06	0.05	0.05	0.05	0.05

Answer:

代价为 3.12。

结构如下:



1.2 算法设计题

1.2.1 15.1-3

Question:

我们对钢条切割问题进行一点修改，除了切割下的钢条段具有不同价格 p_i 外，每次切割还要付出固定的成本 c。这样，切割方案的收益就等于钢条段的价格之和减去切割的成本。设计一个动态规划算法解决修改后的钢条切割问题。

Answer:

修改 BOTTOM-UP-CUT-ROD 算法如下: 除了最后一次迭代 (此时 $i=j$，没有切割)，都需要在循环中的每次迭代中考虑成本 c，令循环在 $j-1$ 处即结束而非 j，保证 c 从候选收入中减去，然后选择当前最佳收入 q 和 $p[j]$ 中的较大值。

1.2.2 15-9

Question:

（字符串拆分）某种字符串处理语言允许程序员将一个字符串拆分为两段。由于此操作需要复制字符串，因此要花费个时间单位来将一个 n 个字符的字符串拆为两段。假定一个程序员希望将一个字符串拆分为多段，拆分的顺序会影响所花费的总时间。例如，假定这个程序员希望将一个 20 个字符的字符串在第 2 个、第 8 个以及第 10 个字符后进行拆分 (字符由左至右，从 1 开始升序编号)。如果她按由左至右的顺序进行拆分，则第一次拆分花费 20 个时间单位，第二次拆分花费 18 个时间单位（在第 8 个字符处拆分 3 20 间的字符串)，而第三次拆分花费 12 个时间单位，共花费 50 个时间单位。但如果她按由右至左的顺序进行拆分，第一次拆分花费 20 个时间单位，第二次拆分花费 10 个时间单位，而第三次拆分花费 8 个时间单位，共花费 38 个时间单位。还可以按其他顺序，比如，她可以首先在第 8 个字符处进行拆分 (时间 20)，接着在左边一段第 2 个字符处进行拆分 (时间 8)，最后在右边一段第 10 个字符处进行拆分 (时间 12)，总时间为 40。
设计算法，对给定的拆分位置，确定最小代价的拆分顺序。更形式化地，给定一个个字符的字符串 S 和一个保存 m 个拆分点的数组 $L[1\ldots m]$，计算拆分的最小代价，以及最优拆分序列。

Answer:

可以根据将数组分割为连续的子数组将问题分为若干个子问题，并尝试使用最小的花费来完成每次可能的分割。因为需要进行 m 次拆分，而且最多有 m^2 中可能的拆分顺序，因此解的复杂度为 $O(m^3)$，另外，因为每次添加数字的复杂度为 $O(n)$，循环中的每次迭代会消耗时间 $O(\lg n+\lg m)$，因此，最终的运行时间为 $O(m^3\lg n)$。该算法会返回拆分的最小代价和最优拆分序列。
例如，对于拆分点数组 $L=[2,5,10]$，$S=25$，调用方法为 Algorithm(L,0,len(L),0,S)。

1.2.3 15-11

Question:

（库存规划）Rinky Dink 公司是一家制造溜冰场冰面修整设备的公司。这种设备每个月的需求量都在变化，因此公司希望设计一种策略来规划生产，需求是给定的，即它虽然是波动的，但是可预测的。公司希望设计接下来 n 个月的生产计划。对第 i 个月，公司知道需求 d, 即该月能够销售出去的设备的数量。令

$$D=\sum_{i=1}^{n} d_i$$

，为后 n 个月的总需求。公司雇用的全职员工，可以提供一个月制造 m 台设备的劳动力。如果公司希望一个月内制造多于 m 台设备，可以雇用额外的兼职劳动力，雇用成本为每制造一台机器付出 c 美元。而且，如果在月末有设备尚未售出，公司还要付出库存成本。保存 j 台设备的成本可描述为一个函数 $h(j)$，$j = 1, 2, \ldots, D$，其中对所有 $1 \leq j \leq D$，$h(j) \geq 0$，对 $1 \leq j \leq D-1$，$h(j) \leq h(j+1)$。
设计库存规划算法，在满足所有需求的前提下最小化成本。算法运行时间应为 n 和 D 的多项式函数。

Answer:
可以根据整数 $i \in [n]$ 和 $j \in [D]$ 划分子问题。i 表示已经过去了多少个月，也即，只需考虑 $(d_i, \ldots, d_n)$，j 表示我们最初储备了多少台设备，然后使用递推式得出从 1 到 $[D]$ 中所有可能的制造设备数。因为子问题数量复杂度为 $O(nD)$，而且当解决某个子问题时，我们只需从 D 种选择中选取花费最小的方案，因此最终的运行时间为 $O(nD^2)$。

1.3 证明题

1.3.1 15.2-5

Question:
令 $R(i,j)$ 表示在一次调用 **MATRIX-CHAIN-ORDER** 过程中，计算其他表项时访问表项 $m[i,j]$ 的次数。证明：

$$\sum_{i=1}^{n}\sum_{j=i}^{n} R(i,j) = \frac{n^3-n}{3}$$

Answer:

$$\begin{aligned}
\sum_{l=2}^{n}\sum_{i=1}^{n-l+1}\sum_{k=i}^{i+l-2} 2 &= \sum_{l=2}^{n}\sum_{i=1}^{n-l+1} 2(l-1) \\
&= \sum_{l=2}^{n} 2(l-1)(n-l+1) \\
&= \sum_{l=1}^{n-1} 2l(n-l) \\
&= 2n\sum_{l=1}^{n-1} l - 2\sum_{l=1}^{n-1} l^2 \\
&= n^2(n-1) - 2\cdot\frac{(n-1)n(2n-1)}{6} \\
&= n^3 - n^2 - \frac{2n^3-3n^2+n}{3} \\
&= \frac{n^3-n}{3}
\end{aligned}$$

1.3.2 15.3-6

Question:

假定你希望兑换外汇，你意识到与其直接兑换，不如进行多种外币的一系列兑换，最后兑换到你想要的那种外币，可能会获得更大收益。假定你可以交易 n 种不同的货币，编号为 $1, 2, \ldots, n$，兑换从 1 号货币开始，最终兑换为 n 号货币。对每两种货币 i 和 j，给定汇率 r_{ij}，意味着你如果有 d 个单位的货币 i，可以兑换 dr_{ij} 个单位的货币 j。进行一系列的交易需要支付一定的佣金，金额取决于交易的次数。令 c_k 表示 k 次交易需要支付的佣金。证明：如果对所有 $k = 1, 2, \ldots, n, c_k = 0$，那么寻找最优兑换序列的问题具有最优子结构性质。然后请证明：如果佣金 c_k 为任意值，那么问题不一定具有最优子结构性质。

Answer:

首先假设佣金为 0，设 k 表示最佳交易序列 s 中出现的货币，取值为 1 n 号，p_k 表示该序列中的第一部分 (货币从 1 号到 k 号)，而 q_k 表示该序列中的其余部分。那么 p_k 和 q_k 分别表示从 1 号到 k 号和从 k 号到 n 号的最佳交易序列。为了证明之，假设 p_k 不是最优的，而 $p_k^{'}$ 是最优的。那么通过序列 $p_k^{'}q_k$ 得到的序列将比 s 更优，这与 s 是最优的相矛盾，q_k 同理可证。
现在假设佣金可以取任意值，假设有 1 号到 6 号的货币，且 $r_{12} = r_{23} = r_{34} = r_{45} = 2, r_{13} = r_{35} = 6$，其余交易均为 100，令 $c_1 = 0, c_2 = 1$，且对于 $k \geq 3$ 有 $c_k = 10$。
在这种情况下的最优解是 $1 \to 3, 3 \to 5$，此时花费为 13。而 $1 \to 3$ 的最优解为 $1 \to 2, 2 \to 3$，此时花费为 5。$3 \to 5$ 的最优解为 $3 \to 4, 4 \to 5$，此时花费为 5。组合这两个子情况最优解时需要花费更多代价，总花费为 18，这不是最优解，因此不具有最优子结构性质。

2 贪心作业

2.1 题目

2.1.1 16.1-4

Question:

假定有一组活动，我们需要将它们安排到一些教室，任意活动都可以在任意教室进行。我们希望使用最少的教室完成所有活动。设计一个高效的贪心算法求每个活动应该在哪个教室进行。(这个问题称为 区间图着色问题(interval-graph color problem)。我们可以构造一个区间图，顶点表示给定的活动，边连接不兼容的活动。要求用最少的颜色对顶点进行着色，使得所有相邻顶点颜色均不相同—这与使用最少的教室完成所有活动的问题是对应的。)

Answer:

用 F 表示可供使用且已经被使用过的一组教室，用 B 表示当前繁忙的一组教室。以开始时间对这些活动进行排序，当遇到一个新的活动时，从 F 中移除一个教室，并在该教室中安排该活动，将其加入 B 中。如果 F 为空，向 F 中添加一个从未被使用的教室，当一个活动结束时，将其从 B 中移除并放到 F 中，这就是最优方案。

2.1.2 16.2-7

Question:

给定两个集合 A 和 B，各包含 n 个正整数。你可以按需要任意重排每个集合。重排后，令 a_i 为集

合 A 的第 i 个元素，b_i 为集合 B 的第 i 个元素。于是你得到回报

$$\prod_{i=1}^{n} a_i^{b_i}$$

设计算法最大化你的回报。证明你的算法是正确的，并分析运行时间。

Answer:

由于集合具有无序性，不影响回报，可假设 A 是升序排列。则可以证明当 B 也是升序排列时回报最大化。假设不是如此，也即，存在 $i < j$ 使得 $a_i < a_j$ 且 $b_i > b_j$，然后只考虑 i 和 j 对回报的贡献，也即，$a_i^{b_i} a_j^{b_j}$，然后，如果交换 b_i 和 b_j 的次序，那么贡献为 $a_i^{b_j} a_j^{b_i}$，可见后者比前者大，因为有 $(\frac{a_j}{a_i})^{b_i - b_j} > 1$，因此，可得在这种次序下得到的回报不是最大化的。

2.1.3 16.3-3

Question:

如下所示，8 个字符对应的出现频率是斐波那契数列的前 8 个数，此频率集合的赫夫曼编码是怎样的？

$a:1 \quad b:1 \quad c:2 \quad d:3 \quad e:5 \quad f:8 \quad g:13 \quad h:21$

你能否推广你的结论，求频率集为前 n 个斐波那契数的最优前缀码？

Answer:

a	1111111
b	1111110
c	111110
d	11110
e	1110
f	110
g	10
h	0

推广:

$T_2.left = c_2, T_2.right = c_1, T_2.freq = c_1.freq + c_2.freq = 2$

$(\forall i; 3 \le i \le n) T_i.left = c_i, T_i.right = T_{i-1}, T_i.freq = c_i.freq + T_{i-1}.freq$

2.1.4 16-1

Question:

（找零问题）考虑用最少的硬币找 n 美分零钱的问题。假定每种硬币的面额都是整数。$a.$ 设计贪心算法求解找零问题，假定有 25 美分、10 美分、5 美分和 1 美分 4 种面额的硬币。证明你的算法

能找到最优解。b. **假定硬币面额是** c **的幂，即面额为** $c^0, C^1, \ldots, c^k, c$ **和** k **为整数，**$c > 1, k \geq 1$。**证明：贪心算法总能得到最优解。**c. **设计一组硬币面额，使得贪心算法不能保证得到最优解。这组硬币面额中应该包含 1 美分，使得对每个零钱值都存在找零方案。**d. **设计一个** $O(nk)$ **时间的找零算法，适用于任何** k **种不同面额的硬币，假定总是包含 1 美分硬币。**

Answer:

a. **总是给最大面额的硬币，重复该过程，直到剩余的零钱为 0。**

b. **假设存在最优解** $(x_0, x_1, \ldots, x_k)$**，其中** x_i **表示面值为** c_i **的零钱的数量，则对于** $i < k$ **应有** $x_i < c$。**假设存在** x_i **使得** $x_i > c$**，那么可以以** c **为递减量减小** x_i**，以 1 为递减量减小** x_{i+1}**，这个硬币的集合的总值相同，却只包含** $c-1$ **个硬币，所以前者方案不是最优的，相互矛盾，因此得证。**

c. **令货币面额为** $1, 3, 4$**，要找 6 美分硬币，贪心算法得到的结果为** $1, 1, 4$**，而最优解决方案应为** $3, 3$。

d. **参考动态规划算法** $MAKE-CHANGE(S, v)$**，由于第一个 for 循环执行** n **次，内部 for 循环执行** k **次，之后的 while 循环执行最多 n 次，因此最终的时间为** $O(nk)$。

2.1.5 思考题

Question:

求以下背包问题的最优解:$n = 7, M = 15, (p_1, \ldots, p_7) = (10, 5, 15, 7, 6, 18, 3), (w_1, \ldots, w_7) = (2, 3, 5, 7, 1, 4, 1)$

Answer:

排序得到 $p_5/w_5 > p_1 > w_1 > p_6/w_6 > p_3/w_3 > p_7/w_7 > p_2/w_2 > p_4/w_4$**，因此得到结果为** $X = (1, \frac{2}{3}, 1, 0, 1, 1, 1)$**，**$F(M) = \frac{166}{3}$。