

## 任选12题

### Problem 7.1 (整数子集问题)

给定一个自然数集合  $A = \{s_1, s_2, \dots, s_n\}$  和自然数  $S$ 。判断是否存在  $A$  的子集，其中元素的和恰好是  $S$ 。请设计一个动态规划算法来解决该问题。

### Problem 7.2

对于输入的一个正整数  $n$ ，我们可以进行下面三种操作：

- 减一操作：  $n := n - 1$ ；
- 除以2操作：  $n := n/2$ ，要求执行前  $n$  是2的倍数；
- 除以3操作：  $n := n/3$ ，要求执行前  $n$  是3的倍数；

请计算将  $n$  变为1最少需要多少个操作。

### Problem 7.3

给定一个自然数数组  $A[1..n]$ ，请设计一个动态规划算法计算  $A$  中最长非递减子序列(子序列元素在数组  $A$  中不一定是连续的)。

### Problem 7.4

给定数组  $A[1..n]$ 。要找到它的子数组  $A[i..j]$ ，使得  $A[i..j]$  中各个元素的乘积最大。

1. 假设  $A$  中的元素皆为非负整数，请给出相应的算法；
2. 假设  $A$  中的元素为任意整数，请给出相应的算法；
3. 假设  $A$  中的元素可以不为整数，请给出相应的算法。

### Problem 7.5

下面是与公共子序列相关的一系列问题：

1. 给定两个字符串  $X = \langle x_1, x_2, \dots, x_m \rangle$  和  $Y = \langle y_1, y_2, \dots, y_n \rangle$ ，请设计一个动态规划算法计算  $X$  和  $Y$  的最长公共子序列；
2. 给定两个字符串  $X$  和  $Y$ 。请给出计算最长子序列的算法，要求在最长子序列中  $X$  中的字符可以重复出现但是  $Y$  中的不可以(算法只需要给出长度不需要具体的子序列)。该问题的一个例子如图7.1所示；
3. 除了  $X, Y$  两个字符串之外，给定一个正整数  $k$ 。问题和上一小题一样，但是子序列中  $X$  中的字符重复出现的次数不超过  $k$ ， $Y$  中的字符不可重复出现。例如，假设  $X, Y$  如图7.1所示， $k = 2$ ，那么在最长子序列中  $X$  中的字符可以出现2次。

### Problem 7.6

令  $A[1..m]$  和  $B[1..n]$  是两个任意的序列。 $A, B$  的公共超序列(supersequence)是一个序列，它包含  $A$  和  $B$  为其子序列。请设计一个高效算法找到  $A, B$  的最短公共超序列(算法最终只需给出序列长度)。

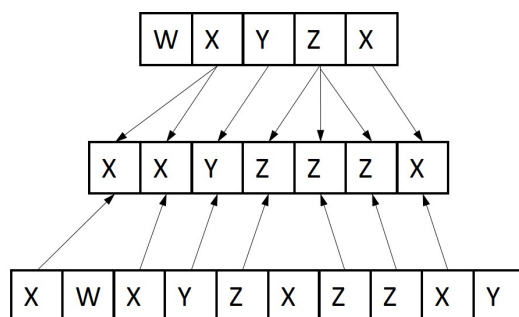


图 7.1: 最长子序列问题的扩展

**Problem 7.7**

这个问题中我们考虑最长公共子序列问题的两个变体。给定三个序列 $X, Y, Z$ ，它们的长度分别为 $|X| = m, |Y| = n, |Z| = k$ ，现在我们的问题是判断序列 $X$ 和 $Y$ 是否可以合并成为一个新的序列 $Z$ 并且不改变其中任何一个序列中元素的相对顺序。例如， $X = \langle ABC \rangle, Y = \langle BACA \rangle$ 可以合并成为 $Z = \langle ABBACCA \rangle$ ，( $Z = \langle A_x, B_x, B_y, A_y, C_x, C_y, A_y \rangle$ )，但是如果 $Z = \langle ABCBAAC \rangle$ ， $X, Y$ 就不能合成 $Z$ 。显然有 $k = m + n$ ，否则 $A, B$ 一定不能合成为 $Z$ 。

1. 假设一个人声称他找到了下面这个简单的算法来解决这个问题。首先，计算 $X$ 和 $Z$ 的LCS(最长公共自序列)，令 $Z'$ 是从 $Z$ 中将LCS中的元素删除得到序列。如果 $Y = Z'$ ，则答案为是，否则为否。请分析上述算法是否正确，并证明你的结论(如果算法正确)或者给出反例(如果算法错误)；
2. 请给出解决这个判定问题的复杂度为 $O(nm)$ 的算法(无论上一问的算法是否正确，这里都不可以使用)；
3. 现在将这个判定问题转化为一个优化问题：从 $X, Y, Z$ 中删除数目最少的元素使得合并成立。例如， $X = \langle ABC \rangle, Y = \langle BACA \rangle, Z = \langle ABCBAAC \rangle$ ，如果从 $Y$ 中删除最后一个元素，从 $Z$ 中删除倒数第二个元素， $X$ 和 $Y$ 就可以合并成为 $Z$ ，优化问题的解是2。算法不仅需要得到最少的元素删除数目还需要输出所删除的元素集合，时间复杂度是 $O(mnk)$ (这里 $k = m + n$ 不再是必要条件)。

**Problem 7.8**

请设计一个高效的算法找到字符串 $T[1...n]$ 中前向和后向相同的最长的连续子串的长度。前向和后向的子串不能够重叠。下面是几个例子：

- 给定输入字符串**ALGORITHM**，你的算法返回0；
- 给定输入字符串**RECURSION**，你的算法返回1，子串是**R**；
- 给定输入字符串**REDIVIDE**，你的算法返回3，子串是**EDI**（前向和后向的字符串不能够重叠）；
- 给定输入字符串**DYNAMICPROGRAMMINGMANYTIMES**，你的算法返回4，子串是**YNAM**。

**Problem 7.9**

给定包含 $n$ 个字符的字符串 $s[1 \dots n]$ 。该字符串可能来自于一本年代久远的书籍，只是由于纸张朽烂的缘故文档中所有的标点符号都不见了(因此该字符串看起来就像这样：“itwasthebestoftimes...”)。现在您希望在字典的帮助下重建这个文档。在此字典表示为一个布尔函数 $dict(\cdot)$ ，对于任意的字符串 $w$ ，

$$dict(w) = \begin{cases} true & w \text{ 是合法的单词} \\ false & \text{其他情况} \end{cases}$$

1. 请给出一个动态规划算法，判断 $s[\cdot]$ 是否能重建为由合法单词组成的序列。假设调用 $dict$ 每次只需一个单位的时间，该算法运行时间应该不超过 $O(n^2)$ ；
2. 若 $s[\cdot]$ 是由合法单词组成的，请输出对应的单词序列。

**Problem 7.10**

当字符串颠倒和原字符串相同时，我们称字符串为回文。例如，**I, DEED, RACECAR**。

1. 描述并分析找到给定字符串的满足回文条件的最长子序列的算法，算法最终只需给出长度即可。例如，**MAHDYNAMICPROGRAMZLETMESHOWYOUTHEM** 的最长回文子序列是**MHYMRORMYHM**，所以算法最终得到的结果是11；
2. 任何一个字符串都可以拆分为一组回文。例如，字符串**BUBBASEESABANANA** (“Bubba sees a banana.”)可以根据下面的步骤分解为回文：

BUB+BASEESAB+ANANA

B+U+BB+A+SEES+ABA+NAN+A

B+U+BB+A+SEES+A+B+NANA

B+U+B+B+A+S+E+E+S+A+B+N+A+N+A

请描述并分析算法对给定字符串可以拆分为的最少的回文数量。例如，给定的输入字符串“**BUBBASEESABANANA**”，你的算法给出的结果应该是3。

**Problem 7.11**

某种字符串处理语言提供了一个将字符串一分为二的基本操作。由于该操作需要拷贝原来的字符串，因此对于长度为 $n$ 的串，无论在其什么位置进行分割，都需要花费 $n$ 个单位的时间。现在假设我们要将一个字符串分割成多段，具体的分割次序会对总的运行时间产生影响。例如，如果要在位置3和位置10分割一个长度为20的串，首先在位置3进行分割产生的总代价为 $20 + 17 = 37$ ；而首先在位置10进行分割产生的总代价为 $20 + 10 = 30$ 。请设计一个动态规划算法，对于给出了 $m$ 个分割位置的长度为 $n$ 的字符串，计算完成所有分割的最小代价。

**Problem 7.12**

1. 给定数量无限的面值分别为 $x_1, x_2, \dots, x_n$ 的硬币，我们希望将金额 $v$ 兑换成零钱，即我们希望找出一堆总值恰好为 $v$ 的硬币。这有时候是不可能的：例如，如果硬币只有5和10两种面值，则我们可以兑换15却不能兑换12。请设计一个 $O(nv)$ 的动态规划算法判断能否兑换金额 $v$ ；

2. 考虑上述零钱兑换问题的一个变体：你有面值 $x_1, x_2, \dots, x_n$ 的硬币，希望兑换的价格为 $v$ ，但是每种面值的硬币最多只能使用一次。举例来说，如果硬币面值为1, 5, 10, 20，则可以兑换的价格包括 $16 = 1 + 15$  和  $31 = 1 + 10 + 20$ ，但是无法兑换40 (因为20不能用两次)。请设计一个 $O(nv)$ 的动态规划算法判断能否兑换金额 $v$ ；
3. 考虑上述零钱兑换问题的另一个变体：给定无限多的面值 $x_1, x_2, \dots, x_n$ 的硬币，我们希望用其中最多 $k$ 枚硬币兑换价格 $v$ 。即，我们需要找到不超过 $k$ 枚的硬币，使其总面值为 $v$ 。这也可能是无法实现的：例如，若面值为5 和10， $k = 6$ ，则我们将可以兑换55，但却不能兑换65。请设计一个的动态规划算法判断能否兑换金额 $v$ 。

### Problem 7.13

图 $G = (V, E)$ 的一个顶点覆盖 $S$ 是 $V$ 的子集，满足： $E$ 中的每条边都至少有一个端点属于 $S$ 。请给出如下问题的一个线性时间的算法：

- 输入：无向树 $T = (V, E)$ ；
- 输出： $T$ 的最小顶点覆盖的大小。

例如对于图7.2中的树，可能的顶点覆盖包括 $\{A, B, C, D, E, F, G\}$ 和 $\{A, C, D, F\}$ ，不包括 $\{C, E, F\}$ 。最小顶点覆盖的大小为3，对应集合为 $\{B, E, G\}$ 。

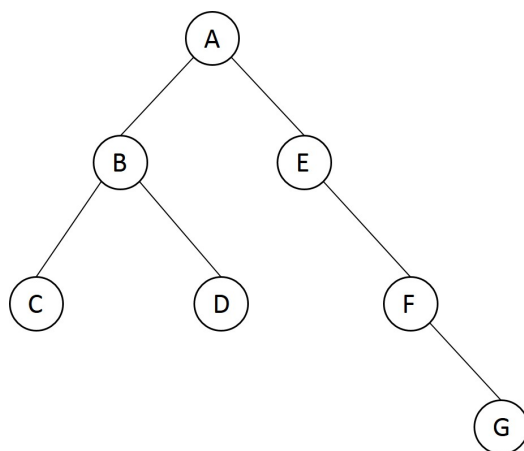


图 7.2: 树的顶点覆盖

### Problem 7.14

现在考虑带权重版本的课程表问题。不同的课程学分不同(学分和课程的时间无关)。你的目标是选择一组互不冲突的课程使得得到的学分最多。

1. 现有贪心算法是按照课程结束时间的先后顺序总是选择最先结束的课程，请证明这个算法并不能够总是得到最优解；
2. 请设计一个在 $O(n^2)$ 时间内解决该问题的算法；
3. 你能否设计一个动态规划算法解决此问题？

**Problem 7.15**

假设你准备开始一次长途旅行。以0公里为起点，一路上共有 $n$ 座旅店，距离起点的公里数分别为 $a_1 < a_2 < \dots < a_n$ 。旅途中，您只能在这些旅店停留。最后一座旅店 $a_n$ 为你的终点。

理想情况下，你每天可以行进200公里，不过考虑到旅店间的实际距离，有时候可能还达不到这么远。如果您某天走了 $x$ 英里，那么您将受到 $(200 - x)^2$ 的惩罚。您需要计划好行程，以使得总惩罚(每天所受惩罚的总和)最小。请设计一个高效的算法，计算一路上最优停留位置序列。

**Problem 7.16**

某公司计划沿着一条高速公路修建一系列酒店。酒店的 $n$ 个可能的选址位于一条直线上，它们距离高速起点的距离(按照升序)依次为 $m_1, m_2, \dots, m_n$ 公里。修建酒店的限制条件如下：

- 在每个地址上最多修建一座酒店。在位置 $i$ 建设酒店可能带来的利润为 $p_i$ ，其中 $i = 1, 2, \dots, n$ ， $p_i > 0$ 。
- 两个酒店间至少间隔 $k$ 公里，其中 $k$ 为正整数。

请设计一个高效的算法，计算修建这些酒店最多能获得的利润总额。

**Problem 7.17**

Vankin's Mile是一个在 $n \times n$ 的棋盘上玩的纸牌游戏。玩家一开始将纸牌放在棋盘的任意一个格子上，然后在每一轮中只可以将纸牌向下或者向右移动一个格子，当纸牌被玩家移动超出棋盘边界时游戏结束。每一个格子有一个数值，可以是正数，负数或者0。一开始，玩家的得分为0；只要玩家的牌未超过边界，就将当前所处格子的数值加到分数中，这个游戏的目标就是使得得分越高越好。

例如，给定图7.17中的棋盘，玩家一开始将纸牌放置在第二行上的8对应的格子，然后将纸牌向下移动，向下移动，向右移动，向右移动，游戏结束。他最终可以得到 $8 - 6 + 7 - 3 + 4 = 10$ 分，但这并不是可以得到的最高得分。

-1	7	-8	10	-5
-4	-9	8	-6	0
5	-2	-6	-6	7
-7	4	7	-3	-3
7	1	-6	4	-9

请设计一个算法来得到 $n \times n$ 的棋盘上玩Vankin's Mile所能得到的最高分数。

**Problem 7.18**

一个送披萨的男孩有一系列的订单要送。这些订单的目的地用坐标 $\{p_1, p_2, \dots, p_n\}$ 来表示， $p_i = (x_i, y_i)$ 。假设 $x$ 坐标是严格递增的，即 $x_1 < x_2 < \dots < x_n$ 。披萨店的位置在 $p_1$ ，并且送外卖的路线满足两个约束条件：首先按照 $x$ 坐标递增的顺序送外卖；然后按照 $x$ 递减的顺序送外卖并且最终回到披萨店(请参考图7.3中的例子)。假设已有函数 $dist(i, j)$ 用来计算 $p_i$ 和 $p_j$ 之间的距离，请给出计算最短外卖路线代价的算法(不要求求解最短路径，只需要计算代价即可)。

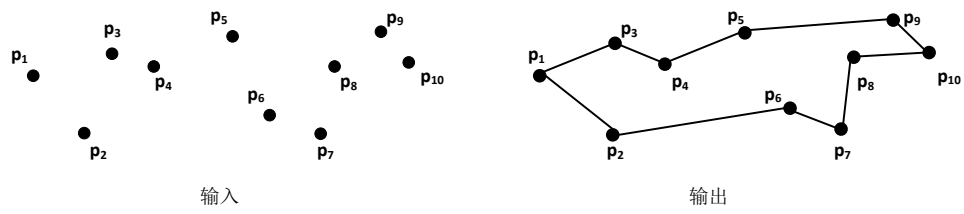


图 7.3: 披萨配送问题