算法分析与设计: 第六次作业

PB20061372　朱云沁　May 16, 2023

15.1-3 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. 请设计一个动态规划算法解决修改后的钢条切割问题, 不仅返回最优收益值, 还返回切割方案. (提醒: 钢条切割问题可以参见课本 P204 页 15.1 小节.)

(1) 证明原问题满足最优性原理;

(2) 写出最优解的递归表达式;

(3) 给出伪代码或在 OJ 系统上实现.

解: 假设一个解将长度为 n 的钢条切割为长度分别为 $i_1, i_2, \cdots, i_k$ 的 k 段, 则原问题可写为

$$\begin{aligned} \max \quad & r_n = \sum_{j=1}^{k} p_{i_j} - (k-1)c \\ \text{s.t.} \quad & \sum_{j=1}^{k} i_j = n,\ 1 \le k \le n, \\ & k, i_1, i_2, \ldots, i_k \in \mathbb{N}^+, \end{aligned}$$

若 $k=1$, 则 $r_n = p_n$; 若 $k \ge 2$, 设 $i_1, i_2, \ldots, i_k$ 是原问题的最优解, 且 $i_1 + i_2 + \cdots + i_l = m$, $1 \le l \le k-1$. 下证 $i_1, i_2, \ldots, i_l$ 是子问题 $\max\ r_m$ 的最优解.

假设 $i_1, i_2, \ldots, i_l$ 不是子问题 $\max\ r_m$ 的最优解, 则存在 $i'_1, i'_2, \ldots, i'_l$ 使得

$$\sum_{j=1}^{l} p_{i'_j} - (l-1)c > \sum_{j=1}^{l} p_{i_j} - (l-1)c,$$

进而

$$\sum_{j=1}^{l} p_{i'_j} + \sum_{j=l+1}^{k} p_{i_j} - (k-1)c > \sum_{j=1}^{k} p_{i_j} - (k-1)c,$$

与 $i_1, i_2, \ldots, i_k$ 是原问题的最优解矛盾, 故 $i_1, i_2, \ldots, i_l$ 是子问题 $\max\ r_m$ 的最优解. 同理可证 $i_{l+1}, i_{l+2}, \ldots, i_k$ 是子问题 $\max\ r_{n-m}$ 的最优解.

递归表达式为

$$r_n = \max\left\{p_n, \max_{1 \le m \le \lfloor \frac{n}{2} \rfloor} \{r_m + r_{n-m}\} - c\right\}, n \ge 2.$$

边界条件为 $r_1 = p_1$. 伪代码如下:

Algorithm 1: 15.1-3

```
Input: rod length n, price table p[1..n], cut cost c.
Output: optimal revenue r_n, optimal cut i_1, i_2, ..., i_k.
for i ← 1 to n
    r_i ← p_i
    for j ← 1 to ⌊i/2⌋
        if r_i < r_j + r_{i−j} − c
            r_i ← r_j + r_{i−j} − c
            s_i ← j
k ← 0
while n > 0
    i_k ← s_n
    n ← n − s_n
```

15.2-1 对矩阵规模序列 $\langle 5, 10, 3, 12, 5, 50, 6 \rangle$, 求矩阵链最优括号化方案. 请参考 P214 页图 15-5 画出算法执行过程的表格.

解: 根据题意, 各矩阵大小如表 1 所示.

Table 1. Matrix sizes for matrix-chain multiplication.

A_1	A_2	A_3	A_4	A_5	A_6
5×10	10×3	3×12	12×5	5×50	50×6

算法执行过程如表 2, 3 所示.

Table 2. Optimal $m[i, j]$ for matrix-chain multiplication.

$m[i,j]$	$j=1$	$j=2$	$j=3$	$j=4$	$j=5$	$j=6$
$i=1$	0	150	330	405	1655	2010
$i=2$		0	360	330	2430	1950
$i=3$			0	180	930	1770
$i=4$				0	3000	1860
$i=5$					0	1500
$i=6$						0

Table 3. Optimal $s[i, j]$ for matrix-chain multiplication.

$s[i,j]$	$j=2$	$j=3$	$j=4$	$j=5$	$j=6$
$i=1$	1	2	2	4	2
$i=2$		2	2	2	2
$i=3$			3	4	4
$i=4$				4	4
$i=5$					5

最优括号化方案为 $((A_1A_2)((A_3A_4)(A_5A_6)))$, 其标量乘法次数为 2010.

15.3-3 考虑矩阵乘法问题的一个变形: 目标改为最大化矩阵序列括号化方案的标量乘法运算次数, 而非最小化. 此问题具有最优子结构性质吗? 请说明.

解: 记矩阵链 $A_1A_2\cdots A_n$ 的最优括号化方案为 $A[1{:}n]$, 分割点在 A_k 和 A_{k+1} 之间, 即 $A[1{:}n] = (A[1{:}k]A[k+1{:}n])$. 下证子链的括号化方案 $A[1{:}k]$ 和 $A[k+1{:}n]$ 也是最优的.

假设子链 $A_1A_2\cdots A_k$ 的最优括号化方案不是 $A[1{:}k]$, 而是 $A'[1{:}k]$, 即 $A'[1{:}k]$ 的标量乘法次数大于 $A[1{:}k]$ 的标量乘法次数. 那么 $(A'[1{:}k]A[k+1{:}n])$ 的标量乘法次数也大于 $A[1{:}n]$ 的标量乘法次数, 与 $A[1{:}n]$ 是最优括号化方案矛盾. 因此, $A[1{:}k]$ 是 $A_1A_2\cdots A_k$ 的最优括号化方案.

同理可证 $A[k+1{:}n]$ 是 $A_{k+1}A_{k+2}\cdots A_n$ 的最优括号化方案. 该问题具有最优子结构性质.

15.4-1 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.

(1) 参考 P225 图 15-8 给出计算表格;

(2) 给出 LCS-Length 带备忘录版本的伪代码.

解: 最长公共子序列为 $\langle 1,0,0,1,1,0 \rangle$. 计算表格如下.

Table 4. Optimal $c[i, j]$ and $b[i, j]$ for longest common subsequence.

	0	$\underline{1}$	$\underline{0}$	1	1	$\underline{0}$	$\underline{1}$	$\underline{1}$	$\underline{0}$
$\underline{1}$	$0\uparrow$	$\underline{1\nwarrow}$	$1\leftarrow$	$1\nwarrow$	$1\nwarrow$	$1\leftarrow$	$1\nwarrow$	$1\nwarrow$	$1\leftarrow$
$\underline{0}$	$1\nwarrow$	$1\uparrow$	$\underline{2\nwarrow}$	$\underline{2\leftarrow}$	$\underline{2\leftarrow}$	$2\nwarrow$	$2\leftarrow$	$2\leftarrow$	$2\nwarrow$
$\underline{0}$	$1\nwarrow$	$1\uparrow$	$2\nwarrow$	$2\uparrow$	$2\uparrow$	$\underline{3\nwarrow}$	$3\leftarrow$	$3\leftarrow$	$3\nwarrow$
$\underline{1}$	$1\uparrow$	$2\nwarrow$	$2\uparrow$	$3\nwarrow$	$3\nwarrow$	$3\uparrow$	$\underline{4\nwarrow}$	$4\nwarrow$	$4\leftarrow$
0	$1\nwarrow$	$2\uparrow$	$3\nwarrow$	$3\uparrow$	$3\uparrow$	$4\nwarrow$	$\underline{4\uparrow}$	$4\uparrow$	$5\nwarrow$
$\underline{1}$	$1\uparrow$	$2\nwarrow$	$3\uparrow$	$4\nwarrow$	$4\nwarrow$	$4\uparrow$	$5\nwarrow$	$\underline{5\nwarrow}$	$5\uparrow$
$\underline{0}$	$1\nwarrow$	$2\uparrow$	$3\nwarrow$	$4\uparrow$	$4\uparrow$	$5\nwarrow$	$5\uparrow$	$5\uparrow$	$\underline{6\nwarrow}$
1	$1\uparrow$	$2\nwarrow$	$3\uparrow$	$4\nwarrow$	$5\nwarrow$	$5\uparrow$	$6\nwarrow$	$6\nwarrow$	$\underline{6\uparrow}$

伪代码如下:

Algorithm 2: 15.4-1

```
function LCS-Length(X, Y)
Input: two sequences X = ⟨x_1, x_2, ..., x_m⟩ and Y = ⟨y_1, y_2, ..., y_n⟩.
Output: the length of a longest common subsequence of X and Y.
let c[0..m, 0..n] and b[1..m, 1..n] be new tables
return LCS-Length-Aux(X, Y, m, n, c, b)

function LCS-Length-Aux(X, Y, i, j, c, b)
if i = 0 or j = 0
    return 0
if c[i, j] > 0
    return c[i, j]
if x_i = y_j
    c[i, j] ← LCS-Length-Aux(X, Y, m − 1, n − 1, c, b) + 1
    b[i, j] ← "up-left"
else if LCS-Length-Aux(X, Y, m − 1, n, c, b) ≥ LCS-Length-Aux(X, Y, m, n − 1, c, b)
    c[i, j] ← LCS-Length-Aux(X, Y, m − 1, n, c, b)
    b[i, j] ← "up"
else
    c[i, j] ← LCS-Length-Aux(X, Y, m, n − 1, c, b)
    b[i, j] ← "left"
return c[i, j]
```

15.4-6 设计一个 $O(n \log n)$ 时间复杂度的算法, 求一个 n 个数的序列的最长单调递增子序列. (提示: 一个长度为 i 的候选子序列的尾元素至少不比一个长度为 $i-1$ 候选子序列的尾元素小. 因此, 可以在输入序列中将候选子序列链接起来.)

(1) 证明该问题满足最优性原理;

(2) 写出最优解的递归表达式;

(3) 给出伪代码或在 OJ 系统上实现.

解: 记序列 $X_n = \langle x_1, x_2, \cdots, x_n \rangle$ 的最长单调递增子序列长度为 k, 一个具有最小尾元素的最长单调递增子序列为 $\langle x_{i_1}, x_{i_2}, \cdots, x_{i_k} \rangle$. 下证 $\langle x_{i_1}, x_{i_2}, \cdots, x_{i_{k-1}} \rangle$ 是 $X_{i_{k-1}} = \langle x_1, x_2, \cdots, x_{i_{k-1}} \rangle$ 的一个具有最小尾元素的长度为 $k-1$ 的单调递增子序列.

显然, $\langle x_{i_1}, x_{i_2}, \cdots, x_{i_{k-1}} \rangle$ 是 $X_{i_{k-1}}$ 的一个长度为 $k-1$ 的单调递增子序列. 假设其不具有最小尾元素, 那么存在一个 $X_{i_{k-1}}$ 的单调递增子序列 $\langle x_{j_1}, x_{j_2}, \cdots, x_{j_{k-1}} \rangle$, 使得 $x_{j_{k-1}} < x_{i_{k-1}}$ 且 $j_{k-1} < i_{k-1}$. 进而, $\langle x_{j_1}, x_{j_2}, \cdots, x_{j_{k-1}}, x_{i_{k-1}}, x_{i_k} \rangle$ 是 X_n 的一个长度为 $k+1$ 的单调递增子序列, 与 X_n 的最长单调递增子序列长度为 k 矛盾. 因此, $\langle x_{i_1}, x_{i_2}, \cdots, x_{i_{k-1}} \rangle$ 是 $X_{i_{k-1}}$ 的一个具有最小尾元素的长度为 $k-1$ 的单调递增子序列. 该问题满足最优性原理.

设前 i 个元素中, 长度为 j 的单调递增子序列的最小尾元素为 $t[i, j]$. 则递归表达式为

$$\forall 1 \le i \le n,\ 1 \le j \le i, \quad t[i,j] = \begin{cases} x_i, & \text{if } t[i-1, j-1] < x_i < t[i-1, j]; \\ t[i-1, j], & \text{otherwise.} \end{cases}$$

边界条件为 $\forall 0 \le i \le n$, $t[i, 0] = -\infty$, $t[i, i+1] = \infty$. 最优解的长度为 $\max\{j \mid t[n, j] \ne \infty\}$. 记录由 $t[i-1, j]$ 到 $t[i, j]$ 的更新过程, 可以在 $O(n)$ 时间内构造出最优解, 详见伪代码.

Algorithm 3: 15.4-6

```
Input: a sequence X = ⟨x_1, x_2, ..., x_n⟩.
Output: the longest increasing subsequence ⟨x_{i_1}, x_{i_2}, ..., x_{i_k}⟩
let t[1..n] and s[1..n] be new arrays
k ← 0
for i ← 1 to n
    t[i] ← ∞
    // binary-search the smallest j such that x_i ≤ t[j]
    j ← Lower-Bound(t, i, x_i)
    t[j] ← x_i
    s[i] ← j
    k ← max{k, j}
l ← k, i ← n
while l > 0
    if s[i] = l
        i_l ← i
        l ← l − 1
    i ← i − 1
```

每个循环均迭代 $O(n)$ 次, 第一个循环中二分查找的时间复杂度为 $O(\log n)$, 因此总时间复杂度为 $O(n \log n)$.