CS 5114
Solutions to Homework Assignment 3

Meghendra Singh

February 17, 2017

[20] 1. CLR Exercise 15.2-1. Give the filled-in m and s tables.
Find an optimal parenthesization of a matrix-chain product whose sequence of dimensions is $\langle 5, 10, 3, 12, 5, 50, 6\rangle$.

The number of matrices (n) for which the optimal paranthesization has to be computed is 6 and the dimensions of these matrices are as follows:

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

The filled in m and s tables are as follows:

i \ $j \rightarrow$	1	2	3	4	5	6
1	0	150	330	405	1655	2010
2		0	360	330	2430	1950
3			0	180	930	1770
4				0	3000	1860
5					0	1500
6						0

Table 1: The $m-$table computed for $n = 6$ and the matrix dimensions $\langle 5, 10, 3, 12, 5, 50, 6\rangle$.

i \ $j \rightarrow$	2	3	4	5	6
1	1	2	2	4	2
2		2	2	2	2
3			3	4	4
4				4	4
5					5

Table 2: The $s-$table for the same matrix dimensions.

In the tables 1 and 2 above, the rows and columns represent i and j respectively, as used in the Matrix-Chain-Order algorithm in CLRS. Using tables 1 and 2, we can obtain an optimal parenthesization (using Print-Optimal-Parens algorithm in CLRS) as: $((A_1A_2)((A_3A_4)(A_5A_6)))$.

---

[20] 2. CLR Exercise 15.4-1. Give the filled-in c and b tables.
Determine an LCS of $X = \langle 1, 0, 0, 1, 0, 1, 0, 1\rangle$ and $Y = \langle 0, 1, 0, 1, 1, 0, 1, 1, 0\rangle$.

The filled in table (b and c combined) is as follows:

			$j \rightarrow$									
			0	1	2	3	4	5	6	7	8	9
			y_j	0	1	0	1	1	0	1	1	0
	0	x_i	0	0	0	0	0	0	0	0	0	0
	1	1	0	$\overset{\uparrow}{\leftarrow}$ 0	$\nwarrow$ 1	$\leftarrow$ 1	$\nwarrow$ 1	$\nwarrow$ 1	$\leftarrow$ 1	$\nwarrow$ 1	$\nwarrow$ 1	$\leftarrow$ 1
	2	0	0	$\nwarrow$ 1	$\overset{\uparrow}{\leftarrow}$ 1	$\nwarrow$ 2	$\leftarrow$ 2	$\leftarrow$ 2	$\nwarrow$ 2	$\leftarrow$ 2	$\leftarrow$ 2	$\nwarrow$ 2
	3	0	0	$\nwarrow$ 1	$\overset{\uparrow}{\leftarrow}$ 1	$\nwarrow$ 2	$\overset{\uparrow}{\leftarrow}$ 2	$\overset{\uparrow}{\leftarrow}$ 2	$\nwarrow$ 3	$\leftarrow$ 3	$\leftarrow$ 3	$\nwarrow$ 3
i	4	1	0	$\uparrow$ 1	$\nwarrow$ 2	$\overset{\uparrow}{\leftarrow}$ 2	$\nwarrow$ 3	$\nwarrow$ 3	$\overset{\uparrow}{\leftarrow}$ 3	$\nwarrow$ 4	$\nwarrow$ 4	$\leftarrow$ 4
$\downarrow$	5	0	0	$\nwarrow$ 1	$\uparrow$ 2	$\nwarrow$ 3	$\overset{\uparrow}{\leftarrow}$ 3	$\overset{\uparrow}{\leftarrow}$ 3	$\nwarrow$ 4	$\overset{\uparrow}{\leftarrow}$ 4	$\overset{\uparrow}{\leftarrow}$ 4	$\nwarrow$ 5
	6	1	0	$\uparrow$ 1	$\nwarrow$ 2	$\uparrow$ 3	$\nwarrow$ 4	$\nwarrow$ 4	$\overset{\uparrow}{\leftarrow}$ 4	$\nwarrow$ 5	$\nwarrow$ 5	$\overset{\uparrow}{\leftarrow}$ 5
	7	0	0	$\nwarrow$ 1	$\uparrow$ 2	$\nwarrow$ 3	$\uparrow$ 4	$\overset{\uparrow}{\leftarrow}$ 4	$\nwarrow$ 5	$\overset{\uparrow}{\leftarrow}$ 5	$\overset{\uparrow}{\leftarrow}$ 5	$\nwarrow$ 6
	8	1	0	$\uparrow$ 1	$\nwarrow$ 2	$\uparrow$ 3	$\nwarrow$ 4	$\nwarrow$ 5	$\overset{\uparrow}{\leftarrow}$ 5	$\nwarrow$ 6	$\nwarrow$ 6	$\overset{\uparrow}{\leftarrow}$ 6

Table 3: The b and c tables combined into one table as computed by the algorithm LCS-LENGTH in CLRS on the sequences $X = \langle 1,0,0,1,0,1,0,1\rangle$ and $Y = \langle 0,1,0,1,1,0,1,1,0\rangle$. The cell for row i and column j contains the value of $c[i,j]$ and the appropriate arrow for the value of $b[i,j]$. To reconstruct the elements of the LCS, follow the $b[i,j]$ arrows from the lower right-hand corner; the sequence is shaded in orange. Each "$\nwarrow$" on the shaded sequence corresponds to a highlighted entry for which $x_i = y_j$, and is a member of an LCS.

Using table - 3, an LCS for X and Y can be computed as: $\langle 1,0,0,1,1,0\rangle$, which has a length of 6 (as evident from the value at $c[8,9]$, i.e. the bottom left corner of the c-table). Also, we might obtain multiple longest common sub-sequences (for example: $\langle 0,0,1,0,1,0\rangle$), because at some cells ($c[i,j]$) we can either select the top cell ($c[i-1,j]$) or the left cell ($c[i,j-1]$), given $x_i \neq y_j$ and both the top and left cells have equal values in them. (for example, we could have selected $c[5,6]$ instead of $c[4,7]$ from $c[5,7]$, while finding the LCS in the shaded example above).

---

[20] 3. CLR Exercise 15.4-2. Give pseudocode for your algorithm to reconstruct the LCS. Give pseudocode to reconstruct an LCS from the completed c table and the original sequences $X = \langle x_1, x_2, ..., x_m\rangle$ and $Y = \langle y_1, y_2, ..., y_n\rangle$ in $O(m+n)$ time, without using the b table.

The pseudocode for an algorithm to reconstruct the LCS (and print it) using a completed c table is as follows:

```
RECONSTRUCT-LCS(c, X, Y)
1  // Call PRINT-LCS with i, j initialized to
2  // the lower right-hand corner of the c table, i.e. i = m and j = n
3  // m and n being the length of the 2 sequences X and Y respectively
4  i = X.length
5  j = Y.length
6  PRINT-LCS(c, X, Y, i, j)
```

```
PRINT-LCS(c, X, Y, i, j)
 1  // If we have either reached the row x_i = 0 or the column y_i = 0,
 2  // we don't need to do anything, but stop reconstructing (printing) the LCS.
 3  if i == 0 or j == 0
 4       return
 5  // If the values of the two sequences are equal (common) for positions i and j, recursively call
 6  // PRINT-LCS on the cell: c[i − 1, j − 1], and print the common value
 7  else if X[i] == Y[j]
 8       PRINT-LCS(c, X, Y, i − 1, j − 1)
 9       print X[i]
10  // Otherwise check whether the top or the left cell has a greater value
11  // and recursively call PRINT-LCS on that cell
12  else if c[i − 1, j] ≥ c[i, j − 1]
13       PRINT-LCS(c, X, Y, i − 1, j)
14  else
15       PRINT-LCS(c, X, Y, i, j − 1)
```

In the above pseudocode, the algorithm RECONSTRUCT-LCS, initializes the values of i and j with the lengths (m and n) of the two sequences X and Y. This is because, the reconstruction of the LCS begins from the bottom right hand corner of the table c, which is $c[m, n]$. RECONSTRUCT-LCS then makes a call to PRINT-LCS using the initialized values of i and j as parameters. PRINT-LCS recurrsively calls itself, and prints a LCS for both the sequences. Recursive calls to PRINT-LCS are always preceded by either a unit decrement in i, j or both i and j, depending on the last two conditions of the recurrence for table c (Equation 15.9 CLRS). PRINT-LCS, basically performs backtracking on the c-table and prints the LCS in the correct order of its appearance in both the sequences X and Y. The recursive calls continue until the base case of i or j equal to 0 is not met. The base case implies that either the "all-zero's" row ($i = 0$) or the "all-zero's" column ($j = 0$) (which were added while constructing the initial c-table) have been reached, and there are no more cells to be considered.

The worst case running time for the above algorithm would result when there aren't any common values between the two sequences X and Y, and the values in the c-table are such that:

$$\begin{aligned} &c[i-1, j] \geq c[i, j-1] \ \forall \ 2 \leq i \leq m \ \& \ j = n \ , and \\ &c[i-1, j] < c[i, j-1] \ \forall \ i = 1 \ \& \ 1 \leq j \leq n \end{aligned} \qquad \text{(case 1)}$$

or,

$$c[i-1,j] < c[i,j-1] \ \forall \ i = m \ \& \ 2 \leq j \leq n \ , and$$
$$c[i-1,j] \geq c[i,j-1] \ \forall \ 1 \leq i \leq m \ \& \ j = 1 \qquad \text{(case 2)}$$

In (case 1) above, PRINT-LCS would make $m-1$ recursive calls to itself by executing lines 12 and 13 in the pseudocode, followed by n recursive calls to itself by executing lines 14 and 15 in the pseudocode, before reaching the base case of i or j equal to 0. Similar explanation applies to (case 2) as well, i.e. PRINT-LCS would make $n-1$ recursive calls to itself by executing lines 14 and 15in the pseudocode, followed by m recursive calls to itself by executing lines 12 and 13 in the pseudocode, before reaching the base case of i or j equal to 0. Considering that the "non-recursive" lines in the pseudocode (lines 1 through 11) execute in constant time, and both the sequences are of comparable lengths, the worst case time complexity for PRINT-LCS (and RECONSTRUCT-LCS) can be written as:

$$O(m-1+n) \ or \ O(n-1+m)$$
$$\implies O(m+n)$$

Where, m and n are the lengths of the two sequences X and Y, for which the LCS has to be computed.
