Computer Science 3719 (Winter 2008):
Assignment #2, Question #2
Answers

2. (40 marks)

a) (10 marks) Determine the longest common subsequence (LCS) of the strings GAAGCCTA and TATCGA using the algorithm given on page 353 of the textbook. In the style of Figure 15.6, show the filled-in dynamic programming matrix, all matrix-cell backpointers, the backpointer path that gives an optimal LCS, and the LCS associated with that path.

Answer: The requested table is given in Table 1. Note that the algorithm on page 353 has a rigid order for selecting backpointers (diagonal-up-across) that assigns only one backpointer per cell, regardless of how many of the three associated previously-computed cells yield an optimal cost. Table 2 shows the result of executing the algorithm such that all optimal-cost backpointers are set up. The original algorithm, by preferring "up" backpointers to "across" backpointers, effectively restricted the derivable LCS to the one that ended furthest to the right in the the given sequence labeling the top of the table. The revised backpointer scheme allows traceback to access a much greater range of the possible LCS for the given sequences.

b) (15 marks) Determine the optimal 0/1 knapsack-load for the set of items $U = \{1, 2, 3, 4, 5, 6\}$ and $B = 6$ using the algorithm given in Lecture #5 of the class notes. The sizes and values of the items in U are given in part (a) of Table 3. Show the filled-in dynamic programming value and backpointer matrices, the "backpointer path" in the backpointer matrix that gives an optimal knapsack-load, and the knapsack-load associated with that path.

Answer: The requested table is given in part (b) of Table 3. The optimal-value knapsack-load derived by the algorithm is $\{2, 3, 6\}$ with summed size 6 and summed value 11.

c) (15 marks) Determine the optimal parenthesization of a matrix-chain product whose sequence of dimensions is $\langle 2, 5, 3, 4, 2, 3, 5\rangle$ using the algorithm given on page 336 of the textbook. In the style of Figure 15.3, show the filled-in dynamic programming matrices m and s , the "backpointer path" in s that gives an optimal parenthesization, and the parenthesization associated with that path.

Answer: The matrices m and s are shown in Figure 1. For the sake of completeness, the computations for each of the cells in m and s are given below:

$$m[1][2] = \min \left\{ \begin{array}{ll} m[1][1] + m[2][2] + (p[0] * p[1] * p[2]) & (k = 1) \\ = 0 + 0 + 30 = 30 & \end{array} \right\} = 30 \ (k = 1)$$

$$m[2][3] = \min \left\{ \begin{array}{ll} m[2][2] + m[3][3] + (p[1] * p[2] * p[3]) & (k = 2) \\ = 0 + 0 + 60 = 60 & \end{array} \right\} = 60 \ (k = 2)$$

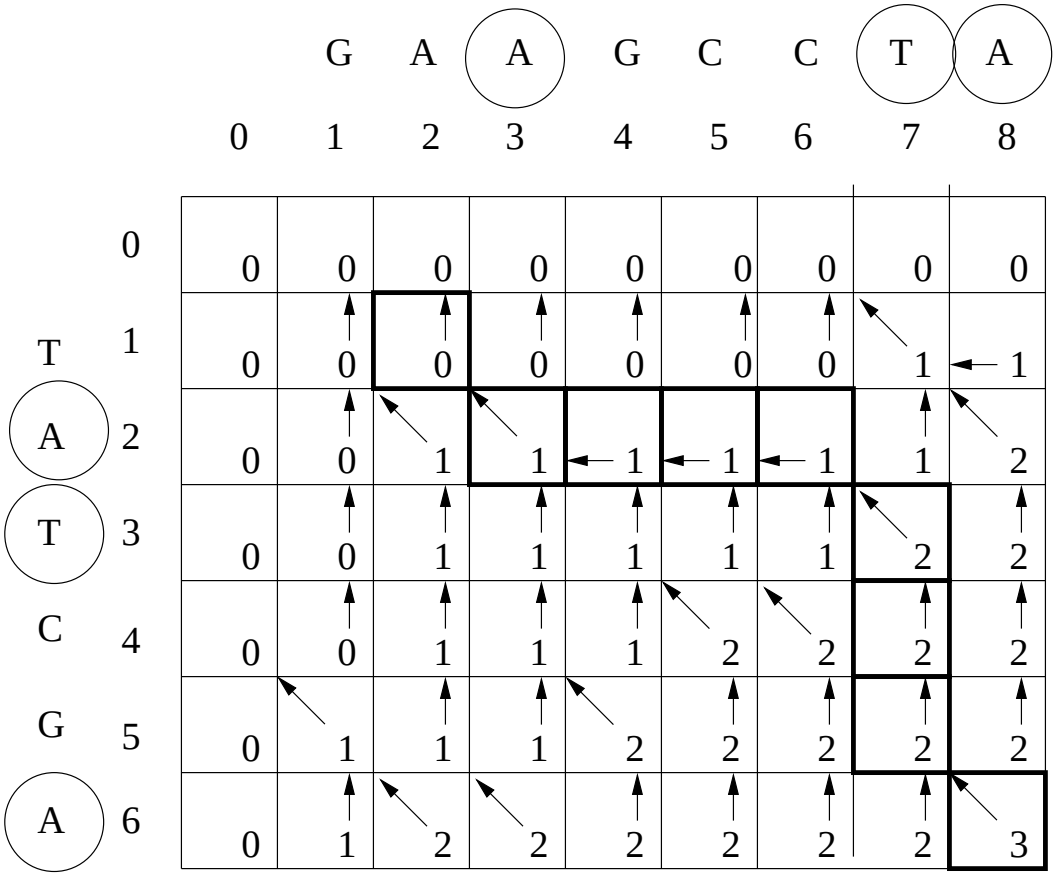

Table 1: Answer for Question #2(a) (textbook-specified backpointers).

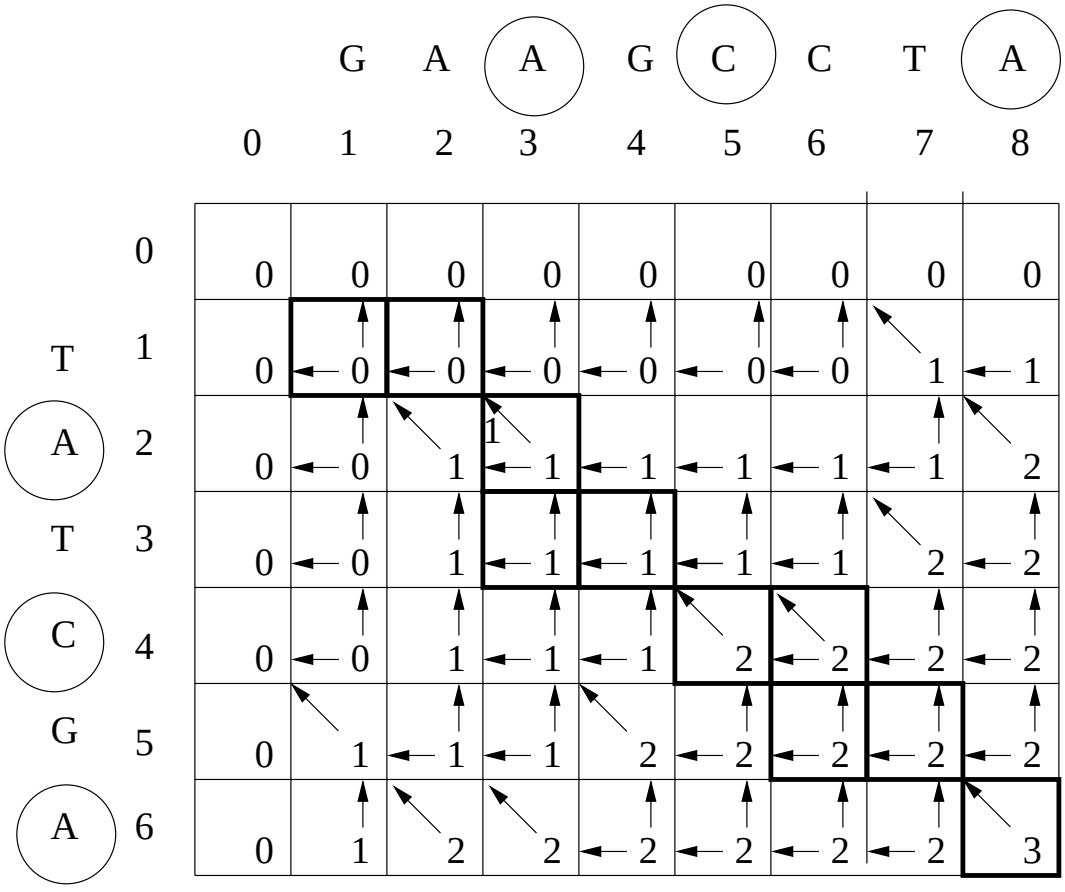

Table 2: Answer for Question #2(a) (all optimal-value backpointers).

(a)

i	$s(i)$	$v(i)$
1	4	5
2	2	4
3	3	4
4	5	4
5	2	1
6	1	3

(b)

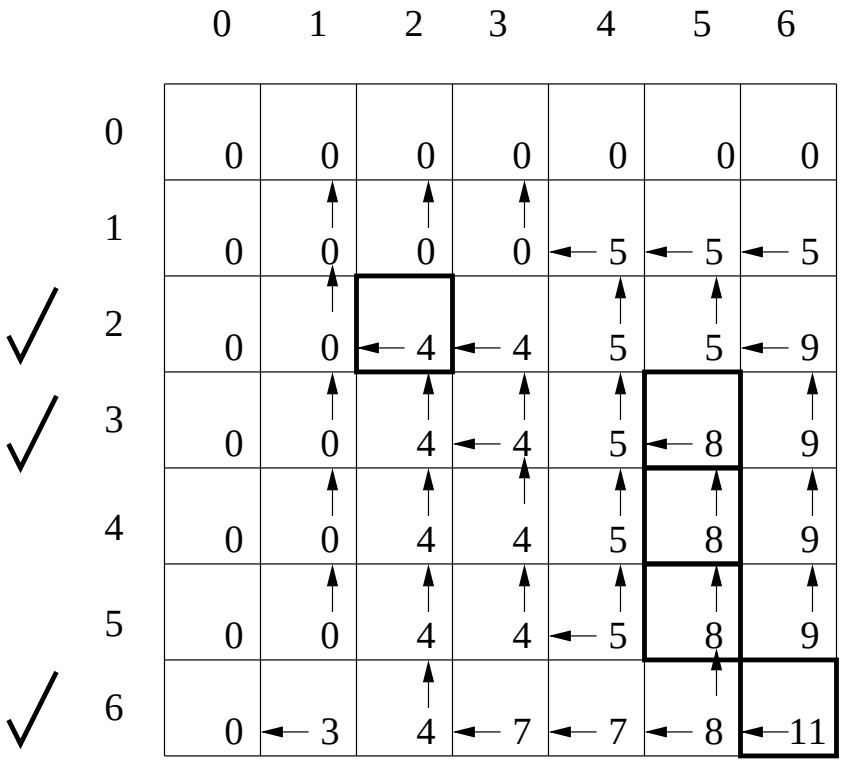

Table 3: Tables for Question #3(b). a) Input item size- and value-function specification. b) Answer.

$$m[3][4] = \min \left\{ \begin{array}{ll} m[3][3] + m[4][4] + (p[2] * p[3] * p[4]) & (k = 3) \\ = 0 + 0 + 24 = 24 & \end{array} \right\} = 24 \ (k = 3)$$

$$m[4][5] = \min \left\{ \begin{array}{ll} m[4][4] + m[5][5] + (p[3] * p[4] * p[5]) & (k = 4) \\ = 0 + 0 + 24 = 24 & \end{array} \right\} = 24 \ (k = 4)$$

$$m[5][6] = \min \left\{ \begin{array}{ll} m[5][5] + m[6][6] + (p[4] * p[5] * p[6]) & (k = 5) \\ = 0 + 0 + 30 = 30 & \end{array} \right\} = 30 \ (k = 5)$$

$$m[1][3] = \min \left\{ \begin{array}{ll} m[1][1] + m[2][3] + (p[0] * p[1] * p[3]) & (k = 1) \\ = 0 + 60 + 40 = 100, & \\ m[1][2] + m[3][3] + (p[0] * p[2] * p[3]) & (k = 2) \\ = 30 + 0 + 24 = 54 & \end{array} \right\} = 54 \ (k = 2)$$

$$m[2][4] = \min \left\{ \begin{array}{ll} m[2][2] + m[3][4] + (p[1] * p[2] * p[4]) & (k = 2) \\ = 0 + 24 + 30 = 54, & \\ m[2][3] + m[4][4] + (p[1] * p[3] * p[4]) & (k = 3) \\ = 60 + 0 + 40 = 100 & \end{array} \right\} = 54 \ (k = 2)$$

$$m[3][5] = \min \left\{ \begin{array}{ll} m[3][3] + m[4][5] + (p[2] * p[3] * p[5]) & (k = 3) \\ = 0 + 24 + 36 = 60, & \\ m[3][4] + m[5][5] + (p[2] * p[4] * p[5]) & (k = 4) \\ = 24 + 0 + 18 = 42 & \end{array} \right\} = 42 \ (k = 4)$$

$$m[4][6] = \min \left\{ \begin{array}{ll} m[4][4] + m[5][6] + (p[3] * p[4] * p[6]) & (k = 4) \\ = 0 + 30 + 40 = 70, & \\ m[4][5] + m[6][6] + (p[3] * p[5] * p[6]) & (k = 5) \\ = 24 + 0 + 60 = 84 & \end{array} \right\} = 70 \ (k = 4)$$

$$m[1][4] = \min \left\{ \begin{array}{ll} m[1][1] + m[2][4] + (p[0] * p[1] * p[4]) & (k = 1) \\ = 0 + 54 + 20 = 74, & \\ m[1][2] + m[3][4] + (p[0] * p[2] * p[4]) & (k = 2) \\ = 30 + 24 + 12 = 66, & \\ m[1][3] + m[4][4] + (p[0] * p[3] * p[4]) & (k = 3) \\ = 54 + 0 + 16 = 70 & \end{array} \right\} = 66 \ (k = 2)$$

$$m[2][5] = \min \left\{ \begin{array}{ll} m[2][2] + m[3][5] + (p[1] * p[2] * p[5]) & (k = 2) \\ = 0 + 42 + 45 = 87, & \\ m[2][3] + m[4][5] + (p[1] * p[3] * p[5]) & (k = 3) \\ = 60 + 24 + 60 = 144, & \\ m[2][4] + m[5][5] + (p[1] * p[4] * p[5]) & (k = 4) \\ = 54 + 0 + 30 = 84 & \end{array} \right\} = 84 \ (k = 4)$$

$$m[3][6] = \min \left\{ \begin{array}{ll} m[3][3] + m[4][6] + (p[2] * p[3] * p[6]) & (k = 3) \\ = 0 + 70 + 60 = 130, & \\ m[3][4] + m[5][6] + (p[2] * p[4] * p[6]) & (k = 4) \\ = 24 + 30 + 30 = 84, & \\ m[3][5] + m[6][6] + (p[2] * p[5] * p[6]) & (k = 5) \\ = 42 + 0 + 45 = 87 & \end{array} \right\} = 84 \ (k = 4)$$

$$m[1][5] = \min \left\{ \begin{array}{ll} m[1][1] + m[2][5] + (p[0] * p[1] * p[5]) & (k = 1) \\ = 0 + 84 + 30 = 114, & \\ m[1][2] + m[3][5] + (p[0] * p[2] * p[5]) & (k = 2) \\ = 30 + 42 + 18 = 90, & \\ m[1][3] + m[4][5] + (p[0] * p[3] * p[5]) & (k = 3) \\ = 54 + 24 + 24 = 102, & \\ m[1][4] + m[5][5] + (p[0] * p[4] * p[5]) & (k = 4) \\ = 66 + 0 + 12 = 78 & \end{array} \right\} = 78 \ (k = 4)$$

$$m[2][6] = \min \left\{ \begin{array}{ll} m[2][2] + m[3][6] + (p[1] * p[2] * p[6]) & (k = 2) \\ = 0 + 84 + 75 = 159, & \\ m[2][3] + m[4][6] + (p[1] * p[3] * p[6]) & (k = 3) \\ = 60 + 70 + 100 = 230, & \\ m[2][4] + m[5][6] + (p[1] * p[4] * p[6]) & (k = 4) \\ = 54 + 30 + 50 = 134, & \\ m[2][5] + m[6][6] + (p[1] * p[5] * p[6]) & (k = 5) \\ = 84 + 0 + 75 = 159 & \end{array} \right\} = 134 \ (k = 4)$$

$$m[1][6] = \min \left\{ \begin{array}{ll} m[1][1] + m[2][6] + (p[0] * p[1] * p[6]) & (k = 1) \\ = 0 + 134 + 50 = 184, & \\ m[1][2] + m[3][6] + (p[0] * p[2] * p[6]) & (k = 2) \\ = 30 + 84 + 30 = 144, & \\ m[1][3] + m[4][6] + (p[0] * p[3] * p[6]) & (k = 3) \\ = 54 + 70 + 40 = 164, & \\ m[1][4] + m[5][6] + (p[0] * p[4] * p[6]) & (k = 4) \\ = 66 + 30 + 20 = 116, & \\ m[1][5] + m[6][6] + (p[0] * p[5] * p[6]) & (k = 5) \\ = 78 + 0 + 30 = 108 & \end{array} \right\} = 108 \ (k = 5)$$

The "backpointer path" (more properly, the backpointer tree) is indicated by dashed arrows linking cells in the s matrix in Figure 1. This tree implies that the optimal parenthesization of the given matrix chain is

$$\begin{array}{l} (A_1 A_2 A_3 A_4 A_5 A_6) \\ \Rightarrow ((A_1 A_2 A_3 A_4 A_5)(A_6))) \\ \Rightarrow (((A_1 A_2 A_3 A_4)(A_5))(A_6))) \\ \Rightarrow ((((A_1 A_2)(A_3 A_4))(A_5))(A_6))) \end{array}$$

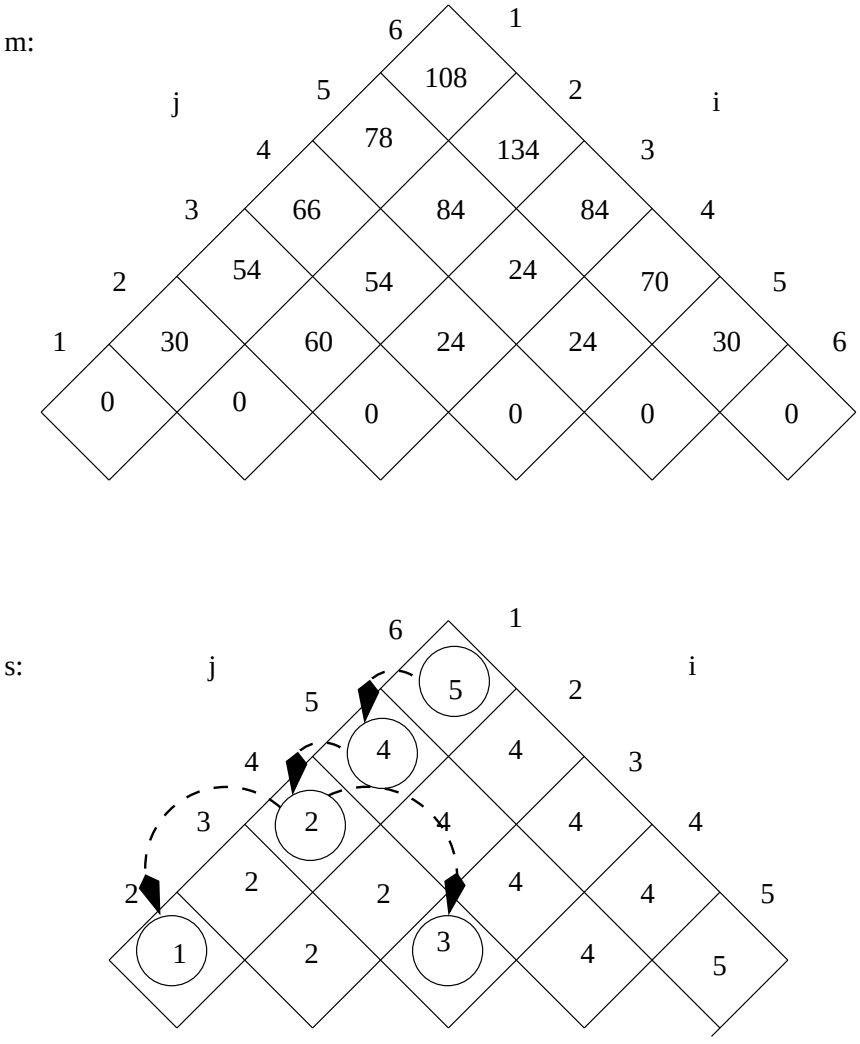

Figure 1: Answer for Question #2(c).