Algorithms -1
Tutorial 5
Solution Sketch

All problems should be done using dynamic programming

1. Find an optimum parenthesization of the matrix chain $A_1A_2A_3A_4A_5$ where $A_1 = 10\times25$, $A_2 = 25\times 100$, $A_3 = 100\times50$, $A_4 = 50\times50$, and $A_5 = 50\times100$ matrices. Show the subproblem definition, the recursive formulation of the solution, and all calculations and steps for finding both the minimum number of scalar multiplications and the actual parenthesis.

   (Practice more such problems, you can take an arbitrary chain)

   Routine

2. Find the Longest Common Subsequence (LCS) of the strings X = AGCGA and Y = CAGATAGAG using the dynamic programming algorithm, First write the subproblem definition and the recursive solution formulation for the length of the LCS. Then show how all the table entries are filled, showing calculations for each entry. Finally report the length of the LCS and the actual LCS from the tables.

   (Practice more such problems, you can take an arbitrary pair of sequences)

   Routine

3. Given a sequence of n integers, design an $O(n^2)$ time algorithm to find the longest increasing subsequence of the sequence (a subsequence is increasing if the integers in it are in non-decreasing order). Solve it both using and not using the solution of the LCS problem directly.

   Let the sequence be $A = a_1a_2a_3...a_n$.

   Using solution of LCS: Sort A in ascending order and store in B. Now find the LCS between A and B.

   Not using solution of LCS:

   Let L[i] store the length of the longest increasing subsequence of A ending at a_i
   Then:

   $L[i] = \max (1 + L(j))$ $0 < j < i$ and $a_j < a_i$
   $= 1$ if no such j exists

   The final answer is $\max(L[i])$, $1 \le i \le n$

4. Given a string of characters, design an $O(n^2)$ time algorithm to find the minimum number of characters to be removed from the string to convert it into a palindrome. Solve it both using and not using the solution of the LCS problem directly.

Let the string be A = $a_1a_2a_3...a_n$.
The minimum number of characters to remove is n – (the length of the longest palindrome subsequence)

Using solution of LCS: Reverse A and store in B. Find the LCS of A and B. Let k be the length of the LCS. The final answer is (n – k) (remove the characters not in the LCS).

Not using the solution of LCS:

Let A(,i j) denote the substring $a_ia_{i+1}a_{i+2}...a_j$

Let L[i, j] = length of longest palindrome subsequence of A(i, j)

Then L[i, j] = 1 if i = j
= L[i+1, j-1] + 2 if $a_i = a_j$
= max(L[i+1, j] , L[i, j-1]) otherwise

Final answer is (n – L[1,n])

5. A sequence is called a good sequence if $a_1 < a_2 > a_3 < a_4 ... a_k$. i.e. $a_i < a_{i+1}$ if i is odd and $a_i > a_{i+1}$ if i is even for all i < k. You are given a sequence A containing n integers. Design an $O(n^2)$ time algorithm to find the length of longest good subsequence of A.

dp[i][0] : Length of the longest good subsequence of the form $a_1 < a_2 > a_3.. < a_i$ ending at i
dp[i][1] : The length of the longest good subsequence of the form $a_1 < a_2 > a_3.. > a_i$ ending at i

$dp[i][0] = \max(1, \max\{dp[j][1] + 1 \mid j < i, A_j < A_i\})$
$dp[i][1] = \max(1, \max\{dp[j][0] + 1 \mid j < i, A_j > A_i\})$

The final answer is $\max(dp[i][0], dp[i][1])$ for $1 \le i \le n$.

6. You are given n intervals $(l_1, r_1); (l_2, r_2) ... (l_n, r_n)$, l_i and r_i are integers and $l_i \le r_i$ for all $1 \le i \le n$. You need to find a set of non-overlapping intervals such the sum of the lengths of the intervals in the set is maximized. Your algorithm should run in $O(n^2)$ time. Can you improve it to O(nlogn)?

This is just the weighted interval scheduling problem discussed in class with the weight of each interval being its length.