Problem 1

Matrix Chain Multiplication

Problem:

Find the optimal way(one that takes the minimum number of multiplications) to calculate the multiplication of n matrices ,

$A_1. A_2. A_3 \ldots\ldots\ldots A_n$

Since we can’t mess up with the sequence in matrix multiplication we basically need to find an integer k such that,

$(A_{1.}\ A_{2\ldots.}\ A_k)_X\ (A_{k+1\ldots\ldots}A_n)$ leads to the Optimal Solution i.e the least number of multiplications .

Filling in The Dp Table :

Step 1: Fill in the Already Known Values/Constants/mark values we that are useless with an ‘X’

Step 2: Choosing How to fill the table

1. Row-wise
2. Column-Wise
3. Diagonal-Wise

This usually depends on how we filled step 1 and the recursive formula.(?)

<u>For this Problem :</u>

I. For all i>j we fill with ‘x’ (since we won't ever need to multiply backwards)

ii. For all i==j we fill with ‘0’

iii. Now , fill each remaining diagonal

-A[0][1]-A[1][2]-A[2][3]-A[3][4] <- marked here>

-A[0][2]-A[1][3]-A[2][4] <* marked here>

-A[0][3]-A[1][4] <! marked>

-A[0][4] <~ marked>

For a test case of 5 matrices,
A1.A2.A3.A4.A5:

i\j	0	1	2	3	4
0	0	-	*	!	Solution
1	x	0	-	*	!
2	x	x	0	-	*
3	x	x	x	0	-
4	x	x	x	x	0

Algorithm : Running Time O(n^3)

Problem 2

LONGEST PALINDROMIC SUBSEQUENCE

Problem Specification:

Given a sequence, find the length of the longest palindromic subsequence in it.
As another example, if the given sequence is "BBABCBCAB", then the output should be 7 as "BABCBAB" is the longest palindromic subsequence in it. "BBBBB" and "BBCBB" are also palindromic subsequences of the given sequence, but not the longest ones.

Problem Solution:

The solution of the problem is very much similar with the previously learnt problems Matrix Chain Multiplication(MCM) and Longest Common Subsequence(LCS)

- The Dp table must be filled up diagonally as in MCM
- The recursive formula is similar to LCS

Filling in The Dp Table :

Step 3.1: Fill in the Already Known Values/Constants/mark values that are useless with an 'X'

Step 3.2: Choosing How to fill the table

1. Row-wise
2. Column-Wise
3. Diagonal-Wise

This usually depends on how we filled step 1 and the recursive formula.

<u>For this Problem we will fill up the dp Table diagonally:</u>

i. For all i>j we fill with 'x' (since we don't need to find the longest palindromic sequence from backwards)

ii. For all i==j we fill with '1'

iii. Now , fill each remaining diagonal

-A[0][1]-A[1][2]-A[2][3]-A[3][4] <- marked here>

-A[0][2]-A[1][3]-A[2][4] <* marked here>

-A[0][3]-A[1][4] <! marked>

-A[0][4] <~ marked>

For a test case of the string S = "$s_1s_2s_3s_4s_5$"

i\j	s_1	s_2	s_3	s_4	s_5
s_1	1	-	*	!	Solution
s_2	x	1	-	*	!
s_3	x	x	1	-	*
s_4	x	x	x	1	-
s_5	x	x	x	x	1

<u>Example :</u>

For a test case of the string S = "MUNUM"

i\j	M	U	N	U	M
M	1	max(1,1)= 1	max(1,1)=1	max(1,3)= 3	(3+2)=5
U	x	1	max(1,1)=1	(1+2)=3	max(3,1)= 3
N	x	x	1	max(1,1)= 1	max(1,1)= 1
U	x	x	x	1	max(1,1)= 1
M	x	x	x	x	1

Therefore , Answer = 5
The palindromic subsequence can be found by backtracking .

Time Complexity of LPS : $O(n^2)$

Problem 3

ROD CUTTING PROBLEM

Problem Description:

Given a rod of length n units, and the prices of all pieces smaller than n, find the most profitable way of cutting the rod.

Let, C(i) = The price of the optimal cut of the rod of length i.

V_k = The price of a cut at length k

Initially, C(0) = 0 ; C(1) = 1

$$C(i) = \max(V_k + C(i-k))_{1 \le k \le i}$$

Or,

$$C(i) = \max(V_i\ ,\ [C(i)+C(i-k)]_{0<k\le ceil(i/2)})$$

Filling in The Dp Table :

The first two rows are given.

Length	0	1	2	3	4	5	6	7	8
Price	0	1	5	8	9	10	17	17	20
C[i]	0	1	5	8	10	13	17		

Problem 4

Segmented Least Squares:

Problem:

As in the discussion above, we are given a set of points P ={(x1,y1),(x2, y2),...,(xn, yn)}, with x1< x2 < ...< xn. We will use pi to denote the point (xi, yi). We must first partition P into some number of segments. Each segment is a subset of P that represents a contiguous set of x-coordinates; that is, it is a subset of the form {pi, pi+1,...,pj−1, pj} for some indices i ≤j. Then, for each segment S in our partition of P, we compute the line minimizing the error with respect to the points in S, according to the formulas above. The penalty of a partition is defined to be a sum of the following terms.

(i) The number of segments into which we partition P, times a fixed, given multiplier C > 0. (ii) For each segment, the error value of the optimal line through that segment.

Goal:

Our goal in the Segmented Least Squares Problem is to find a partition of minimum penalty.

Algorithm :

If the last segment of the optimal partition is pi,...,pn, then the value of the optimal solution is OPT(n) .

For the subproblem on the points p1,...,pj,
OPT(j) = min 1≤i≤j(ei,j + C + OPT(i − 1)), and the segment pi,...,pj is used in an optimum solution for the subproblem if and only if the minimum is obtained using index i.

Pseudo Code:

```
Segmented-Least-Squares(n)
Array M[0...n]
Set M[0]=0
For all pairs i≤j
Compute the least squares error ei,j for the segment pi,...,pj
Endfor
For j =1,2,...,n
Use the recurrence (6.7) to compute M[j]
Endfor
Return M[n]
```