Problem 1. Matrix Chain Multiplication

Matrix chain multiplication is an optimization problem that can be solved using dynamic programming. Given a sequence of matrices, we want to find the most efficient way to multiply these matrices together. The problem is not actually to perform the multiplications, but merely to decide in which order to perform the multiplications.

1. Find an optimal order of a matrix chain multiplication whose sequence of dimensions is $5, 10, 3, 12, 5, 50, 6$.

2. Let $R(i, j)$ be the number of times that table entry $c[i, j]$ is referenced while computing other table entries in a call of MATRIX-CHAIN-ORDER. Show that the total number of references for the entire table is:

$$\sum_{i=1}^{n}\sum_{j=1}^{n}(R(i,j)) = \frac{n^3 - n}{6} \tag{1}$$

```
1   MATRIX-CHAIN-ORDER(p)
2   n  length[p] - 1
3    for i  1 to n
4         do m[i, i]  0
5    for l  2 to n       l is the chain length.
6          do for i  1 to n - l + 1
7                 do j  i + l - 1
8                    m[i, j]
9                    for k  i to j - 1
10                        do q  m[i, k] + m[k + 1, j] + pi-1 pkpj
11                           if q < m[i, j]
12                              then m[i, j]  q
13                                   s[i, j]  k
14  return m and s
```

Problem 2. Longest Common Subsequence

The longest common subsequence (LCS) problem is to find the longest subsequence common to all sequences in a set of sequences (often just two). (Note that a subsequence is different from a substring, for the terms of the former need not be consecutive terms of the original sequence.) It is a classic computer science problem, the basis of file comparison programs such as diff, and has applications in bioinformatics.

1. Determine an LCS of $(1, 0, 0, 1, 0, 1, 0, 1)$and $(0, 1, 0, 1, 1, 0, 1, 1, 0)$.

2. Show how to reconstruct an LCS from the completed c table and the original sequences $X = x_1, x_2, \cdots, x_m$ and $Y = y_1, y_2, \cdots, y_n$ in $O(m+n)$ time, without using the b table.

Problem 3. Assembly Line Scheduling

A car factory has two assembly lines, each with n stations. A station is denoted by $S_{i,j}$ where i is either 1 or 2 and indicates the assembly line the station is on, and j indicates the number of the station. The time taken per station is denoted by $a_{i,j}$. Each station is dedicated to some sort of work like engine fitting, body fitting, painting and so on. So, a

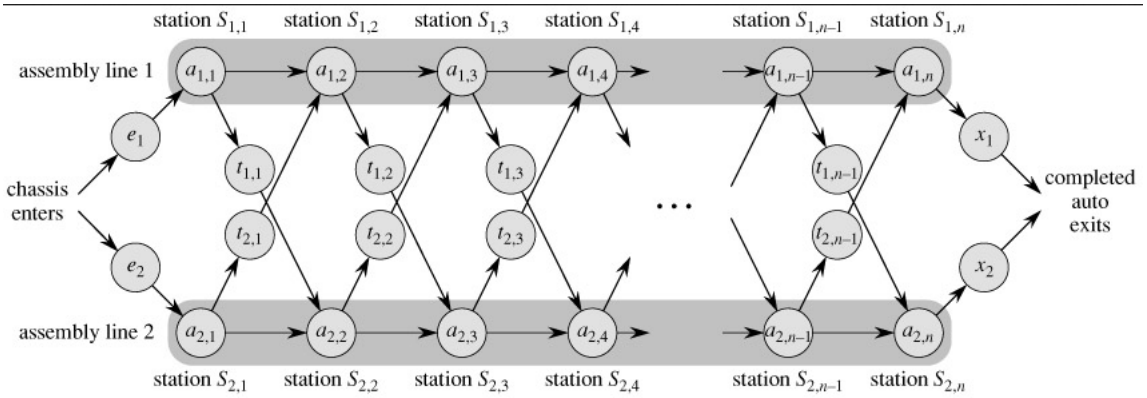

Figure 1: Assembly line scheduling

car chassis must pass through each of the n stations in order before exiting the factory. The parallel stations of the two assembly lines perform the same task. After it passes through station $S_{i,j}$, it will continue to station $S_{i,j+1}$, unless it decides to transfer to the other line. Continuing on the same line incurs no extra cost, but transferring from line i at station $j1$ to station j on the other line takes time $t_{i,j}$. Each assembly line takes an entry time e_i and exit time x_i which may be different for the two lines. Give an algorithm for computing the minimum time it will take to build a car chassis. See figure 1

1. Define an optimal function for the problem.
2. Derive the recursive equation for the optimal function.
3. Write out the pseudo codes for implementing the recursive equation.