Design and Analysis of Algorithms

Lab - 6

Dynamic Programming

Dynamic programming is a computer programming technique where an algorithmic problem is first broken down into sub-problems, the results are saved, and then the sub-problems are optimized to find the overall solution.

Dynamic Programming is mainly an optimization over plain recursion. Wherever we see a recursive solution that has repeated calls for the same inputs, we can optimize it using Dynamic Programming. The idea is to simply store the results of subproblems so that we do not have to re-compute them when needed later. This simple optimization reduces time complexities from exponential to polynomial.

A. Write a C/C++ program for the implementation of Longest Common Subsequence.
B. Write a C/C++ program for the implementation of Matrix Chain Multiplication.

Do the run time analysis and time complexity analysis with the different values of input size. Maintain the tabular data (n, execution time) and plot it graphically using data plotting tools.

Perform the complexity analysis of optimized solution using Dynamic Programming with the plain recursion solution for different size values, n.

Suggestion:

Longest Common Subsequence

The longest common subsequence (LCS) is defined as the longest subsequence that is common to all the given sequences, provided that the elements of the subsequence are not required to occupy consecutive positions within the original sequences.

If S1 and S2 are the two given sequences then, Z is the common subsequence of S1 and S2 if Z is a subsequence of both S1 and S2. Furthermore, Z must be a strictly increasing sequence of the indices of both S1 and S2.

In a strictly increasing sequence, the indices of the elements chosen from the original sequences must be in ascending order in Z.

Example:
S1=”BCDAACD”
S2=”ACDBAC”
Longest subsequence Z is “CDAC”, i.e, that is 4.

Matrix Chain Multiplication

We will be given a sequence of matrices; we must find the most efficient way to multiply these matrices together. The problem is not to actually perform the multiplications but to decide in which order to perform the multiplication.

Example:
Input: arr[] = {40, 20, 30, 10, 30}
There are 4 matrices of dimensions 40×20, 20×30, 30×10, 10×30.
The minimum number of multiplications are obtained by putting parentheses in the following way (A(BC))D. That is 20*30*10 + 40*20*10 + 40*10*30 =26000.