UNIT IV ALGORITHM DESIGN TECHNIQUES

Dynamic Programming: Matrix-Chain Multiplication – Elements of Dynamic Programming – Longest Common Subsequence- Greedy Algorithms: – Elements of the Greedy Strategy- An Activity-Selection Problem - Huffman Coding.

LONGEST COMMON SUBSEQUENCE

Given two strings, S1 and S2, the task is to find the length of the Longest Common Subsequence, i.e. longest subsequence present in both of the strings.

A longest common subsequence (LCS) is defined as the longest subsequence which is common in all given input sequences.

Longest Common Subsequence (LCS) using Memoization:

If we notice carefully, we can observe that the above recursive solution holds the following two properties:

1. Optimal Substructure:

See for solving the structure of L(X[0, 1, . . ., m-1], Y[0, 1, . . . , n-1]) we are taking the help of the substructures of X[0, 1, …, m-2], Y[0, 1,…, n-2], depending on the situation (i.e., using them optimally) to find the solution of the whole.

2. Overlapping Subproblems:

If we use the above recursive approach for strings "BD" and "ABCD", we will get a partial recursion tree as shown below. Here we can see that the subproblem L("BD", "ABCD") is being calculated more than once. If the total tree is considered there will be several such overlapping subproblems.

Approach: Because of the presence of these two properties we can use Dynamic programming or Memoization to solve the problem. Below is the approach for the solution using recursion.

Create a recursive function. Also create a 2D array to store the result of a unique state.

During the recursion call, if the same state is called more than once, then we can directly return the answer stored for that state instead of calculating again.

Longest Common Subsequence Algorithm

Let X=<x1,x2,x3....,xm> and Y=<y1,y2,y3....,ym> be the sequences. To compute the length of an element the following algorithm is used.

Step 1 − Construct an empty adjacency table with the size, n × m, where n = size of sequence X and m = size of sequence Y. The rows in the table represent the elements in sequence X and columns represent the elements in sequence Y.

Step 2 − The zeroth rows and columns must be filled with zeroes. And the remaining values are filled in based on different cases, by maintaining a counter value.

Case 1 − If the counter encounters a common element in both X and Y sequences, increment the counter by 1.

Case 2 − If the counter does not encounter common elements in X and Y sequences at T[i, j], find the maximum value between T[i-1, j] and T[i, j-1] to fill it in T[i, j].

Step 3 − Once the table is filled, backtrack from the last value in the table. Backtracking here is done by tracing the path where the counter incremented first.

Step 4 − The longest common subsequence obtained by noting the elements in the traced path.

Example: