Manish Semwal

204101032

Ans 1:

OPT (m, n) computes the **length of the Longest Common Subsequence** (LCS) present in both the given sequences X and Y of length m and n respectively.

Justification:

In the given algorithm we scan the given sequences X and Y from right to left.

1. **OPT (i, j) = 0 ; if i = 0 or j = 0:**

   The function returns '0', if any of the sequence has been scanned completely.

2. **OPT(i, j) = 1 + OPT (i−1, j−1) ; if i > 0, j > 0 and x_i = y_j:**

   The function returns '1 + the length of the LCS in X and Y before x_i and y_j respectively'. The '1' here shows that x_i (or y_j) has been counted, which means it belongs to one of the common subsequence of X and Y.

3. **OPT(i, j) = max {OPT(i, j − 1), OPT(i − 1, j)} ; otherwise:**

   When 'x_i ≠ y_j' and 'i ≠ 0' and 'j ≠ 0' then, it returns the length which is maximum of the lengths returned by skipping 'x_i and not y_j' and 'y_j and not x_i'.

Hence at every point while scanning the sequences X and Y from right to left the OPT (i, j) is returning the length of the LCS in sequence x_1, x_2, … ,x_i and y_1, y_2, … ,y_j.

Therefore, when i=m and j=n, the function will return the length of the LCS in X and Y.

Ans 2:

In a digraph, OPT (s, t, k) computes the **cost of the highest weighted non-negative walk from 's' using at most k edges.**

Justification:

1. **OPT (u, v, j) = 0 ; if j = 0**

   The function returns '0' if the number of edges that can be included in the walk is '0'.

2. **OPT (u, v, j) = max {OPT (u, v, j−1), $\max_{(u,w)\in E}\{l_{uw}$ +OPT (w, v, j−1)}} ; otherwise**

   The function returns the maximum of two things a) and b) described below:

   a) **OPT (u, v, j−1):** that is decrease an edge count and don't travel to another node, return what has been calculated till now. This will be more than the other argument of max (which is b)) when travelling from the current node to any other node will not be optimal.

   b) **$\max_{(u,w)\in E}\{l_{uw}$ +OPT (w, v, j−1):** that is return the max of the 'optimal walk from all adjacent vertices of 'u' with 'j-1' edges + the edge weight of the edge between 'u' and the corresponding vertex from where the optimum walk has been calculated'.

Hence, OPT (s, t, k) may not reach the 't' vertex but it'll definitely return the cost of the highest weighted walk from 's' node to some node in given digraph.