Design And Analysis Of Algorithms Assignment : 3

Name : Avik Samanta
Roll no. : 204101016

Submission Date : November 02, 2020

1 Ques. : 1

- $X = x_1, x_2, ..., x_m$ and $Y = y_1, y_2, ..., y_n$, and for $1 \le i \le m$ and $1 \le j \le n$:-

$$OPT(i,j) = \begin{cases} 0 & \text{if i = 0 , j = 0} \\ 1 + OPT(i-1, j-1) & \text{if } i > 0, j > 0 \text{ and } x_i = y_j \\ max(OPT(i, j-1), OPT(i-1, j)) & \text{otherwise} \end{cases}$$

- **Claim :** OPT(m, n) computes **the length of Longest Common Sub-sequence (LCS)** of the sequence X and Y.
- **Note** : When we are calling the $OPT(i,j)$, we are considering the prefix sequences of X and Y, *i.e.* $(x_1, x_2, ..., x_i)$ and $(y_1, y_2, ..., y_j)$.
- **Basis** : So, if **any of the sequence is empty**, then there won't be any element to match with the other sequence. definitely **longest common sub-sequence will be of length 0**.
  From the equation OPT(i, 0) = OPT(0, j) = OPT(0, 0) = 0, which is true.
- **Induction Hypothesis :** $OPT(i,j)$ $[\forall(i,j) < (u,v),\ \exists u,v : (1 < u \le m) \text{ and } (1 < v \le n)]$ gives us the length of Longest Common Sub-sequence (LCS), for the sequences $(x_1, x_2, ..., x_i)$ and $(y_1, y_2, ..., y_j)$ [here, $[(i,j) < (u,v)] \equiv [(i \le u, j \le v)$ except $(i = u$ and $j = v)$ together not possible]]
- **Inductive Step :** What does OPT(u, v) give us then?
  - **Case 1** : if **the corresponding x_u and y_v are the same, match them**. Then recursively solve for the remaining part.
    * From Induction Hypothesis, as $(u-1, v-1) < (u,v)$, $OPT(u-1, v-1)$ gives us the length of the LCS for the sequences $(x_1, x_2, ..., x_{u-1})$ and $(y_1, y_2, ..., y_{v-1})$
    * Now as we matched x_u and y_v, the length of the LCS for the sequences $(x_1, x_2, ..., x_u)$ and $(y_1, y_2, ..., y_v)$ will be:-
      [1 (match x_u and y_v) + length of LCS for $(x_1, x_2, ..., x_{u-1})$ and $(y_1, y_2, ..., y_{v-1})$] = $1 + OPT(u-1, v-1)$
    * **Proof for Longest : if $x_u = y_v$, then any LCS Z has $x_u = y_v = l$ as its last symbol**. Justification : suppose that Z' is any longest common sub-sequence that does not end with l, then we can always extend it by appending l to Z' to obtain another legal longer common sub-sequence. This ensures that **resulting common sub-sequence we get from this, will always be the longest**.
  - **Case 2** : Consider the case when **corresponding x_u and y_v values are not the same**. So, they can't be matched. In that case, we will have two cases :-
    * **case 2(a) : Leave x_u unmatched**, or assume x_u won't be a part of the LCS. Then the LCS of **prefix sequences** $(x_1, x_2, ..., x_{u-1})$ **and** $(y_1, y_2, ..., y_v)$ will be the target LCS. As $(u-1, v) < (u,v)$, from the Induction Hypothesis we know that $OPT(u-1, v)$ gives us the length of the LCS for $(x_1, x_2, ..., x_{u-1})$ and $(y_1, y_2, ..., y_v)$
    * **case 2(b) : Leave y_v unmatched**, or assume y_v won't be a part of the LCS. Then the LCS of **prefix sequences** $(x_1, x_2, ..., x_u)$ **and** $(y_1, y_2, ..., y_{v-1})$ will be the target LCS. As $(u, v-1) < (u,v)$, From the Induction Hypothesis we know that $OPT(u, v-1)$ gives us the length of the LCS for $(x_1, x_2, ..., x_u)$ and $(y_1, y_2, ..., y_{v-1})$

And then we will take **the maximum result of those two cases** we considered.
$max(OPT(u, v-1)$ [y_v not matched], $OPT(u-1, v)$[x_u not matched])

- Thus **the $OPT(m, n)$ will give us the length of the Longest Common Sub-sequence (LCS) for the whole sequence X and Y** [**Proved(By Induction)**].

2 Ques. : 2

- Given a graph $G(V, E)$ with integer edge weights l_{uv} for each edge $(u, v) \in E$:-

$$OPT(u,v,j) = \begin{cases} 0 & \text{if j = 0} \\ max[OPT(u,v,j-1), max_{(u,w)\in E}[l_{uw} + OPT(w,v,j-1)]] & \text{otherwise} \end{cases}$$

- If we look closely, the parameter v [in $OPT(u, v, j)$], has no effect / role. Neither in the base case or termination condition, nor in the recursive structure. The recursive structure only depends on u and j [in $OPT(u, v, j)$].
- **Claim :** $OPT(s, t, k)$ computes **maximum weighted walk** (not path, as vertices and edges may be repeated), **starting from** s, **using at most** k **edges**.
- **Base Case :** Max weighted walk we can achieve is 0, without traversing any edge. From the equation above : $OPT(u, v, 0) = 0$ [for any $u, v \in V$ and $j = 0$], which is true.
- **Induction Hypothesis :** $OPT(u, v, j-1)$ gives us the walk with maximum weight, starting from u, using at most $j-1$ edges [with any $u, v \in V$ and $j > 0$].
- **Inductive Step :** Now let's say we are given the chance to use at most j edges. Then we will have two options :-
  - **Case 1 :** We might choose not to use the last edge (j^{th} edge), if we consider, adding another edge does not give us weight greater than what we had with $j-1$ edges. This can happen, as in the question it is mentioned that we can have negative edge weights. Fot that case we have : $OPT(u, v, j-1)$, and from the inductive step, we know, this is what gives us the max weighted walk with $j-1$ edges.
  - **Case 2 :** We might choose to use the last edge (j^{th} edge), if we consider, adding another edge does give us weight greater than what we had with $j-1$ edges. In this case, we will consider all the outgoing edges from u to the adjacent nodes [i.e. w], and take the edge weight l_{uw}. And then take the walk from w with having at most $j-1$ edges. And from the induction hypothesis we know that $OPT(w, v, j-1)$ would give us the max weighted path starting from w using at most $j-1$ edges. So definitely $l_{uw} + OPT(w, v, j-1)$ would definitely give us the max weighted walk with at most j edges, if we take the (u, w) edge to start with. so $max_{(u,w)\in E}[l_{uw} + OPT(w, v, j-1)]$ will explore all the adjacent of u, and give us the maximum from them.
  - Now, what we have to do, we will observe the results from both **case 1** and **case 2** and the maximum of them is what we want.
  - So what we have now :-
    [$OPT(u, v, 0) = 0 =$ walk with maximum weight using at most 0 edges] And
    [$OPT(u, v, j-1) =$ walk with maximum weight using at most $j-1$ edges] **implies** [$OPT(u, v, j)$ = walk with maximum weight using at most j edges] (for any $u, v \in V$ and $j > 0$)
- That means, $OPT(s, t, k)$ will give us **the maximum weighted walk, starting from s, using at most k edges** [If we look closely, the maximum weight can never be negative. Even if we get all the values as negative, as the termination case will always give us 0, then the maximum weighted walk can never be negative]. [**Proved (by induction)**]