LCS (longest common subsequence) summary

- Given two arrays $X[1..n]$ and $Y[1..m]$, find their longest common subsequence.
- Choice of subproblem: Denote by $c(i,j)$ the length of the LCS of X_i and Y_j, where X_i is the array consisting of the first i elements of X, and Y_j is the array consisting of the first j elements of Y. To find the LCS of X and Y we call $c(n,m)$
- Recursive definition of

```
c(i, j)
//returns the length of the LCS of the first i elements of X and the first j elements of Y

    IF (i == 0 or j == 0): return 0

    else

        IF X[i] == Y[j]: return 1 + c(i − 1, j − 1)
        Else: return max{c(i − 1, j), c(i, j − 1)}
```

- Correctness: It tries all possibilities.
- Dynamic programming solution, top-down with memoization: We create $table[0..n][0..m]$, where $table[i][j]$ will store the result of $c(i,j)$. We initialize all entries in the table as 0 and call $cwithDP(n,m)$.

```
cwithDP(i, j)
//returns the length of the LCS of the first i elements of X and the first j elements of Y

    IF (i == 0 or j == 0): return 0

    else

        IF (table[i][j] ≠ 0): RETURN table[i][j]
        IF X[i] == Y[j]: answer 1 + cwithDP(i − 1, j − 1)
        Else: answer= max{cwithDP(i − 1, j), cwithDP(i, j − 1)}
        table[x] = answer
        return answer
```

Running time: $O(m \cdot n)$

- Dynamic programming, bottom-up:
- Computing full solution: