Describe and analyze iterative ***dynamic programming*** algorithms to evaluate the following recurrences from Wednesday's lab. Each recurrence refers to a fixed array $A[1..n]$ of integers.

Each algorithm should compute all possible values for its target function(s), starting with the base case(s) and working toward the final desired value(s), by considering intermediate subproblems in the correct order.

1. **Identify the recursive subproblems.** What are all the different ways that your recursive algorithm can call itself?
2. **Analyze the running time.** Add up the running times of evaluating the function for all possible subproblems, *ignoring recursive calls*.
3. **Choose an appropriate memoization data structure.** For most dynamic programming problems, each recursive subproblem is identified by a few integers, so you can use a multidimensional array; however, some problems require more interesting data structures. ***Do NOT use a hash table.***
4. **Identify dependencies.** Except for the base cases, every recursive subproblem depends on other subproblems—which ones? Draw a picture of your data structure, pick a generic element, and draw arrows from each of the other elements it depends on. Then formalize your picture.
5. **Find a good evaluation order.** Order the subproblems so that each subproblem comes *after* the subproblems it depends on. Typically, you should consider all of the base cases first, then subproblems that depends only on the base cases, and so on. ***Be careful!***
6. **Write down the algorithm.** You know what order to consider the subproblems, and you know how to solve each subproblem. So do that! If your data structure is an array, this usually means writing a few nested for-loops around the recurrence.

---

1. Add a sentinel value $A[0] = -\infty$, and then compute $LIS(0,1)$, where

$$LIS(i,j) = \begin{cases} 0 & \text{if } j > n \\ LIS(i,j+1) & \text{if } j \le n \text{ and } A[i] \ge A[j] \\ \max\{LIS(i,j+1), 1 + LIS(j,j+1)\} & \text{otherwise} \end{cases}$$

2. Add a sentinel value $A[n+1] = \infty$, and then compute $LIS(n,n+1)$, where

$$LIS(i,j) = \begin{cases} 0 & \text{if } i < 1 \\ LIS(i-1,j) & \text{if } i \ge 1 \text{ and } A[i] \ge A[j] \\ \max\{LIS(i-1,j), 1 + LIS(i-1,i)\} & \text{otherwise} \end{cases}$$

3. Compute $\max_i LIS(i)$, where

$$LIS(i) = \begin{cases} 1 & \text{if } A[j] \le A[i] \text{ for all } j > i \\ 1 + \max\left\{LIS(j) \;\middle|\; j > i \text{ and } A[j] > A[i]\right\} & \text{otherwise} \end{cases}$$

4. Add a sentinel value $A[0] = \infty$, and then compute $LDS(0,1)$, where

$$LDS(i,j) = \begin{cases} 0 & \text{if } j > n \\ LDS(i,j+1) & \text{if } j \le n \text{ and } A[i] \le A[j] \\ \max\{LDS(i,j+1), 1 + LDS(j,j+1)\} & \text{otherwise} \end{cases}$$

5. Add a sentinel value $A[0] = -\infty$ and compute $LAS^+(0,1)$, where

$$LAS^+(i,j) = \begin{cases} 0 & \text{if } j > n \\ LAS^+(i,j+1) & \text{if } j \le n \text{ and } A[j] \le A[i] \\ \max\left\{LAS^+(i,j+1), 1 + LAS^-(j,j+1)\right\} & \text{otherwise} \end{cases}$$

$$LAS^-(i,j) = \begin{cases} 0 & \text{if } j > n \\ LAS^-(i,j+1) & \text{if } j \le n \text{ and } A[j] \ge A[i] \\ \max\left\{LAS^-(i,j+1), 1 + LAS^+(j,j+1)\right\} & \text{otherwise} \end{cases}$$

6. Add a sentinel value $A[0] = -\infty$ and compute $LAS^+(0,1)$, where

$$LAS^+(i) = 1 + \max\left\{LAS^-(j) \;\middle|\; j > i \text{ and } A[j] < A[i]\right\}$$

$$LAS^-(i) = 1 + \max\left\{LAS^+(j) \;\middle|\; j > i \text{ and } A[j] > A[i]\right\}$$

To establish base cases, we define $\max \varnothing = 0$.

7. Compute $\max_{1 \le i < j \le n} LCS(i,j)$, where

$$LCS(i,j) = 1 + \max\left\{LCS(j,k) \;\middle|\; j < k \le n \text{ and } A[i] + A[k] > 2A[j]\right\}$$

To establish a base case, we define $\max \varnothing = 0$.

8. Compute $LPS(1,n)$, where

$$LPS(i,j) = \begin{cases} 0 & \text{if } i > j \\ 1 & \text{if } i = j \\ \max\left\{\begin{array}{l} LPS(i+1,j) \\ LPS(i,j-1) \end{array}\right\} & \text{if } i < j \text{ and } A[i] \ne A[j] \\ \max\left\{\begin{array}{l} 2 + LPS(i+1,j-1) \\ LPS(i+1,j) \\ LPS(i,j-1) \end{array}\right\} & \text{otherwise} \end{cases}$$