Two DP solutions.

4. Give a dynamic programming algorithm with running time $O(n^2)$ that finds the longest increasing subsequence in a sequence of numbers $a_1 a_2 \cdots a_n$ of length n. For example, the longest increasing subsequence in the sequence 10,3,5,2,7,1,18,4,12,17,6 is 3,5,7,12,17.

   Solution:

   - SUBPROBLEMS:$L[i]$ = length of the longest increasing subsequence that ends at $A[i]$;
   - ORDER:increasing from 0 to n;
   - START: initialize all $L[i]$ values to 1;
   - CONTINUE: $L[i] = 1 + max(L[j])$ if $A[j] < A[i](1 <= j < i)$;
   - END: return the maximum value of $L[i]$;
   - CORRECTNESS: Suppose that we have already calculated $L[1], L[2], \cdots, L[i-1]$ and we want to calculate $L[i]$. We must consider all indices $j < i$ such that $A[j] < A[i]$ and determine the maximum value of $L[j]$. We then add 1 to this maximum value to get $L[i]$. This value of $L[i]$ is the length of the longest increasing subsequence ending in $A[i]$.
   - RUNNING TIME: $O(n^2)$. For each element $A[i]$ in the input sequence, we need to consider all the previous elements $A[j]$ with indices $j < i$.
   - OPTIMAL OBJECT:To construct the actual longest increasing subsequence we use another array P to store the indices of the previous elements in the subsequence.If max at 'j' then $P[i] = j$, otherwise '$-'$.