- Longest Increasing Subsequence (LIS)

\[ a = [4, 2, 5, 3, 9, 7, 8, 10, 6] \]

\[ b = [2, 5, 7, 8, 10] \text{ length } = 5 \]

- attempt 1

6 is not in LIS, LIS \( [4, 2, 5, 3, 9, 7, 8, 10] \)

6 is in LIS

finding LIS \( [4, 2, 5, 3, 9, 7, 8, 10] \)

\[ \overline{\{2, 5, 7, 8, 10\}} \]

does not work because

\[ \overline{\{2, 5, 7, 8, 10, 6\}} \] is not an increasing subsequence

want: \( LIS = [4, \ldots, 10] \) if every number \(< 6\)

- attempt 2

LIS recursive \( (0) \) return length of LIS ending at \( a[i] \)

\( [4, 2, 5, 3, 6] \)
- recursive search: call \text{LIS\_recursive} (2) multiple times
- dynamic programming

\underline{state:} let \( f[i] \) be the length of LIS ending at \( a[i] \).

\underline{transition function:}

\[
f[i] = \max \left\{ f[j] + 1 \mid \text{for every } j < i, \ a[j] < a[i] \right\}
\]

\[
\begin{align*}
1 & \ 2 \ 3 \ 4 \ 5 \ 6 \ 7 \ 8 \ 9 \\
\{4, 2, 5, 3, 9, 7, 8, 10, 6\} \\
\end{align*}
\]

\[
\begin{array}{cccccccc}
1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\
\hline
f[0] & 1 & 1 & 2 & 2 & 3 & 3 & 4 & 5 & 3 \\
\end{array}
\]

\underline{set up base case}

\underline{for } \ i = 1 \ \text{to} \ \ n
\underline{evaluate the transition function at } f[i]

\underline{code for outputting solution (output } \max \ f[i])

\underline{analyze running time}

\[
\text{running time} = \# \text{states } \times \text{time for evaluating one transition function}
\]

\[
\begin{align*}
\text{LIS: } & \ n \quad O(n) \\
\text{running time: } & \ O(n^2) \\
\text{Knapsack: } & \ nW \quad O(1) \\
\text{running time: } & \ O(nW)
\end{align*}
\]
- Proof for Correctness.

  Use induction.

  Induction hypothesis: "smaller subproblems are computed correctly."

  Before iterating for every \( j < i \), \( f[i, j] \) is length of LIS ending at \( a[i] \).

  Induction: when computing \( f[i, j] \)

  Let \( b[i] \) be the LIS ending at \( a[i] \).

  Case 0: \( b[i] \) has length 1, considered by the 1st case of transition function.

  Case 1: Let \( a[c] \) be the second-to-last number in \( b[i] \), by definition \( j < i \).

  \[
  a[c] \leq a[i, j] \]

  by IH, length \( b[i] \) \( \leq f[i, j] + 1 \)

  \( f[i, j] + 1 \) is considered in transition function.

  Therefore \( f[i, j] \) is also computed correctly.