CS 344: Design and Analysis of Computer Algorithms

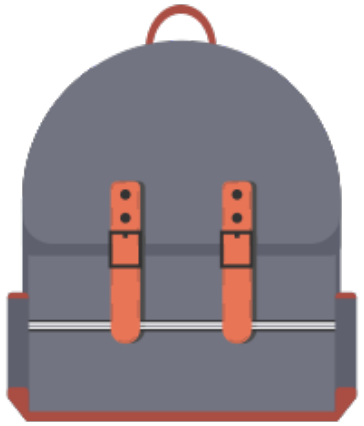
(Spring 2022 — Sections 5,6,7,8)

Lecture 11: Dynamic Programming: Knapsack, Longest Increasing Subsequence

The Knapsack Problem

The Knapsack Problem

- **Input:**
  - A collection of n items with value v_i and weight w_i
  - A knapsack of size W

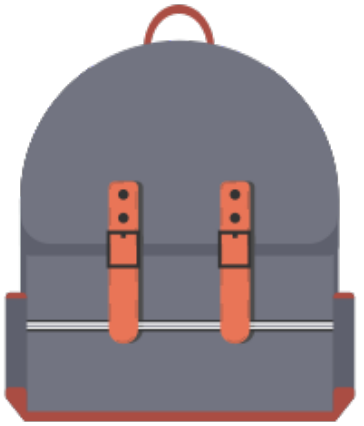


size: 35

value:	**1000**	**5**	**50**	**5**	**100**
weight:	**25**	**5**	**8**	**5**	**15**

The Knapsack Problem

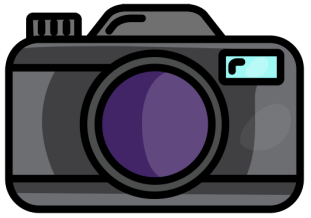
- **Output:**
  - The maximum value we can get by picking a subset S of items
  - The total weight of S should be less than Knapsack size



size: 35

value:	**1000**	**5**	**50**	**5**	**100**
weight:	**25**	**5**	**8**	**5**	**15**





A Dynamic Programming Algorithm

- Step one: **Specification**
  - For every $0 \leq i \leq n \quad , \quad 0 \leq j \leq W$ define:
  - $K(i, j)$: the maximum value we get by picking a subset of items from the first i items when we have a knapsack of size j
  - How to compute the final answer?
    - Return $K(n, W)$

A Dynamic Programming Algorithm

- Step two: **Solution**
- $$K(i,j) = \begin{cases} 0 & \text{if } i = 0 \text{ or } j = 0 \\ K(i-1,j) & \text{if } w_i > j \\ \max\{K(i-1,j-w_i) + v_i \ , \ K(i-1,j)\} & \text{otherwise} \end{cases}$$
- Proof?

A Dynamic Programming Algorithm

- Pick a 2-dimensional array D[1:n][1:W] initialized with 'undefined'
- **MemKnap**(i,j):
  - If D[i][j] $\neq$ undefined return D[i][j]
  - If i=0 or j=0, return D[i][j] = 0
  - If $w_i > j$ let D[i][j] = **MemKnap**$(i-1, j)$
  - Else let D[i][j] = max{**MemKnap**$(i-1, j-w_i) + v_i$, **MemKnap**$(i-1, j)$}
  - Return D[i][j]

A Dynamic Programming Algorithm

- Proof of correctness? This is the same formula as $K(i, j)$ so nothing else to prove
- Runtime?
  - There are $(n + 1) \cdot (W + 1)$ subproblems
  - Each takes $O(1)$ time
  - So total runtime is $O(nW)$

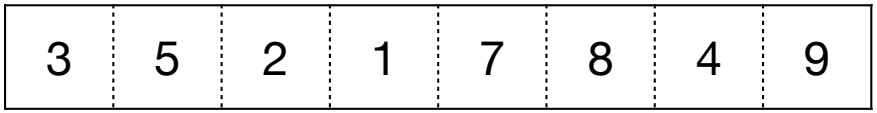
The Longest Increasing Subsequence Problem

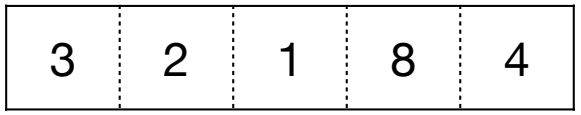
Subsequence of an Array

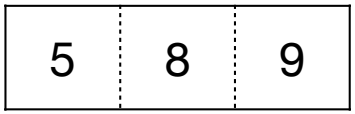
- For an array A[1:n], a subsequence of A:
  - Any array B obtained by removing zero or more entries of A but keeping the ordering intact

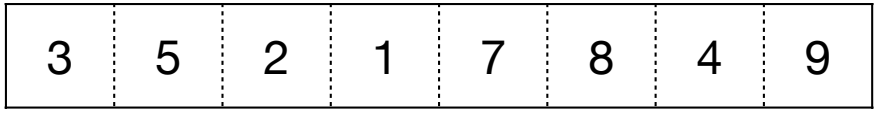
Subsequence of an Array

- For an array A[1:n], a subsequence of A:
  - Any array B obtained by removing zero or more entries of A but keeping the ordering intact
- Example:



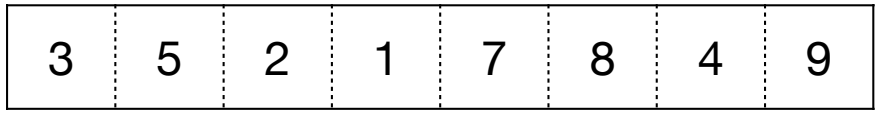


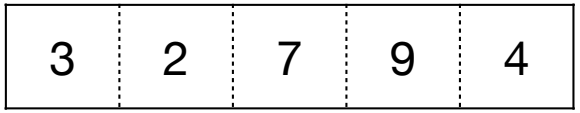


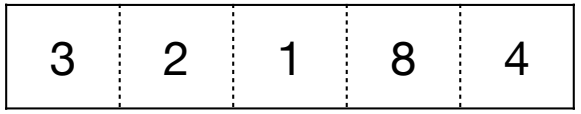


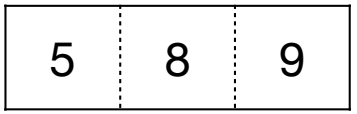
Subsequence of an Array

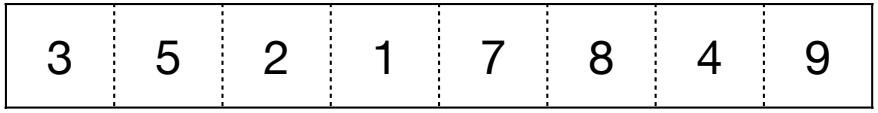
- For an array A[1:n], a subsequence of A:
  - Any array B obtained by removing zero or more entries of A but keeping the ordering intact
- Example:





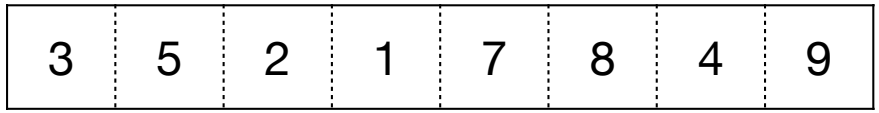


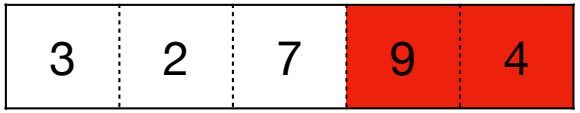


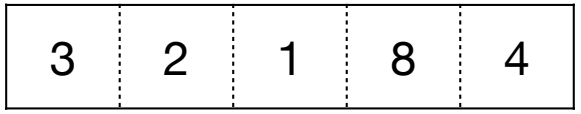


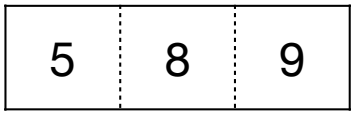
Subsequence of an Array

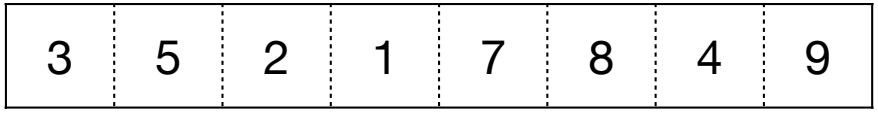
- For an array A[1:n], a subsequence of A:
  - Any array B obtained by removing zero or more entries of A but keeping the ordering intact
- Example:











Increasing Subsequence of an Array

- For an array A[1:n], an increasing subsequence of A:
  - Any subsequence B of A where B[1] < B[2] < B[3] < …

Increasing Subsequence of an Array

- For an array A[1:n], an increasing subsequence of A:
  - Any subsequence B of A where B[1] < B[2] < B[3] < …

- Example:

3	5	2	1	7	8	4	9

3	2	1	8	4

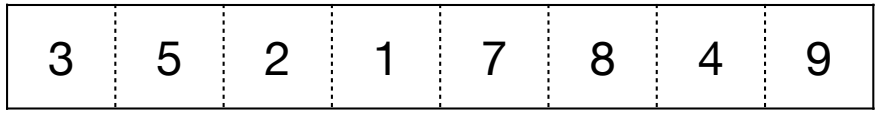
5	8	9

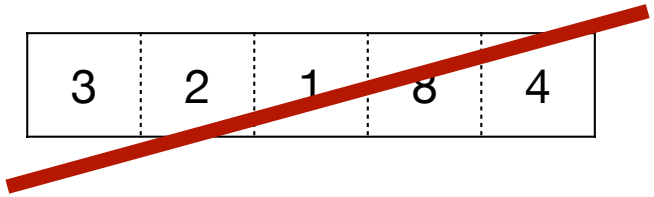
3	5	2	1	7	8	4	9

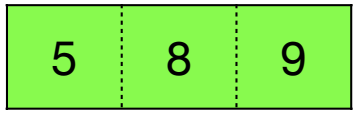
Increasing Subsequence of an Array

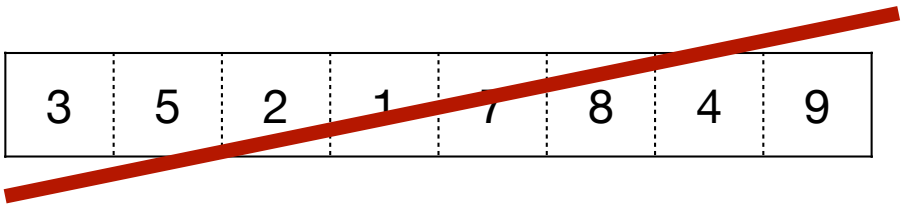
- For an array A[1:n], an increasing subsequence of A:
  - Any subsequence B of A where B[1] < B[2] < B[3] < …

- Example:









Longest Increasing Subsequence

- The Longest Increasing Subsequence (LIS) problem is defined:
- **Input:**
  - An array A[1:n] of n numbers
- **Output:**
  - Length of the longest increasing subsequence of A

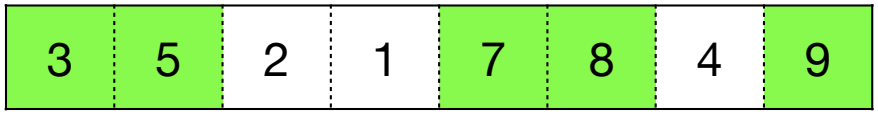
Longest Increasing Subsequence

- The Longest Increasing Subsequence (LIS) problem is defined:
- **Input:**
  - An array A[1:n] of n numbers
- **Output:**
  - Length of the longest increasing subsequence of A
- Example:

3	5	2	1	7	8	4	9

Longest Increasing Subsequence

- The Longest Increasing Subsequence (LIS) problem is defined:
- **Input:**
  - An array A[1:n] of n numbers
- **Output:**
  - Length of the longest increasing subsequence of A
- Example:



The answer is 5

Recursive Formula for LIS

- Step one: **Specification**
- For every $1 \leq i \leq n$, define:
  - $L(i)$: the length of longest increasing subsequence of $A[1 : i]$ **that ends in** $A[i]$

Recursive Formula for LIS

- Step one: **Specification**
- For every $1 \leq i \leq n$, define:
  - $L(i)$: the length of longest increasing subsequence of $A[1 : i]$ **that ends in** $A[i]$
- Example:

3	5	2	1	6	7	4	5

Recursive Formula for LIS

- Step one: **Specification**
- For every $1 \leq i \leq n$, define:
  - $L(i)$: the length of longest increasing subsequence of $A[1 : i]$ **that ends in** $A[i]$
- Example:

3	5	2	1	6	7	4	5

$L(1) = 1$ $L(2) = 2$ $L(3) = 1$ $L(4) = 1$ $L(5) = 3$ $L(6) = 4$ $L(7) = 2$ $L(8) = 3$

Recursive Formula for LIS

- Step one: **Specification**
- For every $1 \leq i \leq n$, define:
  - $L(i)$: the length of longest increasing subsequence of $A[1 : i]$ **that ends in** $A[i]$
- Example:

3	5	2	1	6	7	4	5

$L(1) = 1$ $L(2) = 2$ $L(3) = 1$ $L(4) = 1$ $L(5) = 3$ $L(6) = 4$ $L(7) = 2$ $L(8) = 3$

Recursive Formula for LIS

- Step one: **Specification**
- For every $1 \leq i \leq n$, define:
  - $L(i)$: the length of longest increasing subsequence of $A[1 : i]$ **that ends in** $A[i]$
- Example:

3	5	2	1	6	7	4	5

$L(1) = 1$ $L(2) = 2$ $L(3) = 1$ $L(4) = 1$ $L(5) = 3$ $L(6) = 4$ $L(7) = 2$ $L(8) = 3$

Recursive Formula for LIS

- Step one: **Specification**
- For every $1 \leq i \leq n$, define:
  - $L(i)$: the length of longest increasing subsequence of $A[1 : i]$ **that ends in** $A[i]$
- Example:

3	5	2	1	6	7	4	5

$L(1) = 1$ $L(2) = 2$ $L(3) = 1$ $L(4) = 1$ $L(5) = 3$ $L(6) = 4$ $L(7) = 2$ $L(8) = 3$

Recursive Formula for LIS

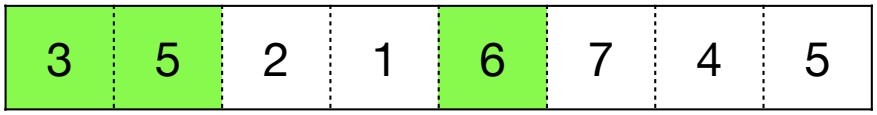
- Step one: **Specification**
- For every $1 \leq i \leq n$, define:
  - $L(i)$: the length of longest increasing subsequence of $A[1 : i]$ **that ends in** $A[i]$
- Example:

3	5	2	1	6	7	4	5

$L(1) = 1$ $L(2) = 2$ $L(3) = 1$ $L(4) = 1$ $L(5) = 3$ $L(6) = 4$ $L(7) = 2$ $L(8) = 3$

Recursive Formula for LIS

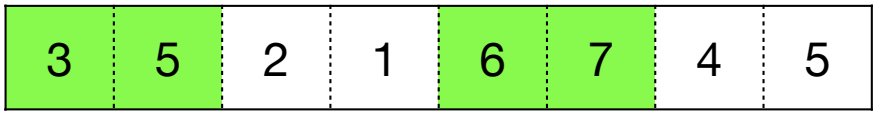
- Step one: **Specification**
- For every $1 \leq i \leq n$, define:
  - $L(i)$: the length of longest increasing subsequence of $A[1 : i]$ **that ends in** $A[i]$
- Example:



$L(1) = 1$ $L(2) = 2$ $L(3) = 1$ $L(4) = 1$ $L(5) = 3$ $L(6) = 4$ $L(7) = 2$ $L(8) = 3$

Recursive Formula for LIS

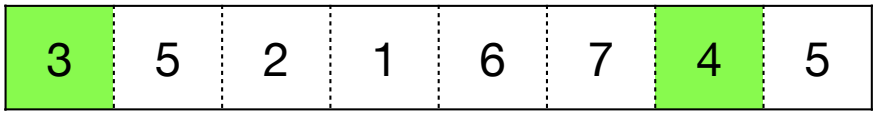
- Step one: **Specification**
- For every $1 \leq i \leq n$, define:
  - $L(i)$: the length of longest increasing subsequence of $A[1 : i]$ **that ends in** $A[i]$
- Example:



$L(1) = 1$ $L(2) = 2$ $L(3) = 1$ $L(4) = 1$ $L(5) = 3$ $L(6) = 4$ $L(7) = 2$ $L(8) = 3$

Recursive Formula for LIS

- Step one: **Specification**
- For every $1 \leq i \leq n$, define:
  - $L(i)$: the length of longest increasing subsequence of $A[1 : i]$ **that ends in** $A[i]$
- Example:



$L(1) = 1$ $L(2) = 2$ $L(3) = 1$ $L(4) = 1$ $L(5) = 3$ $L(6) = 4$ $L(7) = 2$ $L(8) = 3$

Recursive Formula for LIS

- Step one: **Specification**
- For every $1 \leq i \leq n$, define:
  - $L(i)$: the length of longest increasing subsequence of $A[1 : i]$ **that ends in** $A[i]$
- Example:

3	5	2	1	6	7	4	5

$L(1) = 1$ $L(2) = 2$ $L(3) = 1$ $L(4) = 1$ $L(5) = 3$ $L(6) = 4$ $L(7) = 2$ $L(8) = 3$

Recursive Formula for LIS

- Step one: **Specification**
- For every $1 \leq i \leq n$, define:
  - $L(i)$: the length of longest increasing subsequence of $A[1 : i]$ **that ends in** $A[i]$
- Example:

3	5	2	1	6	7	4	5

$L(1) = 1$ $L(2) = 2$ $L(3) = 1$ $L(4) = 1$ $L(5) = 3$ $L(6) = 4$ $L(7) = 2$ $L(8) = 3$

Recursive Formula for LIS

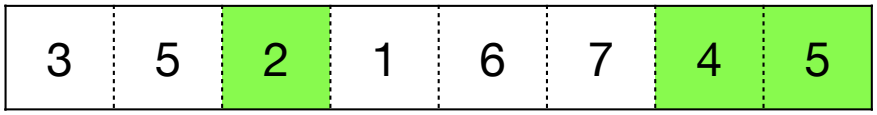
- Step one: **Specification**
- For every $1 \leq i \leq n$, define:
  - $L(i)$: the length of longest increasing subsequence of $A[1 : i]$ **that ends in** $A[i]$
- Example:

3	5	2	1	6	7	4	5

$L(1) = 1$ $L(2) = 2$ $L(3) = 1$ $L(4) = 1$ $L(5) = 3$ $L(6) = 4$ $L(7) = 2$ $L(8) = 3$

Recursive Formula for LIS

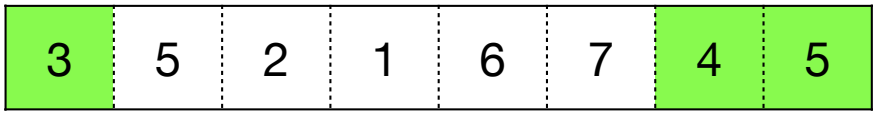
- Step one: **Specification**
- For every $1 \leq i \leq n$, define:
  - $L(i)$: the length of longest increasing subsequence of $A[1 : i]$ **that ends in** $A[i]$
- Example:



$L(1) = 1$ $L(2) = 2$ $L(3) = 1$ $L(4) = 1$ $L(5) = 3$ $L(6) = 4$ $L(7) = 2$ $L(8) = 3$

Recursive Formula for LIS

- Step one: **Specification**
- For every $1 \leq i \leq n$, define:
  - $L(i)$: the length of longest increasing subsequence of $A[1 : i]$ **that ends in** $A[i]$
- Example:



$L(1) = 1$ $L(2) = 2$ $L(3) = 1$ $L(4) = 1$ $L(5) = 3$ $L(6) = 4$ $L(7) = 2$ $L(8) = 3$

Recursive Formula for LIS

- Step one: **Specification**
- For every $1 \leq i \leq n$, define:
  - $L(i)$: the length of longest increasing subsequence of $A[1 : i]$ **that ends in** $A[i]$
- Calculating the final answer?

Recursive Formula for LIS

- Step one: **Specification**
- For every $1 \leq i \leq n$, define:
  - $L(i)$: the length of longest increasing subsequence of $A[1 : i]$ **that ends in** $A[i]$
- Calculating the final answer?
  - Return the largest value among $L(1), L(2), \ldots, L(n)$

Recursive Formula for LIS

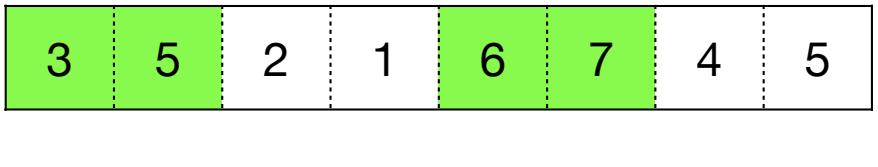
- Step one: **Specification**
- For every $1 \leq i \leq n$, define:
  - $L(i)$: the length of longest increasing subsequence of $A[1 : i]$ **that ends in** $A[i]$
- Calculating the final answer?
  - Return the largest value among $L(1), L(2), \ldots, L(n)$
- Example:

3	5	2	1	6	7	4	5

$L(1) = 1$ $L(2) = 2$ $L(3) = 1$ $L(4) = 1$ $L(5) = 3$ $L(6) = 4$ $L(7) = 2$ $L(8) = 3$

Recursive Formula for LIS

- Step one: **Specification**
- For every $1 \leq i \leq n$, define:
  - $L(i)$: the length of longest increasing subsequence of $A[1 : i]$ **that ends in** $A[i]$
- Calculating the final answer?
  - Return the largest value among $L(1), L(2), \ldots, L(n)$
- Example:



$L(1) = 1$ $L(2) = 2$ $L(3) = 1$ $L(4) = 1$ $L(5) = 3$ $L(6) = 4$ $L(7) = 2$ $L(8) = 3$

Recursive Formula for LIS

- Step two: **Solution**
- For every $1 \leq i \leq n$, define:

$$L(i) = \begin{cases} 1 & \textbf{if } i = 1 \\ \max \begin{cases} \max\{1 + L(j) \mid 1 \leq j < i \textbf{ and } A[j] < A[i]\} \\ 1 \end{cases} & \textbf{otherwise} \end{cases}$$

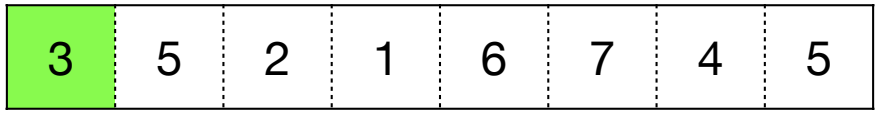
Example

$$L(i) = \begin{cases} 1 & \text{if } i = 1 \\ \max \begin{cases} \max\{1 + L(j) \mid 1 \leq j < i \text{ and } A[j] < A[i]\} \\ 1 \end{cases} & \text{otherwise} \end{cases}$$

3	5	2	1	6	7	4	5

Example

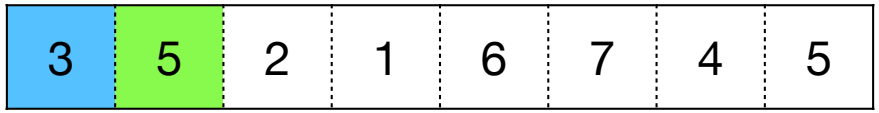
$$L(i) = \begin{cases} 1 & \text{if } i = 1 \\ \max \begin{cases} \max\{1 + L(j) \mid 1 \le j < i \text{ and } A[j] < A[i]\} \\ 1 \end{cases} & \text{otherwise} \end{cases}$$



$L(1) = 1$

Example

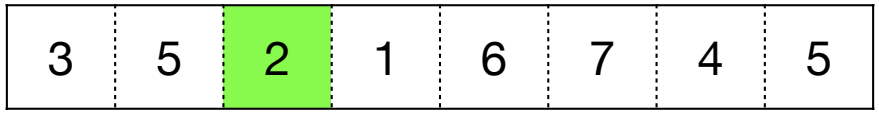
$$L(i) = \begin{cases} 1 & \text{if } i = 1 \\ \max \begin{cases} \max\{1 + L(j) \mid 1 \le j < i \text{ and } A[j] < A[i]\} \\ 1 \end{cases} & \text{otherwise} \end{cases}$$



$L(1) = 1 \quad L(2) = 2$

Example

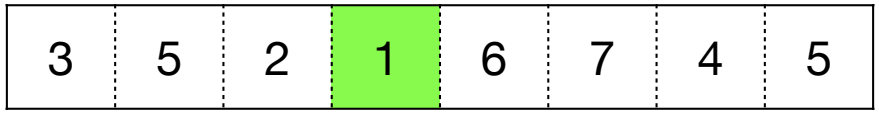
$$L(i) = \begin{cases} 1 & \text{if } i = 1 \\ \max \begin{cases} \max\{1 + L(j) \mid 1 \le j < i \text{ and } A[j] < A[i]\} \\ 1 \end{cases} & \text{otherwise} \end{cases}$$



$L(1) = 1 \quad L(2) = 2 \quad L(3) = 1$

Example

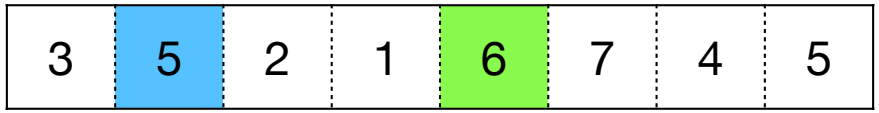
$$L(i) = \begin{cases} 1 & \text{if } i = 1 \\ \max \begin{cases} \max\{1 + L(j) \mid 1 \leq j < i \text{ and } A[j] < A[i]\} \\ 1 \end{cases} & \text{otherwise} \end{cases}$$



$L(1) = 1 \quad L(2) = 2 \quad L(3) = 1 \quad L(4) = 1$

Example

$$L(i) = \begin{cases} 1 & \text{if } i = 1 \\ \max \begin{cases} \max\{1 + L(j) \mid 1 \le j < i \text{ and } A[j] < A[i]\} \\ 1 \end{cases} & \text{otherwise} \end{cases}$$

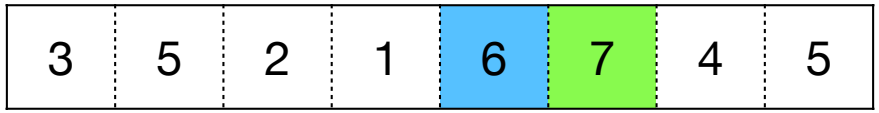


$L(1) = 1$ $L(2) = 2$ $L(3) = 1$ $L(4) = 1$

$L(5) = 3$

Example

$$L(i) = \begin{cases} 1 & \text{if } i = 1 \\ \max \begin{cases} \max\{1 + L(j) \mid 1 \le j < i \text{ and } A[j] < A[i]\} \\ 1 \end{cases} & \text{otherwise} \end{cases}$$

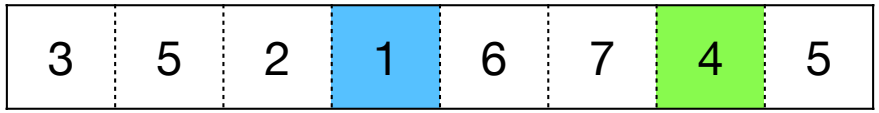


$L(1) = 1$ $L(2) = 2$ $L(3) = 1$ $L(4) = 1$

$L(5) = 3$ $L(6) = 4$

Example

$$L(i) = \begin{cases} 1 & \text{if } i = 1 \\ \max \begin{cases} \max\{1 + L(j) \mid 1 \leq j < i \text{ and } A[j] < A[i]\} \\ 1 \end{cases} & \text{otherwise} \end{cases}$$

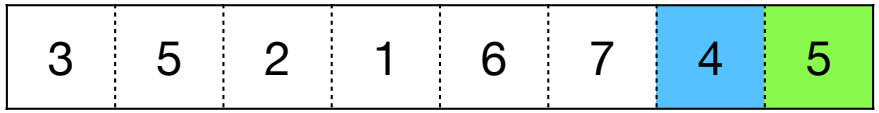


$L(1) = 1$ $L(2) = 2$ $L(3) = 1$ $L(4) = 1$

$L(5) = 3$ $L(6) = 4$ $L(7) = 2$

Example

$$L(i) = \begin{cases} 1 & \text{if } i = 1 \\ \max \begin{cases} \max\{1 + L(j) \mid 1 \le j < i \text{ and } A[j] < A[i]\} \\ 1 \end{cases} & \text{otherwise} \end{cases}$$



$L(1) = 1$ $L(2) = 2$ $L(3) = 1$ $L(4) = 1$

$L(5) = 3$ $L(6) = 4$ $L(7) = 2$ $L(8) = 3$

Example

$$L(i) = \begin{cases} 1 & \text{if } i = 1 \\ \max\left\{ \begin{array}{l} \max\{1 + L(j) \mid 1 \le j < i \text{ and } A[j] < A[i]\} \\ 1 \end{array} \right. & \text{otherwise} \end{cases}$$

3	5	2	1	6	7	4	5

$L(1) = 1$ $L(2) = 2$ $L(3) = 1$ $L(4) = 1$

$L(5) = 3$ $L(6) = 4$ $L(7) = 2$ $L(8) = 3$

Proof of Correctness

$$L(i) = \begin{cases} 1 & \text{if } i = 1 \\ \max \begin{cases} \max\{1 + L(j) \mid 1 \leq j < i \text{ and } A[j] < A[i]\} \\ 1 \end{cases} & \text{otherwise} \end{cases}$$

A Dynamic Programming Algorithm

- $$L(i) = \begin{cases} 1 & \text{if } i = 1 \\ \max \begin{cases} \max\{1 + L(j) \mid 1 \le j < i \text{ and } A[j] < A[i]\} \\ 1 \end{cases} & \text{otherwise} \end{cases}$$

- **DynLIS**(A[1:n])
  - Pick a 1-dimensional array T[1:n] initialized with 'undefined'
  - Let T[1] = 1. For i=2 to n:
    - Let T[i] = 1. For j=1 to i-1: if A[j] < A[i], let T[i] = max{T[i],T[j]+1}
  - Return maximum of T[1], T[2], …, T[n]

A Dynamic Programming Algorithm

- Proof of correctness?
  - Evaluation order ensures that each T[i] is calculated from previously computed T[j]
  - The correctness follows from correctness of recursive formula
- Runtime analysis:
  - An outer-loop with n iterations
    - For each iteration i of out-loop, an inner-loop with i iterations
  - Total runtime is $O(n^2)$

Summary of Dynamic Programming

Dynamic Programming

- Dynamic programming is just smart recursion
- Dynamic programming is NOT blindly filling up some tables

Dynamic Programming

- Every time you do dynamic programming:
  - Clearly specify subproblems in plain English
  - Design a recursive formula for solving subproblems from smaller ones
  - Prove the correctness of your recursive formula
  - Turn the formula into a dynamic programming algorithm using either memoization or bottom-up approach
  - Analyze the runtime of your algorithm