Dynamic Programming
and LIS

Longest increasing subsequence:

Given \( n \) numbers \( A_1, \ldots, A_n \), find increasing subsequence:

\[
S_1, \ldots, S_k \in \mathbb{N}^n
\]

\[
S_1 < S_2 < \ldots < S_k
\]  
(subsequence)

\[
A_{S_1} < A_{S_2} < \ldots < A_{S_k}
\]  
(increasing)

or maximum length \( k \).

will convert to longest path on DAG
$O(n^2)$ bottom-up approach:

Suppose you build your solution $S_1, \ldots, S_n$ from left to right.

Walking a path: $(S_i, A_i) \rightarrow (S_j, A_j) \rightarrow \ldots$

on the DAG $(i, A_i) \rightarrow (j, A_j) \quad i < j \land A_i < A_j$

Start at $(\infty, \infty)$ and end at $(0, 0)$

Answer = longest path in DAG $\rightarrow (-1)$

because:
Any path is an increasing sequence
any IS is a path
$\implies$ longest path = longest IS
Memoized View

\[ f_A(i) := \text{LIS ending at } A_i \]
\[ = \max_{j < i} f_A(j) + 1 \quad \text{if } A_j < A_i \]

Answer = \max_{i \in [n]} f_A(i)

OR (for poly time)

if \( f(A) \) is LIS of \( A \).

- let \( i = \text{argmax } A_i \)
- LIS either contains \( i \) or not

\[ f(A) = \max \left( f(A \setminus A_i), \left( f(A_1 \ldots A_{i-1}, A'_{i-1}) + 1 \right) \right) \]

How many possible inputs?

box to LL
\[ \Rightarrow n^2 \text{ options} \]
\[ + n \text{ time per option} \]
\[ = n^3 \]
$O(n \log n)$ Version

Given two possible starts on $A_{1..m}$:

$S_i, S_j$

When is one clearly superior?
When are they equivalent?

Answer: only will care about length & last value

$f_m(k) := \text{minimal last value of length-} k \text{ subsequence on } 1..m$

$f_{m+1}(k) = \min \{ f_m(k) \} \text{ s.t. } A_{m+1} \text{ is } f_m(k-1) < A_{m+1}$

$f_m: [z_1, z_2, \ldots, z_t, z_{t+1}, \ldots, z_k]$

If $A_{m+1} \in (z_t, z_{t+1})$:

$f_{m+1}: [z_1, z_2, \ldots, z_t, A_{m+1}, z_{t+1}, \ldots, z_k]$

$\Rightarrow$ update is $O(\log n)$ binary search

$\Rightarrow O(n \log n)$ time