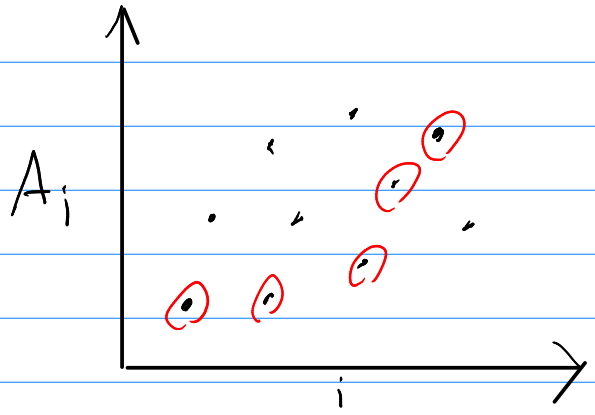
Dynamic Programming and LIS

Longest increasing Subsequence:



Given n numbers $A_1, \ldots, A_n$

find increasing subsequence:

$s_1, \ldots, s_k \in [n]$

$s_1 < s_2 < \ldots < s_k$ (subsequence)

$A_{s_1} < A_{s_2} < \ldots < A_{s_k}$ (increasing)

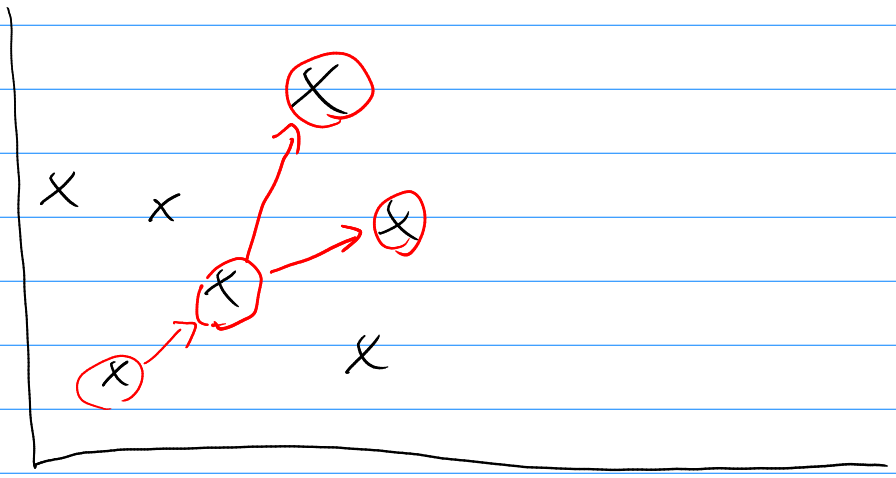
of maximum length k.

---

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: $(S_1, A_{S_1}) \to (S_2, A_{S_2}) \to \ldots$

on the DAG $(i, A_i) \to (j, A_j) \iff i < j \;\&\; A_i < A_j$

start at $(-\infty, \infty)$ end at (∞, ∞)

Answer = longest path on DAG (-1)

because:

Any path is an increasing sequence

any IS is a path

$\Rightarrow$ longest path = longest IS

Memoized View

$$f_A(i) := \text{LIS ending at } A_i$$

$$= \max_{j<i,\ A_j<A_i} f_A(j) + 1$$

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

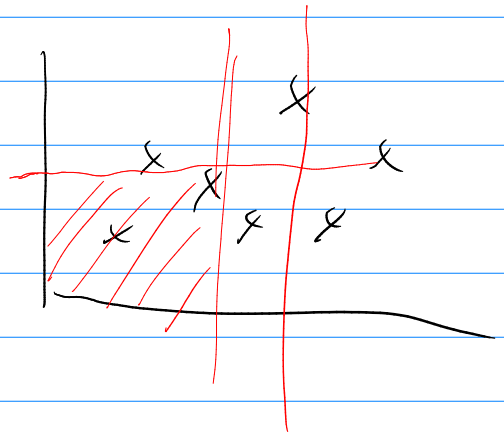
OR (for poly time)

if $f(A) = \text{LIS of } A$.

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

$$f(A) = \max\left(f(A \setminus A_i),\ f(A_{1,\dots,} A'_{i-1}) + 1 \right)$$

How many possible inputs?



box to LL

$\Rightarrow n^2$ options

$+ n$ time per option

$= n^3$

$O(n \log n)$ version

Given two possible starts, on $A_1, \ldots, A_n$:

$s_1, \ldots, s_i$

$s'_1, \ldots, s'_j$

When is one clearly superior?
When are they equivalent?

Answer: only will care about length & last value

$f_n(K) :=$ minimal last value of length-K subsequence on $1, \ldots, n$

$f_{n+1}(K) = \min(f_n(K),$ ← don't use A_{n+1}
A_{n+1} if $f_n(K-1) < A_{n+1})$

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

if $A_{n+1} \in (z_t, z_{t+1})$:

f_{n+1}: $[z_1, z_2, \ldots, z_t, A_{n+1}, z_{t+2}, \ldots, z_\ell]$

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