Recursion $\longrightarrow$ Dynamic Programming

Longest Increasing Subsequence

$3\ \cancel{1}\ 4\ \cancel{1}\ 5\ \cancel{9}\ \cancel{2}\ 6\ \cancel{5}\ \cancel{3}\ \cancel{5}\ \cancel{5}\ \big|\ 8\ 9\ 7\ 9\ 3\ 2\ 3\ 8\ 4\ 6\ 2\ 6$ $\qquad A[1..n]$

global var

Is this next?

$$\begin{array}{c|cccccccccccc} i & j & & & & & & & & & & & n \\ 6 & 8 & 9 & 7 & 9 & 3 & 2 & 3 & 8 & 4 & 6 & 2 & 6 \end{array}$$

Is this next?

$LIS(i,j)$ = length of longest increasing subsequence of $A[j..n]$

where all #s are bigger than $A[i]$

$$LIS(i,j) = \begin{cases} 0 & \text{if } j > n \\ LIS(i, j+1) & \text{if } A[i] \geq A[j] \\ \max \begin{Bmatrix} 1 + LIS(j, j+1) \\ LIS(i, j+1) \end{Bmatrix} & \text{if } A[i] < A[j] \end{cases}$$

$$LISbigger(i,j) = \begin{cases} 0 & \text{if } j > n \\ LISbigger(i, j+1) & \text{if } A[i] \geq A[j] \\ \max \left\{ \begin{array}{c} LISbigger(i, j+1) \\ 1 + LISbigger(j, j+1) \end{array} \right\} & \text{otherwise} \end{cases}$$

$O(n^2)$ different ways to call

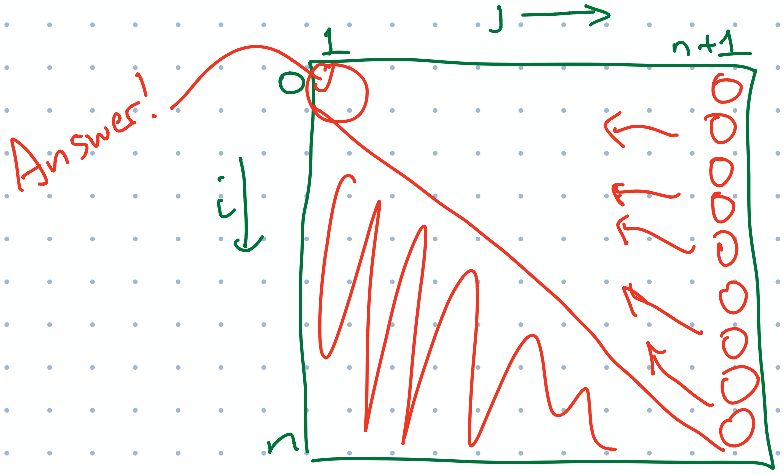
```
LISBIGGER(i, j):
  if j > n
      return 0
  else if A[i] ≥ A[j]
      return LISBIGGER(i, j + 1)
  else
      skip ← LISBIGGER(i, j + 1)
      take ← LISBIGGER(j, j + 1) + 1
      return max{skip, take}
```

$T(n) =$

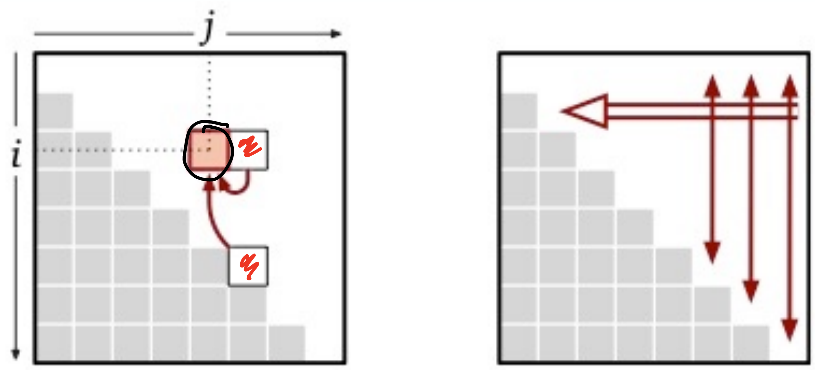
$T(n-1)$
$+ T(n-1)$
$+ O(1)$

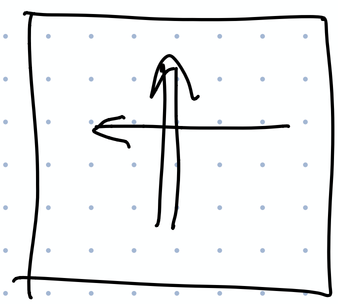
```
LIS(A[1..n]):
  A[0] ← −∞
  return LISBIGGER(0, 1)
```

Memoization — Remember what you've done



LIS[0..n, 1..n+1]





```
FastLIS(A[1..n]):
  A[0] ← −∞                          ⟨⟨Add a sentinel⟩⟩
  for i ← 0 to n                     ⟨⟨Base cases⟩⟩
      LISbigger[i, n+1] ← 0
  for j ← n down to 1
      for i ← 0 to j−1               ⟨⟨...or whatever⟩⟩
          keep ← 1 + LISbigger[j, j+1]
          skip ← LISbigger[i, j+1]
          if A[i] ≥ A[j]
              LISbigger[i, j] ← skip
          else
              LISbigger[i, j] ← max{keep, skip}
  return LISbigger[0, 1]
```

evaluation order

recursive call

recurrence

i

6 ~~5~~ ~~3~~ ~~5~~ 8 9 7 9 ~~3~~ ~~2~~ ~~3~~ ---

What's next?

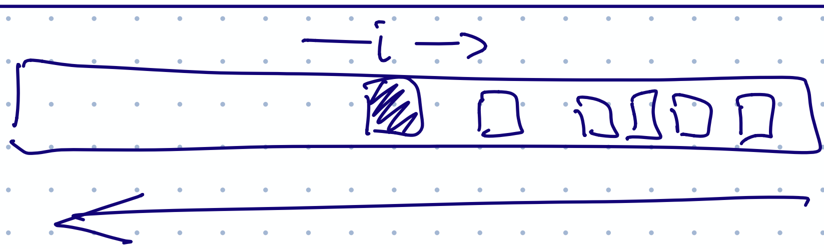
LIS2(i) = length of the longest incr subseq of $A[i..n]$ starting with $A[i]$?

$$\text{LIS2}(i) = 1 + \max\left\{ \text{LIS2}(j) \;\middle|\; \begin{array}{c} i < j \le n \\ A[j] > A[i] \end{array} \right\}$$

$$\max \varnothing = 0$$

```
LISFirst(i):
    best ← 0
    for j ← i + 1 to n
        if A[j] > A[i]
            best ← max{best, LISFirst(j)}
    return 1 + best
```

```
LIS(A[1..n]):
    A[0] ← −∞
    return LISFirst(0) − 1
```



```
FastLIS2(A[1..n]):
  A[0] = −∞                                    ⟨⟨Add a sentinel⟩⟩
  for i ← n downto 0
      LISfirst[i] ← 1
      for j ← i+1 to n                         ⟨⟨...or whatever⟩⟩
          if A[j] > A[i] and 1 + LISfirst[j] > LISfirst[i]
              LISfirst[i] ← 1 + LISfirst[j]
  return LISfirst[0] − 1                       ⟨⟨Don't count the sentinel⟩⟩
```

$\boxed{O(n^2)}$