374 10/6/22

More dynamic programming

Longest increasing subsequence:
A[1...n] real numbers
Compute LIS (longest increasing subsequence).

A[1...n] $A[n+1] = +\infty$

$L(i,j)$ ≡ length of LIS in $A[1...i]$ with all numbers selected to the subsequence smaller than $A[j]$ $\boxed{i < j}$

$L(n, n+1)$

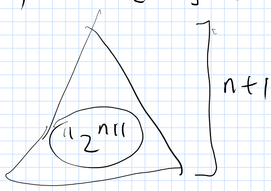
$$L(i,j) = \begin{cases} 0 & i=0 \\ L(i-1,j) & A[i] > A[j] \\ \max\left(L(i-1,i)+1,\ L(i-1,j)\right) & A[i] < A[j] \end{cases}$$

```
LIS(i,j):
  if i=0 then return 0
  if A[i] > A[j] then return LIS(i-1,j)
  x ← LIS(i-1,i)+1
  y ← LIS(i-1,j)
  return max(x,y).
```

If $(i,j) \in C$ then return C.val(i,j) (C ← cache)
store value returned in cache C.

$O(n^2)$

LIS(n, n+1) $A[1...n]$ $A[n+1] = +\infty$



$\boxed{L(10,15) = 7}$

memoization

$i = 0, \dots, n$
$j = 1, \dots, n+1$
$(n+1)^2$

Binom(n, i) ≡ # of ways to pick i elements out of n if order doesn't matter

$\binom{n}{i} = \binom{n-1}{i} + \binom{n-1}{i-1}$

$\binom{n-1}{i-1}$ $\binom{n-1}{i}$

$\binom{n}{n} = 1$ $\binom{n}{0} = 1$

$\binom{n}{i} \sim O(n^2)$

Fibonacci

$F_n = F_{n-1} + F_{n-2}$ $F_0 = 0$ $F_1 = 1$

$F_n \approx \phi^n$

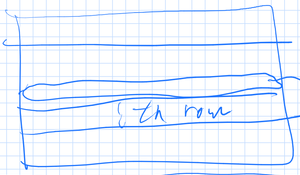
Explicit memoization

Implement the memoization using an array/stack directly

M[n+1, n+1] ← null

```
LISM(i,j):
  if M[i,j] ≠ null then return M[i,j]
  if i=0 then
    M[i,j] ← 0, return 0
  x ← LISM(i-1,j)
  if A[i] > A[j] then
    M[i,j] ← x, return x
  y = LISM(i-1,i)+1
  M[i,j] ← max(x,y)
  return M[i,j]
```

Explicit filling of the table with no recursion and with loops ≡ dynamic programming



$O(n^2)$ space $\Rightarrow O(n)$

t[a]