CS3000
5/28 - Weds

Admin

- HW3 due Fri 9pm
- APP5 due 11:30am 5/29

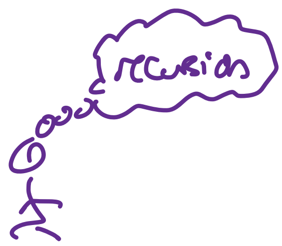
Agenda

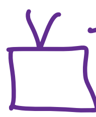
1. DP summary/review
2. LCS problem
3. DP bottom-up
4. APP5

1. DP Summary

- optimization problem
  - pick "best" valid solution
- DP approach
  1. Define recursive equation
  2. When computing value of subproblem...
     - already solved? look it up
     - o/w, solve and save

- value of opt. solution
  - ex: $ from selling rods
- actual opt. solution
  - ex: cuts in rod to get to our optimal soln



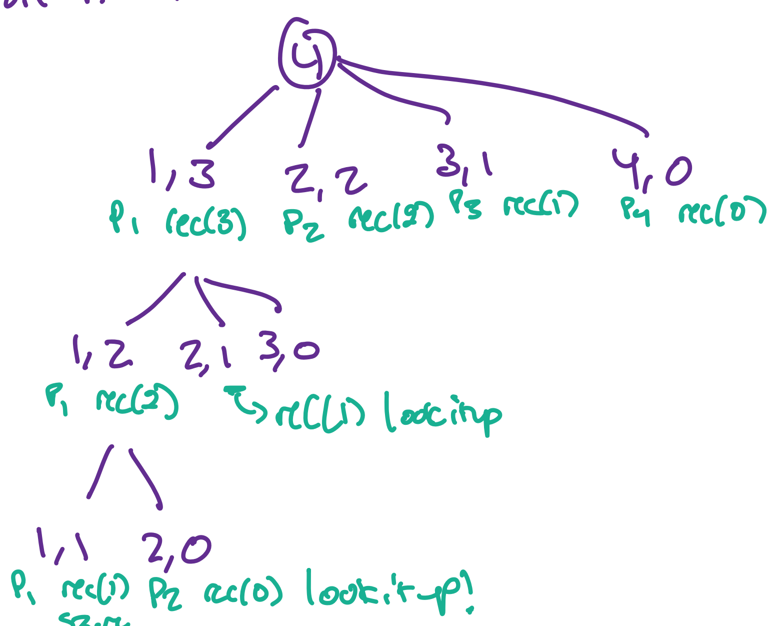


- same recursion?
- saves solutions

yesterday... top-down DP

- still recursive
- don't recurse unless we need to!

ex rec() start n=4



1,0
rec(0)
base case!
save

r store values of opt. solns

$\langle 0, \boxed{\ }, \boxed{\ }, \boxed{\ }, \boxed{\ } \rangle$
0 1 2 3 4

DP...

- expect: optimization
- need: overlapping subproblems

$\leadsto$ Bottom-up instead?

- asymptotic run-times are same!
- IRL, bottom up a little faster
- generally, pick your fave :)

r store values of solutions

- solve r[0] and save
- use r[0] to solve r[1], save it
- use r[0] and/or r[1] to solve r[2], save it

↳ recursive equation
overlapping subproblems
are iterative, w/o actual recursion

* bottom up
compare in directionality

S_3 NOR S_4 NORT

2. LCS Problem

↳ Longest Common Subsequence sequence S = ⟨NORTHEASTERN⟩

- subsequence of S: S with 0 or more elements left out

Subsequences of S: NORTH, NTH, HT, EER

not subseqs of S: TRO, NOO, NNE

Given two sequences X, Y

$X = \langle x_1, x_2, \ldots, x_m \rangle$

$Y = \langle y_1, y_2, \ldots, y_n \rangle$

Want: common subsequence

- subsequence of X, Y
- optimal: <u>longest</u> one!

Brute force approach:

- find all subseqs of X → length m 2^m subseqs
- find all subseqs of Y → length n 2^n subseqs
- of the ones in common, keep the longest

Brute-force run time: exponential :(

ex finding the LCS by hand

	X = CBA, Y = LIBR	X = CBA, Y = LIBRARY	X = CBAWY, Y = LIBRARY
value:	1	2	3
actual:	B	BA	BAY

notice: soln to bigger problem contained in soln to subproblems

LIBR ⇒ LIBRARY
B ⇒ BA

CBA ⇒ CBAWY
BA ⇒ BAY

good candidate for DP! :)

Approach

1. Recursive equation
2. when computing value of soln...
   - did we solve it? look it up!
   - o/w, solve and save

Recursive Equation

- c table
- let X_i subsequence from x_1 to x_i
- let x_i be character at pos i
- let i be tied to X
- let j be tied to Y
- X_0, Y_0 are empty sequences

→ $c[i,j]$ length of $LCS(X_i, Y_j)$

- what is base case?
  what is $c[i,j]$ in base case?
- If $x_i == y_j$ then what's $c[i,j]$?
- otherwise, what's $c[i,j]$? → ok (normal to assume we have solutions to smaller problems)

Base case: X_0 (empty) $\quad c[i,j] = 0$
Y_0 (empty)

If $x_i == y_j$ they have a character in common!
LCS +1

otherwise LCS stays the same

need to know what it was before... already stored!

12:43

3. DP Bottom-Up

formalize recursive equation:

$$c[i,j] = \begin{cases} \emptyset & \text{if } i = \emptyset \text{ or } j = \emptyset \\ c[i-1, j-1] + 1 & \text{if } x_i == y_j \\ \max\{c[i-1,j],\ c[i,j-1]\} & \text{otherwise} \end{cases}$$

ex $X = A\ B\ C\ D \qquad m = 4$
$Y = A\ E\ B\ D\ H \qquad n = 5$

↳ LCS(X, Y)
- create c table, base case

for $i = 1$ to m
  for $j = 1$ to n
    if $x_i == y_j$
      $c[i,j] = c[i-1, j-1] + 1$
    else
      $c[i,j] = \max(c[i-1,j],\ c[i,j-1])$

x_i \ y_j		A	E	B	D	H
x_i	0	0	0	0	0	0
A	0	1	1	1	1	1
B	0	1	1	2	2	2
C	0	1	1	2	2	2
D	0	1	1	2	3	3

$i=1, j=1$ A vs. A
$i=1$ $j=2$ A vs. AE
$j=3$ A vs AEB
$j=4$ A vs. AEBD
$j=5$ A vs. AEBDH

where is the answer?
↳ bottom right! LCS = 3

$i=2$ $j=1$ AB vs A

$j=2$ AB vs. AE

$j=3$ AB vs. AEB

$j=4$ AB vs. AEBD

$j=5$ AB vs. AEBDH

Actual optimal solution!

↳ construct while computing value

leave clues to piece back together

LCS actual solution: B table

- B table tells us a path

← go left

↑ go up

} character not part of solution

↖ go up diagonal and print character!

	A	E	B	D	H
A	↖	←	←	←	←
B	↑	↑	↖	←	←
C	↑	↑	↑	↑	↑
D	↑	↑	↑	↖	←

A B D

* will it work with duplicate characters? APPS!