CS3000

5/17 - Weds

Admin

- Short HW2 due tom. 9pm
- Long HW2 out tom, due Tues.
- Exam #1 next thurs 5/25
- Fun optional recitation tomorrow

Agenda

1. Dynamic Programming (DP)
2. Longest common Subsequence (LCS)
3. DP LCS solution

1. Dynamic Programming

Approaches
- naive – linear search, selection sort
- divide and conquer - binary search, karatsuba, mergesort, quick sort
- DP - LCS

Start with D+C, and then... ⋮ can we do better? ⋮

When is DP a good fit?
- optimization problems
- optimal substructure
- overlapping subproblems

Optimization Problem
- easy to find a solution
- we want the best of many/all solutions
- optimize (max/min) of some component

ex

problem: need coffee
work in meserve
class in Shillman

poss. solutions:
- sbux in CSC
- Tatte
- Basement
- DD in Richards
- DD in Shillman
- sneak coffee maker into office

All valid!

But, what are we optimizing for?

- Time: Shillman — one optimal solns
- $; make my own — one optimal solns
- Taste: either DD — two optimal solns

In general, optimization problem can be solved naively (brute force)

- compute the value of every valid solution
- see which is best

↳ but, this is slowwwww :(

"Can we do better?"

↳ try D+C … might be a slow implementation :(

"Can we make a faster implementation"

DP is good when

- optimization
- optimal substructure: solution to smaller problem is part of solution to bigger problem
- overlapping subproblems: recursive solution solves the same subproblems multiple times

2. Longest Common Subsequence

LCS

sequence: like a string

ex: $X = \langle n, o, r, t, h, e, a, s, t, e, r, n \rangle$

$x_1 \qquad \qquad \qquad x_n$

subsequence of X: X with 0 or more elements removed

ex: subsequences of X

$\langle n \rangle \quad \langle n, n \rangle \quad \langle o, h, e \rangle \quad \langle e, a, e, r \rangle \quad \langle \rangle$

ex: invalid subsequences of X

$\langle r, o \rangle \quad \langle n, n, n \rangle$

How many subsequences of X are there? 2^n

ex: $X = \langle A, L, G, O \rangle$ $\qquad 2^4 = 16$

$\underline{2} \ \underline{2} \ \underline{2} \ \underline{2}$

LCS: given two sequences X, Y

$X = \langle x_1, x_2, \ldots, x_m \rangle \qquad Y = \langle y_1, y_2, \ldots, y_n \rangle$

An LCS(X, Y):

- is a subsequence of X
- is a subsequence of Y
- is an optimal solution
  - ↳ longest!

For today: LCS(X, Y) is the length

Approach #1: naive (Brute Force)

- Find all subsequences of X (2^m)
- Find all subsequences of Y (2^n)
- Find the ones they have in common
- choose the longest

Solves the problem? ☑ Runtime: $2^m + 2^n$!!??

!!!
Can we do better?
!!! ↳ try D+C

$\boxed{\text{Approach \#2: Divide + Conquer}}$

$X = \langle A, G, A, T \rangle$ $m = 4$
$Y = \langle G, X, B, T \rangle$ $n = 4$

- work our way backwards from x_m and y_n
- just focus on length

$\boxed{D+C}$ → start at end

A, G, A, $\boxed{T}$ → A, G, A
G, X, B $\boxed{T}$ → G, X, B

1 LCS(AGAT, GXBT) = LCS(AGA, GXB) + 1

$\boxed{\text{next}}$ LCS(AGA, GXB)

A, G, A
G, X, B

not the same !!?

2
LCS(AGA, GXB) = LCS(AG, GXB)
or 3
LCS (AGA, GX)

→ whichever is longer

repeated work!

$\boxed{\text{next}}$ LCS(AG, GXB)

A, G
G, X, B

LCS(AGA, GX) = LCS(AG, GX) or LCS(AGA, G)

4

LCS(AG, GXB) = LCS(AG, GX) or LCS(A, GXB) → whichever is longer

5

(more...)

10:46

- remove one element from X, Y
- remove one from X but not Y
- remove one from Y but not X

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

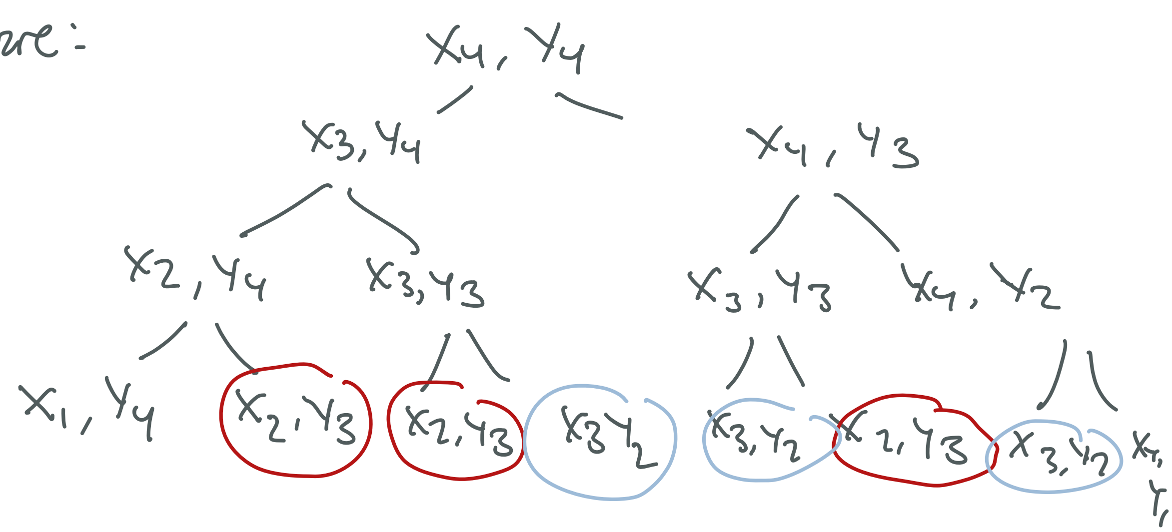
$Y = \langle y_1, y_2, \ldots y_n \rangle$

worst case: no elements in common

X = sequence $X_i = \langle x_1, x_2, \ldots, x_i \rangle$

Y = sequence $Y_j = \langle y_1, y_2, \ldots, y_j \rangle$

compare:

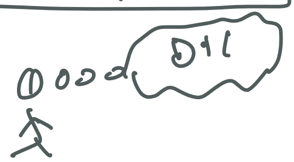


so much overlap!

eventually compare every $x_1, x_2, \ldots x_m$ to every $y_1, y_2, \ldots y_n$

$\to \Theta(2^m)$:(same as naive solution

3. DP approach



Implementation is better

- once we solve a subproblem, never solve it again
- instead, <u>remember</u> it!
- tradeoff of time / space

D+C
↳ DP

Recurrence for the value of an optimal solution

$c[i,j]$ = length of an LCS of X_i and Y_j

$$c[i,j] = \begin{cases} \text{Base case: } \emptyset \text{ when } i=0 \text{ or } j=0 \\ \text{if } x_i == y_j \quad c[i-1,j-1]+1 \\ \text{if } x_i \neq y_j \quad \max\{c[i-1,j],\ c[i,j-1]\} \end{cases}$$

↳ make a table/matrix
"C" table

X = <A, G, A, T>
Y = <G, X, B, T>

ex $C[0,0] = LCS("", "") = 0$

$C[1,1] = LCS(A, G) = 0$

$C[3,1] = LCS(AGA, G) = 1$

$C[1,3] = LCS(A, GXB) = 0$

$C[4,4] = LCS(AGAT, GXBT)$ = actual answer!

C table → length

	(Y) $j=0$	G $j=1$	X $j=2$	B $j=3$	T $j=4$
(X) $i=0$	0 BC	0 BC	0 BC	0 BC	0 BC
A $i=1$	0 BC	A, G 0	A, GX 0	A, GXB 0	A, GXBT 0
G $i=2$	0 BC	AG, G 1	AG, GX 1	AG, GXB 1	AG, GXBT 1
A $i=3$	0 BC	AGA, G 1	AGA, GX 1	AGA, GXB 1	AGA, GXBT 1
T $i=4$	0 BC	AGAT, G 1	AGAT, GX 1	AGAT, GXB 1	AGAT, GXBT 2

Final answer!

(answers to
subproblems)