We didn't finish the slides "L14" titled:

Dynamic Programming, Part Deux

- These are updated slides on the LCS problem

Today's Keywords

- Dynamic Programming
- Longest Common Subsequence
- Bottom-up vs. Top-down solutions

CLRS Readings

- Chapter 15
  - **Section 15.4, longest common subsequence**

Reminders: Dynamic Programming

- Requires Optimal Substructure
  – Solution to larger problem contains the solutions to smaller ones
- Avoid extra work due to overlapping subproblems
- Idea:
  1. Identify the recursive structure of the problem
     - What is the “last thing” done?
  2. Save the solution to each subproblem in memory
  3. Select a good order for solving subproblems
     - “Top Down”: Solve each recursively
     - “Bottom Up”: Iteratively solve smallest to largest

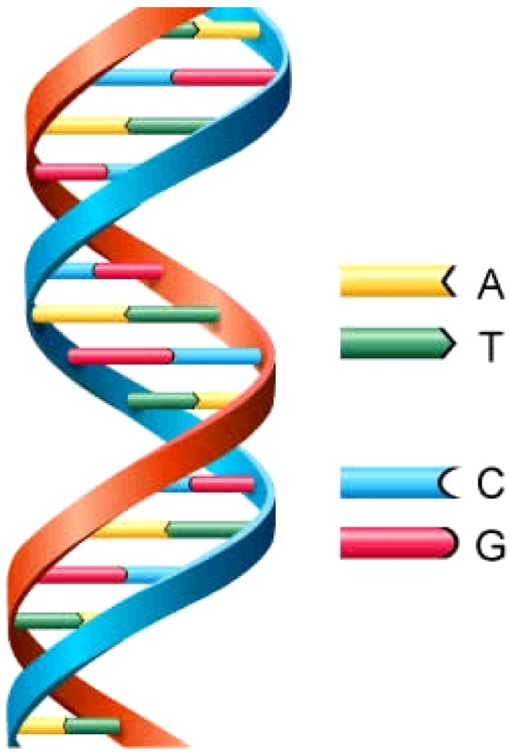
Longest Common Subsequence

Given two sequences X and Y, find the length of their longest common subsequence

Example:
X = ATCTGAT
Y = TGCATA
LCS = TCTA

X = AT C TGAT
Y = TGCAT A

Brute force: Compare every subsequence of X with Y: $\Omega(2^n)$



Dynamic Programming

- Requires Optimal Substructure
  - Solution to larger problem contains the solutions to smaller ones
- Avoid extra work due to overlapping subproblems
- Idea:
  1. Identify the recursive structure of the problem
     - What is the "last thing" done?
  2. Save the solution to each subproblem in memory
  3. Select a good order for solving subproblems
     - "Top Down": Solve each recursively
     - "Bottom Up": Iteratively solve smallest to largest

1. Identify Recursive Structure

Let $LCS(i, j)$ = length of the LCS for the first i characters of X, first j character of Y

Find $LCS(i, j)$:

Case 1: $X[i] = Y[j]$

X = $ATCTGCGT$

Y = $TGCATAT$

$LCS(i, j) = LCS(i - 1, j - 1) + 1$

Case 2: $X[i] \neq Y[j]$

X = $ATCTGCGA$

Y = $TGCATAC$

$LCS(i, j) = LCS(i, j - 1)$

X = $ATCTGCGA$

Y = $TGCATAC$

$LCS(i, j) = LCS(i - 1, j)$

$$LCS(i, j) = \begin{cases} 0 & \text{if } i = 0 \text{ or } j = 0 \\ LCS(i - 1, j - 1) + 1 & \text{if } X[i] = Y[j] \\ \max(LCS(i, j - 1), LCS(i - 1, j)) & \text{otherwise} \end{cases}$$

Dynamic Programming

- Requires Optimal Substructure
  - Solution to larger problem contains the solutions to smaller ones
- Avoid extra work due to overlapping subproblems
- Idea:
  1. Identify the recursive structure of the problem
     - What is the "last thing" done?
  2. Save the solution to each subproblem in memory
  3. Select a good order for solving subproblems
     - "Top Down": Solve each recursively
     - "Bottom Up": Iteratively solve smallest to largest

1. Identify Recursive Structure

Let $LCS(i, j)$ = length of the LCS for the first i characters of X, first j character of Y

Find $LCS(i, j)$:

Case 1: $X[i] = Y[j]$

X = $ATCTGCGT$
Y = $TGCATAT$
$LCS(i, j) = LCS(i - 1, j - 1) + 1$

Case 2: $X[i] \neq Y[j]$

X = $ATCTGCGA$
Y = $TGCATAC$
$LCS(i, j) = LCS(i, j - 1)$

X = $ATCTGCGA$
Y = $TGCATAC$
$LCS(i, j) = LCS(i - 1, j)$

$$LCS(i, j) = \begin{cases} 0 & \text{if } i = 0 \text{ or } j = 0 \\ LCS(i-1, j-1) + 1 & \text{if } X[i] = Y[j] \\ \max(LCS(i, j-1), LCS(i-1, j)) & \text{otherwise} \end{cases}$$

Read from M[i,j] if present

Save to M[i,j]

Dynamic Programming

- Requires Optimal Substructure
  - Solution to larger problem contains the solutions to smaller ones
- Avoid extra work due to overlapping subproblems
- Idea:
  1. Identify the recursive structure of the problem
     - What is the "last thing" done?
  2. Save the solution to each subproblem in memory
  3. Select a good order for solving subproblems
     - "Top Down": Solve each recursively
     - "Bottom Up": Iteratively solve smallest to largest

3. Solve in a Good Order

$$LCS(i,j) = \begin{cases} 0 & \text{if } i = 0 \text{ or } j = 0 \\ LCS(i-1, j-1) + 1 & \text{if } X[i] = Y[j] \\ \max(LCS(i, j-1), LCS(i-1, j)) & \text{otherwise} \end{cases}$$

$Y =$	$X =$	0	A 1	T 2	C 3	T 4	G 5	A 6	T 7
	0	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
T	1	**0**	**0**	**1**	**1**	**1**	**1**	**1**	**1**
G	2	**0**	**0**	**1**	**1**	**1**	**2**	**2**	**2**
C	3	**0**	**0**	**1**	**2**	**2**	**2**	**2**	**2**
A	4	**0**	**1**	**1**	**2**	**2**	**2**	**3**	**3**
T	5	**0**	**1**	**2**	**2**	**3**	**3**	**3**	**4**
A	6	**0**	**1**	**2**	**2**	**3**	**3**	**4**	**4**

To fill in cell (i,j) we need cells $(i-1, j-1), (i-1, j), (i, j-1)$

Fill from Top->Bottom, Left->Right (with any preference)

LCS Length Algorithm

LCS-Length(X, Y) // Y for M's rows, X for its columns

1. n = length(X) // get the # of symbols in X
2. m = length(Y) // get the # of symbols in Y
3. for i = 1 to m M[i,0] = 0 // special case: Y_0
4. for j = 1 to n M[0,j] = 0 // special case: X_0
5. for i = 1 to m // for all Y_i
6. for j = 1 to n // for all X_j
7. if (X[i] == Y[j])
8. M[i,j] = M[i-1,j-1] + 1
9. else M[i,j] = max(M[i-1,j], M[i,j-1])
10. return M[m,n] // return LCS length for Y and X

Run Time?

$$LCS(i,j) = \begin{cases} 0 & \text{if } i = 0 \text{ or } j = 0 \\ LCS(i-1, j-1) + 1 & \text{if } X[i] = Y[j] \\ \max(LCS(i, j-1), LCS(i-1, j)) & \text{otherwise} \end{cases}$$

$X =$, $Y =$

		0	A 1	T 2	C 3	T 4	G 5	A 6	T 7
	0	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
T	1	**0**	**0**	**1**	**1**	**1**	**1**	**1**	**1**
G	2	**0**	**0**	**1**	**1**	**1**	**2**	**2**	**2**
C	3	**0**	**0**	**1**	**2**	**2**	**2**	**2**	**2**
A	4	**0**	**1**	**1**	**2**	**2**	**2**	**3**	**3**
T	5	**0**	**1**	**2**	**2**	**3**	**3**	**3**	**4**
A	6	**0**	**1**	**2**	**2**	**3**	**3**	**4**	**4**

Run Time: $\Theta(n \cdot m)$ (for $|X| = n$, $|Y| = m$)

Reconstructing the LCS

$$LCS(i,j) = \begin{cases} 0 & \text{if } i = 0 \text{ or } j = 0 \\ LCS(i-1, j-1) + 1 & \text{if } X[i] = Y[j] \\ \max(LCS(i, j-1), LCS(i-1, j)) & \text{otherwise} \end{cases}$$

$Y =$ \ $X =$		0	A 1	T 2	C 3	$\boxed{T}$ 4	$\boxed{G}$ 5	$\boxed{A}$ 6	$\boxed{T}$ 7
	0	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
$\boxed{T}$	1	**0**	0	1	1	1	1	1	1
$\boxed{G}$	2	**0**	0	1	1	1	2	2	2
C	3	**0**	0	1	2	2	2	2	2
$\boxed{A}$	4	**0**	1	1	2	2	2	3	3
$\boxed{T}$	5	**0**	1	2	2	3	3	3	4
A	6	**0**	1	2	2	3	3	4	4

Start from bottom right,

if symbols matched, print that symbol then go diagonally

else go to largest adjacent

Reconstructing the LCS

$$LCS(i,j) = \begin{cases} 0 & \text{if } i = 0 \text{ or } j = 0 \\ LCS(i-1, j-1) + 1 & \text{if } X[i] = Y[j] \\ \max(LCS(i, j-1), LCS(i-1, j)) & \text{otherwise} \end{cases}$$

$Y =$ \ $X =$		0	A 1	$\boxed{T}$ 2	$\boxed{C}$ 3	T 4	G 5	$\boxed{A}$ 6	$\boxed{T}$ 7
	0	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
$\boxed{T}$	1	**0**	**0**	**1**	**1**	**1**	**1**	**1**	**1**
G	2	**0**	**0**	**1**	**1**	**1**	**2**	**2**	**2**
$\boxed{C}$	3	**0**	**0**	**1**	**2**	**2**	**2**	**2**	**2**
$\boxed{A}$	4	**0**	**1**	**1**	**2**	**2**	**2**	**3**	**3**
$\boxed{T}$	5	**0**	**1**	**2**	**2**	**3**	**3**	**3**	**4**
A	6	**0**	**1**	**2**	**2**	**3**	**3**	**4**	**4**

Start from bottom right,
if symbols matched, print that symbol then go diagonally
else go to largest adjacent

Reconstructing the LCS

$$LCS(i,j) = \begin{cases} 0 & \text{if } i = 0 \text{ or } j = 0 \\ LCS(i-1,j-1)+1 & \text{if } X[i] = Y[j] \\ \max(LCS(i,j-1), LCS(i-1,j)) & \text{otherwise} \end{cases}$$

$Y =$ \ $X =$		0	A 1	$\boxed{T}$ 2	$\boxed{C}$ 3	$\boxed{T}$ 4	G 5	$\boxed{A}$ 6	T 7
	0	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
$\boxed{T}$	1	**0**	0	1	1	1	1	1	1
G	2	**0**	0	1	1	1	2	2	2
$\boxed{C}$	3	**0**	0	1	2	2	2	2	2
A	4	**0**	1	1	2	2	2	3	3
$\boxed{T}$	5	**0**	1	2	2	3	3	3	4
$\boxed{A}$	6	**0**	1	2	2	3	3	4	4

Start from bottom right,

if symbols matched, print that symbol then go diagonally

else go to largest adjacent

Top-Down Solution with Memoization

We need two functions; one will be recursive.

LCS-Length(X, Y) // Y is M's cols.
1. n = length(X)
2. m = length(Y)
3. Create table M[m,n]
4. Assign -1 to all cells M[i,j]
// get value for entire sequences
5. return **LCS-recur**(X, Y, M, m, n)

LCS-recur(X, Y, M, i, j)
1. if (i == 0 || j == 0) return 0
// have we already calculated this subproblem?
2. if (M[i,j] != -1) return M[i,j]
3. if (X[i] == Y[j])
4.   M[i,j] = **LCS-recur**(X, Y, M, i-1, j-1) + 1
5. else
6.   M[i,j] = max(**LCS-recur**(X, Y, M, i-1, j),
            LCS-recur(X, Y, M, i, j-1))
7. return M[i,j]

Another LCS Example

Let's see how LCS algorithm works on the following example:

- X = ABCB
- Y = BDCAB

What is the Longest Common Subsequence of X and Y?

LCS(X, Y) = BCB

X = A **B** **C** **B**

Y = **B** D **C** A **B**

LCS Example (0)

ABCB
BDCAB

i		j	0	1	2	3	4	5
			Yj	**B**	**D**	**C**	**A**	**B**
0	Xi							
1	**A**							
2	**B**							
3	**C**							
4	**B**							

X = ABCB; m = |X| = 4
Y = BDCAB; n = |Y| = 5
Allocate array M[5,4]

Note: In this example, X is M's rows, Y is the columns. Opposite from earlier example.

LCS Example (1)

ABCB
BDCAB

i		j	0	1	2	3	4	5
			Yj	**B**	**D**	**C**	**A**	**B**
0	Xi		**0**	**0**	**0**	**0**	**0**	**0**
1	**A**		**0**					
2	**B**		**0**					
3	**C**		**0**					
4	**B**		**0**					

for i = 1 to m   M[i,0] = 0
for j = 1 to n   M[0,j] = 0

LCS Example (2)

ABCB
BDCAB

i		j	0	1	2	3	4	5
			Yj	**B**	**D**	**C**	**A**	**B**
0	Xi		**0**	**0**	**0**	**0**	**0**	**0**
1	**A**		**0**	**0**				
2	**B**		**0**					
3	**C**		**0**					
4	**B**		**0**					

if (X[i] == Y[j])
M[i,j] = M[i-1,j-1] + 1
else M[i,j] = max(M[i-1,j], M[i,j-1])

LCS Example (3)

ABCB

BDCAB

	j	0	1	2	3	4	5
i		Yj	**B**	**D**	**C**	**A**	**B**
0	Xi	**0**	**0**	**0**	**0**	**0**	**0**
1	**A**	**0**	**0**	**0**	**0**		
2	**B**	**0**					
3	**C**	**0**					
4	**B**	**0**					

if (X[i] == Y[j])

M[i,j] = M[i-1,j-1] + 1

else M[i,j] = max(M[i-1,j], M[i,j-1])

LCS Example (4)

ABCB
BDCAB

i		j	0	1	2	3	4	5
			Yj	**B**	**D**	**C**	**A**	**B**
0	Xi		**0**	**0**	**0**	**0**	**0**	**0**
1	**A**		**0**	**0**	**0**	**0**	**1**	
2	**B**		**0**					
3	**C**		**0**					
4	**B**		**0**					

if (X[i] == Y[j])
M[i,j] = M[i-1,j-1] + 1
else M[i,j] = max(M[i-1,j], M[i,j-1])

LCS Example (5)

ABCB
BDCAB

	j	0	1	2	3	4	5
i		Yj	**B**	**D**	**C**	**A**	**B**
0	Xi	**0**	**0**	**0**	**0**	**0**	**0**
1	**A**	**0**	**0**	**0**	**0**	**1** →	**1**
2	**B**	**0**					
3	**C**	**0**					
4	**B**	**0**					

if (X[i] == Y[j])
M[i,j] = M[i-1,j-1] + 1
else M[i,j] = max(M[i-1,j], M[i,j-1])

LCS Example (6)

ABCB
BDCAB

i		j	0	1	2	3	4	5
			Yj	B	D	C	A	B
0	Xi		0	0	0	0	0	0
1	A		0	0	0	0	1	1
2	B		0	1				
3	C		0					
4	B		0					

if (X[i] == Y[j])
M[i,j] = M[i-1,j-1] + 1
else M[i,j] = max(M[i-1,j], M[i,j-1])

LCS Example (7)

ABCB
BDCAB

		j	0	1	2	3	4	5
i			Yj	**B**	**D**	**C**	**A**	**B**
0	Xi		**0**	**0**	**0**	**0**	**0**	**0**
1	**A**		**0**	**0**	**0**	**0**	**1**	**1**
2	**B**		**0**	**1**	**1**	**1**	**1**	
3	**C**		**0**					
4	**B**		**0**					

if (X[i] == Y[j])
M[i,j] = M[i-1,j-1] + 1
else M[i,j] = max(M[i-1,j], M[i,j-1])

LCS Example (8)

ABCB
BDCAB

i	j	0	1	2	3	4	5
		Yj	**B**	**D**	**C**	**A**	**B**
0	Xi	**0**	**0**	**0**	**0**	**0**	**0**
1	**A**	**0**	**0**	**0**	**0**	**1**	**1**
2	**B**	**0**	**1**	**1**	**1**	**1**	**2**
3	**C**	**0**					
4	**B**	**0**					

if (X[i] == Y[j])
    M[i,j] = M[i-1,j-1] + 1
else M[i,j] = max(M[i-1,j], M[i,j-1])

LCS Example (10)

ABCB
BDCAB

	j	0	1	2	3	4	5
i		Yj	**B**	**D**	**C**	**A**	**B**
0	Xi	**0**	**0**	**0**	**0**	**0**	**0**
1	**A**	**0**	**0**	**0**	**0**	**1**	**1**
2	**B**	**0**	**1**	**1**	**1**	**1**	**2**
3	**C**	**0**	**1**	**1**			
4	**B**	**0**					

if (X[i] == Y[j])
M[i,j] = M[i-1,j-1] + 1
else M[i,j] = max(M[i-1,j], M[i,j-1])

LCS Example (11)

ABCB
BDCAB

	j	0	1	2	3	4	5
i		Yj	**B**	**D**	**C**	**A**	**B**
0	Xi	**0**	**0**	**0**	**0**	**0**	**0**
1	**A**	**0**	**0**	**0**	**0**	**1**	**1**
2	**B**	**0**	**1**	**1**	**1**	**1**	**2**
3	**C**	**0**	**1**	**1**	**2**		
4	**B**	**0**					

if (X[i] == Y[j])
M[i,j] = M[i-1,j-1] + 1
else M[i,j] = max(M[i-1,j], M[i,j-1])

LCS Example (12)

ABCB
BDCAB

i		j	0	1	2	3	4	5
			Yj	**B**	**D**	**C**	**A**	**B**
0	Xi		**0**	**0**	**0**	**0**	**0**	**0**
1	**A**		**0**	**0**	**0**	**0**	**1**	**1**
2	**B**		**0**	**1**	**1**	**1**	**1**	**2**
3	**C**		**0**	**1**	**1**	**2**	**2**	**2**
4	**B**		**0**					

if (X[i] == Y[j])
M[i,j] = M[i-1,j-1] + 1
else M[i,j] = max(M[i-1,j], M[i,j-1])

LCS Example (13)

ABCB
BDCAB

i		j	0	1	2	3	4	5
			Yj	**B**	**D**	**C**	**A**	**B**
0	Xi		**0**	**0**	**0**	**0**	**0**	**0**
1	**A**		**0**	**0**	**0**	**0**	**1**	**1**
2	**B**		**0**	**1**	**1**	**1**	**1**	**2**
3	**C**		**0**	**1**	**1**	**2**	**2**	**2**
4	**B**		**0**	**1**				

if (X[i] == Y[j])
M[i,j] = M[i-1,j-1] + 1
else M[i,j] = max(M[i-1,j], M[i,j-1])

LCS Example (14)

ABCB

BDCAB

i		j	0	1	2	3	4	5
			Yj	B	D	C	A	B
0	Xi		0	0	0	0	0	0
1	A		0	0	0	0	1	1
2	B		0	1	1	1	1	2
3	C		0	1	1	2	2	2
4	B		0	1	1	2	2	

if (X[i] == Y[j])
M[i,j] = M[i-1,j-1] + 1
else M[i,j] = max(M[i-1,j], M[i,j-1])

LCS Example (15)

ABCB
BDCAB

i		j	0	1	2	3	4	5
			Yj	**B**	**D**	**C**	**A**	**B**
0	Xi		**0**	**0**	**0**	**0**	**0**	**0**
1	**A**		**0**	**0**	**0**	**0**	**1**	**1**
2	**B**		**0**	**1**	**1**	**1**	**1**	**2**
3	**C**		**0**	**1**	**1**	**2**	**2**	**2**
4	**B**		**0**	**1**	**1**	**2**	**2**	**3**

if (X[i] == Y[j])
M[i,j] = M[i-1,j-1] + 1
else M[i,j] = max(M[i-1,j], M[i,j-1])

Practice!

- X = [G, D, V, E, G, T, A] and
  Y = [G, V, C, E, K, S, T]
- Find the LCS, show the table M
- Can you reconstruct the LCS from M?