Design and Analysis of Algorithm

Dynamic Programming
(Longest Common Subsequence)

Lecture – 57

Overview

- Dynamic Programming is a general algorithm design technique for solving problems defined by recurrences with overlapping subproblems
- Invented by American mathematician “Richard Bellman) in the year 1950s to solve optimization problems and later assimilated by Computer Science.
- “Programming” here means “planning”

Dynamic Programming

- "Method of solving complex problems by breaking them down into smaller sub-problems, solving each of those sub-problems just once, and storing their solutions."
- The problem solving approach looks like Divide and conquer approach.(which is not true)

Dynamic Programming

Difference between Dynamic programming and Divide and Conquer approach.

Divide & Conquer	Dynamic Programming
1. Partitions a problem into independent smaller sub-problems	1. Partitions a problem into overlapping sub-problems
2. Doesn't store solutions of sub-problems. (Identical sub-problems may arise - results in the same computations are performed repeatedly.)	2. Stores solutions of sub-problems: thus avoids calculations of same quantity twice
3. Top down algorithms: which logically progresses from the initial instance down to the smallest sub-instances via intermediate sub-instances.	3. Bottom up algorithms: in which the smallest sub-problems are explicitly solved first and the results of these used to construct solutions to progressively larger sub-instances

Dynamic Programming

Is a Four-step methods

1. Characterize the structure of an optimal solution.
2. Recursively define the value of an optimal solution.
3. Compute the value of an optimal solution, typically in a bottom-up fashion.
4. Construct an optimal solution from computed information.

Dynamic Programming

Problems:

1. 0/1 Knapsack Problem
2. Floyd-Warshall Algorithm
3. Longest Common Sub-sequence
4. Matrix Chain Multiplication

Dynamic Programming

Problem 3: Longest Common Subsequences (LCS)

Problem:

"Given two sequences $X = \langle x_1, x_2, \dots\dots, x_m \rangle$ and $Y = \langle y_1, y_2, \dots\dots, y_n \rangle$. Find a subsequence common to both whose length is longest. A subsequence doesn't have to be consecutive, but it has to be in order."

Dynamic Programming

Problem 3: Longest Common Subsequences (LCS)

Example:

Springtime

pioneer

$LCS : pine$

horseback

snowflake

$LCS : oak$

maelestrom

becalm

$LCS : elm$

heroically

scholarly

$LCS : holl$

Dynamic Programming

Problem 3: Longest Common Subsequence

- It is used, when the solution can be recursively described in terms of solutions to subproblems (optimal substructure)
- Algorithm finds solutions to subproblems and stores them in memory for later use
- More efficient than "brute-force methods", which solve the same subproblems over and over again

Dynamic Programming

Problem 3: Longest Common Subsequence

- Application: comparison of two DNA strings
- Example: X= {A B C B D A B }, Y= {B D C A B A}

  Longest Common Subsequence:

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

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

- Brute force algorithm would compare each subsequence of X with the symbols in Y

Dynamic Programming

Problem 3: Longest Common Subsequence

- if |X| = m, |Y| = n, then there are 2^m subsequences of x; we must compare each with Y (n comparisons)
- So the running time of the brute-force algorithm is $O(n\ 2^m)$
- Notice that the LCS problem has *optimal substructure*: solutions of subproblems are parts of the final solution.
- Subproblems: "find LCS of pairs of *prefixes* of X and Y"

Dynamic Programming

Problem 3: Longest Common Subsequence

Step 1: Characterize the structure of an optimal solution

- Define X_i, Y_j to be the prefixes of X and Y of length i and j respectively
- Define $c[i, j]$ to be the length of LCS of X_i and Y_j
- Then the length of LCS of X and Y will be $c[m, n]$.
- We start with $i = j = 0$ (i.e empty substrings of x and y)
- Since X_0 and Y_0 are empty strings, their LCS is always empty (i.e. $c[0,0] = 0$)
- LCS of empty string and any other string is empty, so for every i and j: c[0, j] = c[i,0] = 0

Dynamic Programming

Problem 3: Longest Common Subsequence

Step 1: Characterize the structure of an optimal solution

- In the process of calculation of $c[i,j]$, there are two cases:
- $First\ case$: $x[i] = y[j]$: one more symbol in strings X and Y matches, so the length of LCS X_i and Y_j equals to the length of LCS of smaller strings X_{i-1} and Y_{j-1}, plus 1.
- $Second\ case$: $x[i]\ ! =\ y[j]$: As symbols don't match, our solution is not improved, and the length of $LCS(X_i, Y_j)$ is the same as before (i.e. maximum of $LCS(X_i, Y_{j-1})$ and $LCS(X_{i-1}, Y_j)$)

Dynamic Programming

Problem 3: Longest Common Subsequence

Step 2: Recursively define the value of optimal solution.

- Define $c[i,j]$ to be the length of LCS of X_i and Y_j. Then the length of LCS of X and Y will be calculated as $c[m,n]$.

$$c[i,j] = \begin{cases} 0 & , \quad if\ i = 0\ or\ j = 0 \\ c[i-1, j-1] + 1 & , \quad if\ i,j > 0\ and\ X_i = Y_j \\ \max(c[i-1,j], c[i,j-1]) & , \quad if\ i,j > 0\ and\ X_i \neq Y_j \end{cases}$$

Dynamic Programming

Problem 3: Longest Common Subsequence

Step 3: Compute optimal solution cost.

LCS-Length(X, Y)

1. $m = length(X)$ // get the # of symbols in X
2. $n = length(Y)$ // get the # of symbols in Y
3. $for\ i = 1\ to\ m$

   $c[i, 0] = 0$ // special case: Y_0
4. $for\ j = 1\ to\ n$

   $c[0, j] = 0$ // special case: X_0
5. $for\ i = 1\ to\ m$ // for all X_i
6. $for\ j = 1\ to\ n$ // for all Y_j
7. $if\ (X_i == Y_j)$
8. $c[i, j] = c[i-1, j-1] + 1\ and\ b[i, j] = \text{“} \nwarrow \text{"}$
9. $else\ c[i, j] = \max(c[i-1, j], c[i, j-1])\ and$

   $b[i, j] = \text{“} \uparrow \text{“}\ (if\ \max is\ c[i-1, j])$

   $b[i, j] = \text{“} \leftarrow \text{“}\ (if\ \max is\ c[i, j-1])$
10. $return\ c$ and b

Dynamic Programming

Problem 3: Longest Common Subsequence

Step 3: Compute optimal solution cost.

Example 1: What do ABCB and BDCAB have in common?

$X = A\ B\ C\ B$

$Y = B\ D\ C\ A\ B$

What is the Longest Common Subsequence $LCS\ (X, Y)$?

$X = A\ \boldsymbol{B}\ \boldsymbol{C}\ \boldsymbol{B}$

$Y = \boldsymbol{B}\ D\ \boldsymbol{C}\ A\ \boldsymbol{B}$

Hence,

$LCS\ (X, Y) = BCB$

Note: The demonstration of this problem is given in the next page.

Dynamic Programming

Problem 3: Longest Common Subsequence

Step 3: Compute optimal solution cost.

Example 1: What do ABCB and BDCAB have in common?

$X = A\ B\ C\ B$

$Y = B\ D\ C\ A\ B$

	j	0	1	2	3	4	5
i		Y_j	B	D	C	A	B
0	X_I						
1	A						
2	B						
3	C						
4	B						

X = ABCB; m = |X| = 4
Y = BDCAB; n = |Y| = 5
Allocate array c[5,6]

Dynamic Programming

Problem 3: Longest Common Subsequence

Step 3: Compute optimal solution cost.

Example 1: What do ABCB and BDCAB have in common?

$X = A\ B\ C\ B$
$Y = B\ D\ C\ A\ B$

	j	0	1	2	3	4	5
i		Y_j	B	D	C	A	B
0	X_I	0	0	0	0	0	0
1	A	0					
2	B	0					
3	C	0					
4	B	0					

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

Dynamic Programming

Problem 3: Longest Common Subsequence

Step 3: Compute optimal solution cost.

Example 1: What do ABCB and BDCAB have in common?

$X = A\ B\ C\ B$

$Y = B\ D\ C\ A\ B$

	j	0	1	2	3	4	5
i		Y_j	B	D	C	A	B
0	X_I	0	0	0	0	0	0
1	A	0	0				
2	B	0					
3	C	0					
4	B	0					

$if\ (X_i == Y_j)$

$c[i,j] = c[i-1, j-1] + 1\ and\ b[i,j] = \text{“} \searrow \text{"}$

$else\ c[i,j] = \max(c[i-1,j], c[i,j-1])\ \ and$

$b[i,j] = \text{“} \downarrow \text{“}\ (if \max is\ c[i-1,j])$

$b[i,j] = \text{“} \rightarrow \text{“}\ (if \max is\ c[i,j-1])$

Dynamic Programming

Problem 3: Longest Common Subsequence

Step 3: Compute optimal solution cost.

Example 1: What do ABCB and BDCAB have in common?

$X = A\ B\ C\ B$

$Y = B\ D\ C\ A\ B$

	j	0	1	2	3	4	5
i		Y_j	B	D	C	A	B
0	X_I	0	0 ↓	0 ↓	0	0	0
1	A	0	0	0			
2	B	0					
3	C	0					
4	B	0					

$if\ (X_i == Y_j)$

$c[i,j] = c[i-1, j-1] + 1\ and\ b[i,j] = \text{“} \searrow \text{"}$

$else\ c[i,j] = \max(c[i-1,j], c[i,j-1])\ \ and$

$b[i,j] = \text{“} \downarrow \text{“}\ (if \max is\ c[i-1,j])$

$b[i,j] = \text{“} \rightarrow \text{“}\ (if \max is\ c[i,j-1])$

Dynamic Programming

Problem 3: Longest Common Subsequence

Step 3: Compute optimal solution cost.

Example 1: What do ABCB and BDCAB have in common?

	j	0	1	2	3	4	5
i		Y_j	B	D	C	A	B
0	X_I	0	0	0	0	0	0
1	A	0	0 ↓	0 ↓	0 ↓		
2	B	0					
3	C	0					
4	B	0					

$X = A\ B\ C\ B$

$Y = B\ D\ C\ A\ B$

$if\ (X_i == Y_j)$

$c[i,j] = c[i-1, j-1] + 1\ and\ b[i,j] = \text{“} \searrow \text{"}$

$else\ c[i,j] = \max(c[i-1,j], c[i,j-1])\ \ and$

$b[i,j] = \text{“} \downarrow \text{“}\ (if \max is\ c[i-1,j])$

$b[i,j] = \text{“} \rightarrow \text{“}\ (if \max is\ c[i,j-1])$

Dynamic Programming

Problem 3: Longest Common Subsequence

Step 3: Compute optimal solution cost.

Example 1: What do ABCB and BDCAB have in common?

	j	0	1	2	3	4	5
i		Y_j	B	D	C	A	B
0	X_I	0	0	0	0	0	0
1	A	0	0 ↓	0 ↓	0 ↓	1 ↘	
2	B	0					
3	C	0					
4	B	0					

$X = A\ B\ C\ B$

$Y = B\ D\ C\ A\ B$

$if\ (X_i == Y_j)$

$c[i,j] = c[i-1,j-1] + 1\ and\ b[i,j] = \text{“} \searrow \text{"}$

$else\ c[i,j] = \max(c[i-1,j], c[i,j-1])\ \ and$

$b[i,j] = \text{“} \downarrow \text{“}\ (if \max is\ c[i-1,j])$

$b[i,j] = \text{“} \rightarrow \text{“}\ (if \max is\ c[i,j-1])$

Dynamic Programming

Problem 3: Longest Common Subsequence

Step 3: Compute optimal solution cost.

Example 1: What do ABCB and BDCAB have in common?

$X = A\ B\ C\ B$

$Y = B\ D\ C\ A\ B$

	j	0	1	2	3	4	5
i		Y_j	B	D	C	A	B
0	X_I	0	0	0	0	0	0
1	A	0	↓ 0	↓ 0	↓ 0	↘ 1	→ 1
2	B	0					
3	C	0					
4	B	0					

$if\ (X_i == Y_j)$

$c[i,j] = c[i-1, j-1] + 1\ and\ b[i,j] = \text{“}\searrow\text{"}$

$else\ c[i,j] = \max(c[i-1,j], c[i,j-1])\ \ and$

$b[i,j] = \text{“}\downarrow\text{“}\ (if\ \max is\ c[i-1,j])$

$b[i,j] = \text{“}\rightarrow\text{“}\ (if\ \max is\ c[i,j-1])$

Dynamic Programming

Problem 3: Longest Common Subsequence

Step 3: Compute optimal solution cost.

Example 1: What do ABCB and BDCAB have in common?

$X = A\ B\ C\ B$

$Y = B\ D\ C\ A\ B$

	j	0	1	2	3	4	5
i		Y_j	B	D	C	A	B
0	X_I	0	0	0	0	0	0
1	A	0	0 ↓	0 ↓	0 ↓	1 ↘	1 →
2	B	0	1 ↘				
3	C	0					
4	B	0					

$if\ (X_i == Y_j)$

$c[i,j] = c[i-1,j-1] + 1\ and\ b[i,j] = \text{“} \searrow \text{"}$

$else\ c[i,j] = \max(c[i-1,j], c[i,j-1])\ \ and$

$b[i,j] = \text{“} \downarrow \text{“}\ (if \max is\ c[i-1,j])$

$b[i,j] = \text{“} \rightarrow \text{“}\ (if \max is\ c[i,j-1])$

Dynamic Programming

Problem 3: Longest Common Subsequence

Step 3: Compute optimal solution cost.

Example 1: What do ABCB and BDCAB have in common?

$X = A\ B\ C\ B$

$Y = B\ D\ C\ A\ B$

	j	0	1	2	3	4	5
i		Y_j	B	D	C	A	B
0	X_I	0	0	0	0	0	0
1	A	0	0 ↓	0 ↓	0 ↓	1 ↘	1 →
2	B	0	1 ↘	1 →			
3	C	0					
4	B	0					

$if\ (X_i == Y_j)$

$c[i,j] = c[i-1,j-1] + 1\ and\ b[i,j] = \text{“} \searrow \text{"}$

$else\ c[i,j] = \max(c[i-1,j], c[i,j-1])\ \ and$

$b[i,j] = \text{“} \downarrow \text{“}\ (if \max is\ c[i-1,j])$

$b[i,j] = \text{“} \rightarrow \text{“}\ (if \max is\ c[i,j-1])$

Dynamic Programming

Problem 3: Longest Common Subsequence

Step 3: Compute optimal solution cost.

Example 1: What do ABCB and BDCAB have in common?

$X = A\ B\ C\ B$

$Y = B\ D\ C\ A\ B$

	j	0	1	2	3	4	5
i		Y_j	B	D	C	A	B
0	X_I	0	0	0	0	0	0
1	A	0	↓ 0	↓ 0	↓ 0	↘ 1	→ 1
2	B	0	↘ 1	→ 1	→ 1		
3	C	0					
4	B	0					

$if\ (X_i == Y_j)$

$c[i,j] = c[i-1, j-1] + 1\ and\ b[i,j] = \text{“}\searrow\text{"}$

$else\ c[i,j] = \max(c[i-1,j], c[i,j-1])\ \ and$

$b[i,j] = \text{“}\downarrow\text{“}\ (if \max is\ c[i-1,j])$

$b[i,j] = \text{“}\rightarrow\text{“}\ (if \max is\ c[i,j-1])$

Dynamic Programming

Problem 3: Longest Common Subsequence

Step 3: Compute optimal solution cost.

Example 1: What do ABCB and BDCAB have in common?

	j	0	1	2	3	4	5
i		Y_j	B	D	C	A	B
0	X_I	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					

$X = A\ B\ C\ B$

$Y = B\ D\ C\ A\ B$

$if\ (X_i == Y_j)$

$c[i,j] = c[i-1, j-1] + 1\ and\ b[i,j] = \text{“} \searrow \text{"}$

$else\ c[i,j] = \max(c[i-1,j], c[i,j-1])\ \ and$

$b[i,j] = \text{“} \downarrow \text{“}\ (if \max is\ c[i-1,j])$

$b[i,j] = \text{“} \rightarrow \text{“}\ (if \max is\ c[i,j-1])$

Dynamic Programming

Problem 3: Longest Common Subsequence

Step 3: Compute optimal solution cost.

Example 1: What do ABCB and BDCAB have in common?

$X = A\ B\ C\ B$

$Y = B\ D\ C\ A\ B$

	j	0	1	2	3	4	5
i		Y_j	B	D	C	A	B
0	X_I	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					

$if\ (X_i == Y_j)$

$c[i,j] = c[i-1, j-1] + 1\ and\ b[i,j] = \text{"}\searrow\text{"}$

$else\ c[i,j] = \max(c[i-1,j], c[i,j-1])\ \ and$

$b[i,j] = \text{"}\downarrow\text{"}\ (if\ \max is\ c[i-1,j])$

$b[i,j] = \text{"}\rightarrow\text{"}\ (if\ \max is\ c[i,j-1])$

Dynamic Programming

Problem 3: Longest Common Subsequence

Step 3: Compute optimal solution for cost.

Example 1: What do ABCB and BDCAB have in common?

$X = A\ B\ C\ B$

$Y = B\ D\ C\ A\ B$

	j	0	1	2	3	4	5
i		Y_j	B	D	C	A	B
0	X_I	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				
4	B	0					

$if\ (X_i == Y_j)$

$c[i,j] = c[i-1,j-1] + 1\ and\ b[i,j] = \text{“} \searrow \text{"}$

$else\ c[i,j] = \max(c[i-1,j], c[i,j-1])\ \ and$

$b[i,j] = \text{“} \downarrow \text{“}\ (if \max is\ c[i-1,j])$

$b[i,j] = \text{“} \rightarrow \text{“}\ (if \max is\ c[i,j-1])$

Dynamic Programming

Problem 3: Longest Common Subsequence

Step 3: Compute optimal solution cost.

Example 1: What do ABCB and BDCAB have in common?

$X = A\ B\ C\ B$

$Y = B\ D\ C\ A\ B$

	j	0	1	2	3	4	5
i		Y_j	B	D	C	A	B
0	X_I	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					

$if\ (X_i == Y_j)$

$c[i,j] = c[i-1, j-1] + 1\ and\ b[i,j] = \text{“} \searrow \text{"}$

$else\ c[i,j] = \max(c[i-1,j], c[i,j-1])\ \ and$

$b[i,j] = \text{“} \downarrow \text{“}\ (if\ \max is\ c[i-1,j])$

$b[i,j] = \text{“} \rightarrow \text{“}\ (if\ \max is\ c[i,j-1])$

Dynamic Programming

Problem 3: Longest Common Subsequence

Step 3: Compute optimal solution cost.

Example 1: What do ABCB and BDCAB have in common?

$X = A\ B\ C\ B$

$Y = B\ D\ C\ A\ B$

	j	0	1	2	3	4	5
i		Y_j	B	D	C	A	B
0	X_I	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					

$if\ (X_i == Y_j)$

$\quad c[i,j] = c[i-1, j-1] + 1\ and\ b[i,j] = \text{“} \searrow \text{"}$

$else\ c[i,j] = \max(c[i-1,j], c[i,j-1])\ \ and$

$\qquad b[i,j] = \text{“} \downarrow \text{“}\ (if \max is\ c[i-1,j])$

$\qquad b[i,j] = \text{“} \rightarrow \text{“}\ (if \max is\ c[i,j-1])$

Dynamic Programming

Problem 3: Longest Common Subsequence

Step 3: Compute optimal solution cost.

Example 1: What do ABCB and BDCAB have in common?

$X = A\ B\ C\ B$

$Y = B\ D\ C\ A\ B$

	j	0	1	2	3	4	5
i		Y_j	B	D	C	A	B
0	X_I	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	
4	B	0					

$if\ (X_i == Y_j)$

$c[i,j] = c[i-1, j-1] + 1\ and\ b[i,j] = \text{“}\searrow\text{"}$

$else\ c[i,j] = \max(c[i-1,j], c[i,j-1])\ \ and$

$b[i,j] = \text{“}\downarrow\text{“}\ (if \max is\ c[i-1,j])$

$b[i,j] = \text{“}\rightarrow\text{“}\ (if \max is\ c[i,j-1])$

Dynamic Programming

Problem 3: Longest Common Subsequence

Step 3: Compute optimal solution cost.

Example 1: What do ABCB and BDCAB have in common?

$X = A\ B\ C\ B$

$Y = B\ D\ C\ A\ B$

	j	0	1	2	3	4	5
i		Y_j	B	D	C	A	B
0	X_I	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					

$if\ (X_i == Y_j)$

$c[i,j] = c[i-1,j-1] + 1\ and\ b[i,j] = \text{“}\searrow\text{"}$

$else\ c[i,j] = \max(c[i-1,j], c[i,j-1])\ \ and$

$b[i,j] = \text{“}\downarrow\text{“}\ (if \max is\ c[i-1,j])$

$b[i,j] = \text{“}\rightarrow\text{“}\ (if \max is\ c[i,j-1])$

Dynamic Programming

Problem 3: Longest Common Subsequence

Step 3: Compute optimal solution cost.

Example 1: What do ABCB and BDCAB have in common?

$X = A\ B\ C\ B$

$Y = B\ D\ C\ A\ B$

	j	0	1	2	3	4	5
i		Y_j	B	D	C	A	B
0	X_I	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				

$if\ (X_i == Y_j)$

$c[i,j] = c[i-1,j-1] + 1\ and\ b[i,j] = \text{“} \searrow \text{"}$

$else\ c[i,j] = \max(c[i-1,j], c[i,j-1])\ \ and$

$b[i,j] = \text{“} \downarrow \text{“}\ (if \max is\ c[i-1,j])$

$b[i,j] = \text{“} \rightarrow \text{“}\ (if \max is\ c[i,j-1])$

Dynamic Programming

Problem 3: Longest Common Subsequence

Step 3: Compute optimal solution cost.

Example 1: What do ABCB and BDCAB have in common?

$X = A\ B\ C\ B$

$Y = B\ D\ C\ A\ B$

	j	0	1	2	3	4	5
i		Y_j	B	D	C	A	B
0	X_I	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			

$if\ (X_i == Y_j)$

$c[i,j] = c[i-1, j-1] + 1\ and\ b[i,j] = \text{“} \searrow \text{"}$

$else\ c[i,j] = \max(c[i-1,j], c[i,j-1])\ \ and$

$b[i,j] = \text{“} \downarrow \text{“}\ (if \max is\ c[i-1,j])$

$b[i,j] = \text{“} \rightarrow \text{“}\ (if \max is\ c[i,j-1])$

Dynamic Programming

Problem 3: Longest Common Subsequence

Step 3: Compute optimal solution cost.

Example 1: What do ABCB and BDCAB have in common?

$X = A\ B\ C\ B$

$Y = B\ D\ C\ A\ B$

	j	0	1	2	3	4	5
i		Y_j	B	D	C	A	B
0	X_I	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	D	0	1	2	2		

$if\ (X_i == Y_j)$

$c[i,j] = c[i-1,j-1] + 1\ and\ b[i,j] = \text{“}\searrow\text{"}$

$else\ c[i,j] = \max(c[i-1,j], c[i,j-1])\ \ and$

$b[i,j] = \text{“}\downarrow\text{“}\ (if \max is\ c[i-1,j])$

$b[i,j] = \text{“}\rightarrow\text{“}\ (if \max is\ c[i,j-1])$

Dynamic Programming

Problem 3: Longest Common Subsequence

Step 3: Compute optimal solution cost.

Example 1: What do ABCB and BDCAB have in common?

$X = A\ B\ C\ B$

$Y = B\ D\ C\ A\ B$

	j	0	1	2	3	4	5
i		Y_j	B	D	C	A	B
0	X_I	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	

$if\ (X_i == Y_j)$

$\quad c[i,j] = c[i-1,j-1] + 1\ and\ b[i,j] = \text{“}\searrow\text{"}$

$else\ c[i,j] = \max(c[i-1,j], c[i,j-1])\ \ and$

$\quad b[i,j] = \text{“}\downarrow\text{“}\ (if\ \max is\ c[i-1,j])$

$\quad b[i,j] = \text{“}\rightarrow\text{“}\ (if\ \max is\ c[i,j-1])$

Dynamic Programming

Problem 3: Longest Common Subsequence

Step 3: Compute optimal solution cost.

Example 1: What do ABCB and BDCAB have in common?

$X = A\ B\ C\ B$

$Y = B\ D\ C\ A\ B$

	j	0	1	2	3	4	5
i		Y_j	B	D	C	A	B
0	X_I	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

$if\ (X_i == Y_j)$

$\quad c[i,j] = c[i-1,j-1] + 1\ and\ b[i,j] = \text{“}\searrow\text{"}$

$else\ c[i,j] = \max(\ c[i-1,j], c[i,j-1]\ )\quad and$

$\qquad b[i,j] = \text{“}\downarrow\text{“}\ (if\ \max is\ c[i-1,j])$

$\qquad b[i,j] = \text{“}\rightarrow\text{“}\ (if\ \max is\ c[i,j-1])$

Dynamic Programming

Problem 3: Longest Common Subsequence

Step 3: Compute optimal solution cost.

Example 1: What do ABCB and BDCAB have in common?

	j	0	1	2	3	4	5
i		Y_j	B	D	C	A	B
0	X_I	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

$$X = A\ B\ C\ B$$
$$Y = B\ D\ C\ A\ B$$

$if\ (X_i == Y_j)$

$c[i,j] = c[i-1,j-1] + 1\ and\ b[i,j] = \text{“}\searrow\text{"}$

$else\ c[i,j] = \max(c[i-1,j], c[i,j-1])\ \ and$

$b[i,j] = \text{“}\downarrow\text{“}\ (if \max is\ c[i-1,j])$

$b[i,j] = \text{“}\rightarrow\text{“}\ (if \max is\ c[i,j-1])$

Dynamic Programming

Problem 3: Longest Common Subsequence

Step 3: Compute optimal solution cost.

Example 1: What do ABCB and BDCAB have in common?

$X = A\ B\ C\ B$

$Y = B\ D\ C\ A\ B$

	j	0	1	2	3	4	5
i		Y_j	B	D	C	A	B
0	X_I	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

$if\ (X_i == Y_j)$

$c[i,j] = c[i-1,j-1] + 1\ and\ b[i,j] = \text{“} \searrow \text{"}$

$else\ c[i,j] = \max(c[i-1,j], c[i,j-1])\ \ and$

$b[i,j] = \text{“} \downarrow \text{“}\ (if\ \max is\ c[i-1,j])$

$b[i,j] = \text{“} \rightarrow \text{“}\ (if\ \max is\ c[i,j-1])$

The running time= $O(m * n)$ since each $c[i,j]$ is calculated in constant time, and there are m*n elements in the array

Dynamic Programming

Problem 3: Longest Common Subsequence

Step 4: Construct / print the optimal solution.

Example 1: What do ABCB and BDCAB have in common?

$Print - LCS(b, X, i, j)$
$if (i == 0) and\ (j == 0)$
$\quad return$
$if\ b[i, j] ==$ " ↘ "
$\qquad\qquad Print - LCS(b, X, i-1, j-1)$
$\qquad\qquad Print\ x_i$
$if\ b[i, j] ==$ " ↓ "
$\qquad\qquad Print - LCS(b, X, i-1, j)$
$else Print - LCS(b, X, i, j-1)$

Dynamic Programming

Problem 3: Longest Common Subsequence

Step 4: Construct / print the optimal solution.

Example 1: What do ABCB and BDCAB have in common?

$Print-LCS(b,X,i,j)$

$if\,(i==0) and\ (j==0)$

$\quad return$

$if\ b[i,j]==\text{“}\searrow\text{“}$

$\qquad Print-LCS(b,X,i-1,j-1)$

$\qquad Print\ x_i$

$if\ b[i,j]==\text{“}\downarrow\text{“}$

$\qquad Print-LCS(b,X,i-1,j)$

$else Print-LCS(b,X,i,j-1)$

	j	0	1	2	3	4	5
i		Y_j	B	D	C	A	B
0	X_I	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

B

Dynamic Programming

Problem 3: Longest Common Subsequence

Step 4: Construct / print the optimal solution.

Example 1: What do ABCB and BDCAB have in common?

$Print-LCS(b,X,i,j)$

$if\,(i==0) and\ (j==0)$

$\quad return$

$if\ b[i,j] ==$ " ↘ "

$\qquad Print-LCS(b,X,i-1,j-1)$

$\qquad Print\ x_i$

$if\ b[i,j] ==$ " ↓ "

$\qquad Print-LCS(b,X,i-1,j)$

$else Print-LCS(b,X,i,j-1)$

	j	0	1	2	3	4	5
i		Y_j	B	D	C	A	B
0	X_I	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

C B

Dynamic Programming

Problem 3: Longest Common Subsequence

Step 4: Construct / print the optimal solution.

Example 1: What do ABCB and BDCAB have in common?

$Print - LCS(b, X, i, j)$
$if\,(i == 0) and\ (j == 0)$
$\quad return$
$if\ b[i, j] ==$ “ ↘ “
$\qquad Print - LCS(b, X, i-1, j-1)$
$\qquad Print\ x_i$
$if\ b[i, j] ==$ “ ↓ “
$\qquad Print - LCS(b, X, i-1, j)$
$else Print - LCS(b, X, i, j-1)$

	j	0	1	2	3	4	5
i		Y_j	B	D	C	A	B
0	X_I	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

B C B

Dynamic Programming

Problem 3: Longest Common Subsequence

Step 4: Construct / print the optimal solution.

Example 1: What do ABCB and BDCAB have in common?

$Print - LCS(b, X, i, j)$
$if\,(i == 0) and\ (j == 0)$
$\quad return$
$if\ b[i,j] == \text{“}\searrow\text{“}$
$\qquad Print - LCS(b, X, i-1, j-1)$
$\qquad Print\ x_i$
$if\ b[i,j] == \text{“}\downarrow\text{“}$
$\qquad Print - LCS(b, X, i-1, j)$
$else Print - LCS(b, X, i, j-1)$

	j	0	1	2	3	4	5
i		Y_j	B	D	C	A	B
0	X_I	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

$$print\ reverse(B\ C\ B) = B\ C\ B$$

Dynamic Programming

Problem 3: Longest Common Subsequence

Step 4: Construct / print the optimal solution.

Example 1: What do ABCB and BDCAB have in common?

The initial call is $Print - LCS(b, X, X.length, Y.length)$

$Print - LCS(b, X, i, j)$
$if\,(i == 0) and\ (j == 0)$
$\quad return$
$if\ b[i, j] ==$ " ↘ "
$\qquad Print - LCS(b, X, i-1, j-1)$
$\qquad Print\ x_i$
$if\ b[i, j] ==$ " ↓ "
$\qquad Print - LCS(b, X, i-1, j)$
$else Print - LCS(b, X, i, j-1)$

This algorithm required
$\Theta(m+n)$ time for execution

Dynamic Programming

Problem 3: Longest Common Subsequence

Example 2: What do ABCBDAB and BDCABA have in common?

$X = A\ B\ C\ B\ D\ A\ B$

$Y = B\ D\ C\ A\ B\ A$

What is the Longest Common Subsequence $LCS\ (X, Y)$?

Example 3: What do AGGTA and GXTYAY have in common?

$X = A\ G\ G\ T\ A$

$Y = G\ X\ T\ Y\ A\ Y$

What is the Longest Common Subsequence $LCS\ (X, Y)$?

Self practice

Thank u