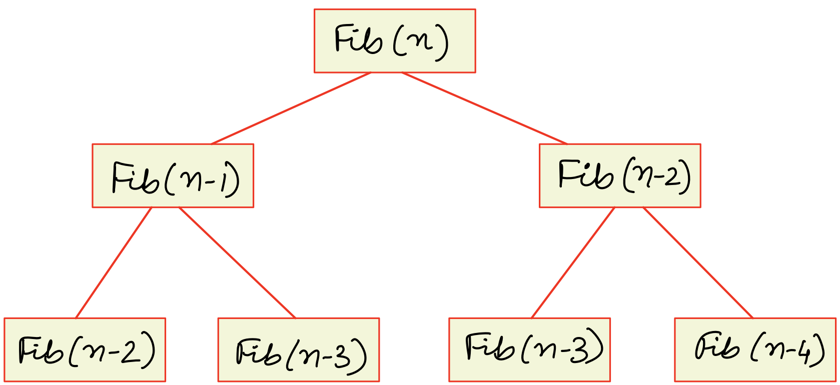
COL 351:
Analysis and Design of Algorithms

Lecture 11

Dynamic Programming

- Break problems into subproblems
- store solution of subproblems as it can be needed multiple times.

Eg Fib(n)




```
Fib(n):
{
 If n ∈ {0,1} : Return 1
 Else:
 Return Fib(n-1) + Fib(n-2)
}
```

- Fib(n-2) is called 2 times
- Fib(n-3) is called 3 times

Longest Common Subsequence (LCS)

$X = c\ a\ b\ a\ c\ d$

$Y = a\ c\ b\ c\ c\ d$

$(c\ \ b\ \ c\ \ d) \leftarrow$ subsequence of Y

$LCS(X, Y) = (c, b, c, d)$

OR

(a, b, c, d)

Solving LCS

Input : Two sequences $X = (x_1 \ldots x_m)$ and $Y = (y_1 \ldots y_n)$

Def^n : $X_i = (x_1 \ldots x_i)$

$Y_j = (y_1 \ldots y_j)$

$T[i,j]$:

$$\boxed{\text{length of } LCS(X_i, Y_j)}$$

To compute $|LCS[X_i, Y_j]|$ we will use entry of 3 cells

		y_1	y_2	y_3	y_4	y_5	y_6
	0	0	0	0	0	0	0
x_1	0						
x_2	0						
x_3	0						
x_4	0						
x_5	0						
x_6	0						

Table of size $(m+1) \times (n+1)$

Ques: How to compute

$LCS(X_i, Y_j)$?

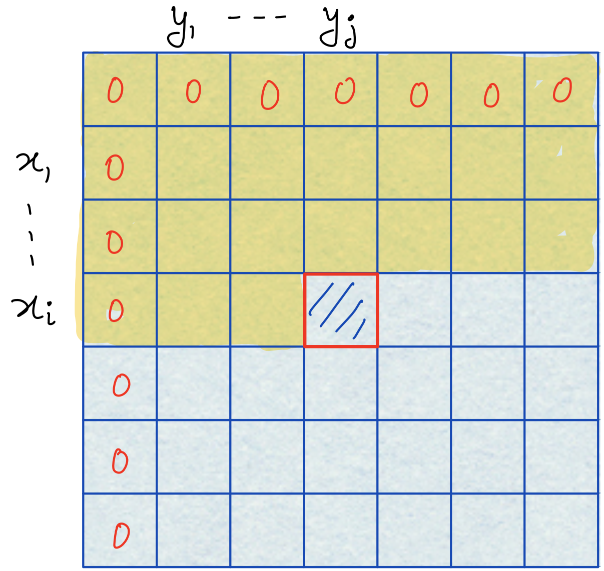


Table of size

$(m+1) \times (n+1)$

Case 1: $x_i = y_j$

$LCS(X_{i-1}, Y_{j-1}) \cdot x_i$

Case 2: $x_i \neq y_j$

Case 2a

last char of

$LCS(X_i, Y_j) \neq x_i$

OUTPUT

$LCS(X_{i-1}, Y_j)$

Case 2b

last char of

$LCS(X_i, Y_j) \neq y_j$

OUTPUT

$LCS(X_i, Y_{j-1})$

Computing table T

Input : Two sequences $X = (x_1 \ldots x_m)$ and $Y = (y_1 \ldots y_n)$

Def^n : $X_i = (x_1 \ldots x_i)$
$Y_j = (y_1 \ldots y_j)$

1. Create a 2D-array "T" of size $(m+1) \times (n+1)$.
2. For $i = 0$ to m : $T[i, 0] = 0$
3. For $j = 0$ to n : $T[0, j] = 0$
4. For $i = 1$ to m :
   - For $j = 1$ to n :
     - If $(x_i = y_j)$: $T[i, j] = T[i-1, j-1] + 1$
     - Else $T[i, j] = \max(T[i-1, j], T[i, j-1])$

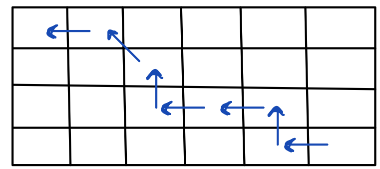
Space $= O(mn)$ Time $= O(mn)$

Computing LCS from table T

Input: Two sequences $X = (x_1 \dots x_m)$ and $Y = (y_1 \dots y_n)$

Def^n: $X_i = (x_1 \dots x_i)$

$Y_j = (y_1 \dots y_j)$



Back tracing

LCS(i,j)

1. If $(i=0$ or $j=0)$ return ϕ
2. If $(x_i = y_j)$ return $\langle LCS(x_{i-1}, y_{j-1}) \cdot x_i \rangle$

   Else if $(T[i,j] = T[i-1,j])$: return $LCS(i-1, j)$

   Else return $LCS(i, j-1)$

* Time to compute LCS(X,Y) is $O(m+n)$ given table T.

CHALLENGE PROBLEM

Question 1: Suppose we are interested in computing length of LCS of X, Y.

Can you achieve this in O(min{m, n}) space in the same time?

Question 2: Suppose we are interested in computing LCS of X, Y.

Can you achieve this in O(m+n) space in polynomial time?

Can we have recursive solution for LCS?

If $(x_i = y_j)$:
  Return $LCS(X_{i-1}, Y_{j-1}) \cdot x_i$

$ans1 = LCS(X_{i-1}, Y_j)$
$ans2 = LCS(X_i, Y_{j-1})$
If $LEN(ans1) \geq LEN(ans2)$: Return ans1
Else: Return ans2

What is time complexity of this algorithm?

H.W. $\Omega(2^{m+n})$

Matrix Chain Multiplication

$A_{10 \times 3}$ $B_{3 \times 5}$ $C_{5 \times 4}$

Goal: Find $P = ABC$ in best possible way
(using simple airthmetics)

- $A\underbrace{\underbrace{(BC)}_{3\times5\times4}}_{10 \times 3 \times 4}$

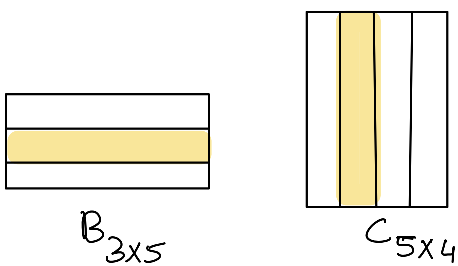
$$\begin{array}{r} 60 \\ 120 \\ \hline 180 \end{array}$$

- $\underbrace{\underbrace{(AB)}_{150}\,C}_{200}$

$$\begin{array}{r} 150 \\ 200 \\ \hline 350 \end{array}$$

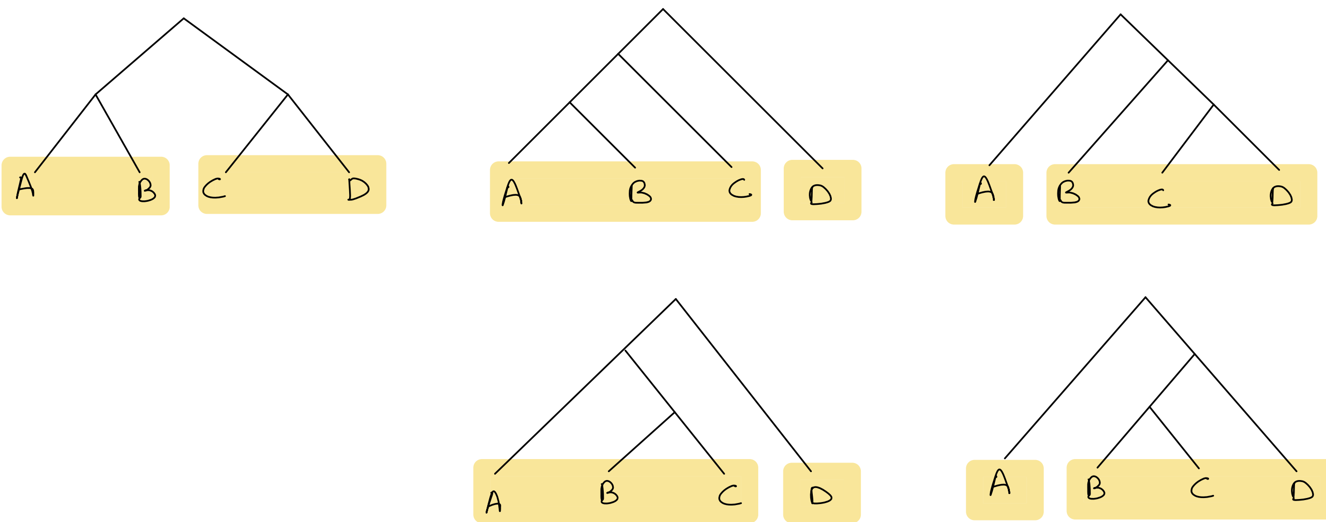
Eg:

Steps
$= 3 \times 5 \times 4$



Matrix Chain Multiplication

Ques:
In how many ways can you multiply four matrices ABCD?



Remark: Though we have 5 possible splits, the number of choices for last split is $4-1 = 3$.

Matrix Chain Multiplication

Given: n- matrices $A_1 \quad A_2 \quad \cdots \quad A_i \quad \cdots \quad A_n$

$d_0 \times d_1$ $\quad$ $d_{i-1} \times d_i$ $\quad$ $d_{n-1} \times d_n$

Goal: Find best possible way for computing $P = A_1 \cdots A_n$

Input: Dimension vector $D = (d_0 \dots d_n)$ for $A_1\ A_2 \dots A_n$.

What should be the subproblems?

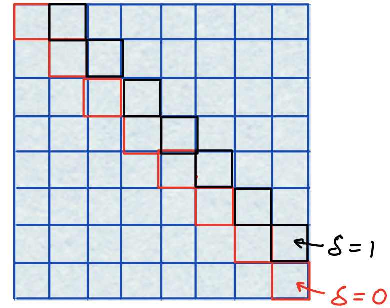
Create a matrix M of size $n \times n$:

Store in $M[i,j]$ BEST POSSIBLE time to multiply matrices $(A_i \dots A_j)$

$$M[i,i] = 0$$

$$M[i,j] = \min_{i \le k < j} \left\{ \underbrace{M[i,k]} + \underbrace{M[k+1,j]} + \underbrace{d_{i-1} \cdot d_k \cdot d_j} \right\}$$

Algorithm to compute matrix M



For $\delta = 0$ to $(n-1)$:

  For $i = 1$ to $(n - \delta)$:

    $j = i + \delta$

    If $(i = j)$: $M[i,j] = 0$

    Else:

$$M[i,j] = \min_{i \le k < j} \left\{ M[i,k] + M[k+1,j] + d_{i-1} \cdot d_k \cdot d_j \right\}$$

Time $= O(n^3)$ Space $= O(n^2)$