

CSE 221 Algorithms: Ch5

Instructor: Saraf Anika

CSE, SoSET, EDU

Fall 2021

Dynamic Programming(DP) Intro

Those who cannot remember the past
are condemned to repeat it.

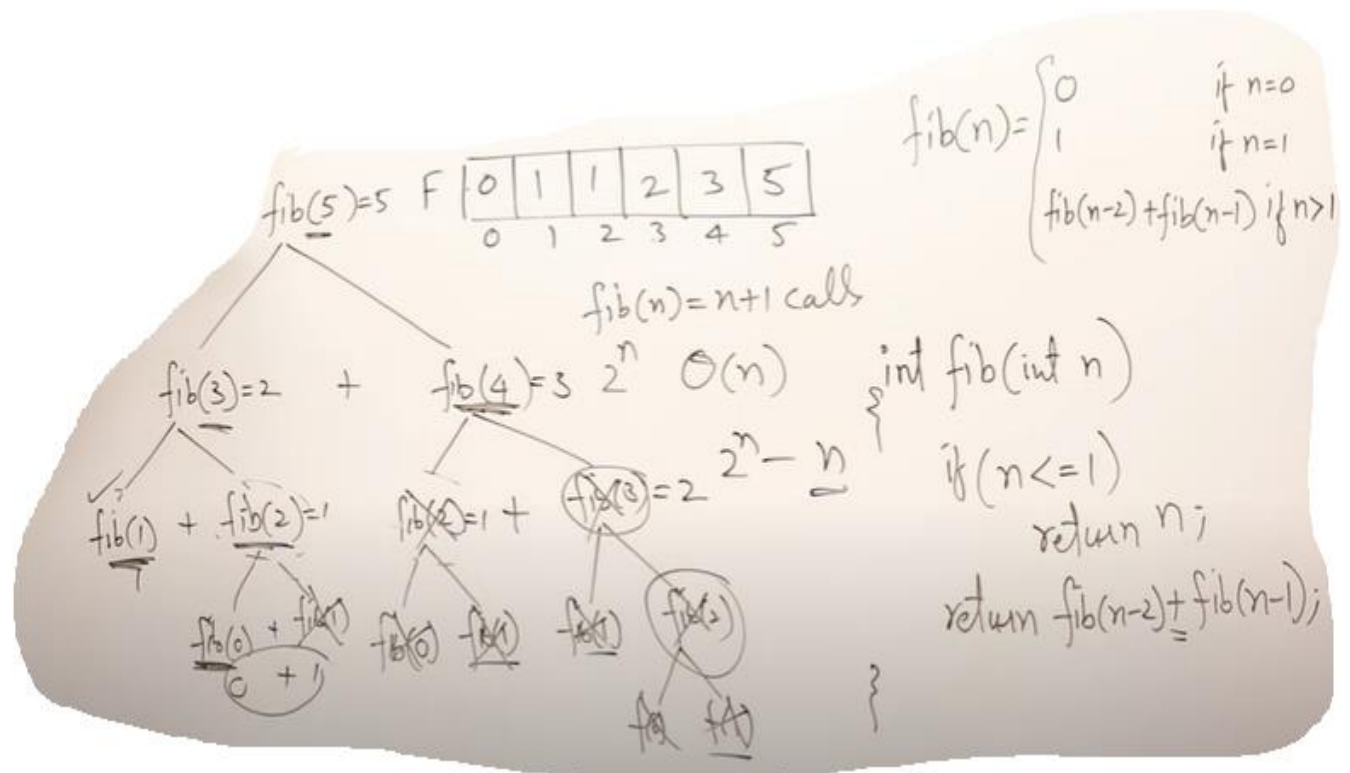
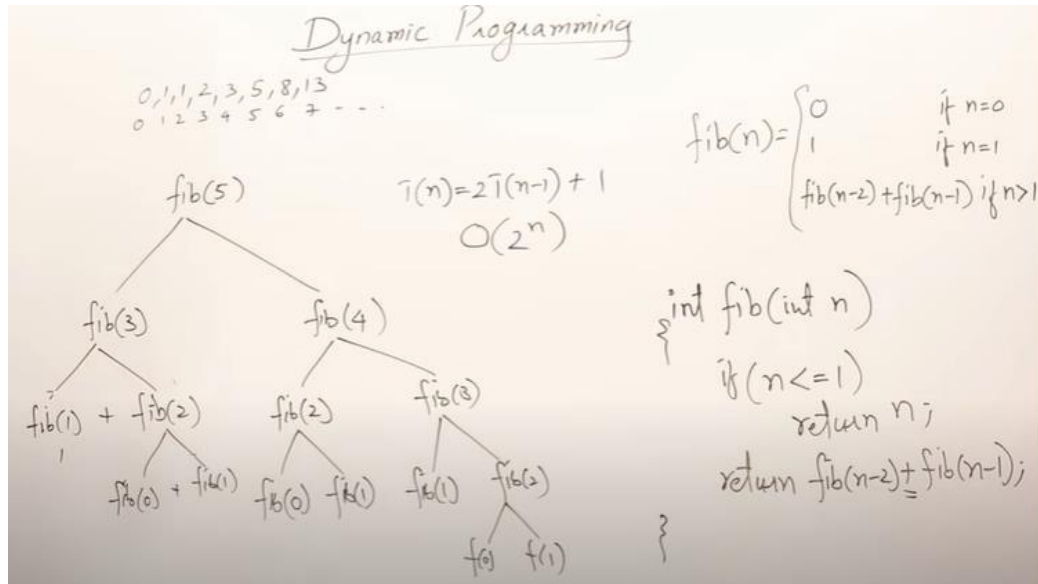
-Dynamic Programming

- ✓ Dynamic Programming (DP) is an algorithmic technique for solving an optimization problem by breaking it down into simpler sub-problems and utilizing the fact that the optimal solution to the overall problem depends upon the optimal solution to its sub-problems.
- Let's take the example of the **Fibonacci numbers**. As we all know, Fibonacci numbers are a series of numbers in which each number is the sum of the two preceding numbers. The first few Fibonacci numbers are 0, 1, 1, 2, 3, 5, and 8, and they continue on from there.
- If we are asked to calculate the nth Fibonacci number, we can do that with the following equation

Dynamic Programming(DP)

- Top-down with Memoization :**

In this approach, we try to solve the bigger problem by recursively finding the solution to smaller sub-problems. Whenever we solve a sub-problem, we cache its result so that we don't end up solving it repeatedly if it's called multiple times. Instead, we can just return the saved result. This technique of storing the results of already solved subproblems is called Memoization.



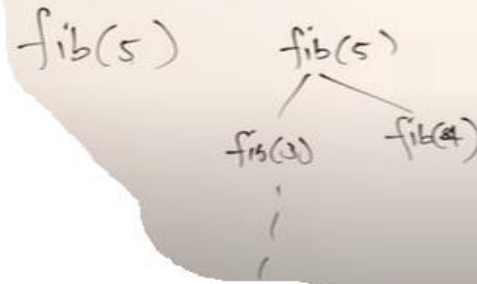
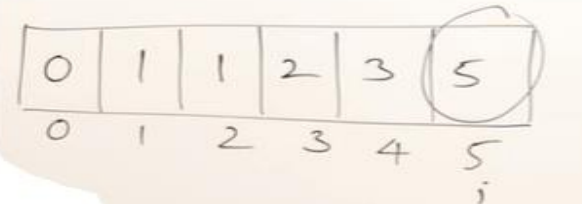
Dynamic Programming(DP)

- **Bottom-up with Tabulation:**

Tabulation is the opposite of the top-down approach and avoids recursion. In this approach, we solve the problem “bottom-up” (i.e. by solving all the related sub-problems first). This is typically done by filling up an n-dimensional table. Based on the results in the table, the solution to the top/original problem is then computed.

Tabulation is the opposite of Memoization, as in Memoization we solve the problem and maintain a map of already solved sub-problems. In other words, in memoization, we do it top-down in the sense that we solve the top problem first (which typically recurses down to solve the sub-problems).

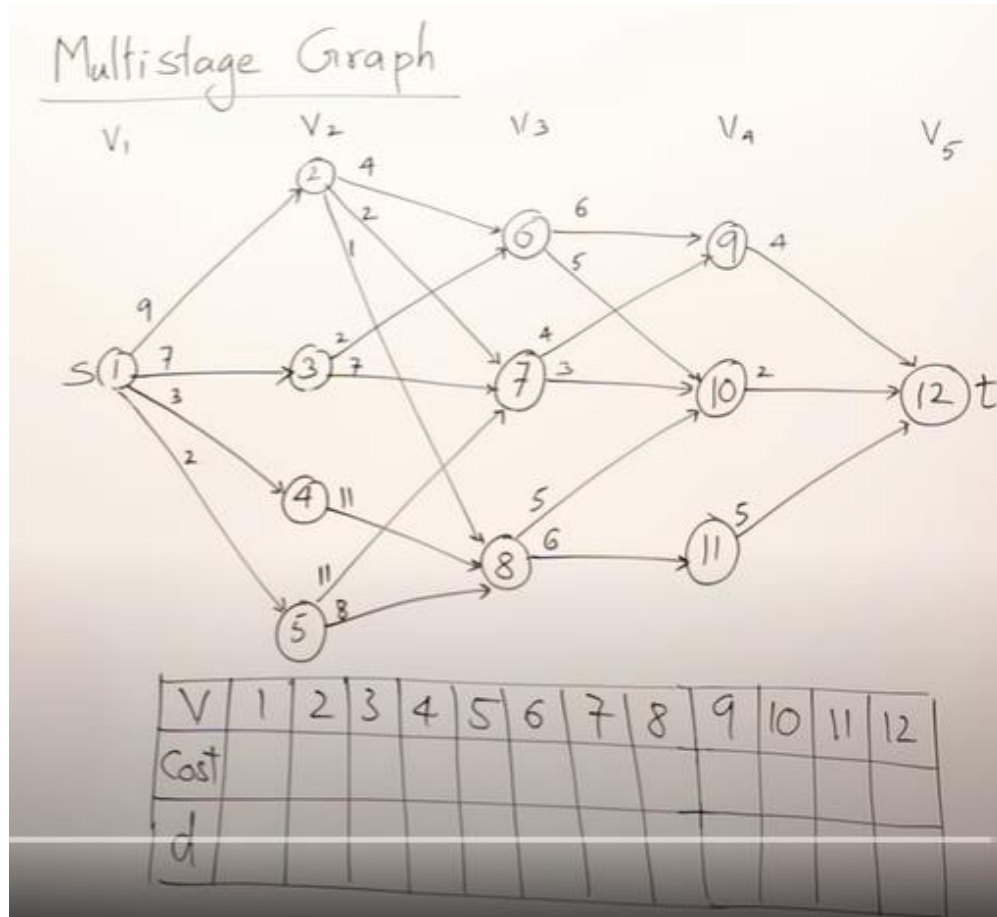
Tabulation VS Memorization



$$\text{fib}(n) = \begin{cases} 0 & \text{if } n=0 \\ 1 & \text{if } n=1 \\ \text{fib}(n-2) + \text{fib}(n-1) & \text{if } n>1 \end{cases}$$

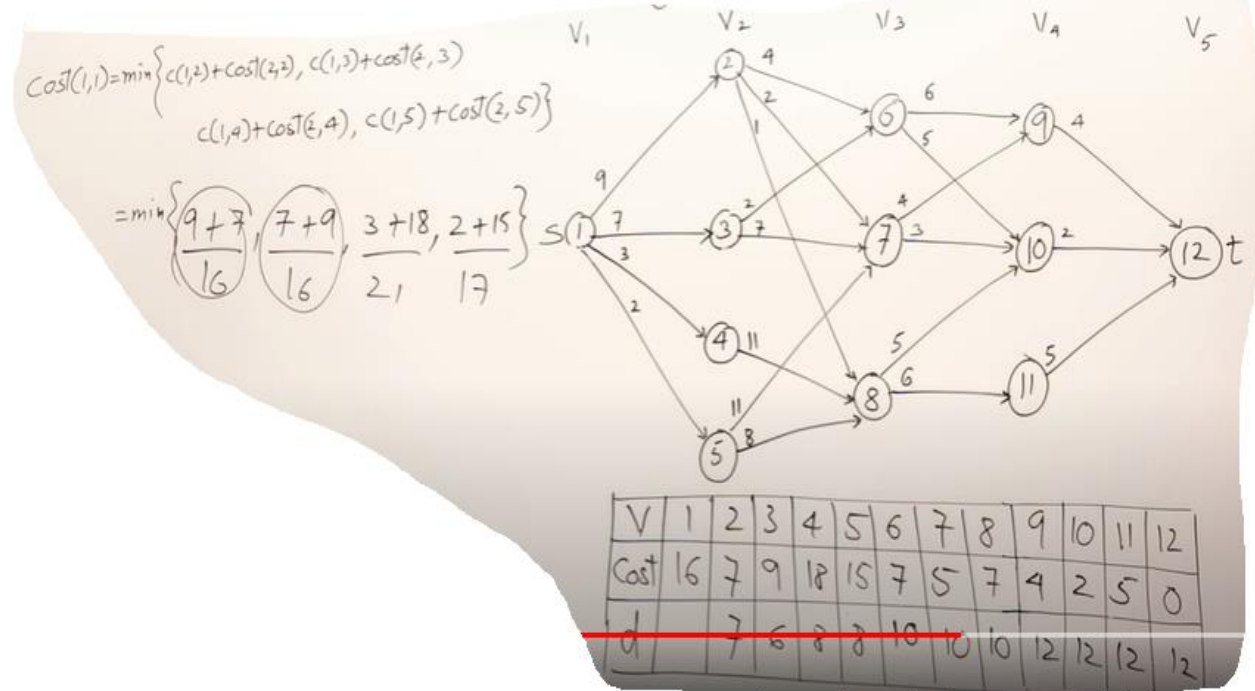
```
int fib(int n)
{
    if (n <= 1)
        return n;
    F[0] = 0; F[1] = 1;
    for (int i = 2; i <= n; i++)
    {
        F[i] = F[i-2] + F[i-1];
    }
    return F[n];
}
```

Multistage Graph(Bottom-Up Approach)



$$\text{Cost}(3,6) = \min \left\{ \begin{aligned} &c(6,9) + \text{Cost}(4,9) \\ &c(6,10) + \text{Cost}(4,10) \end{aligned} \right\}$$

$$= \min \{ 6+4, \underline{5+2} \} = 7$$



Multistage Graph(Bottom-Up Approach)

stage vertex no

$$Cost(i, j) = \min_{\substack{< j, l > \in E \\ l \in V_{i+1}}} \{c(j, l) + Cost(i+1, l)\}$$

stage vertex

$$d(1, \underline{1}) = 2$$

$$d(2, \underline{2}) = 7$$

$$d(3, \underline{7}) = 10$$

$$d(4, \underline{10}) = 12$$

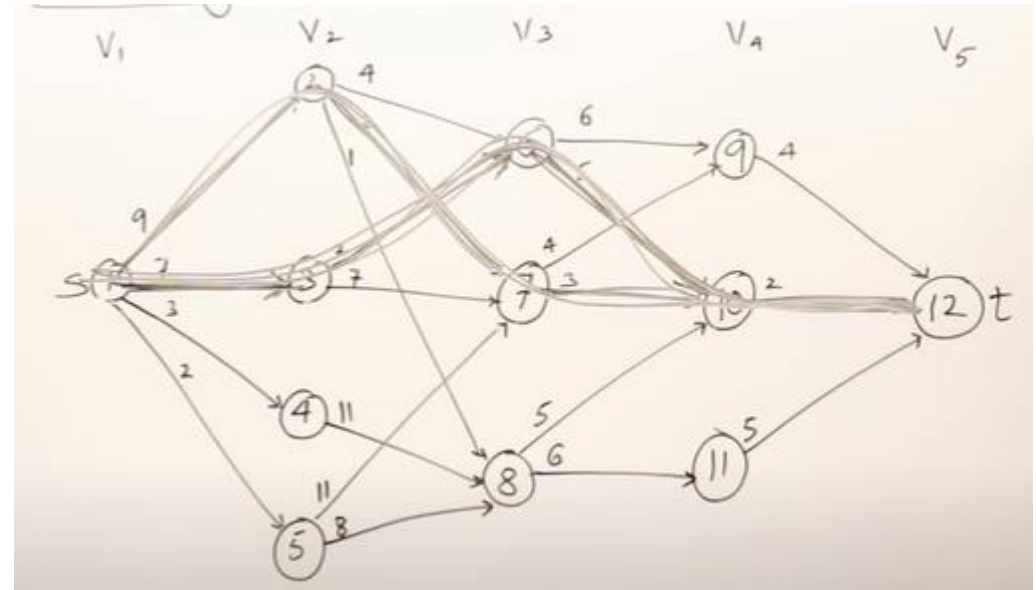
stage vertex

$$d(1, \underline{1}) = 3$$

$$d(2, \underline{3}) = 6$$

$$d(3, \underline{6}) = 10$$

$$d(4, \underline{10}) = 12$$



Matrix Chain Multiplication

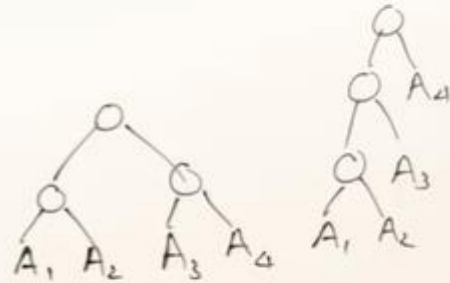
Matrix Chain Multiplication

$$A_1 \cdot A_2 \cdot A_3 \cdot A_4$$

5x4 4x6 6x2 2x7

$$((A_1 \cdot A_2) \cdot A_3) \cdot A_4$$

$$(A_1 \cdot A_2) \cdot (A_3 \cdot A_4)$$



$$T(n) = \frac{2n(n-1)}{n+1}$$

$$T(3) = \underline{\underline{5}}$$

$$A_1 \cdot A_2 \cdot A_3 \cdot A_4$$

5x4 4x6 6x2 2x7

m	1	2	3	4
1				
2				
3				
4				

S	1	2	3	4
1				
2				
3				
4				

Matrix Chain Multiplication

Matrix Chain Multiplication

$A_1 \cdot A_2 \cdot A_3 \cdot A_4$
 $5 \times 4 \quad 4 \times 6 \quad 6 \times 2 \quad 2 \times 7$

$m[1,2]$
 $A_1 \cdot A_2$
 $5 \times 4 \quad 4 \times 6 = 120$

$m[3,4]$
 $A_3 \cdot A_4$
 $6 \times 2 \quad 2 \times 7 = 12$

$m[2,3]$
 $A_2 \cdot A_3$
 $4 \times 6 \quad 6 \times 2 = 12$

m	1	2	3	4
1	0	120		
2		0	48	
3			0	84
4				0

s	1	2	3	4
1		1		
2				
3				
4				

$A_1 \cdot A_2 \cdot A_3 \cdot A_4$
 $5 \times 4 \quad 4 \times 6 \quad 6 \times 2 \quad 2 \times 7$

$m[1,3] = 88$

min { $A_1 \cdot (A_2 \cdot A_3)$, $(A_1 \cdot A_2) \cdot A_3$ }

$m[1,1] + m[2,3] + 5 \times 4 \times 2$ $m[1,2] + m[3,3] + 5 \times 6 \times 2$

$0 + 48 + 40$ $120 + 0 + 60$

88 180

m	1	2	3	4
1	0	120	88	
2		0	48	
3			0	84
4				0

s	1	2	3	4
1		1		
2				
3				
4				

Matrix Chain Multiplication

$A_1 \cdot A_2 \cdot A_3 \cdot A_4$
 $5 \times 4 \quad 4 \times 6 \quad 6 \times 2 \quad 2 \times 7$

$m[2,4]$

$A_2 \cdot (A_3 \cdot A_4) \quad (A_2 \cdot A_3) \cdot A_4$
 $4 \times 6 \quad 6 \times 2 \quad 2 \times 7 \quad 4 \times 6 \quad 6 \times 2 \quad 2 \times 7$

$m[2,2] + m[3,4] + 4 \times 6 \times 7 \quad m[2,3] + m[4,4] + 4 \times 2 \times 7$
 $0 + 84 + 168 \quad 48 + 0 + 56$
 $252 \quad 104$

m	1	2	3	4
1	0	120	88	
2		0	48	
3			0	84
4				0

S	1	2	3	4
1		1	1	
2			2	
3				3
4				

$A_1 \cdot A_2 \cdot A_3 \cdot A_4$
 $5 \times 4 \quad 4 \times 6 \quad 6 \times 2 \quad 2 \times 7$
 $d_0 \quad d_1 \quad d_2 \quad d_3 \quad d_4$

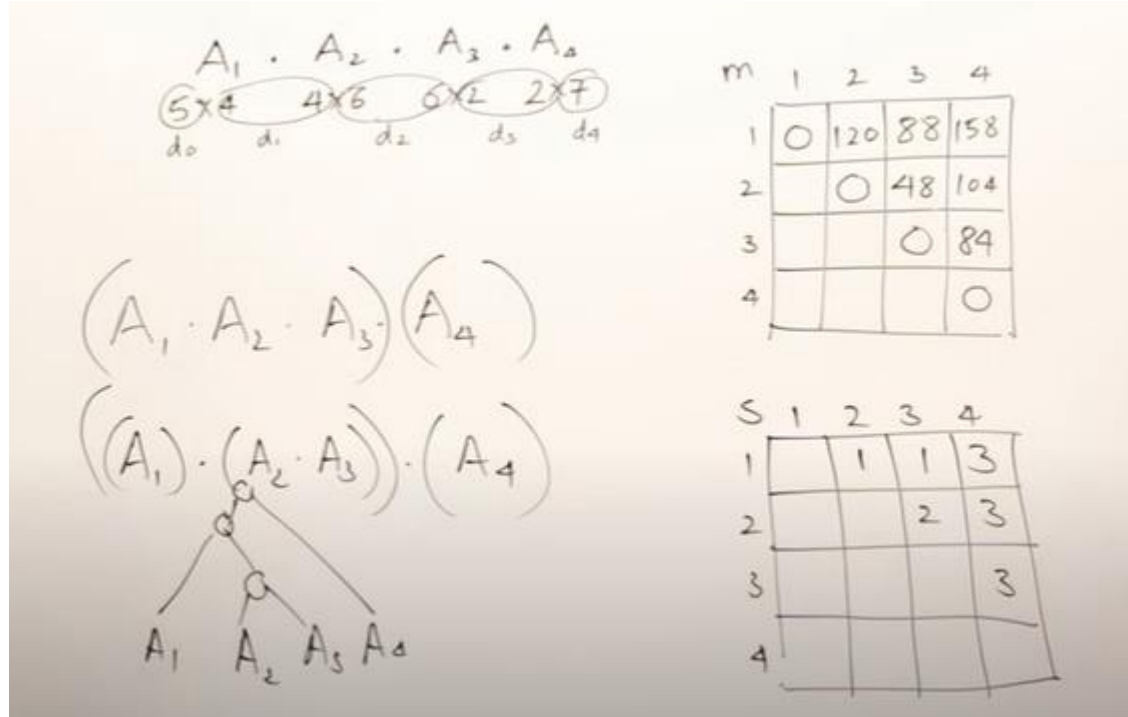
$m[1,4] = \min \left\{ \begin{aligned} &m[1,1] + m[2,4] + 5 \times 4 \times 7 \\ &0 \quad 104 + 140 \\ &m[1,2] + m[3,4] + 5 \times 6 \times 7 \\ &120 + 84 + 210 \\ &m[1,3] + m[4,4] + 5 \times 2 \times 7 \\ &88 + 0 + 70 \end{aligned} \right\}$

$m[i,j] = \min \left\{ m[i,k] + m[k+1,j] + d_{i-1} * d_k * d_j \right\}$

m	1	2	3	4
1	0	120	88	158
2		0	48	104
3			0	84
4				0

S	1	2	3	4
1		1	1	3
2			2	3
3				3
4				

Matrix Chain Multiplication



Time Complexity:

$$\frac{n(n-1)}{2} = n^2 \times n = n^3$$
$$O(n^3)$$

LCS

Longest Common Subsequence (LCS)

Longest Common Subsequence (LCS)

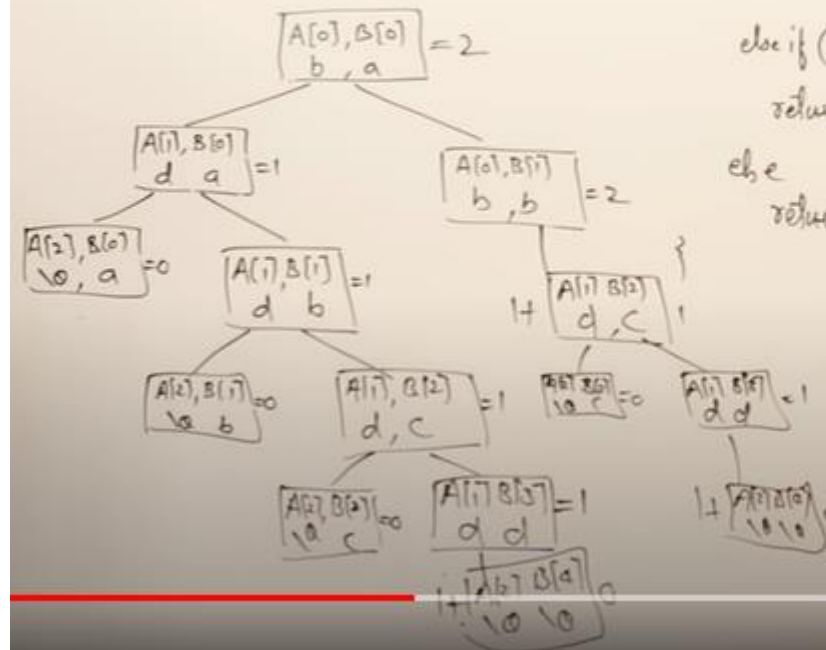
String1: a b c d e f g h i j
String2: c d g i ✓

String1: a b c d e f g h i j
String2: e c d g i ✗

Longest Common Subsequence (LCS)

Without DP

A [b | d | \0]
0 1 2
B [a | b | c | d | \0]
0 1 2 3 4



LCS

A

b	d
---	---

B

a	b	c	d
---	---	---	---

1 2 3 4

a b c d

	0	1	2	3	4
0					
b 1					
d 2					

if ($A[i] = B[j]$)

$$LCS[i, j] = 1 + LCS[i-1, j-1]$$

else

$$LCS[i, j] = \max(LCS[i-1, j], LCS[i, j-1])$$

A

b	d
---	---

B

a	b	c	d
---	---	---	---

a b c d

	0	1	2	3	4
0	0	0	0	0	0
b 1	0	0	1	1	1
d 2	0	0	1	1	2

b	d
---	---

if ($A[i] = B[j]$)

$$LCS[i, j] = 1 + LCS[i-1, j-1]$$

else

$$LCS[i, j] = \max(LCS[i-1, j], LCS[i, j-1])$$

LCS

str1: stone
str2: longest

		l	o	n	g	e	s	t	
		0	1	2	3	4	5	6	7
s	0	0	0	0	0	0	0	0	0
t	1	0							
o	2	0							
n	3	0							
e	4	0							
s	5	0							

if (A[i] = B[j])
 $LCS[i, j] = 1 + LCS[i-1, j-1]$
else
 $LCS[i, j] = \max(LCS[i-1, j], LCS[i, j-1])$

LCS

m str1: stone
n str2: longest (one)
3

		l	o	n	g	e	s	t	
		0	1	2	3	4	5	6	7
0		0	0	0	0	0	0	0	0
s	1	0	0	0	0	0	0	1	1
t	2	0	0	0	0	0	0	1	2
o	3	0	0	1	1	1	1	1	2
n	4	0	0	1	2	2	2	2	2
e	5	0	0	1	2	2	3	3	3

o n e

if $(A[i] = B[j])$

$$LCS[i, j] = 1 + LCS[i-1, j-1]$$

else

$$LCS[i, j] = \max(LCS[i-1, j], LCS[i, j-1])$$

m x n

$O(mn)$

0/1 Knapsack Problem: 2 Methods (DP)

Breakdown of the formula:

★ V : DP table which stores values of subproblems

★ $V[i, w]$: Maximum profit by considering the first 'i' elements in a bag of weight 'w'

★ $V[i-1, w]$: Case when the current object is not included (~0) and the bag with current weight 'w' must be filled with the maximum profit possible (stored at 'i-1')

★ $V[i-1, w-w[i]] + p[i]$: Case when the current object is included (~1) and the remaining part of the bag 'w-w[i]' must be filled with the maximum profit possible (stored at 'i-1')

Bottom Up Method

O/1 Knapsack Problem

$m=8$
 $n=4$ ✓

$P=\{1, 2, 5, 6\}$ ✓
 $w=\{2, 3, 4, 5\}$ ✓

P_i	w_i	0	1	2	3	4	5	6	7	8
1	2	0	0	1	1	1	1	1	1	1
2	3	0	0	1	2	3	3	3	3	3
5	4	0	0	1	3	5	5	6	7	7
6	5	0	0	1	2	5	6	6	7	8

Profit

1+1+1+1+1+1+1+1

Formula, (Bottom Up Method)

$$V[i, j] = \max\{V[i-1, w], V[i-1, w-w[i]] + P[i]\}$$

Where w is the values of the rows on the top of the table, $w[i]$ is the associated weights given in the question.

0/1 Knapsack Problem: 2 Methods

0/1 Knapsack Problem

$P = \{1, 2, 5, 6\}$
 $W = \{2, 3, 4, 5\}$

		W									
		$\sqrt{}$	0	1	2	3	4	5	6	7	8
P_i	W_i	0	0	0	0	0	0	0	0	0	0
1	2	1	0	0	1	1	1	1	1	1	1
2	3	2	0	0	1	2	3	3	3	3	3
5	4	3	0	0	1	2	5	5	6	7	7
6	5	4	0	0	1	2	5	6	6	7	8

x_1, x_2, x_3, x_4
 $\{0, 1, 0, 1\}$

$8 - 6 = 2$
 $2 - 2 = 0$

SET METHOD:

0/1 Knapsack Problem

$m=8$
 $n=4$
 $P = \{1, 2, 5, 6\}$
 $W = \{2, 3, 4, 5\}$
 (P, W)

$S^0 = \{(0, 0)\}$
 $S^1 = \{(1, 2)\}$
 $S^2 = \{(0, 0), (1, 2)\}$
 $S^3 = \{(2, 3), (3, 5)\}$
 $S^4 = \{(0, 0), (1, 2), (2, 3), (3, 5)\}$
 $S^5 = \{(5, 4), (6, 6), (7, 7), (8, 9)\}$
 $S^6 = \{(0, 0), (1, 2), (2, 3), (3, 5), (5, 4), (6, 6), (7, 7)\}$
 $S^7 = \{(6, 5), (7, 7), (8, 8), (9, 9), (12, 11), (13, 13)\}$
 $S^8 = \{(0, 0), (1, 2), (2, 3), (5, 4), (6, 6), (7, 7), (8, 8)\}$

$(8, 8) \in S^4$
 but $(8, 8) \notin S^3 \therefore x_4 = 1$
 $(8 - 6, 8 - 5) = (2, 3)$

$(2, 3) \in S^3 \therefore x_3 = 0$
 and $(2, 3) \in S^2$

$(2, 3) \in S^2 \therefore x_2 = 1$
 but $(2, 3) \notin S^1$

$(2 - 2, 3 - 1) = (0, 0)$
 $(0, 0) \in S^0$ and $(0, 0) \in S^0 \therefore x_1 = 0$