CS & IT ENGINEERING

Algorithm

Dynamic Programming

Lecture No.- 03

By- Dr. Khaleel Khan sir

Recap of Previous Lecture

Topic Multistage Graph

Topic Travelling Salesperson Problem

Topics to be Covered

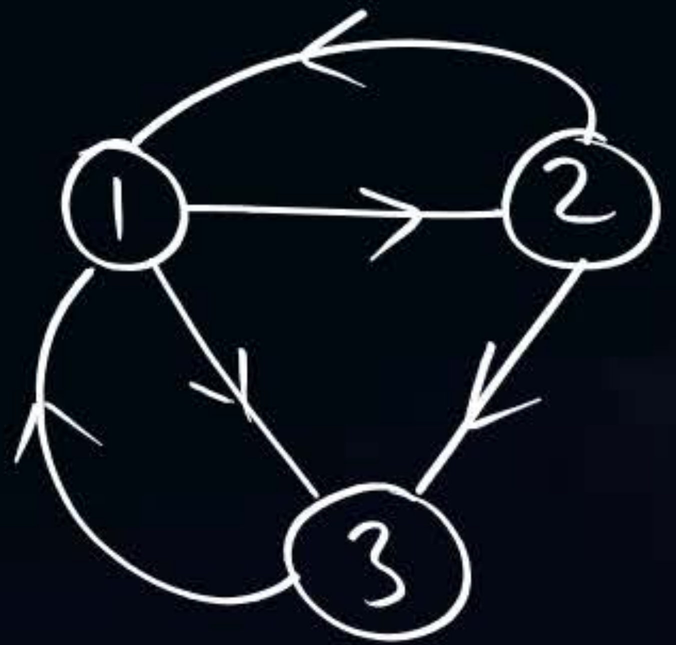
Topic: All Pairs Shortest Paths

Topic: 0/1 Knapsack

LCS

Topic : Dynamic Programming: (DP)

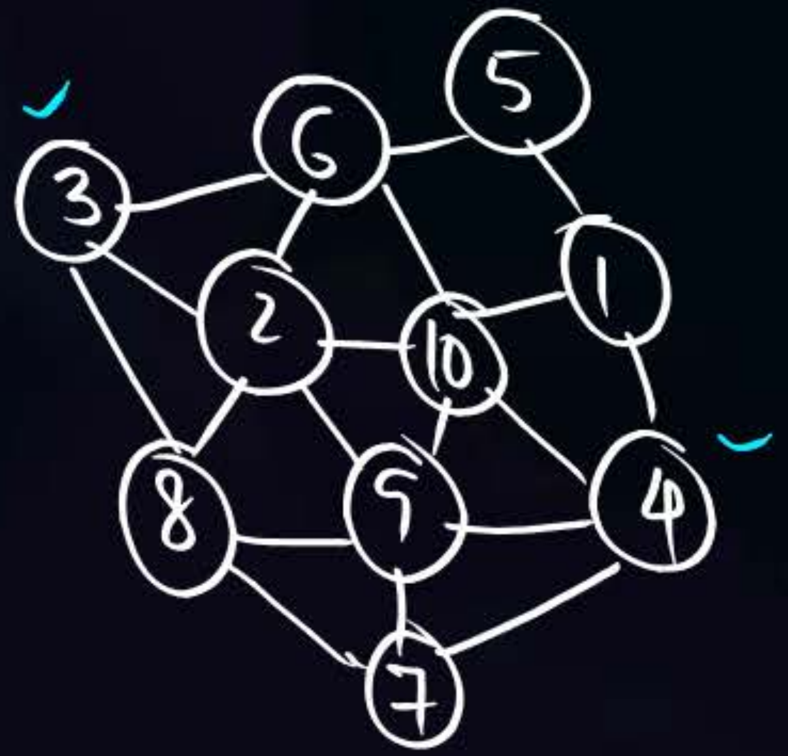
3) All-Pairs Shortest Paths <Floyd-Warshall's Algo>



C	1	2	3
1	0	4	11
2	6	0	2
3	3	∞	0

Let $A^K(i,j)$ repr. cost of the Path from vertex 'i' (src) to vertex 'j' (dest.), with 'K' being the highest Intermediate vertex along the path;

$$\boxed{A^K(i,j) : i \cdots K \cdots j}$$



src   dest

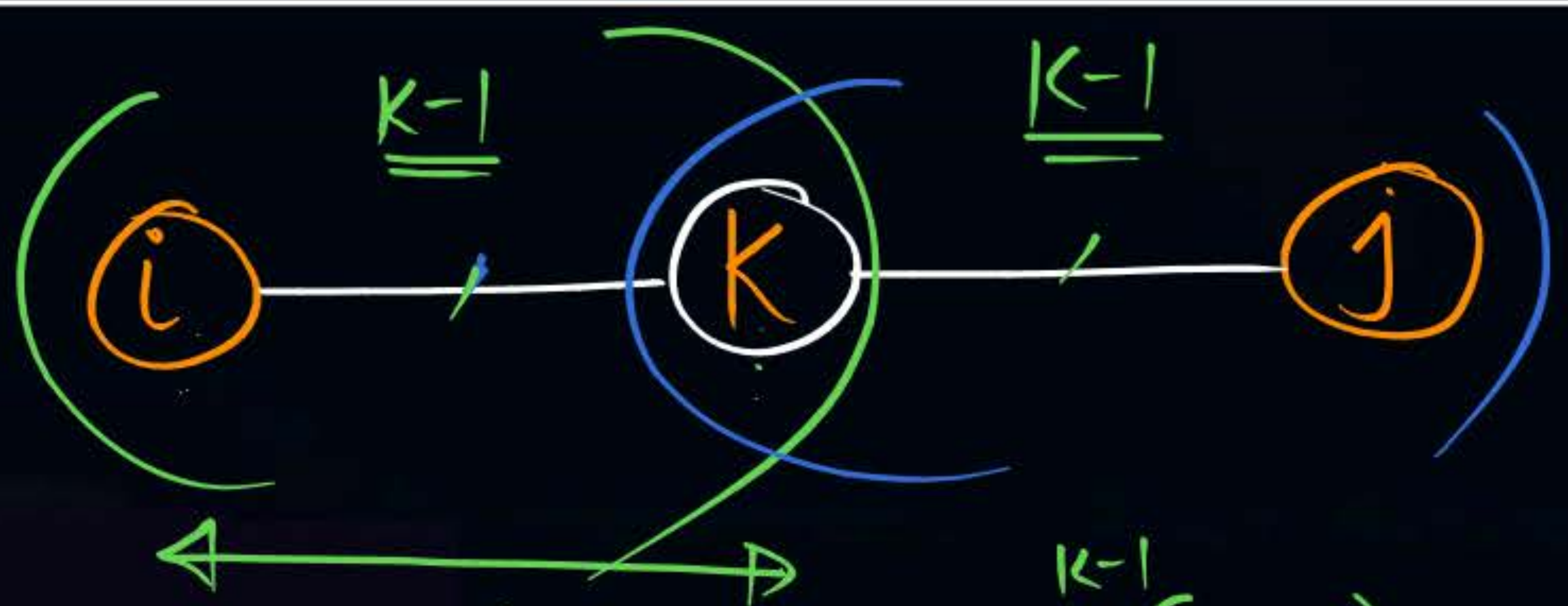
3 - - - - 4

3-6-5-1-4

3-2-10-1-4

3-8-9-10-4

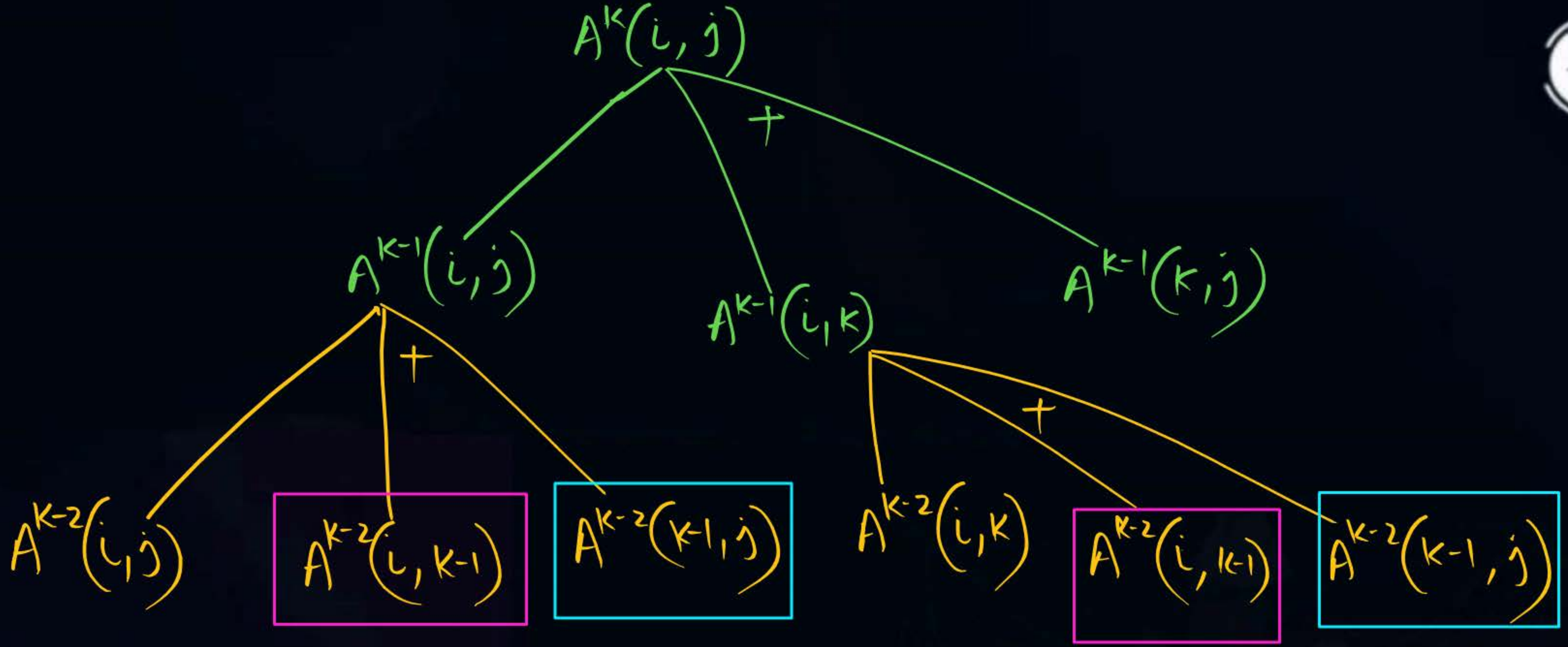
$$A^{k}(i,j) = \min_{1 \le k \le n} \left\{ A^{k-1}(i,k) + A^{k-1}(k,j) \;,\; A^{k-1}(i,j) \right\}$$



$$A^{k-1}(i,k) + A^{k-1}(k,j)$$

$$A^{0}(i,j) = c(i,j)$$

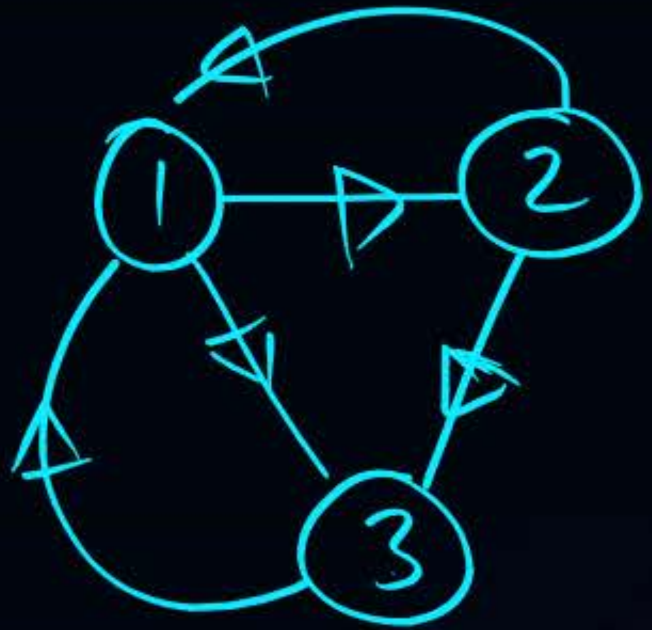
= edge



$A^{k}(i, j)$
$A^{k-1}(i, j)$
+
$A^{k-1}(i, k)$
$A^{k-1}(k, j)$
$A^{k-2}(i, j)$
+
$A^{k-2}(i, k-1)$
$A^{k-2}(k-1, j)$
$A^{k-2}(i, k)$
+
$A^{k-2}(i, k-1)$
$A^{k-2}(k-1, j)$

$$A^K(i,j) = \min\{A^{K-1}(i,j),\ A^{K-1}(i,K) + A^{K-1}(K,j)\}$$

$$A^3 \Rightarrow A^2 \Rightarrow A^1 \Rightarrow A^0$$



C	1	2	3
1	0	4	11
2	6	0	2
3	3	∞	0

A^0	1	2	3
1	0	4	11
2	6	0	2
3	3	∞	0

A^1	1	2	3
1	0	4	11
2	6	0	2
3	3	7	0

A^2	1	2	3
1	0	4	6
2	6	0	2
3	3	7	0

A^3	1	2	3
1	0	4	6
2	5	0	2
3	3	7	0

$$A^1(1,2) = \min\{\overset{4}{A^0(1,2)},\ \overset{0}{A^0(1,1)} \overset{+4}{+} A^0(1,2)\}$$

$A^1(2,3)$

$2 - 1 - 3$

$(6+11) = 17$

$A^1(3,2)$

$3 - 1 - 2$

Algorithm FLOYD-WARSHAL(G, n, e, C
integer $C[1..n, 1..n]$

{

integer $A[1..n, 1..n]$

Time: $O(n^3)$

Space: $O(n^2)$

$$\boxed{\begin{array}{l} 1.\ \text{for } i \leftarrow 1 \text{ to } n \\ \quad \text{for } j \leftarrow 1 \text{ to } n \\ \quad\quad A[i,j] = C[i,j]; \end{array}}$$

2. for $K \leftarrow 1$ to n : Intermediate vertex
    for $i \leftarrow 1$ to n : src
        for $j \leftarrow 1$ to n : dest

$$A[i,j] = \min\{\underline{A[i,j],\ A[i,K] + A[K,j]}\}$$

2m

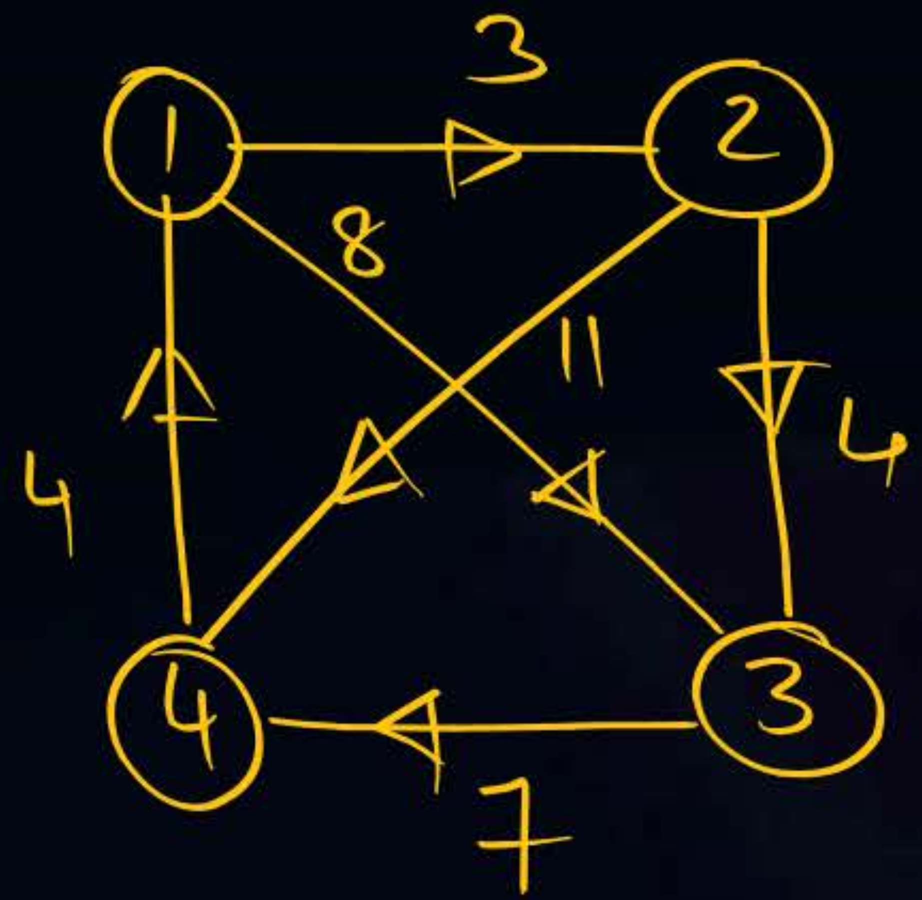
}

→ Transitive closure of a Matrix (representing a graph)

Flyod-warshall's Algo. can be used to obtain Transitive of a Matrix (repr. the graph)

└→ $O(n^3)$

$i \to j$
$j \to k$ $\Longrightarrow i \to k$



1
2
3
4
3
8
11
4
4
7

4) 0|1 KNAPSACK (Binary Knapsack)

→ KNAP Cap : M

→ no. of objects : n

(O_i)

w_i P_i x_i

$$\text{Max} \sum_{i=1}^{n} P_i x_i$$

$$\text{S.T.C} \sum_{i=1}^{n} w_i \cdot x_i \leq M$$

$$x_i = 0|1$$

Soln Space : $O(2^n)$

$\langle x_1\, x_2\, x_3 \dots x_n \rangle$

Let 0|KNAP(n, M) repr. profit with 'n'-objects & KNAP of Capacity M;

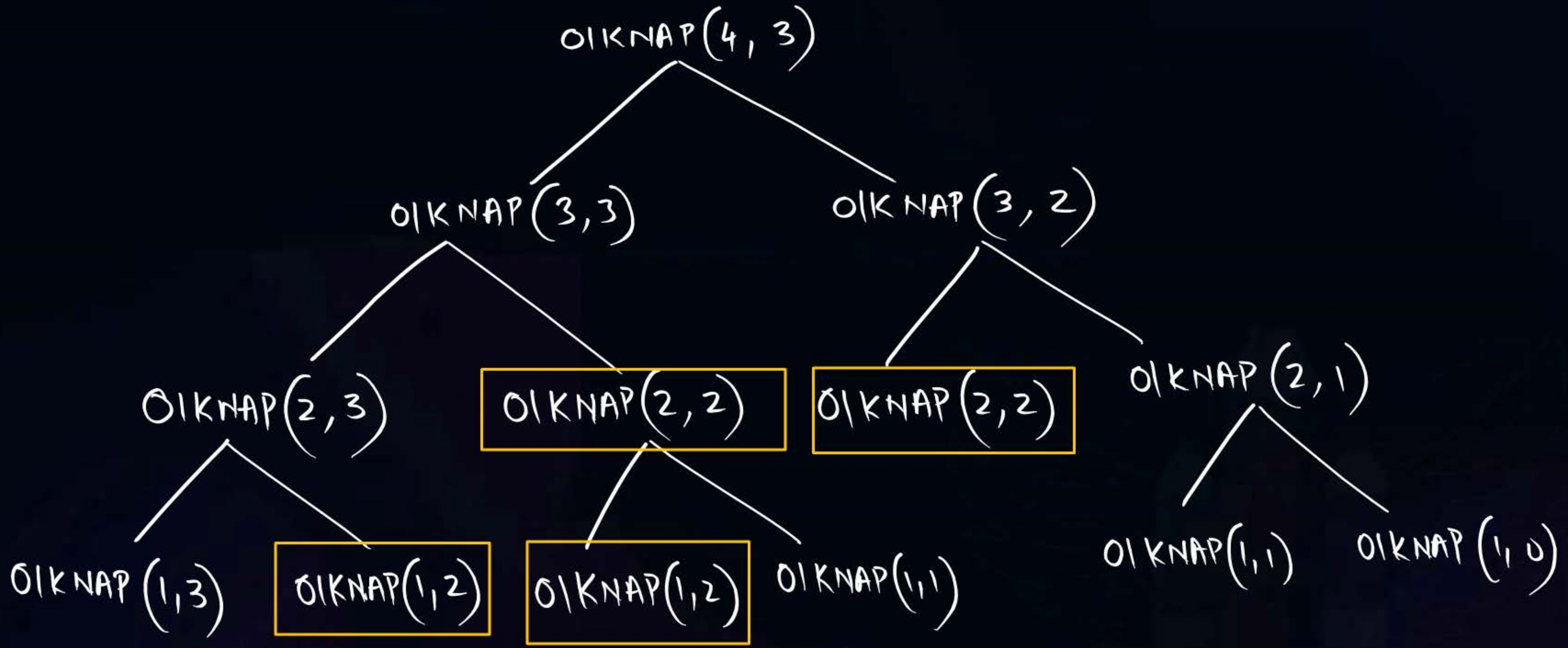
$$0|\text{KNAP}(n,M) = 0|\text{KNAP}(n-1,M), \quad w_n > M$$

$$= \text{Max}\left\{ \underbrace{0|\text{KNAP}(n-1,M)}^{x_n=0},\ \underbrace{0|\text{KNAP}(n-1, M-w_n)+P_n}^{x_n=1} \right\}, \quad w_n \leq M$$

$$0|\text{KNAP}(n,M) = 0, \quad n=0 \text{ or } M=0$$

$n=4$; $M=3$; $\langle w_1 \cdots w_4 \rangle = \langle 1, 1, 1, 1 \rangle$

$\langle P_1 \cdots P_4 \rangle = \langle 10, 20, 30, 40 \rangle$



Ex: $n = 4$; $M = 8$; $\langle w_1 \cdots w_4 \rangle = \langle 2, 3, 4, 5 \rangle$, $\langle P_1 \cdots P_4 \rangle = \langle 1, 2, 5, 6 \rangle$

Let $x[0 \ldots n, 0 \ldots M]$ be an array of which $x[n, M] = $ Profit

$x[4, 8] =$

j (Bottom-up-Tabulation)

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

i

$\langle x_1\, x_2\, x_3\, x_4 \rangle$

0 1 0 1

$x_4 = 1$

$8 - 6 = 2$

$x_3 = 0$

2

$x_2 = 1$

0 $x_1 = 0$

$$X[1,2] = \max\left\{X[0,2],\ X[0,1] + P_1\right\}$$

$$0,\quad 0+1$$

$n = 3;\ M = 6;$

$$\langle P_1, P_2, P_3 \rangle = \langle 1, 2, 5 \rangle$$

$$\langle w_1, w_2, w_3 \rangle = \langle 2, 3, 4 \rangle$$

Time : O(n * M)
Space : O(n * M)

Algo 01KNAP (M, n, ,W, P)

P [1.... n], W [1...... n] ; Input

```
{
    Integer   X[ 0..n, 0..M];
1.  for i ← 0 to n
        for j ← 0 to m
            If (i = 0  or  j = 0)
*               x [i,j] = 0;
```

Boundary Condition

```
else
    if (W[i] ≤ j)
    {
        x[i, j] = max {x [i–1, j ], x[i–1, j-W [i]]+ P_i }
    }
    else
        x [i, j] = x [i–I , j];
}
```

5) Longest Common Subsequence (LCS) < String Manipulation >

1) String : a group of one/more characters,

Substring

A group of 1/more characters taken in contiguous from the given string,

Subsequence

A group of one/more characters taken from the given string, that may not be contiguous, but however their relative order is maintained

Ex: $\langle A\ B\ C \rangle = 3$

$(1+2+3)$

$\left.\begin{array}{l} \langle A \rangle \\ \langle B \rangle \\ \langle C \rangle \end{array}\right\} 3$

$\left.\begin{array}{l} \langle A\ B \rangle \\ \langle B C \rangle \end{array}\right\} 2$

$\left.\langle A B C \rangle\right\} 1$

Q) Given a string of length n-characters, then the no of substrings possible are $\underline{O(n^2)}$,

1) $\underline{\langle A\ B\ C \rangle}$

✓ $\langle A \rangle$; $\langle B \rangle$; $\langle B \rangle$

✓ $\langle A B \rangle$; $\langle B C \rangle$; $\langle A\ C \rangle$

✓ $\langle A B C \rangle$

$\langle C A \rangle$ ✗

$\langle C B \rangle$ ✗

$\langle A\ B\ C\ D \rangle$

$\langle 1+2+3+4 \rangle$

$\langle \overset{n}{\ldots\ldots\ldots} \rangle$

$$1+2+3+\cdots+n$$

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

$$= O(n^2)$$

String : <C A B D A B B C D>

<B B D> ✓

<B A C> ✓

<C A D C> ✓

<A D A B D> ✓

<C D C A> ✗

Every Substring is also a Subsequence;

Every Subsequence is Not a Substring

Q2) Given a String of length n-characters, the no. of Subsequences possible are $O(\underline{\ 2^n\ })$;

$\longrightarrow$ (Common Subsequence)

THANK - YOU