CS & IT ENGINEERING

Algorithm

Dynamic Programming

Lecture No.- 05

By- Dr. Khaleel Khan sir

Recap of Previous Lecture

Topic Longest Common Subsequence

Topic MCP

Topics to be Covered

Topic — Matrix Chain Product

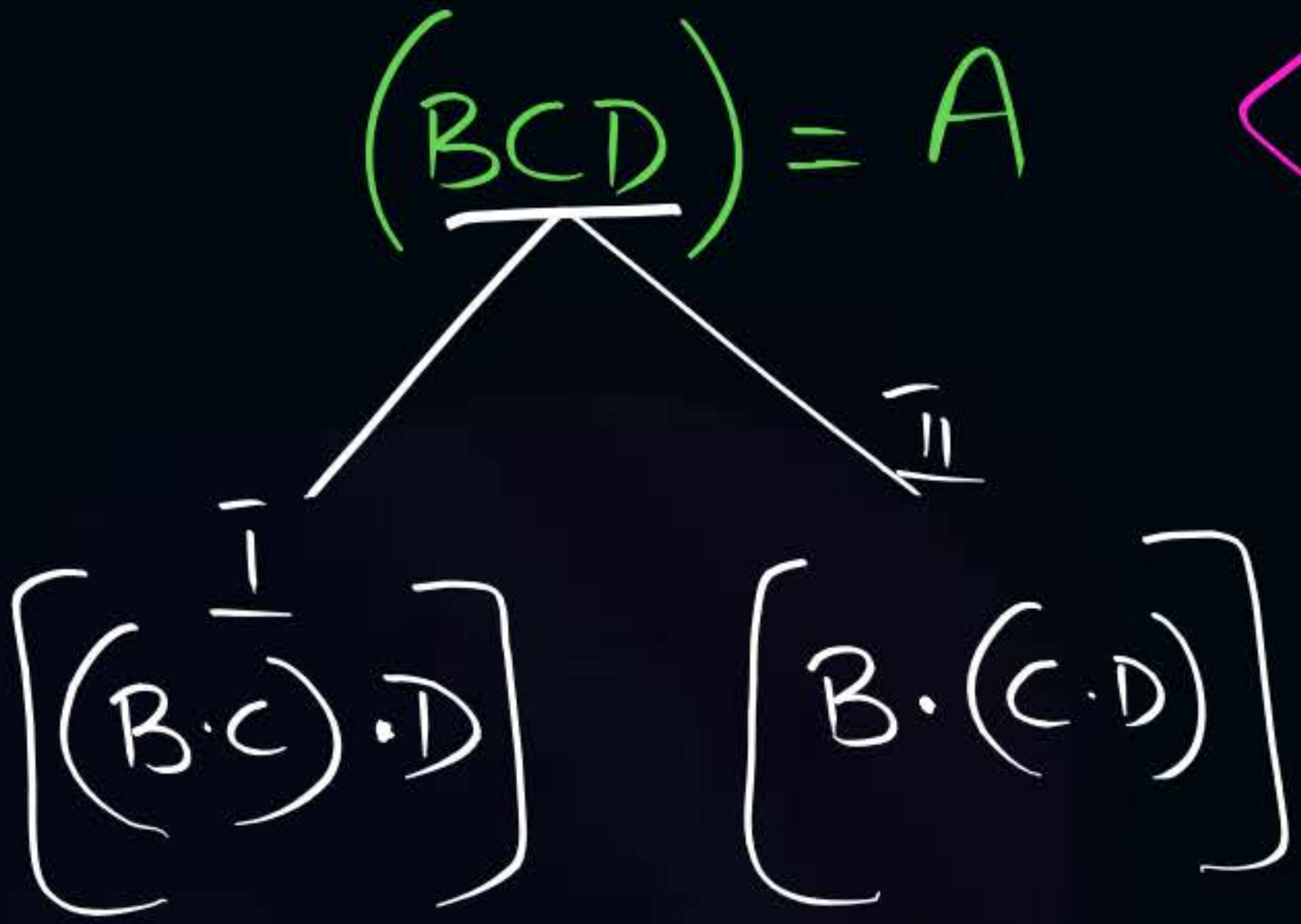
Topic — Sum of Subsets

Topic : Dynamic Programming: (DP)

Matrix chain Product (MCP):

⟨Compatible⟩

$$A_{n \times m} \times B_{m \times k}$$



(Paranthesize)

(No. of Scalar Multiplications)

Defn: Given a chain q Matrices $\langle A_1 A_2 A_3 \ldots A_n \rangle$ where Matrix A_i is q dimension $P_{i-1} \times P_i$

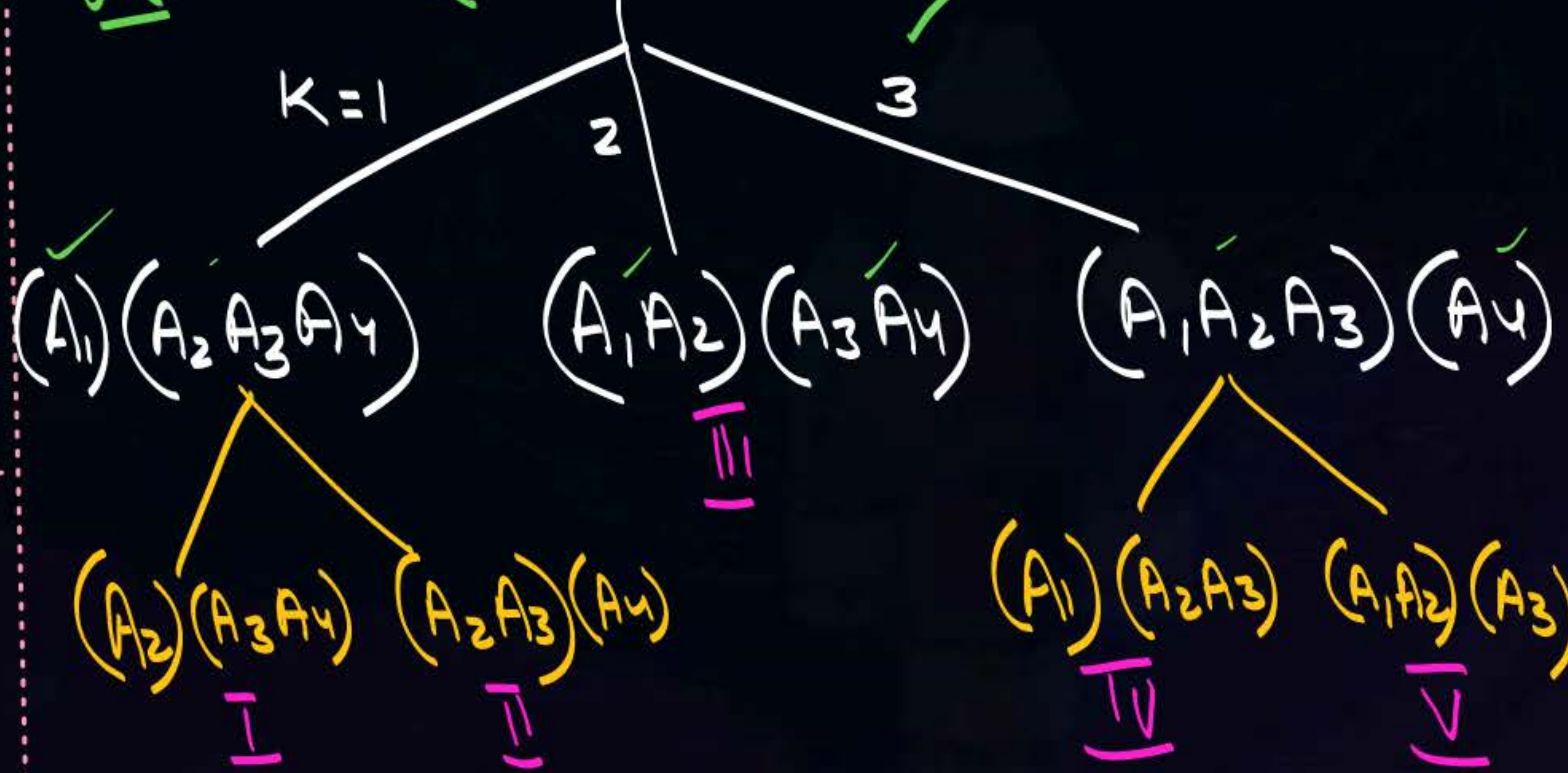
The Problem q M.C.P is to fully Parenthesize the chain, Such that the total no. q Scalar Multiplications is Minimum;

Ex1: $\langle BCD \rangle$

Catalan No: $\frac{1}{(n+1)} {}^{2n}C_n$

$\Omega(2^n)$

Ex2: $\langle A_1 A_2 A_3 A_4 \rangle$



Derivation of D.P based Recurrence for M.C.P

→ Let the resultant Matrix $A_{i..j}$ be of the Product $\langle A_i \cdot A_{i+1} \cdot A_{i+2} \cdots A_j \rangle$

Ex: $A_{1..n} : \langle A_1 A_2 A_3 \cdots A_n \rangle$

$k = 1, 2, 3, \ldots n-1$

→ Any optimal Parenthesization, must Split the chain about the Matrix A_k & A_{k+1} S.T the total No. of Scalar Multiplications is Min.

→ Let $m[i,j]$ denote the no. of Scalar multiplications to get the Matrix $A_{i..j}$

$$\langle A_i A_{i+1} \cdots A_j \rangle = A_{i..j} = m[i,j]$$

$$\langle \underbrace{(A_i A_{i+1} \cdots A_k)}_{} \underbrace{(A_{k+1} \cdots A_j)}_{} \rangle$$

$$(A_{i..k}) \quad (A_{k+1..j})$$

$$[\, i \le k \le j-1 \,]$$

$$\boxed{m[i,j] = \min_{i \le k < j} \left\{ m[i,k] + m[k+1,j] + P_{i-1} * P_k * P_j \right\}}$$

$$m[i,j] = \min_{i \le K < j} \{ m[i,K] + m[K+1,j] + P_{i-1} \times P_K \times P_j \}$$

$$m[i,i] = 0$$

$S[i,j] = K$ ⟨Point of Split⟩

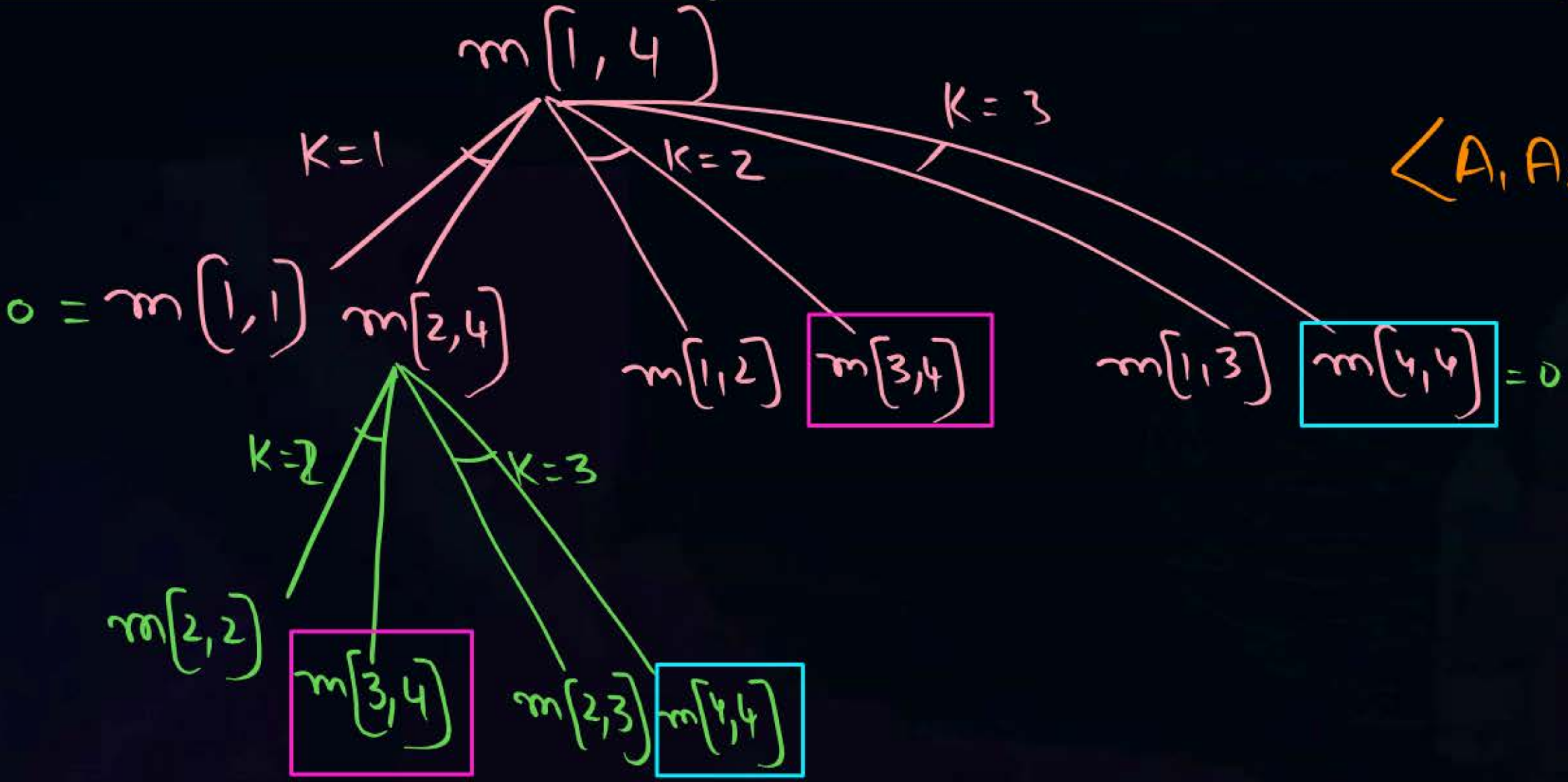
$j - i = 0$

$j - i = 1$

$j - i = 2$

⋮

$\langle A_1 A_2 A_3 A_4 \rangle = A_{1..4}$



$$A_{1..6} = \langle \underbrace{(\overset{A_i}{A_1} A_2 A_3)}_{I}\underbrace{(\overset{A_K}{A_4} A_5 \overset{A_j}{A_6})}_{II} \rangle$$

2×6

$$(A_{1..3}) \leftrightarrow (A_{4..6})$$

$$\frac{2 \times 8}{P_0 \times P_3} \qquad \frac{8 \times 6}{P_3 \times P_6}$$

$$m[1,3] + m[4,6] + \underline{P_0 \cdot P_3 \cdot P_6}$$

$$P_{i-1} * P_K * P_j$$

$A_1 \rightarrow P_0 \times P_1 \rightarrow 2 \times 3$

$A_2 \rightarrow P_1 \times P_2 \rightarrow 3 \times 5$

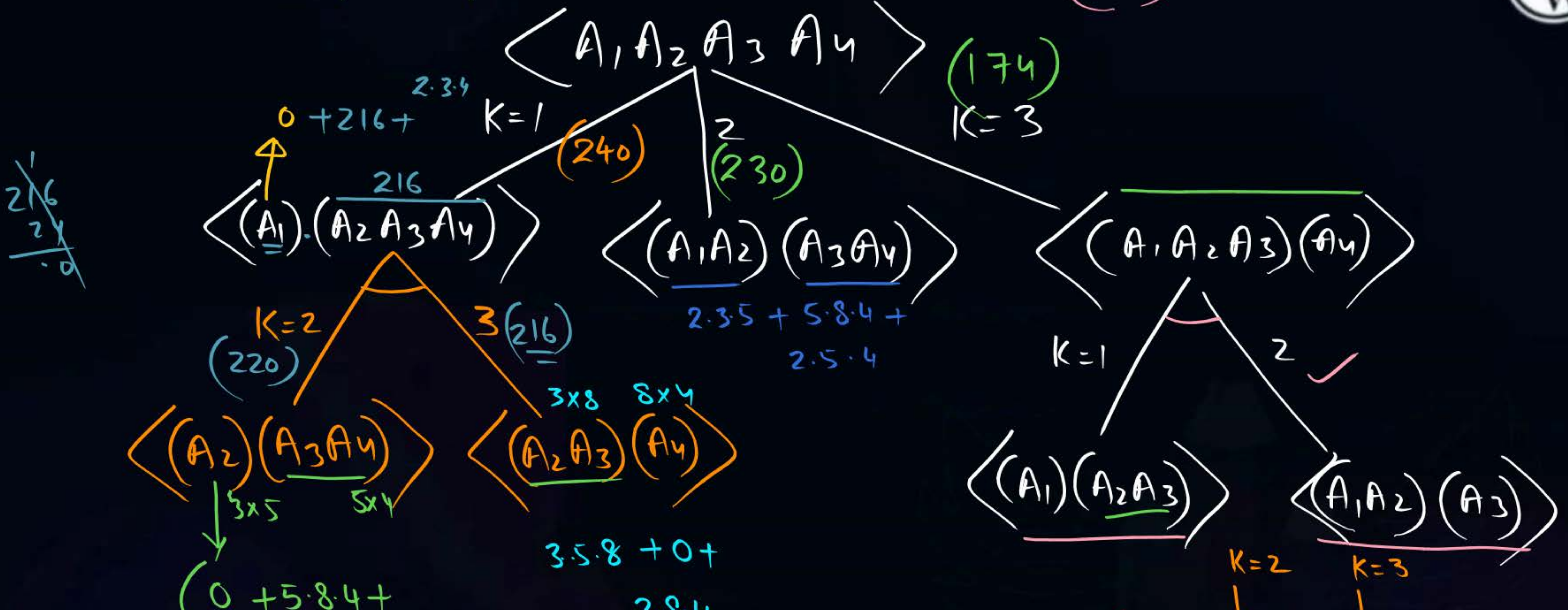
$A_3 \rightarrow P_2 \times P_3 \rightarrow 5 \times 8$

$A_4 \rightarrow P_3 \times P_4 \rightarrow 8 \times 4$

$A_5 \rightarrow P_4 \times P_5 \rightarrow 4 \times 7$

$A_6 \rightarrow P_5 \times P_6 \rightarrow 7 \times 6$

$\langle A_1 A_2 A_3 A_4 \rangle = \langle 2\times3;\ 3\times5;\ \underline{5\times8;\ 8\times4} \rangle = (174)$ ✓



$\langle A_1 A_2 A_3 A_4 \rangle$
(174)
K=3
K=1
(240)
2
(230)
2·3·4
0 +216+
216
$\langle (A_1)\cdot(A_2 A_3 A_4) \rangle$
$\langle (A_1 A_2)(A_3 A_4) \rangle$
2·3·5 + 5·8·4 + 2·5·4
$\langle (A_1 A_2 A_3)(A_4) \rangle$
216
24
K=2
(220)
3 (216)
3×8
8×4
$\langle (A_2)(A_3 A_4) \rangle$
3×5
5×4
$\langle (A_2 A_3)(A_4) \rangle$
(0 +5·8·4+ ·3·5·4)
3·5·8 +0+ 3·8·4
160
60
220
120
96
216
K=1
2 ✓
$\langle (A_1)(A_2 A_3) \rangle$
$\langle (A_1 A_2)(A_3) \rangle$
K=2
K=3

Final Parenthes $= \langle ((A_1 A_2)\cdot(A_3))\cdot(A_4) \rangle$

$30 + 80 + 2\cdot8\cdot4 = \left.\begin{matrix}110\\64\end{matrix}\right] 174$

$$l=4 \quad \frac{m[1,4]}{j-i=3} = \min_{K=1,2,3} \left\{ \begin{array}{l} \underbrace{m[1,1]}_{l=1} + \underbrace{m[2,4]}_{l=3} + P_0 \cdot P_1 \cdot P_4 , \\ \underbrace{m[1,2]}_{l=2} + \underbrace{m[3,4]}_{l=2} + P_0 \cdot P_2 \cdot P_4 , \\ \underbrace{m[1,3]}_{l=3} + \underbrace{m[4,4]}_{l=1} + P_0 \cdot P_3 \cdot P_4 \end{array} \right\}$$

$l=1 \Rightarrow \text{Bound}\ (m[i,i])$

$l=2 \Rightarrow j-i=1$

$l=3 \Rightarrow j-i=2$

$l=4 \Rightarrow j-i=3$

Topic : Matrix Chain Product

Algorithm Matrix-Chain-Product (p)

```
n ← length[p] — 1
for i ← 1 to n
        do m[i, i] ← 0                      } Base Cond.   l=1  j-i=0
n  for l ← 2 to n          l is the chain length.
n    for i ← 1 to n - l + 1
         (j ← i + l - 1)
            m[i, j] ← ∞
n           for k ← i to j - 1     < Pts. of Split >
               { q ← m[i, k] + m[k + 1, j] + $P_{i-1}P_kP_j$
                    if (q < m[i, j] )
                       then m[i, j] ← q
                            s[i, j] ← k
return m and s
```

→ $O(n^3)$

Space: $O(n^2)$

l=2

i = 1 to 3

j = ~~1+2~~ 12

order of Matrices

$P = \langle P_0\ P_1\ P_2 \ldots P_n \rangle$ order

$\langle A_1, A_2 \ldots A_n \rangle$

n = (No. of Matrices in the chain)

l=2

m[1,2] m[2,3] m[3,4]

l=3

m[1,3] m[2,4]

$\langle A_1\ A_2\ A_3\ A_4 \rangle$

n=4

i=2

j=2+2-1

=3

7) Sum of Subsets (SoS):

Defn: Given a Set of n-elements (integers) & also another Element (number) 'M'. The problem of S.O.S is to determine, if there exists a Subset of the given elements whose Sum Equals to 'M';

T/F

Y/N

Ex: $n = 5$; $A: \left\langle \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \\ 10; & 20; & 30; & 40; & 50 \end{array} \right\rangle$; $M = 50$

Brute Force: $O(2^n)$

1) {1, 4}

2) {2, 3}

3) {5}

⟨Decision Problem⟩

Derivation of D.P based recurrence for S.o.S

n; $A\langle \overline{A_1\ A_2 \cdots \underline{A_n}} \rangle$, M; $X:\langle 1 \cdots n \rangle$

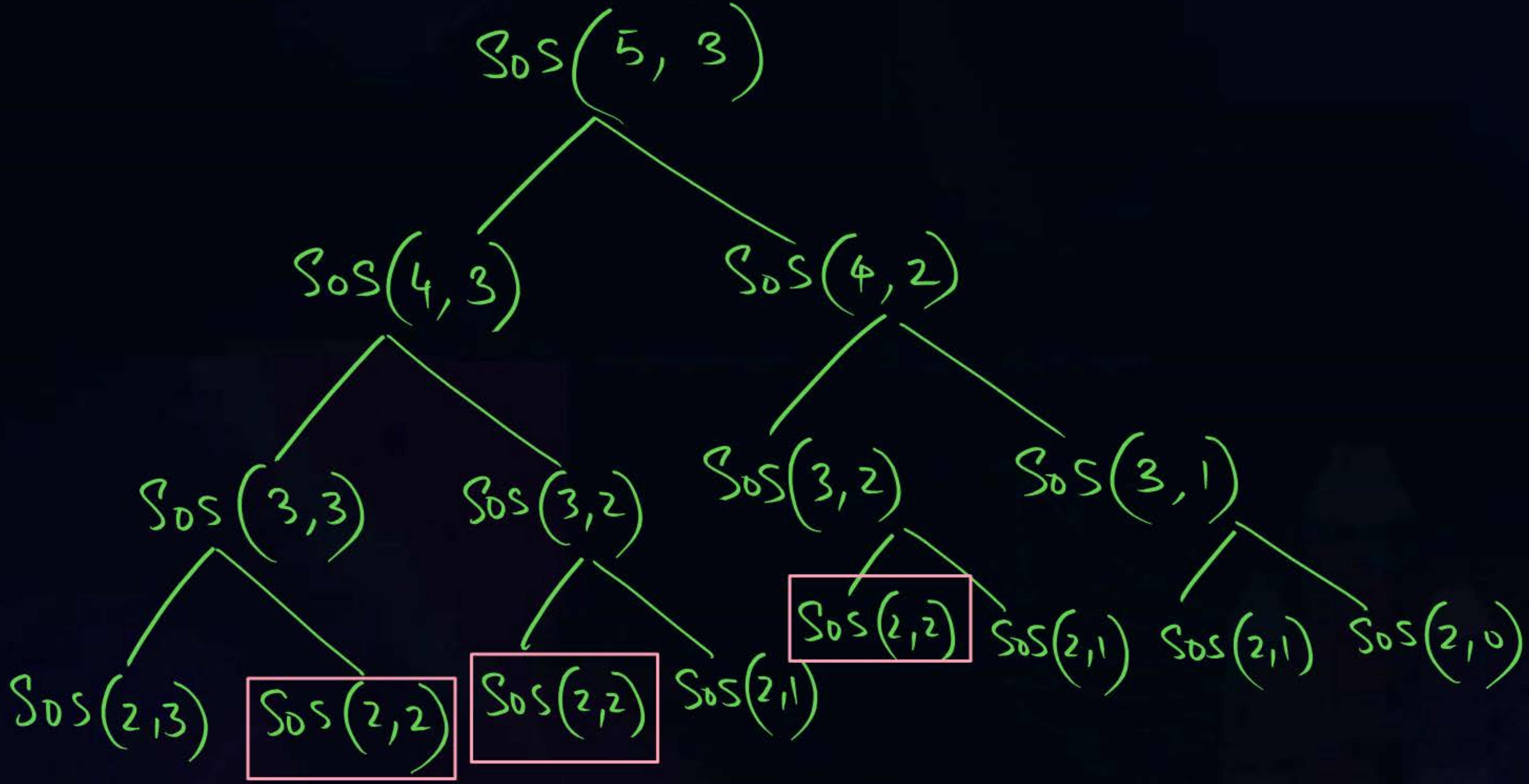
Let $SoS(n, M)$ repr. the soln to S.o.S, with n, numbers & Element M

$SoS(n,M) = T/F$ [Whethere there exists a Subset of given 'n' Elements that sum to M]

$$SoS(n,M) = \begin{cases} SoS(n-1,M), & A_n > M \\ \left[SoS(n-1,M) \text{ or } SoS(n-1, M-A_n) \right], & A_n \leq M \\ F, & n=0,\ M>0 \\ T, & n \geq 0;\ M=0 \end{cases}$$

Base Cond

1) $n = 5$; $A: \langle 1, 1, 1, 1, 1 \rangle$; $M = 3$



$n=5$; $M=6$; A: $\langle 2,8,4,11,9 \rangle$

$X\langle 0..5,\ 0..6 \rangle$

$X[0..n,\ 0..M]$

$X[\underline{n}, \underline{M}]$

$X[i,j] = T/F$

<Whether there exists a subset from the given 'i' elements, whose sum is 'j'>

A_i	i \ j	0	1	2	3	4	5	6
	0	T	F	F	F	F	F	F
2	1	T	F	T	F	F	F	F
8	2	T	F	T	F	F	F	F
4	3	T	F	T	F	T	F	T
11	4	T	F	T		T		
9	5	T	F	T		T		

Topic : Sum of Subsets

if M is very large like 2^n

$\therefore$ Time: $O(n \cdot 2^n)$

SOS can be implemented using bottom-up DP with Tabulation:

Time: $O(n * M)$

Space: $O(n * M)$

```
Algorithm SOS (n , M, A)
            A [1.....n]
{
  integer X[0..n , 0..M];

1.    for i ← 0 to n
        for j ← 0 to M
          if (i ≥ 0 and j = 0)
            X [i , j]= T
          else
            if (i = 0 and j > 0)
             X [i, j] = F;
            else
              if (A [i] > j )
                X [i,j] = X [i –1, j]
              else
                X [i, j] = X [i–1, j] V X [i–1, j –A [i] ]
}
```

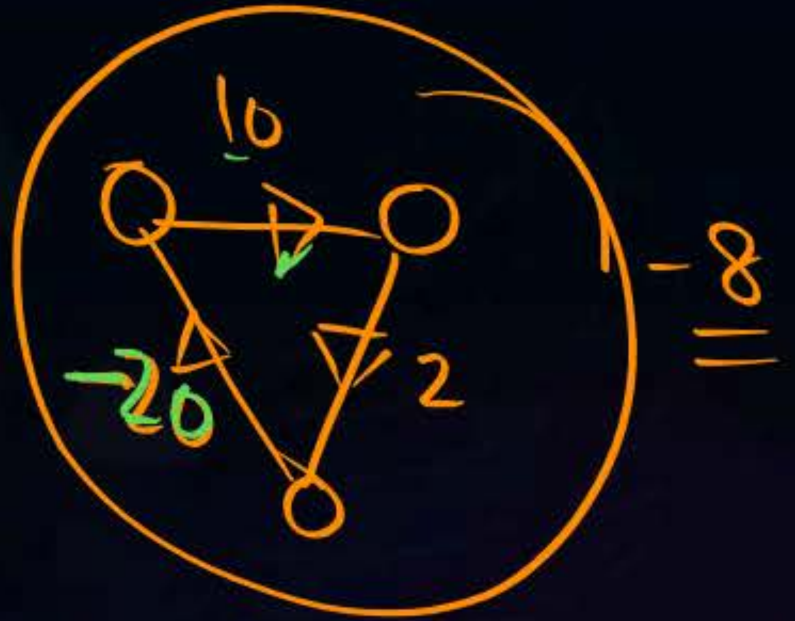
Base

Slide 6

8) Bellman-Ford Algorithm [Single Source Shortest Paths]

(i) Dijkstra's Algo [Greedy Method] always works, provided the graph has +ve wt. edges;

(ii) If the graph has -ve wt. edges but no -ve wt. cycle then Dijkstra's Algo. MAY|MAY NOT work;



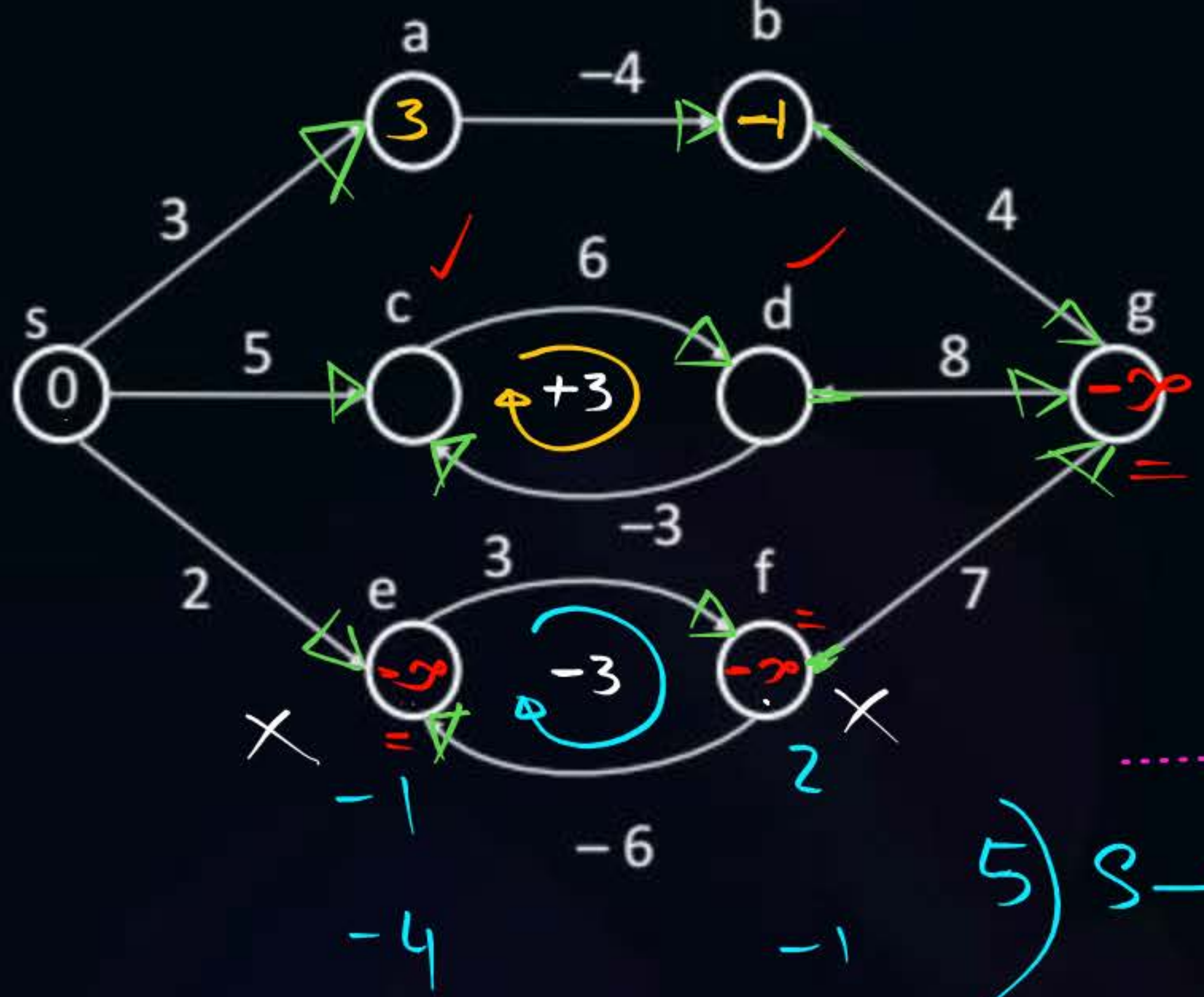
(iii) If the graph has -ve wt. edges & no -ve wt. cycle then Bellman-Ford Algo always works correctly

⟨D.P⟩

(Temperature)

(iv) If the graph has -ve wt. cycle reachable from Source then NO-ALGO works;

Bellman Ford Algorithm



1) S → a = 3 ✓

2) S → b = -1 ✓

3) S → g = 3 (?) ✗

4) S → c = $\boxed{5}$ ✓

S → c → d → c = 8

= 11

= 14

(S → c → D) = $\boxed{11}$ ✓

(S → c → d → c → d) = 14

= 17

= 20

5) S → e = 2

= -1

= -4

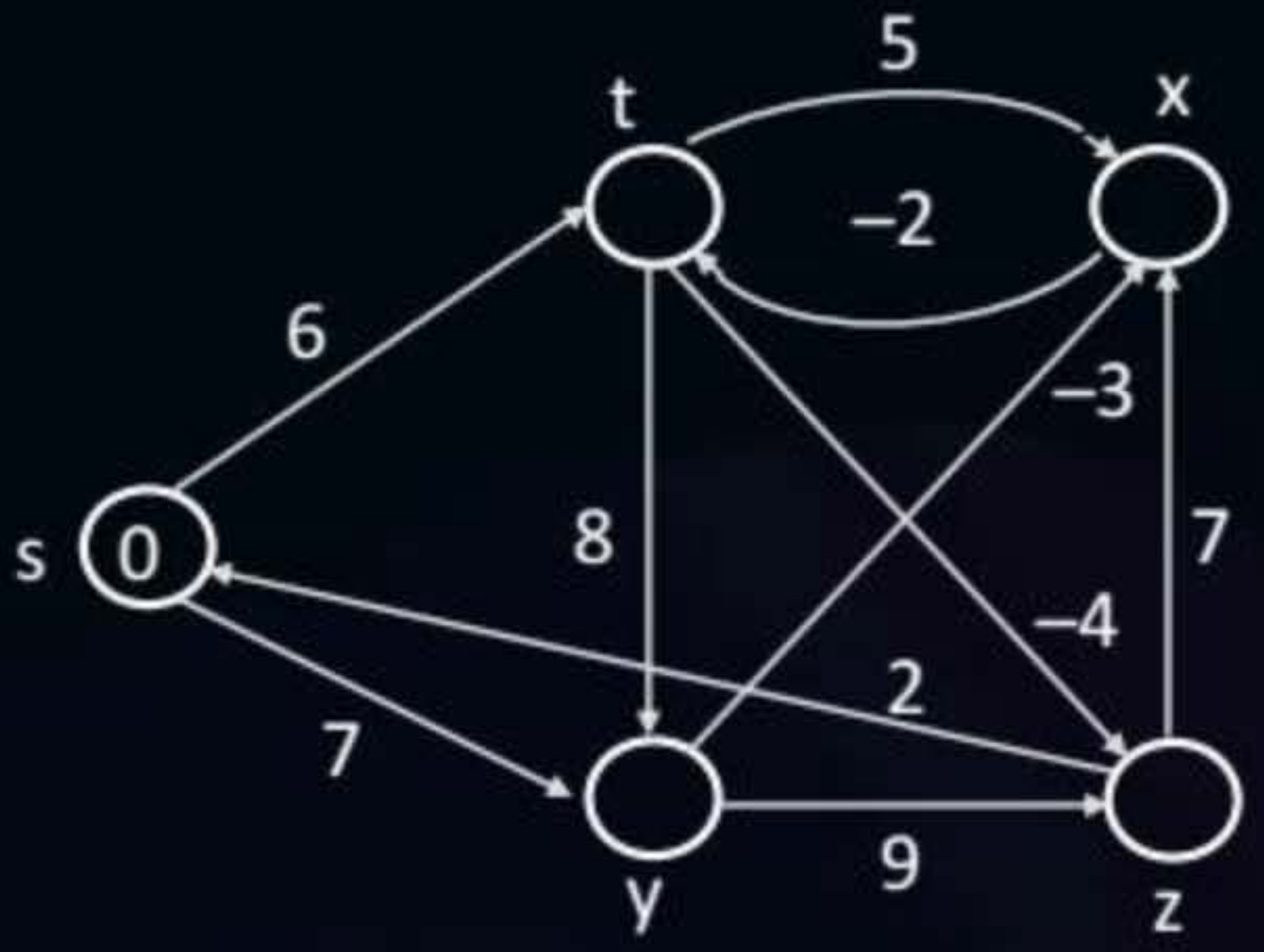
6) S → e → f = 5

= 2

= -1

Slide 7

Bellman Ford Algo



1) $(t \to x)$

2) $(t \to z \to x)$

Apply Dijkstra's S.S.S.P Algo

$v_0 = s$;

(d-values)

	s	t	x	y	z
{s}	–	✓	✓	✓	✓

Slide 9

THANK - YOU