



AKTU

B.Tech 5th Sem



CS IT & CS Allied

DAA : Design & Analysis Of Algorithm

ONE SHOT Revision

(Crash Course)

Unit-4 : Dynamic Programming

Backtrack

Branch-and-Bound



By Dr. Nidhi Parashar Ma'am

- M.Tech Gold Medalist
- Net Qualified



AKTU

B.Tech 5th Sem



CS IT & CS Allied

DAA : Design & Analysis Of Algorithm

Unit-4 : Lecture-1

Today's Target

- Introduction to dynamic programming ✓
 - Principle of Optimality
- Optimal Resource Allocation Problem
- AKTU PYQs



By Dr. Nidhi Parashar Ma'am

- M.Tech Gold Medalist
- Net Qualified

(BCS-503 - Design and Analysis of Algorithm)

AKTU : Syllabus

Unit-IV : Dynamic Programming

Dynamic Programming with Examples Such as Knapsack. All Pair Shortest Paths – Warshal's and Floyd's Algorithms, Resource Allocation Problem. Backtracking, Branch and Bound with Examples Such as Travelling Salesman Problem, Graph Coloring, n-Queen Problem, Hamiltonian Cycles and Sum of Subsets.

Dynamic Programming

A mathematical technique that deals with optimization of multistage decision problems.

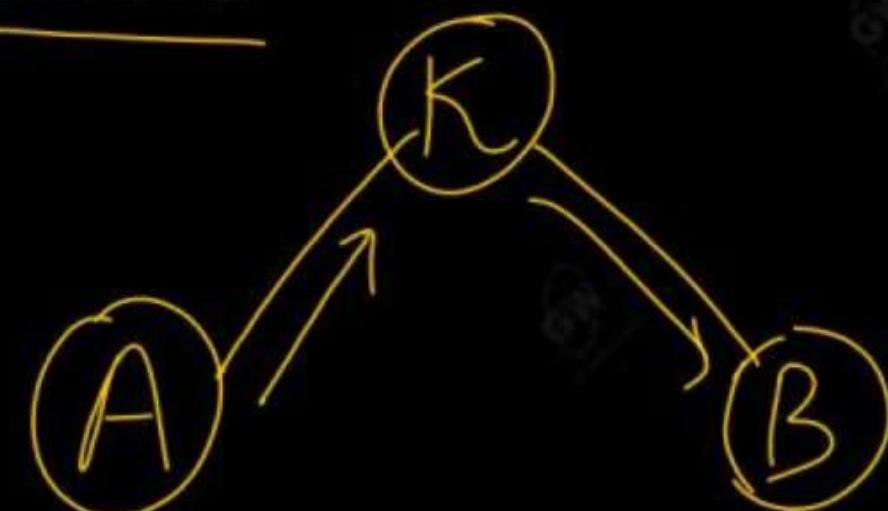
Formal Procedure :-

- (1) Define problem variables, objective function and set up the constraints.
- (2) Determine stages of the problem.
- (3) Develop the recursive method.
- (4) Make tabular representation
- (5) Find optimal decision at each stage and then overall optimal

① ②

Iteration

Recursion

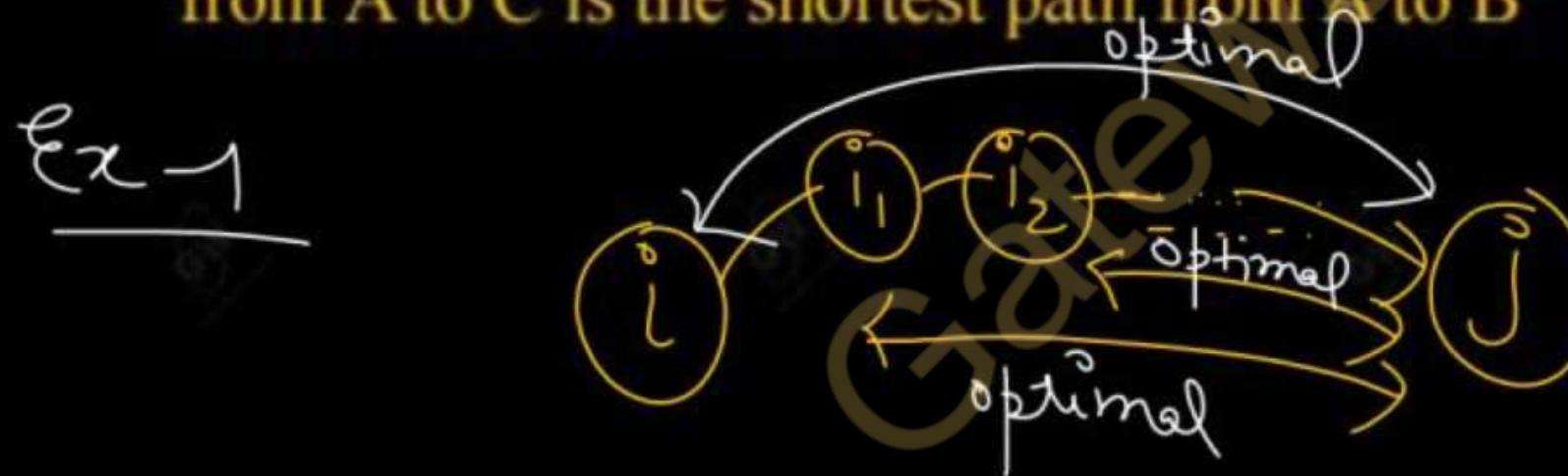


Principle of optimality (Imp)

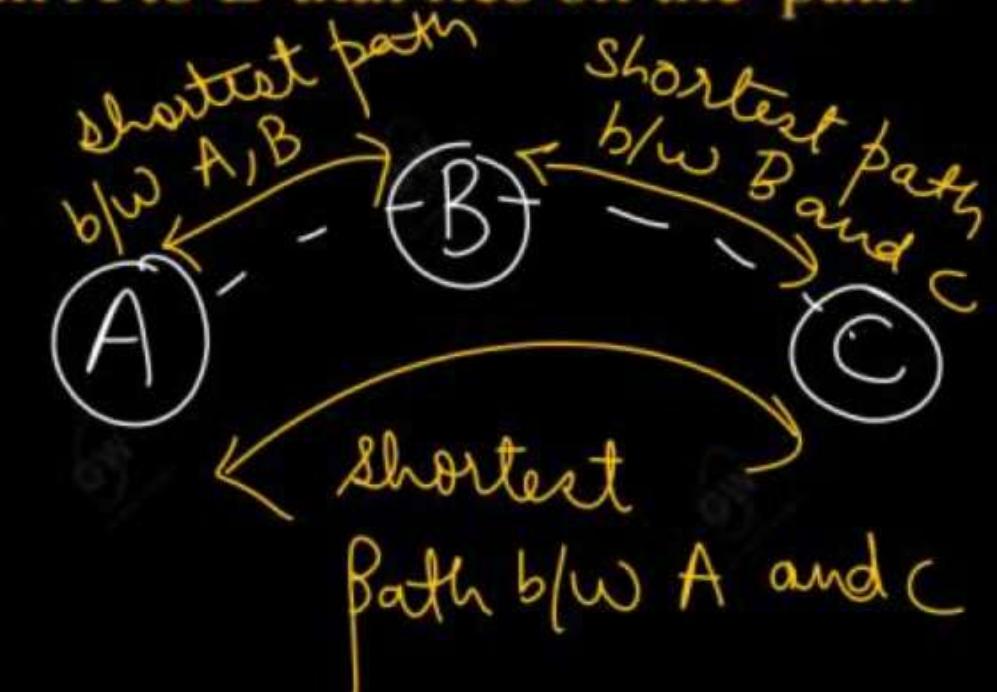
Suppose that in solving a problem, we have to make a sequence of decisions D_1, D_2, \dots, D_n

- If this sequence is optimal, then the last k decisions, $1 \leq k \leq n$ must be optimal.
- Ex-1 ➤ In the shortest path problem, If i, i_1, i_2, \dots, j is a shortest path from i to j , then i_1, i_2, \dots, j must be a shortest path from i to j
- Ex-2 ➤ To Search for best path: Dynamic programming principle (Principle of Optimality) says.....

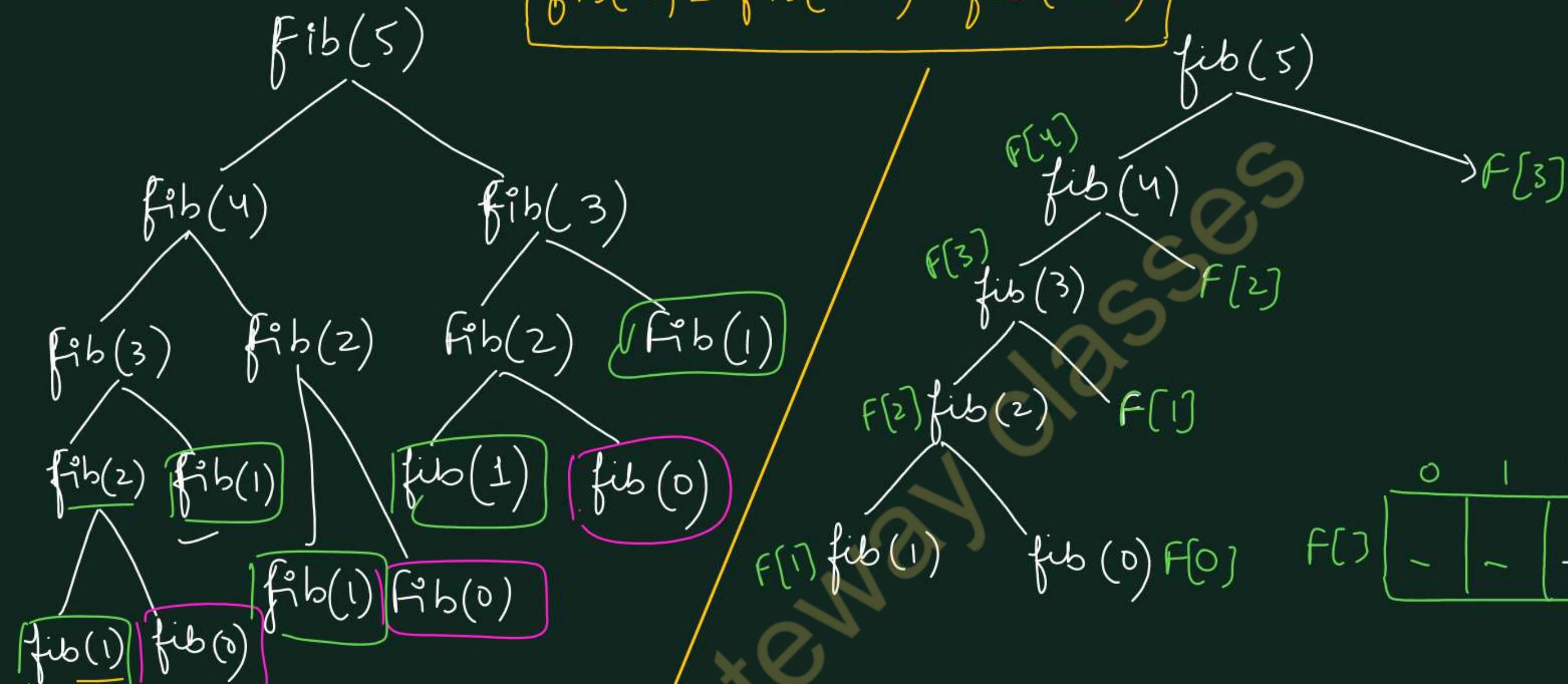
“If B is on the shortest path from A to C , then the path from A to B that lies on the path from A to C is the shortest path from A to B ”



Ex-2



$$\boxed{\text{fib}(n) = \text{fib}(n-1) + \text{fib}(n-2)}$$



divide and conquer
(Recursive Solution)

dynamic programming

0	1	2	3	4	5
-	-	-	-	-	-

the main

(1) Break problem into subproblems and each subproblem is referred to as a stage.

(2) Variables that specify the condition of decision process and summarize the current ‘status’ of system are called state variable.

Optimal Resource Allocation using Dynamic Programming

Problem: - An owner of a chain of four grocery stores purchased six crates of strawberries. Find allocation of six crates so as to maximize the profits. The owner does not wish to split crates between stores, but it is willing to make zero allocation.

Maximization Problem

6

No. of boxes	Estimated Profiles			
	Store 1	Store 2	Store 3	Store 4
0	0	0	0	0
1	4	2	6	2
2	6	4	8	3
3	7	6	8	4
4	7	8	8	4
5	7	9	8	4
6	7	10	8	4

Solution :-

Stage - 1 : → Store - 1

Stage - 2 : → store - 1 and store 2

Stage - 3 : → store 1, store 2, and store 3

Stage - 4 : → store 1, store 2, store 3, and store 4

Let $f_1(x_1), f_2(x_2), f_3(x_3), f_4(x_4)$ are respective profit from four stores respectively.

Given, x_1, x_2, x_3, x_4 are the no. of crates to be allocated to four stores respectively.

Mathematical Problem Formulation

$$\text{Maximize } z = [f_1(x_1) + f_2(x_2) + f_3(x_3) + f_4(x_4)]$$

Subjected to, $x_1 + x_2 + x_3 + x_4 \leq 6$,

Where $x_1, x_2, x_3, x_4 \geq 0$

x_1, x_2, x_3 and x_4 can not be fraction.

Solution :-

Stage - 1 (Store 1 is considered)

No. of boxes	0	1	2	3	4	5	6
Profit $f_1(x_1)$	0	4	6	7	7	7	7

Stage - 2 (store 1 and store 2 are considered)

Stage 1
table

Store	x_1	0	1	2	3	4	5	6
1	$f_1(x_1)$	0	4	6	7	7	7	7

Store 2

x_2

$$\text{Max} \left\{ f_1(x_1) + f_2(x_2) \right\}$$

store 2

0	0	0	4	6	7	7	7	7
1	2	2	6	8	9	9	9	9
2	4	4	8	10	11	11		
3	6	6	10	12	13			
4	8	8	12	14				
5	9	9	13					
6	10	10						

INVALID

Stage - 3 : (store 1 + store 2) + store 3 are considered.

optimal result of stage 2

No. of boxes	0	1	2	3	4	5	6
Max. of $f_1(x_1) + f_2(x_2)$	0	4	6	8	10	12	14
Boxes in store 1 + store 2	0+0	1+0	2+0 1+1	2+1 1+2	2+2 1+3	2+3 <u>1+4</u>	2+4
Store 3							
x_3	$f_3(x_3)$						
0	0	0	4	6	8	10	12
1	6	6	10	12	14	16	18
2	8	8	12	14	16	18	
3	8	8	12	14	16		
4	8	8	12	14			
5	8	8	12				
6	8	8					

Max. of $[f_1(x_1) + f_2(x_2)] + f_3(x_3)$

Stage - 4: $(\text{store 1} + \text{store 2} + \text{store 3}) + \text{store 4}$ is considered.

No. of boxes	0	1	2	3	4	5	6
Max. of $f_1(x_1) + f_2(x_2) + f_3(x_3)$	0	6	10	12	14	16	18
Boxes in store 1 + store 2 + store 3 <i>optimal solution of stage 3</i>	0+0+0	0+0+1	1+0+1	2+0+1 1+1+1 1+0+2	2+1+1 1+2+1 1+1+2	2+2+1 1+3+1 2+1+2 1+2+2	2+3+1 1+4+1 2+2+2 1+3+2
Boxes in store 4	6	5	4	3	2	1	0
$f_4(x_4)$	4	4	4	4	3	2	0
$f_1(x_1) + f_2(x_2) +$ $f_3(x_3) + f_4(x_4)$	4	10	14	16	17	18	18

Max Profit

Possible optimal allocation

Store 1	Store 2	Store 3	Store 4
2	2	1	1
1	3	1	1
2	1	2	1
1	2	2	1
2	3	1	0
1	4	1	0
2	2	2	0
1	3	2	0

optimal no. of crates to be allocated for Max Profit = 18

1. Define the principle of optimality. (AKTU 2022-23)

2. How dynamic programming is different from the greedy approach. (AKTU)



AKTU

B.Tech 5th Sem



CS IT & CS Allied

DAA : Design & Analysis Of Algorithm

Unit-4 : Lecture-2

Today's Target

- Finding All Pair Shortest path
- Floyd Warshall's Algorithm
- AKTU PYQs



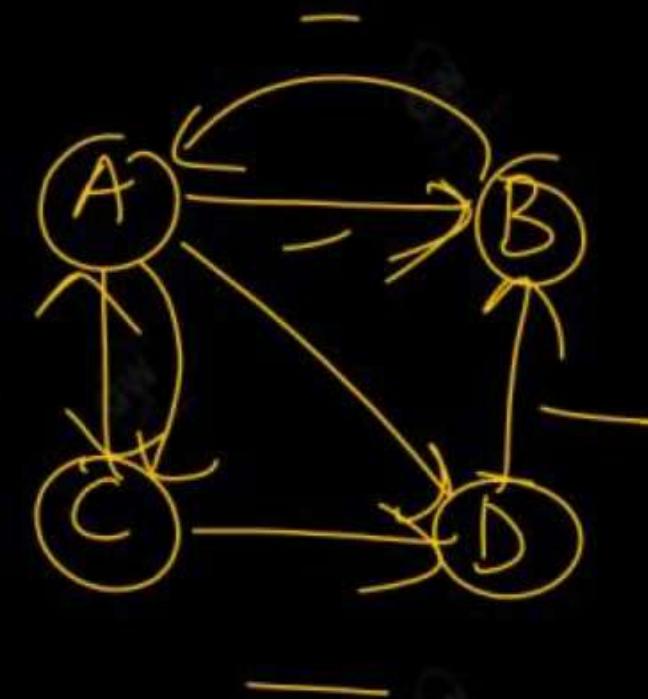
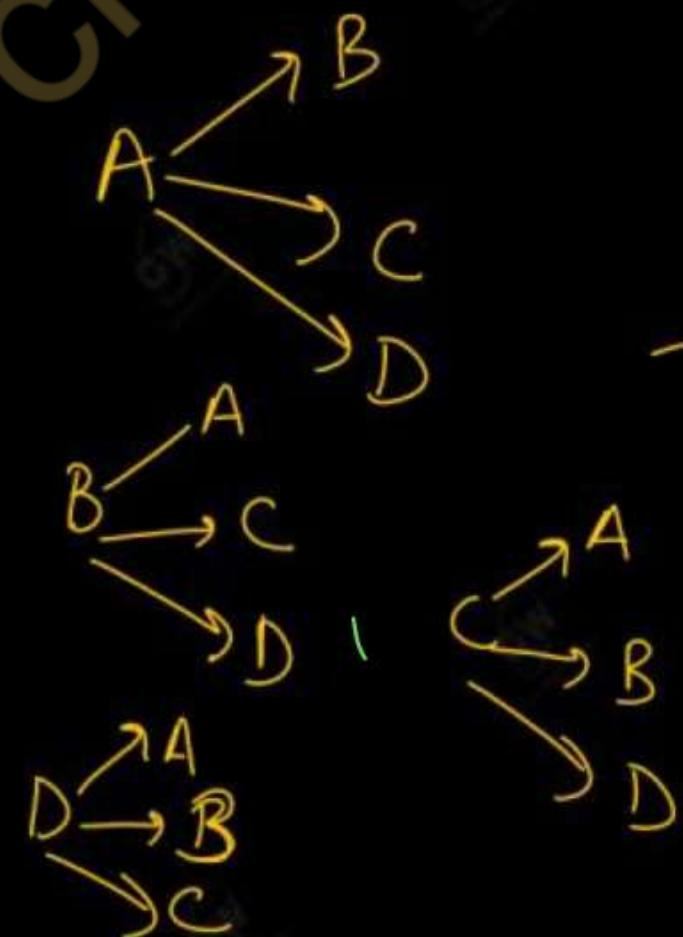
By Dr. Nidhi Parashar Ma'am

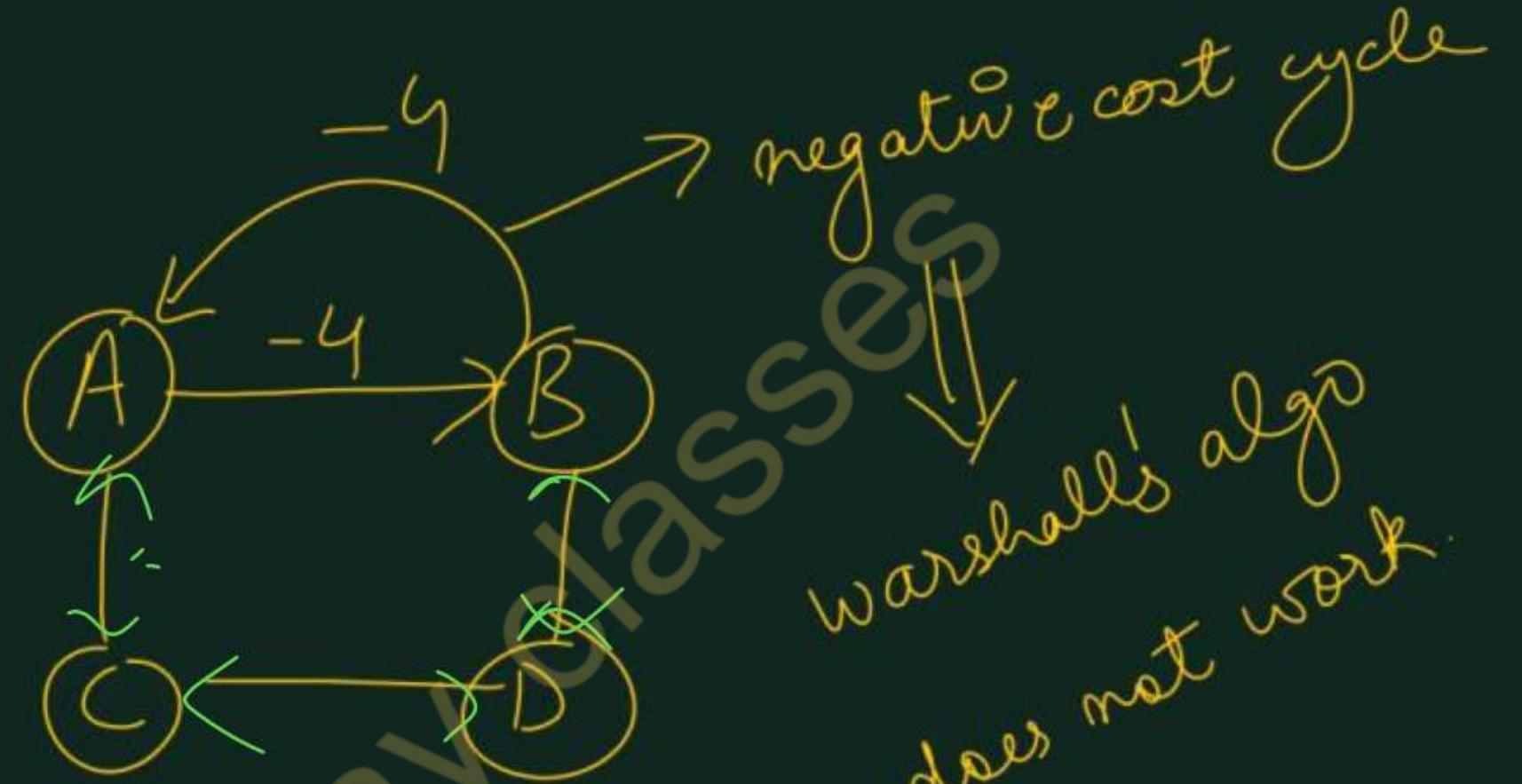
- M.Tech Gold Medalist
- Net Qualified

Floyd Warshall Algorithm

- It is a dynamic programming based algorithm so it breaks the problem into Smaller subproblems and then solves each sub problem. After that it combines the result to solve bigger problem.
- Also called as Floyd's Algo.
- It is all - pair shortest path algorithm used to find the shortest path or shortest distance between every pair of Vertices in the given graph.
- Applicable to both directed / undirected graph.
- It fails if the graph have negative cost cycle.

but it works correctly on
a graph with -ve cost edges.





Algorithm

- Make a matrix for the given input graphs.
- Create another solution matrix as the input graph Matrix.
- Update the solution Matrix while considering all vertices as an intermediate vertex.
- Choose each vertex one by one and update all the shortest paths that includes the selected vertex as an intermediate vertex in the as an intermediate vertex in the shortest path.

Pseudocode

Floyd-Warshall(w)

n = no. of rows in w

D⁰ = w

for k = 1 to n // for n number of nodes (one vertex chosen as intermediate)
 for i = 1 to n // rows

for j = 1 to n // cols

$$D^K(i, j) = \min(D^{k-1}[i, k] + D^{k-1}[k, j])$$

return Dⁿ

$$D^{K-1}(i, j) \}$$

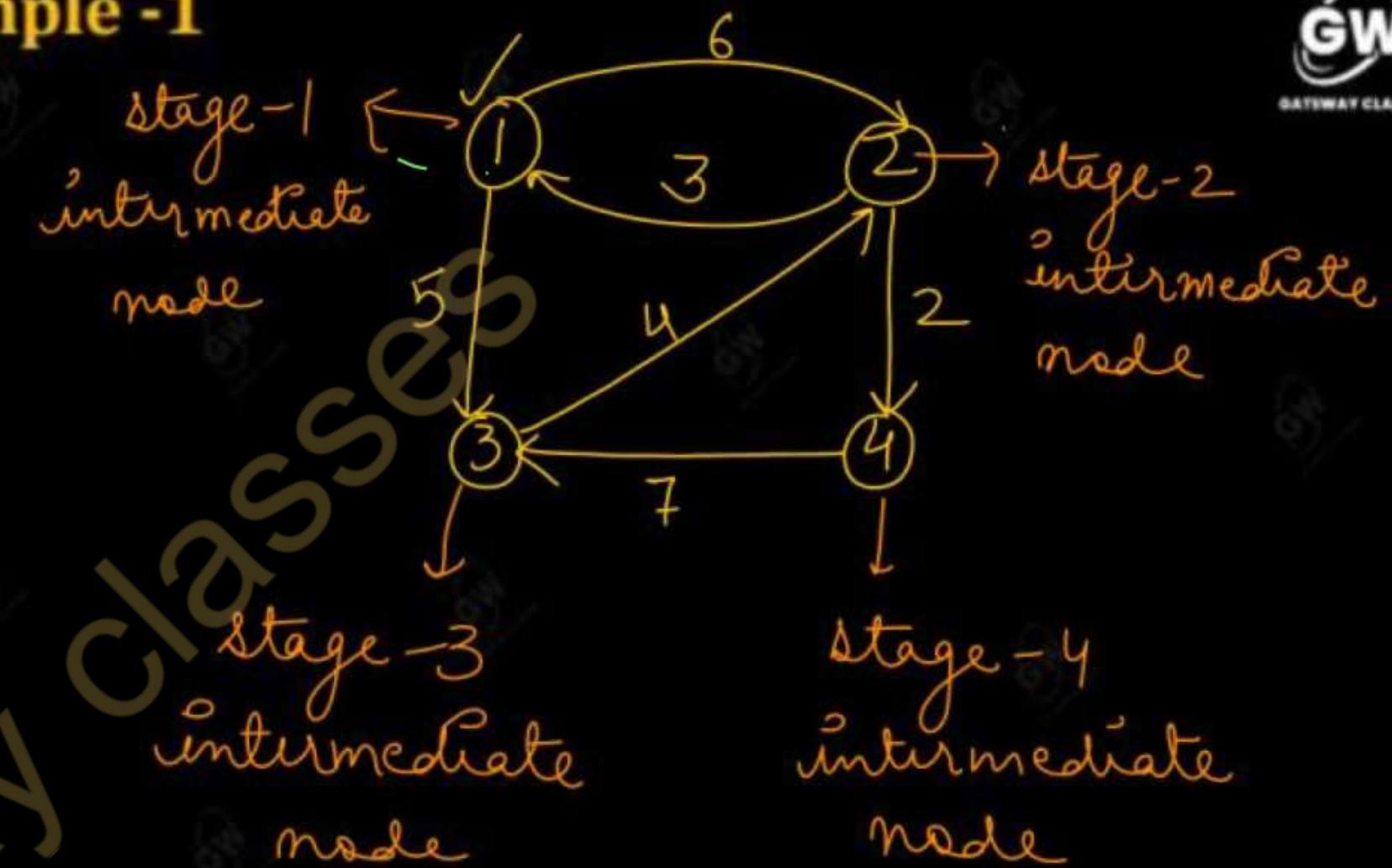
$$T(n) = O(n^3)$$

Example - 1

$$\omega = \begin{matrix} & \begin{matrix} 1 & 2 & 3 & 4 \end{matrix} \\ \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \end{matrix} & \left[\begin{array}{cccc} 0 & 6 & 5 & \infty \\ 3 & 0 & \infty & 2 \\ \infty & 4 & 0 & \infty \\ \infty & \infty & 7 & 0 \end{array} \right] \end{matrix}$$

Intermediate node

$$D = \begin{matrix} & \begin{matrix} 1 & 2 & 3 & 4 \end{matrix} \\ \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \end{matrix} & \left[\begin{array}{cccc} 0 & 6 & 5 & \infty \\ 3 & 0 & \infty & 2 \\ \infty & 4 & 0 & \infty \\ \infty & \infty & 7 & 0 \end{array} \right] \end{matrix}$$



$$D^k(i,j) = \min \left\{ \underbrace{D^{k-1}(i,j)}_{\text{existing cost}}, \underbrace{(D^{k-1}(i,k) + D^{k-1}(k,j))}_{\text{cost via } k} \right\}$$

Example - 1

$$D^1(2,3) = \min \{ D^0(2,3), D^0(2,1) + D^0(1,3) \} = \min \{ \infty, 3 + 5 \} = 8$$

$$D^1(2,4) = \min \{ D^0(2,4), D^0(2,1) + D^0(1,4) \} = \min \{ 2, 3 + \infty \} = 2$$

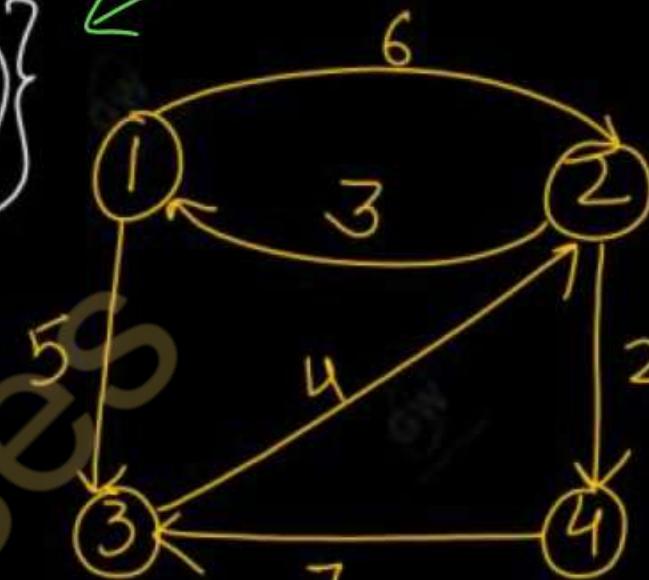
$$D^1(3,2) = \min \{ D^0(3,2), D^0(3,1) + D^0(1,2) \} = \min \{ 4, \infty + 6 \} = 4$$

$$D^1(3,4) = \min \{ D^0(3,4), D^0(3,1) + D^0(1,4) \} = \min \{ \infty, \infty + \infty \} = \infty$$

$$D^1(4,2) = \min \{ D^0(4,2), D^0(4,1) + D^0(1,2) \} = \min \{ \infty, \infty + 6 \} = \infty$$

$$D^1(4,3) = \min \{ D^0(4,3), D^0(4,1) + D^0(1,3) \} = \min \{ 7, \infty + 5 \} = 7$$

optimal solution of
stage 1



D^0	1	2	3	4
1	0	6	5	∞
2	3	0	∞	2
3	∞	4	0	∞
4	∞	∞	7	0

D^1	1	2	3	4
1	0	6	5	∞
2	3	0	8	2
3	∞	4	0	∞
4	∞	∞	7	0

Example -1

$$D^2(1,3) = \min \{ D^1(1,3), D^1(1,2) + D^1(2,3) \} = \min \{ 5, 6+8 \} = 5$$

$$D^2(1,4) = \min \{ D^1(1,4), D^1(1,2) + D^1(2,4) \} = \min \{ \infty, 6+2 \} = 8$$

$$D^2(3,1) = \min \{ D^1(3,1), D^1(3,2) + D^1(2,1) \} = \min \{ \infty, 4+3 \} = 7$$

$$D^2(3,4) = \min \{ D^1(3,4), D^1(3,2) + D^1(2,4) \} = \min \{ \infty, 4+2 \} = 6$$

$$D^2(4,1) = \min \{ D^1(4,1), D^1(4,2) + D^1(2,1) \} = \min \{ \infty, \infty + 3 \} = \infty$$

$$D^2(4,3) = \min \{ D^1(4,3), D^1(4,2) + D^1(2,3) \} = \min \{ 7, \infty + 8 \} = 7$$

	1	2	3	4
1	0	6	5	∞
2	3	0	8	2
3	∞	4	0	∞
4	∞	∞	7	0

	1	2	3	4
1	0	6	5	8
2	3	0	8	2
3	7	4	0	6
4	∞	∞	7	0

Optimal solution of stage-2

Example -1

$$D^3(1,2) = \min\{D^2(1,2), D^2(1,3) + D^2(3,2)\} = \min\{6, 5+4\} = 6 \quad (6)$$

$$D^3(1,4) = \min\{D^2(1,4), D^2(1,3) + D^2(3,4)\} = \min\{8, 5+6\} = 8 \quad (8)$$

$$D^3(2,1) = \min\{D^2(2,1), D^2(2,3) + D^2(3,1)\} = \min\{3, 8+7\} = 3 \quad (3)$$

$$D^3(2,4) = \min\{D^2(2,4), D^2(2,3) + D^2(3,4)\} = \min\{2, 8+6\} = 2$$

$$D^3(4,1) = \min\{D^2(4,1), D^2(4,3) + D^2(3,1)\} = \min\{\infty, 7+7\} = 14 \quad (14)$$

$$D^3(4,2) = \min\{D^2(4,2), D^2(4,3) + D^2(3,2)\} = \min\{\infty, 7+4\} = 11 \quad (11)$$

$$D^2 = \begin{matrix} & \begin{matrix} 1 & 2 & 3 & 4 \end{matrix} \\ \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \end{matrix} & \begin{bmatrix} 0 & 6 & 5 & 8 \\ 3 & 0 & 8 & 2 \\ 7 & 4 & 0 & 6 \\ \infty & \infty & 7 & 0 \end{bmatrix} \end{matrix}$$

$$D^3 = \begin{matrix} & \begin{matrix} 1 & 2 & 3 & 4 \end{matrix} \\ \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \end{matrix} & \begin{bmatrix} 0 & 6 & 5 & 8 \\ 3 & 0 & 8 & 2 \\ 7 & 4 & 0 & 6 \\ 14 & 11 & 7 & 0 \end{bmatrix} \end{matrix}$$

optimal solution till stage-3

Example -1

$$D^4(1,2) = \min\{D^3(1,2), D^3(1,4) + D^3(4,2)\} = \min\{6, 8 + 11\} = 6$$

$$D^4(1,3) = \min\{D^3(1,3), D^3(1,4) + D^3(4,3)\} = \min\{5, 8 + 7\} = 5$$

$$D^4(2,1) = \min\{D^3(2,1), D^3(2,4) + D^3(4,1)\} = \min\{3, 2 + 14\} = 3$$

$$D^4(2,3) = \min\{D^3(2,3), D^3(2,4) + D^3(4,3)\} = \min\{8, 2 + 7\} = 8$$

$$D^4(3,1) = \min\{D^3(3,1), D^3(3,4) + D^3(4,1)\} = \min\{7, 6 + 14\} = 7$$

$$D^4(3,2) = \min\{D^3(3,2), D^3(3,4) + D^3(4,2)\} = \min\{4, 6 + 11\} = 4$$

$$D^3 = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 0 & 6 & 5 & 8 \\ 3 & 0 & 8 & 2 \\ 7 & 4 & 0 & 6 \\ 14 & 11 & 7 & 0 \end{bmatrix}$$

$$D^4 = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 0 & 6 & 5 & 8 \\ 3 & 0 & 8 & 2 \\ 7 & 4 & 0 & 6 \\ 14 & 11 & 7 & 0 \end{bmatrix}$$

final solution

Example -2

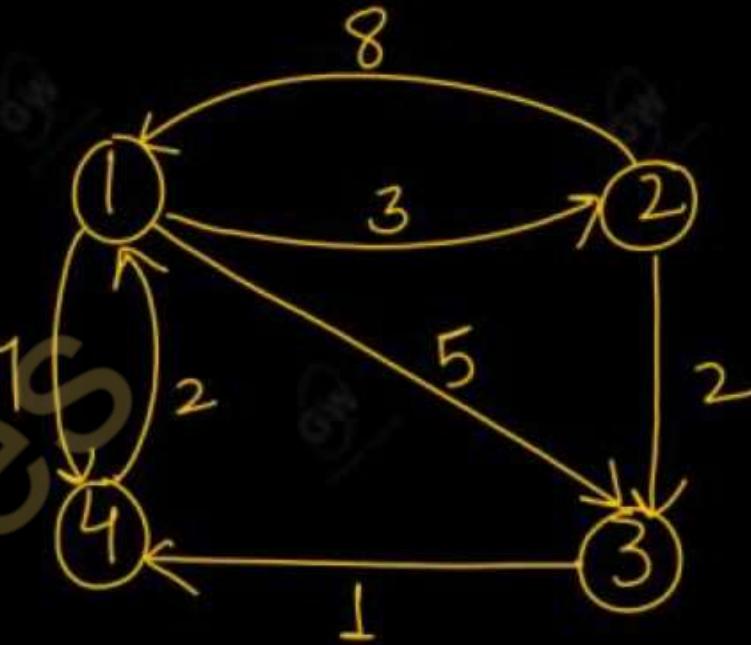
Solution —

$$D^1 = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 0 & 3 & \infty & 7 \\ 2 & 8 & 0 & 2 \\ 3 & 5 & 8 & 0 \\ 4 & 2 & 5 & \infty \end{bmatrix}$$

$$D^2 = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 0 & 3 & 5 & 7 \\ 2 & 8 & 0 & 2 \\ 3 & 5 & 8 & 0 \\ 4 & 2 & 5 & 7 \end{bmatrix}$$

$$D^3 = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 0 & 3 & 5 & 6 \\ 2 & 7 & 0 & 2 \\ 3 & 5 & 8 & 0 \\ 4 & 2 & 5 & 7 \end{bmatrix}$$

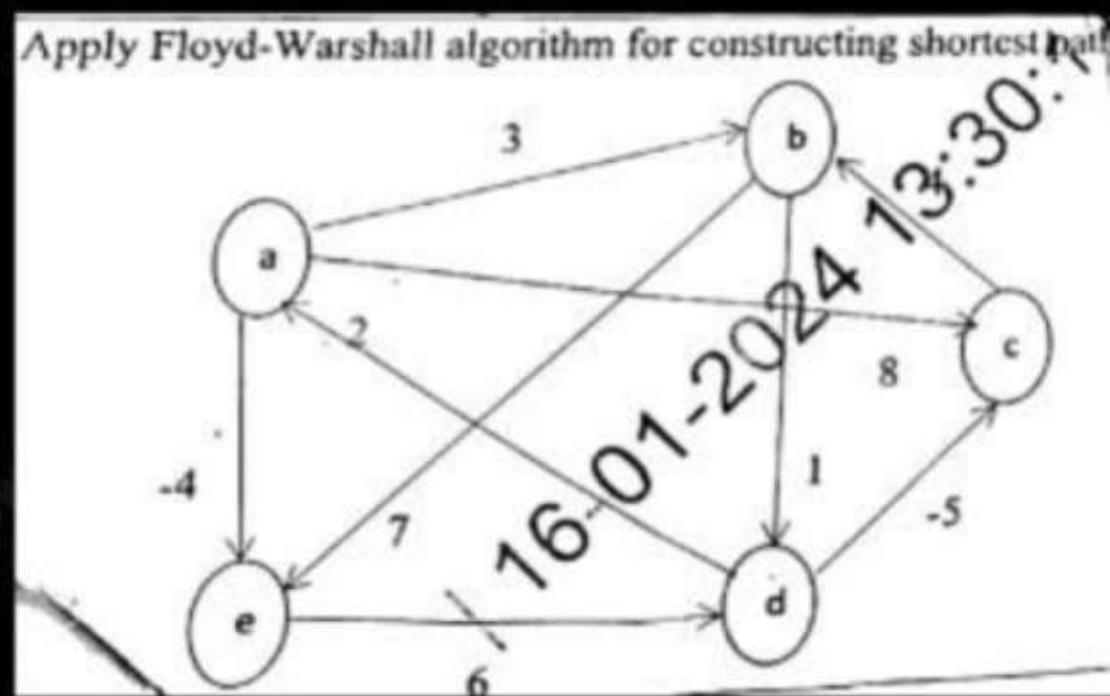
$$D^4 = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 0 & 3 & 5 & 6 \\ 2 & 5 & 0 & 2 \\ 3 & 3 & 6 & 0 \\ 4 & 2 & 5 & 7 \end{bmatrix}$$



$$D^0 = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 0 & 3 & \infty & 7 \\ 2 & 8 & 0 & 2 \\ 3 & 5 & \infty & 0 \\ 4 & 2 & \infty & 0 \end{bmatrix}$$

Application-

- 1) In networking devices
- 2) To calculate transitive closure of directed graphs.
- 3) In routing data packets

1.

(AKTU 2023-24)

2. Give Floyd Warshall algorithm to find the shortest path for all pairs of vertices in a graph. Give the complexity of the algorithm. Explain with example. (AKTU 2018-19)
3. Define dynamic programming. How dynamic programming approach can be used to find the shortest path? Illustrate with an example.(AKTU 2018-19)



AKTU

B.Tech 5th Sem



CS IT & CS Allied

DAA : Design & Analysis Of Algorithm

Unit-4 : Lecture-3

[PART-1 + PART-2]

Today's Target

- 0/1 knapsack problem
- AKTU PYQs

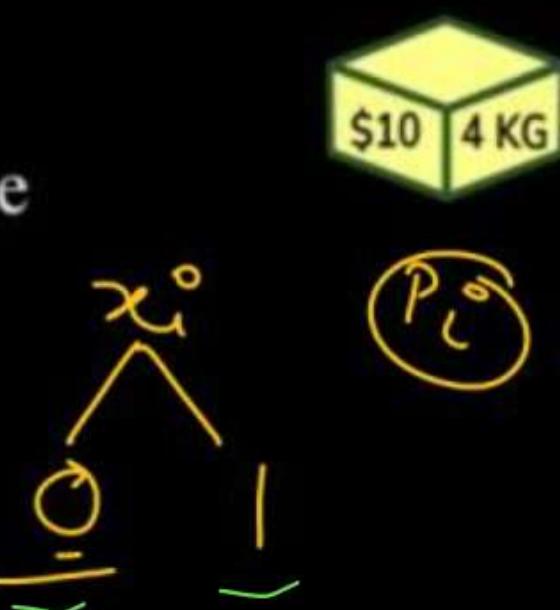


By Dr. Nidhi Parashar Ma'am

- M.Tech Gold Medalist
- Net Qualified

0/1 Knapsack Problem (Imp)

- There are given some objects and a knapsack/bag.
- Object i has a weight w_i and knapsack has a capacity M .
- If object i is placed into the knapsack, a profit of $p_i x_i$ is earned.
- Either an object is included as a whole or not included at all.
- Here x_i is 0 if object i is not included and x_i is 1 if the object i is ~~not~~ included.
- The objective is to fill the knapsack in a way that maximises the total profit earned.
- Since the knapsack capacity is M , we require the total weight of all chosen objects to be at most M .



Formally, the problem can be stated as follows:

Objective function \rightarrow

$$\text{Maximize} \left(\sum_{1 \leq i \leq n} p_i x_i \right) \rightarrow \text{No. of objects}$$

subjected to

constraint \Rightarrow

$$\sum_{1 \leq i \leq n} w_i x_i \leq M \rightarrow \text{Knapsack Maximum capacity}$$

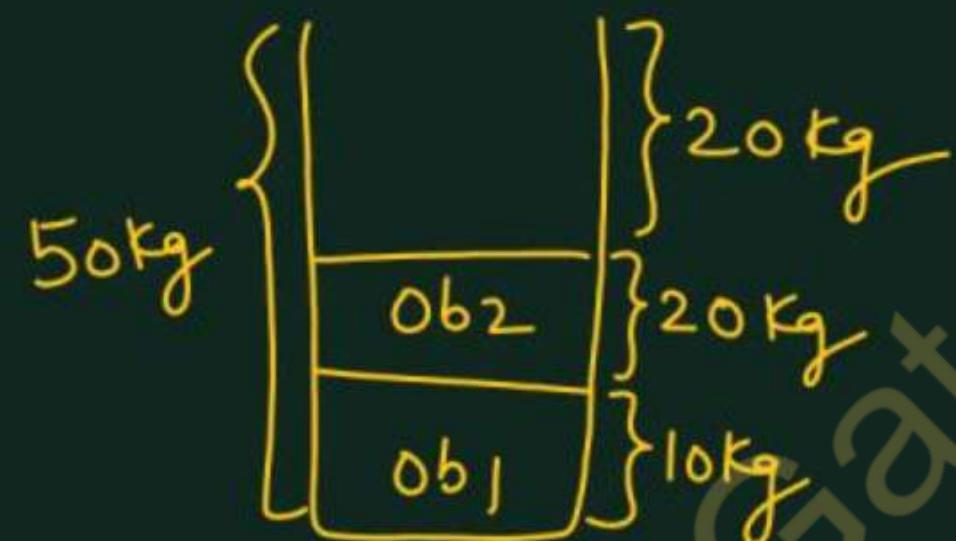
and,

$$x_i = \begin{cases} 0 & 1 \leq i \leq n \end{cases}$$

How greedy Algorithm fails to solve 0/1 knapsack problem?

Ex-1

Objects	1	2	3
$M=50\text{kg}$	50	80	90
w	10	20	30
P/w	5	4	3



$$\text{Profit} = 50 + 80 = 130$$

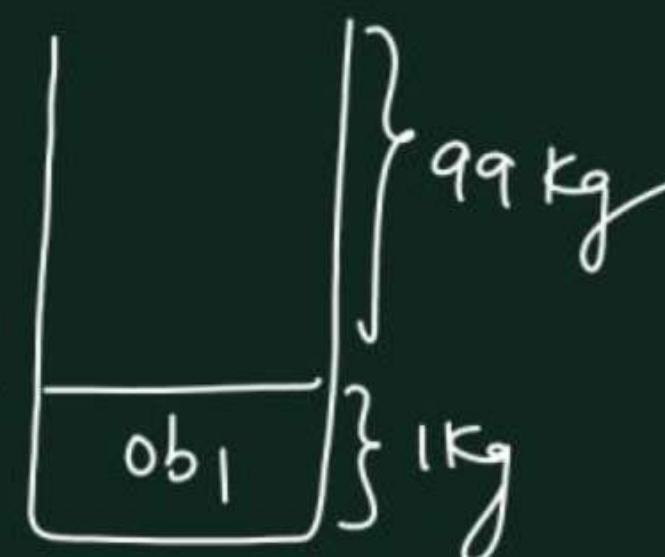
It should have been = 170

Ex-2

Objects	1	2
P	2	100
w	1	100

$$P = 2$$

It would have
been $P=100$



if ob2 was included first.

Brute Force Approach

Generate all subsets and check which subset generates the maximum benefit while maintaining the total weight.

- 1 denotes that an item has been considered
- 0 denotes that item has not been considered

there are N items and each item can take two values 0 or 1,

Number of combinations:

2^N possible combinations

Dynamic Programming approach

- Build a table consisting of rows of objects and columns of weights.
- Starting from the first row and column, we keep on filling the table till the desired value is obtained.
- Let the table be represented as $V[n, w]$ where n is the number of objects to be included and w is the left over space in the bag.

For every object, there can be two possibilities:

- if the object's weight is greater than the leftover space in the bag, then

$$V[n, w] = V[n - 1, w]$$

- else, the object might be taken or left out.

- if it is taken, the total value of the bag becomes:

$$V[n, w] = \underline{P_n} + \overline{V[n - 1, w - w_n]} \text{ where ' } P_n \text{ ' is profit of the } n^{\text{th}} \text{ object and}$$

w_n is its weight.

- if the object is not included:

$$V[n, w] = V[n - 1, w]$$

$V[n, w]$

available capacity

\uparrow

\downarrow

n^{th} object

$= V[n-1, w]$, if $w_n \leq w$ \rightarrow object can not be included in the bag.

$= \text{Max} (V[n-1, w], P_n + V[n-1, w - w_n])$ otherwise

Profit without considering this object

Note \Rightarrow Arrange objects in increasing order of their weights

Q-1 Solve following 0/1 knapsack problem $M=7$, $n=4$, $P=(1,4,5,7)$ and $W=(1,3,4,5)$.

Table Dimensions-

$(n+1) \times (M+1)$
rows cols

$$V[2,3] =$$

$$\max \{ V[1,3], 4 + V[1,3-3] \}$$

$$= \max \{ 1, 4 + 0 \}$$

$$= 4$$

$$V[2,4] = \max [V[1,4], 4 + V[1,4-3]]$$

$$= \max [1, 4 + 1] \Rightarrow 5$$

		Objects:				1	2	3	4	
		P	1	4	5	7				
		W	1	3	4	5				
										Remaining bag capacity
			P	W	n	w ≠ 0	w=1	w=2	w=3	w=4
			0	0	n=0	0	0	0	0	0
			1	!	n=1	0	1	1	1	1
			4	3	n=2	0	1	1	4	5
			5	4	n=3	0			5	5
			7	5	n=4	0			5	5

$$V[2,5] = \max [V[1,5], 4 + V[1,5-3]]$$

$$= \max [1, 4 + 1] \Rightarrow 5$$

Q-1 Solve following 0/1 knapsack problem $M=7$, $n=4$, $P=(1,4,5,7)$ and $W=(1,3,4,5)$.

Table Dimensions-

$\underbrace{(n+1)}_{\text{rows}} \times \underbrace{(M+1)}_{\text{cols}}$

similarly $V[2,6]$ and $V[2,7]$
will give profit = 5 only

$V[3,1]$, $V[3,2]$, and $V[3,3]$

are copied from above cells

as in these cases, available capacity
is less than the weight of 3rd object.

Objects :	1	2	3	4	
P :	1	4	5	7	
w :	1	3	4	5	

P	w_n	n	$w \neq 0$	$w=1$	$w=2$	$w=3$	$w=4$	$w=5$	$w=6$	$w=7$
0	0	$n=0$	0	0	0	0	0	0	0	0
1	1	$n=1$	0	1	1	1	1	1	1	1
4	3	$n=2$	0	1	1	4	5	5	5	5
5	4	$n=3$	0	1	1	4	5	6	6	9
7	5	$n=4$	0							

$$\begin{aligned}
 V[3,4] &= \max[V[2,4], 5 + V[2,4-4]] \\
 &= \max[5, 5 + 0] \Rightarrow 5
 \end{aligned}$$

Q-1 Solve following 0/1 knapsack problem $M=7$, $n=4$, $P=(1,4,5,7)$ and $W=(1,3,4,5)$.

$$v[3,5] = \max [v[2,5], 5 + v[2,1])$$

Objects:	1	2	3	4	
P:	1	4	5	7	
W:	1	3	4	5	

↑ remaining bag capacity

$$= \max[5, 5+1] = \textcircled{6}$$

$$V[3, \zeta] = \max(V[2, 6], 5 + V[2, 27])$$

$$= \max[5, 5+1] = 6$$

$$v[3,7] = \max(v[2,7], 5 + v[2,3])$$

$$= \max[5, 5+4] = 9$$

$$v[4,5] = \max \{ v[3,5], 7 + v[3,0] \}$$

$$= \max\{6, 7+0\} = 7$$

$$v[4,6] = \max\{v[3,4], 7 + v[3,1]\}$$

$$= \max\{6, 7+1\} = 8$$

P	w	n	$w \neq 0$	$w=1$	$w=2$	$w=3$	$w=4$	$w=5$	$w=6$	$w=7$
0	0	$n=0$	○	○	○	○	○	○	○	○
1	1	$n=1$	○	1	1	1	X	1	1	1
4	3	$n=2$	○	1	1	4	5	5	5	5
5	4	$n=3$	○	1	1	4	5	6	6	9
7	5	$n=4$	○	1	1	4	5	7	8	9

$$v[4,7] = \max \{ v[3,7], 7 + v[3,2] \}$$

$$= \max \{9, 7+1\} = 9$$

Q-1 Solve following 0/1 knapsack problem $M=7$, $n=4$, $P=(1,4,5,7)$ and $W=(1,3,4,5)$.

$$x = \{0, 1, 1, 0\}$$

x_1 , x_2 , x_3 , x_4

$$\max \text{Profit} = 9$$

$$\begin{aligned} \text{Remaining Profit} &= q - P_{\text{third object}} \\ &= q - 5 \Rightarrow 4 \end{aligned}$$

Remaining Profit $4-4 \Rightarrow 0$

Ans -

Object 3 and object 4 are added in the bag

To gain Max Profit of 9

Dynamic-0-1-knapsack (P, w, N, M)

```
{ for w = 0 to M    for all columns of first row  
    V[0, w] = 0  
  
    for n = 1 to N // for all rows of first col.  
        V[n, 0] = 0  
  
    for w = 1 to M  
        if wn ≤ w // if object under consideration is less than/equal to  
            if Pn + V[n-1, w-wn] > V[n-1, w]  
                V[n, w] = Pn + V[n-1, w-wn]  
            else  
                V[n, w] = V[n-1, w]  
        else  
            V[n, w] = V[n-1, w]  
  
    if object weight  
    is greater  
    than remaining  
    capacity  
    return V[N, M]
```

weight

if object weight is greater than remaining capacity

*Time Complexity: O(N * M)*

Q-2 Solve following 0/1 knapsack problem $M=8$, $n=4$, $P=(1,2,5,6)$ and $W=(2,3,4,5)$.

$$\begin{array}{c} \text{Ob : } 1 \ 2 \ 3 \ 4 \\ \hline P : 1 \ 2 \ 5 \ 6 \end{array} \boxed{M=8}$$

$$w : 2 \ 3 \ 4 \ 5$$

$$V[2,4] = \max \left\{ V[1,4], 2 + V[1,1] \right\} \\ = \max \left\{ 1, 2+0 \right\} \\ = 2$$

$$V[2,6] = \max \left\{ V[1,6], 2 + V[1,3] \right\} \\ = \max \left\{ 1, 2+1 \right\}$$

= 3

$$X = \left\{ \begin{matrix} 0 & 1 & 0 & 1 \\ x_1 & x_2 & x_3 & x_4 \end{matrix} \right\}^{(M+1) \times (M+1)}$$

$$\boxed{\text{Max} = 8}$$

P	w	n	$w=0$	$w=1$	$w=2$	$w=3$	$w=4$	$w=5$	$w=6$	$w=7$	$w=8$
0	0	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

Q-3 Solve following 0/1 knapsack problem $M=8$, $n=4$, $P=(2,3,1,4)$ and $W=(3,4,6,5)$.

First arrange weights in \uparrow order.

$x_1 \ x_2 \ x_3 \ x_4$

			$w=0$	$w=1$	$w=2$	$w=3$	$w=4$	$w=5$	$w=6$	$w=7$	$w=8$
$v[4,6]$	$= \max[v[3,6], 1+v[3,0]]$	P	w	n	$w=0$	$w=1$	$w=2$	$w=3$	$w=4$	$w=5$	$w=6$
		0	0	$n=0$	0	0	0	0	0	0	0
		2	3	$n=1$	0	0	0	2	2	2	2
		3	4	$n=2$	0	0	0	1	2	3	5
		4	5	$n=3$	0	0	0	2	3	4	5
		1	6	$n=4$	0	0	0	2	3	4	6

$$\mathcal{X} = \{x_1, x_2, x_3, x_4\}$$

MAX PROFIT = 6

Q-4 Solve 0/1 knapsack problem $c=20$, $n=4$, $P=(11, 21, 31, 33)$, $W=(2, 11, 22, 15)$. (AKTU 2019-20)

ρ	$w=0$	$w=1$	$w=2$	$w=3$	$w=4$	$w=5$	$w=6$	$w=7$	$w=8$	$w=9$	$w=10$	$w=11$	$w=12$	$w=13$	$w=14$	$w=15$	$w=16$	$w=17$	$w=18$	$w=19$	$w=20$
0	0 0	0 0	0 0	0 0	0 0	0 0	0 0	0 0	0 0	0 0	0 0	0 0	0 0	0 0	0 0	0 0	0 0	0 0	0 0	0 0	0 0
11	2 1	0 0	//	//	//	//	//	//	//	//	//	//	//	//	//	//	//	//	//	//	//
21	1 2	0 0	1 1	1 1	1 1	1 1	1 1	1 1	1 1	1 1	2 1	2 1	3 2	3 2	3 2	3 2	3 2	3 2	3 2	3 2	3 2
33	1 5 3	0 0	//	//	//	//	//	//	//	//	2 1	2 1	3 2	3 2	3 2	3 3	4 4	4 4	4 4	4 4	4 4
31	2 2 4	0 0	1 1	1 1	1 1	1 1	1 1	1 1	1 1	1 1	2 1	2 1	3 2	3 2	3 3	4 4	4 4	4 4	4 4	4 4	4 4

$$x = \{ |_{x_1}, 0, |_{x_2}, |_{x_3}, |_{x_4} \}$$

$$\text{MAX PROFIT} = \boxed{44}$$

AKTU PYQs

1. Solve following 0/1 knapsack problem $c=20$. $n=4$

$P=(11,21,31,33)$ and $W=(2,11,22,15)$. (AKTU 2019-20)

Gateway classes



AKTU

B.Tech 5th Sem



CS IT & CS Allied

DAA : Design & Analysis Of Algorithm

Unit-4 : Lecture-4

Today's Target

- Solving Traveling Salesman problem using dynamic programming
- AKTU PYQs



By Dr. Nidhi Parashar Ma'am

- M.Tech Gold Medalist
- Net Qualified

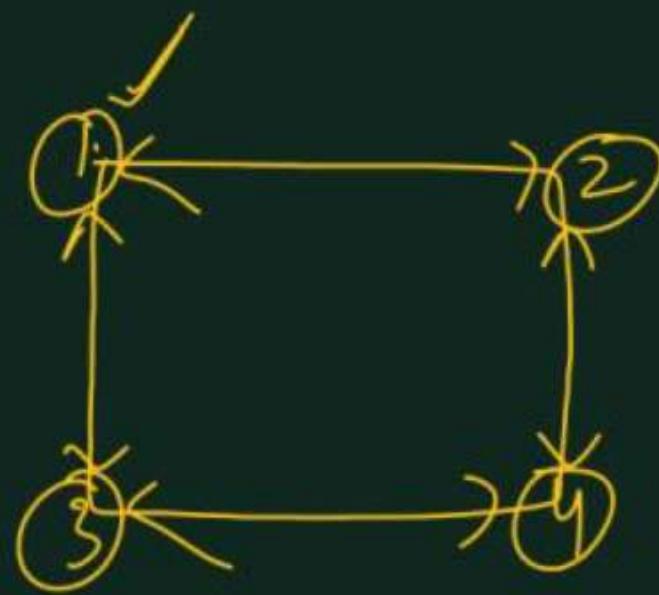
Travelling Salesman Problem

(Using dynamic Programming)

Problem Statement :-

A Salesman has to visit every city exactly once and return to the home city (Any chosen vertex as starting /Home City) forming a cycle and the challenge is that the salesman needs to minimize total length of the trip.





If ' n ' nodes are present in the graph.

$$(n-1)!$$

$$\Rightarrow n! = \boxed{O(n^n)}$$

Brute-force Approach

- ✓ 1 → 2 → 3 → 4 → 1
- ✓ 1 → 2 → 4 → 3 → 1
- | → 3 → 2 → 4 → |
- | → 3 → 4 → 2 → |
- | → 4 → 2 → 3 → |
- | → 4 → 3 → 2 → |

$$\frac{(3!)^4}{4}$$

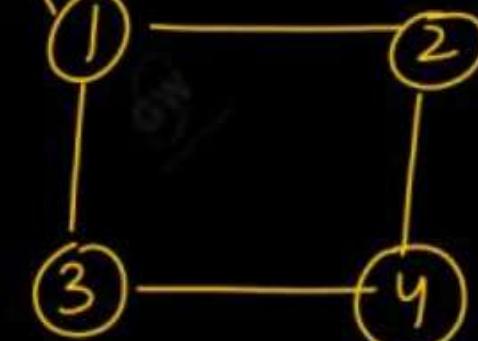
Recursive Equation :-

$$\text{TSP}(1, \{2, 3, 4\})$$

starting vertex

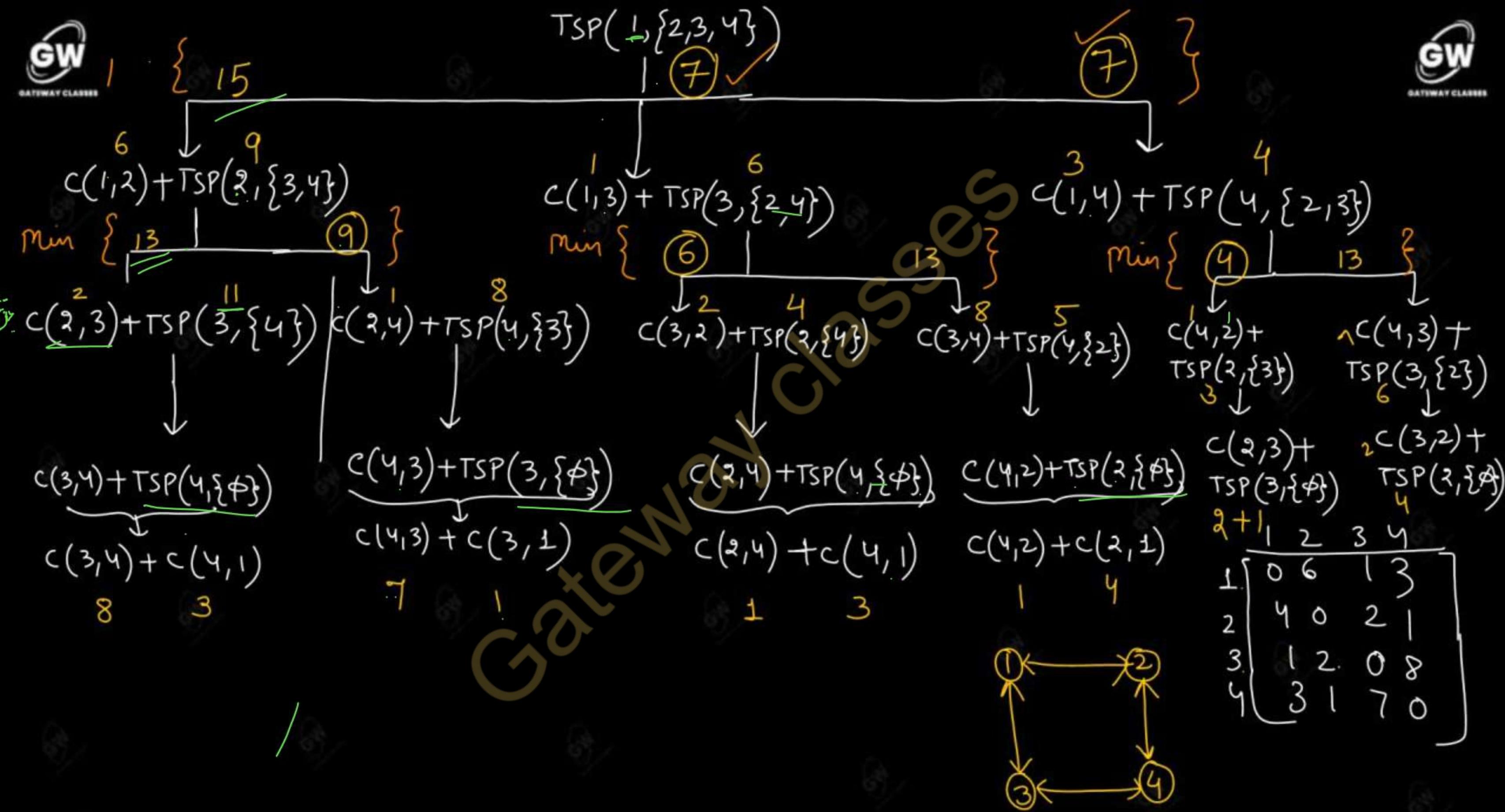
set of remaining vertices

$$= \text{Min} \left\{ \begin{array}{l} C(1, 2) + \text{TSP}(2, \{3, 4\}) \\ C(1, 3) + \text{TSP}(3, \{2, 4\}) \\ C(1, 4) + \text{TSP}(4, \{2, 3\}) \end{array} \right.$$

**General Recursive Equation**

$$\text{TSP}(i, S) = \begin{cases} \text{Min } k \in S \{C(i, k) + \text{TSP}(k, S - \{k\})\} & \text{when } S \neq \emptyset \\ C(i, 1) & \text{when } S = \emptyset \end{cases}$$

\nearrow chosen next vertex



There are two optimal solutions with cost = ⑦

solⁿ 1 ⇒

$$1 \rightarrow 3 \rightarrow 2 \rightarrow 4 \rightarrow 1$$

solⁿ 2 ⇒

$$1 \rightarrow 4 \rightarrow 2 \rightarrow 3 \rightarrow 1$$

Time Complexity -

$$T(n) = (n-1) \cdot 2^{(n-2)}$$

$$T(n) \approx n \cdot 2^n$$

still it is exponential time taking problem for large 'n'

$$n \cdot 2^n < n^n$$

In case of brute force approach

Total function calls $\Rightarrow 15$ ✓
But $TSP(4, \{\phi\})$, $TSP(3, \{\phi\})$, $TSP(2, \{\phi\})$ called twice

Distinct function calls

$$= 15 - 3 = 12$$

$$\text{No. of distinct fun^n calls} = (n-1)2^{n-2}$$

$$= (4-1)2^2$$

$$= 3 \cdot 4 \Rightarrow 12$$

$$TSP(1, \{2, 3, 4\}) = \min \left\{ \begin{array}{l} c(1, 2) + TSP(2, \{3, 4\}) \\ c(1, 3) + TSP(3, \{2, 4\}) \\ c(1, 4) + TSP(4, \{2, 3\}) \end{array} \right\}$$

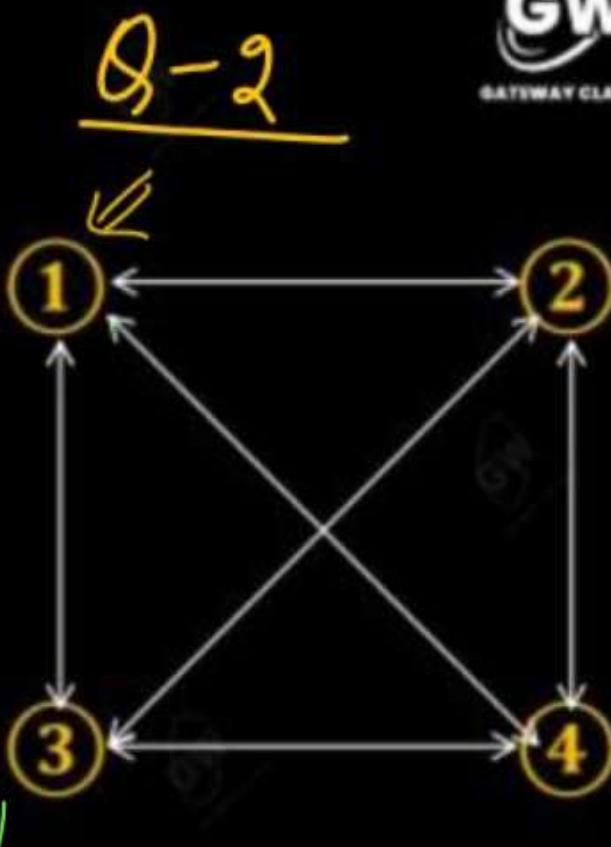
$$\underline{TSP(2, \{3, 4\})} = \min \left\{ \begin{array}{l} \cancel{c(2, 3)} + TSP(3, \{4\}) \\ \cancel{c(2, 4)} + TSP(4, \{3\}) \end{array} \right\}$$

$$\underline{TSP(3, \{4\})} = c(3, 4) + TSP(4, \{\phi\})$$

$$= c(3, 4) + c(4, 1) = 12 + 8 \Rightarrow \underline{\underline{20}}$$

$$\underline{TSP(4, \{3\})} = c(4, 3) + TSP(3, \{\phi\})$$

$$= c(4, 3) + c(3, 1) = 9 + 6 = \underline{15}$$



	1	2	3	4
1	0	10	15	20
2	4	0	9	10
3	6	13	0	12
4	8	8	9	0

$$\text{TSP}(2, \{3, 4\}) = \min \left\{ 9+20, 10+15 \right\}$$

$$= \underline{25}$$

$$\text{TSP}(3, \{2, 4\}) = \min \left\{ \begin{array}{l} C(3, 2) + \text{TSP}(2, \{4\}) \\ 25 \\ C(3, 4) + \text{TSP}(4, \{2\}) \end{array} \right\}$$

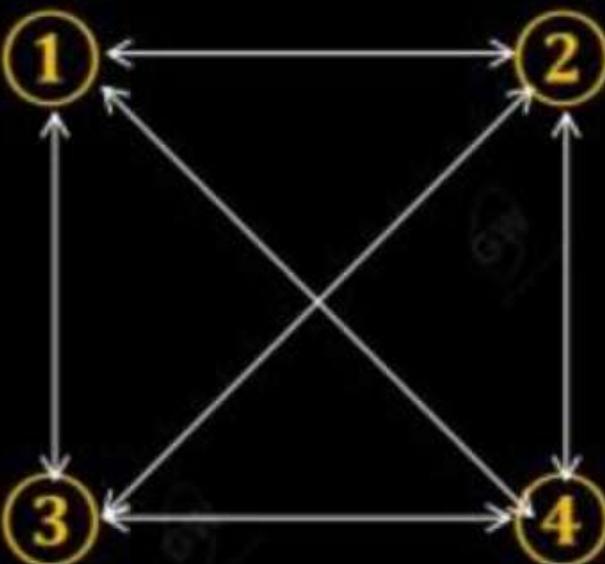
$$\text{TSP}(2, \{4\}) = C(2, 4) + \text{TSP}(4, \{\phi\})$$

$$= C(2, 4) + C(4, 1)$$

$$= 10 + 8 = \underline{28}$$

$$\text{TSP}(4, \{2\}) = C(4, 2) + \text{TSP}(2, \{\phi\})$$

$$= C(4, 2) + C(2, 1) = 8 + 5 \Rightarrow \underline{13}$$



	1	2	3	4
1	0	10	15	20
2	5	0	9	10
3	6	13	0	12
4	8	8	9	0

$$T(3, \{2, 4\}) = \min \{ 13 + 28, 12 + 13 \}$$

$$= \min \{ 41, 25 \} = \textcircled{25}$$

$$\underline{TSP(4, \{2, 3\})} = \min \left\{ \begin{array}{l} C(4, 2) + TSP(2, \{3\}) \\ C(4, 3) + TSP(3, \{2\}) \end{array} \right\}$$

$$TSP(2, \{3\}) = C(2, 3) + TSP(3, \{\phi\})$$

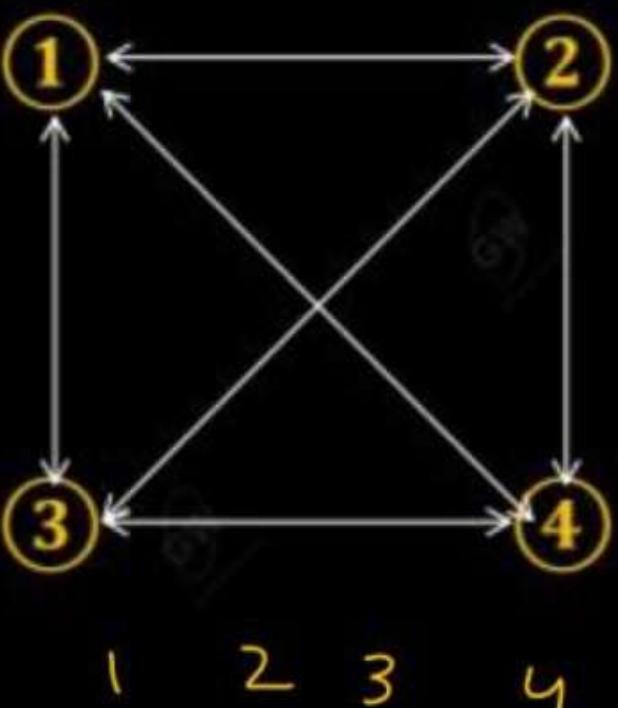
$$= C(2, 3) + C(3, 1)$$

$$= 9 + 6 = \underline{15}$$

$$TSP(3, \{2\}) = C(3, 2) + TSP(2, \{\phi\})$$

$$= C(3, 2) + C(2, 1)$$

$$= 13 + 5 = \underline{18}$$



	1	2	3	4
1	0	10	15	20
2	5	0	9	10
3	6	13	0	12
4	8	8	9	0

$$\begin{aligned} TSP(4, \{2, 3\}) &= \min \{ 8+15, 9+18 \} \\ &= \min \{ 23, 27 \} = 23 \end{aligned}$$

$$\begin{aligned} TSP(1, \{2, 3, 4\}) &= \min \left\{ \frac{10+25}{}, \frac{15+25}{}, \frac{20+23}{}, \dots \right\} \\ &= \min \{ 35, 40, 43 \} \end{aligned}$$

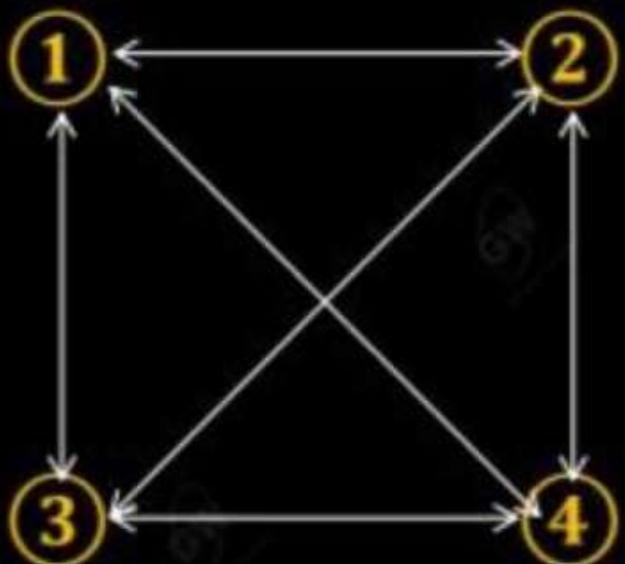
Min cost = 35

Ans

Optimal Route

$1 \rightarrow 2 \rightarrow 4 \rightarrow 3 \rightarrow 1$ ✓

$$10 + 10 + 9 + 6$$



	1	2	3	4
1	0	10	15	20
2	5	0	9	10
3	6	13	0	12
4	8	8	9	0

Q- Solve traveling salesman problem using dynamic programming.

(AKTU 2021-22)

starting vertex

1	0	1	15	6
2	2	0	7	3
3	9	6	0	12
4	10	4	8	0

$$n = 4$$



AKTU

B.Tech 5th Sem



CS IT & CS Allied

DAA : Design & Analysis Of Algorithm

Unit-4 : Lecture-5

Today's Target

- Optimal Reliability allocation problem (unit -3)
- AKTU PYQs



By Dr. Nidhi Parashar Ma'am

- M.Tech Gold Medalist
- Net Qualified

Reliability Design

We have to setup a system (combinations of devices D_1, D_2, D_3, D_4).

devices - $D_1 - D_2 - D_3 - D_4$

Cost - $C_1 - C_2 - C_3 - C_4$

Reliability - $r_1 = r_2 = r_3 = r_4 = 0.9$

Chances that D_1 works perfectly

We have to develop a system with Max. Reliability

Reliability of working perfectly = 65%

Chances that system may fail = 35%

suppose for D_1 , $r_1 = 0.9$

$$\text{System Reliability} = \frac{\prod r_i}{4} \\ = (0.9)^4 = 0.6561$$

= 65% Reliable

Max. Reliability
↓
Minimization problem

↓
Can be solved using dynamic programming

	D_1	D_2	D_3	D_4
	C_1	C_2	C_3	C_4

	r_1	r_2	r_3	r_4
	0.9	0.9	0.9	0.9

Suppose

Reliability of each device = 0.9

Reliability of the entire system = $\pi_{R_i} = (0.9)^4 = 0.6561$
 $\Rightarrow 65\%$

Let's have multiple copies of each device in parallel.

Stages	<u>S_1</u>	<u>S_2</u>	<u>S_3</u>	<u>S_4</u>
	<u>D_1</u>	D_2	D_3	D_4
	<u>D_1</u>	D_2	D_3	D_4
	<u>D_1</u>	D_2	D_3	D_4

General Formula for
Reliability calculations

$$\left(1 - \left(1 - r_i\right)^n\right)$$

Let's take device D_1

Reliability of each device = 0.9

Probability of not failing working good = $(1 - 0.9)$

Probability for all three copies have failed. = $(1 - 0.9)^3$

$$(0.1)^3 = \frac{1}{1000}$$

Reliability of three copies (for correct functioning) = $(1 - 0.001)$

$$= 0.999 \Rightarrow 99.9\% \text{ Reliable system}$$

Total copies considered for a particular device i .

Problem - Statement

If we have C cost in hand and we need to setup a system and buy devices. How many copies of each device I should buy within that cost such that the total reliability of the systems is maximized.

Example

$$\boxed{C = 105}$$

Total Reliability =

$$\pi_{x_i}$$

$$0.9 \times 0.8 \times 0.5$$

$$= 0.36$$

= 36% Reliable

D_i	C_i	r_i	U_i
D_1	$\frac{30}{}$	0.9	
D_2	$\frac{15}{}$	0.8	
D_3	$\frac{20}{}$	0.5	

At least one copy of each device is to be included.

$$\sum C_i = \underline{30} + \underline{15} + \underline{20} = \underline{\underline{65}}$$

The remaining amount we can spend on buying multiple copies of other devices.

$$\text{Remaining amount} = \underbrace{[C - \sum C_i]}_{\text{Remaining amount}} = \underline{(105 - 65)} = \underline{40} \$$$

In these 40 \$ will calculate how many copies of each device we can buy of all 40 \$ are spent on one device only.

For $D_1 = \left\lfloor \frac{40}{30} \right\rfloor = 1 + 1 \Rightarrow 2$ Upper bounds

For $D_2 = \left\lfloor \frac{40}{15} \right\rfloor = 2 + 1 = 3$

For $D_3 = \left\lfloor \frac{40}{20} \right\rfloor = 2 + 1 = 3$

These are upper bounds of each and every device (Max. How many copies we can take of that particular device by spending all) money.

So, upper bound $\boxed{U_i = \left\lfloor \frac{c - \sum c_i}{c_i} \right\rfloor + 1}$

We start with a set of order pair

$$S_0 = \{ (1, 0) \}$$

Reliability \rightarrow cost

(R, C)

no device is included

1st stage

for D_1

device no.

S_1

$$= \{ (0.9 \times 1, 0 + 30) \} = \{ (0.9, 30) \}$$

$D_1 = 1$ copy of no. copies

S_1

$$= \{ (0.99 \times 1, 0 + 60) \} = \{ (0.99, 60) \}$$

$$S^1 = \{ (0.9, 30) (0.99, 60) \}$$

\downarrow \downarrow

$D_1 = 1$ copy

D_l	C_l	r_l	U_l
D_1	30	0.9 ✓	2
D_2	15	0.8	3
D_3	20	0.5	3

$$C = 105$$

for 2 copies of D_1

reliability =

$$1 - (1 - r_1)^2$$

$$= 1 - (1 - 0.9)^2 =$$

$$= 1 - (0.1)^2 = 1 - 0.01$$

$$= 0.99 ✓$$

$$\text{cost} = \underline{60}$$

For D_2

$$S^1 = \{ (\underline{0.9}, 30) (\underline{0.99}, 60) \}$$

$$S^2_1 = \{ (0.72, 45) (0.79, 75) \}$$

$$S^2_2 = \{ (\underline{0.86}, 60) (0.95, \cancel{90}) \}$$

After spending 90\$ upto first two devices, we are having insufficient amount of money to purchase at least one copy of $D_3 \Rightarrow$ so we will discard this order pair.

D_t	C_t	r_t	U_t
D_1	30	0.9	2
D_2	15	0.8	3
D_3	20	0.5	3

For 2 copies of D_2

$$1 - (1 - 0.8)^2$$

$$= 1 - (0.2)^2$$

$$= 1 - 0.04$$

$$= \underline{\underline{0.96}} \quad \text{Cost} = \underline{\underline{30}}$$

For D_2 = ① $S^1 = \{(0.9, 30) (0.99, 60)\}$

$S^2 = \{(0.72, 45) (0.79, 75)\}$

S^2 ② $= \{(0.86, 60) (0.95, 90)\}$

S^2 ③ $= \{(0.89, 75) (0.98, 105)\}$ Insufficient amount
to purchase even a single copy of D_3 .

If Reliability \uparrow then cost should also \uparrow . otherwise
the order-pair with higher cost is not considered:

$S^2 = \{(0.72, 45) (0.86, 60) (0.89, 75)\}$

$D_2 = 2$ copies

D_t	c_t	r_t	U_t
D_{t1}	30	0.9	2
D_{t2}	15	0.8	3
D_{t3}	20	0.5	3

For 2 copies of D_2

$$1 - (1 - 0.8)^2$$

$$= 1 - (0.2)^2$$

$$= 1 - 0.04$$

$$= \underline{0.96} \quad \text{Cost} = \underline{30}$$

For 3 copies of D_2

$$1 - (1 - 0.8)^3$$

$$= 1 - (0.2)^3 = 1 - 0.008$$

$$= 0.992$$

For D_3 $\boxed{S^2} = \{(0.72, 45) \boxed{(0.86, 60)} (0.89, 75)\}$

$\underline{\underline{S^3}} = \{(0.36, 65) (0.43, 80) (0.44, 95)\}$

$\textcircled{2} S^3 = \{(0.54, 85) \boxed{(0.64, 100)} (0.66, \cancel{75+40})\}$

$\cancel{S^3} = \{(0.62, 105) (0.74, \cancel{120}) (0.77, \cancel{75+60})\}$

$S^3 = \{(0.36, 65) (0.43, 80) (0.44, 95) (0.54, 85) (0.62, \cancel{105})\}$

$S^3 = \{(0.36, 65) (0.43, 80) (0.54, 85) \boxed{0.64, 100}\}$

$D_3 = 2$
copies

D_t	C_t	r_t	U_t
D_1	30	0.9	2
D_2	15	0.8	3
D_3	20	0.5	3

Reliability for 2 copies of

D_3

$$1 - (1 - 0.5)^2 \\ = 1 - (0.5)^2 = 1 - 0.25$$

$$= \frac{0.75}{C} = 40$$

for 3 copies of D_3

$$1 - (1 - 0.5)^3 \\ = 1 - (0.5)^3 = 0.875 \\ C = 60$$

Final Solution -

$D_1 \Rightarrow 1$ copy

$D_2 \Rightarrow 2$ copies

$D_3 \Rightarrow 2$ copies

Max Reliability = 0.64

- 64% Reliable system

D_i	C_i	r_i	U_i
D_1	30	0.9	2
D_2	15	0.8	3
D_3	20	0.5	3

Total Reliability of the system
with single copies for
all devices = $\prod r_i$

$$\prod r_i = 36\%$$

improved to



AKTU

B.Tech 5th Sem



CS IT & CS Allied

DAA : Design & Analysis Of Algorithm

Unit-4 : Lecture-6

Today's Target

- Introduction to Backtracking
Solving N-Queens Problem using backtracking
- AKTU PYQs

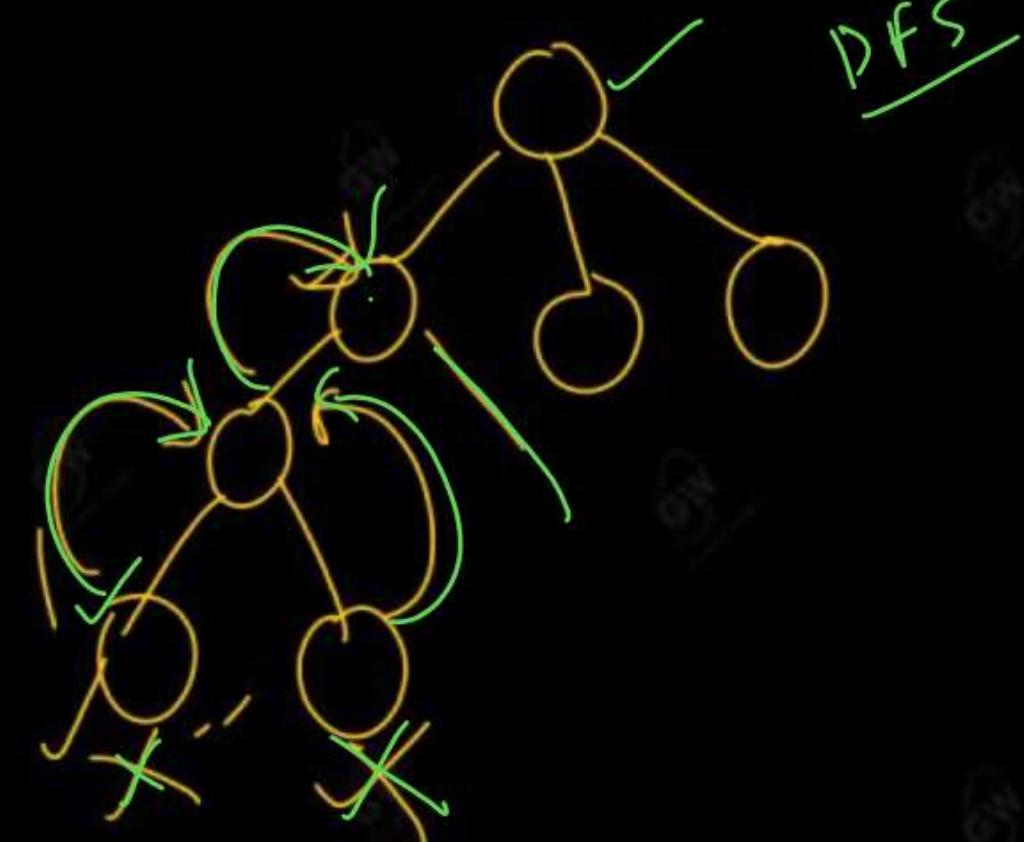


By Dr. Nidhi Parashar Ma'am

- M.Tech Gold Medalist
- Net Qualified

Backtracking (AKT U)

- It is a problem-solving algorithm that uses brute force approach for finding the desired output.
- Backtracking is used when we have multiple solutions and we need all of them.
- The Brute force approach tries out all the possible solutions and chooses the desired/best X solutions.
- Computational problem solved using backtracking usually have some constraints.
- if the current solution is not suitable, then backtrack and try other solutions. Thus, recursion is used in this approach.
- Backtracking is not used to solve optimization problem.
- Only the solutions that satisfy these constraints are considered.
- It follows depth first search.



Pseudo-code for backtracking problems:

explore(choices):

 if there are no more choices to make: stop.

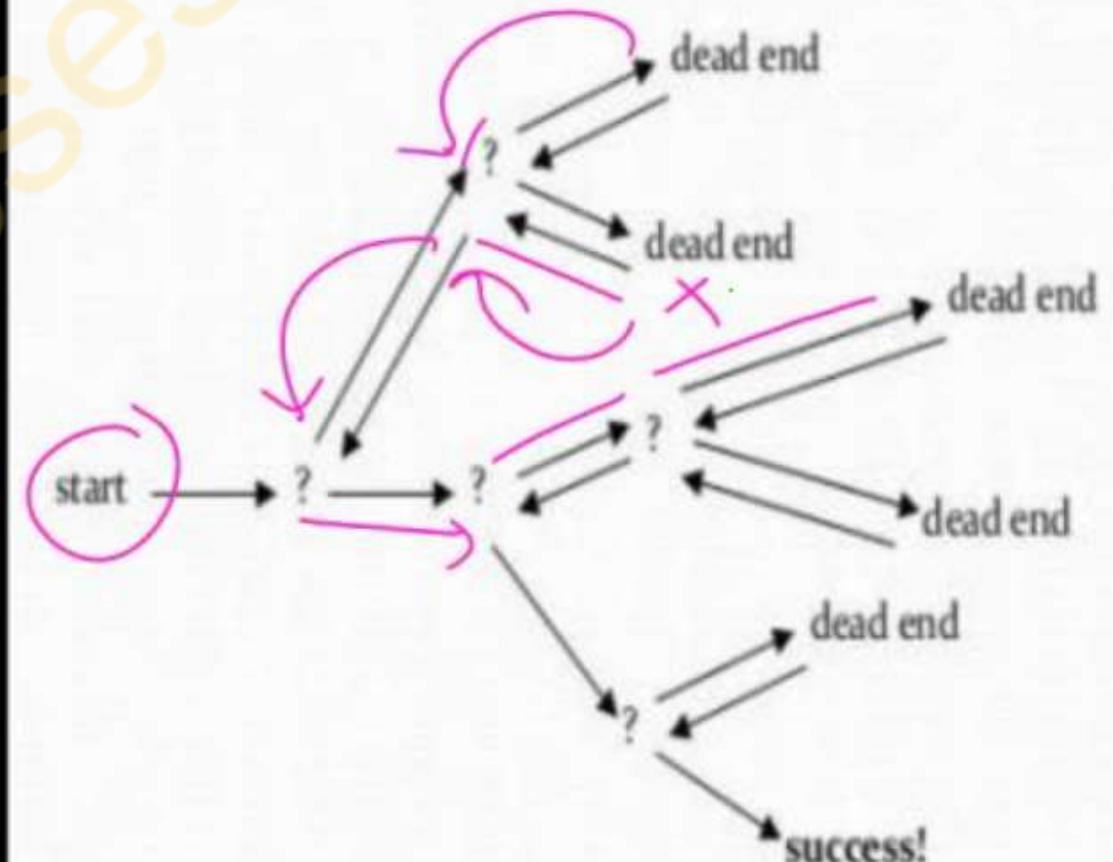
 else:

- Make a single choice C from the set of choices.
----Remove C from the set of choices.

- explore the remaining choices.

- Un-make choice C.

----Backtrack!



A space state tree is a tree representing all the possible states (solution or nonsolution) of the problem from the root as an initial state to the leaf as a terminal state.

Suppose we need to sort three numbers 8 6 7

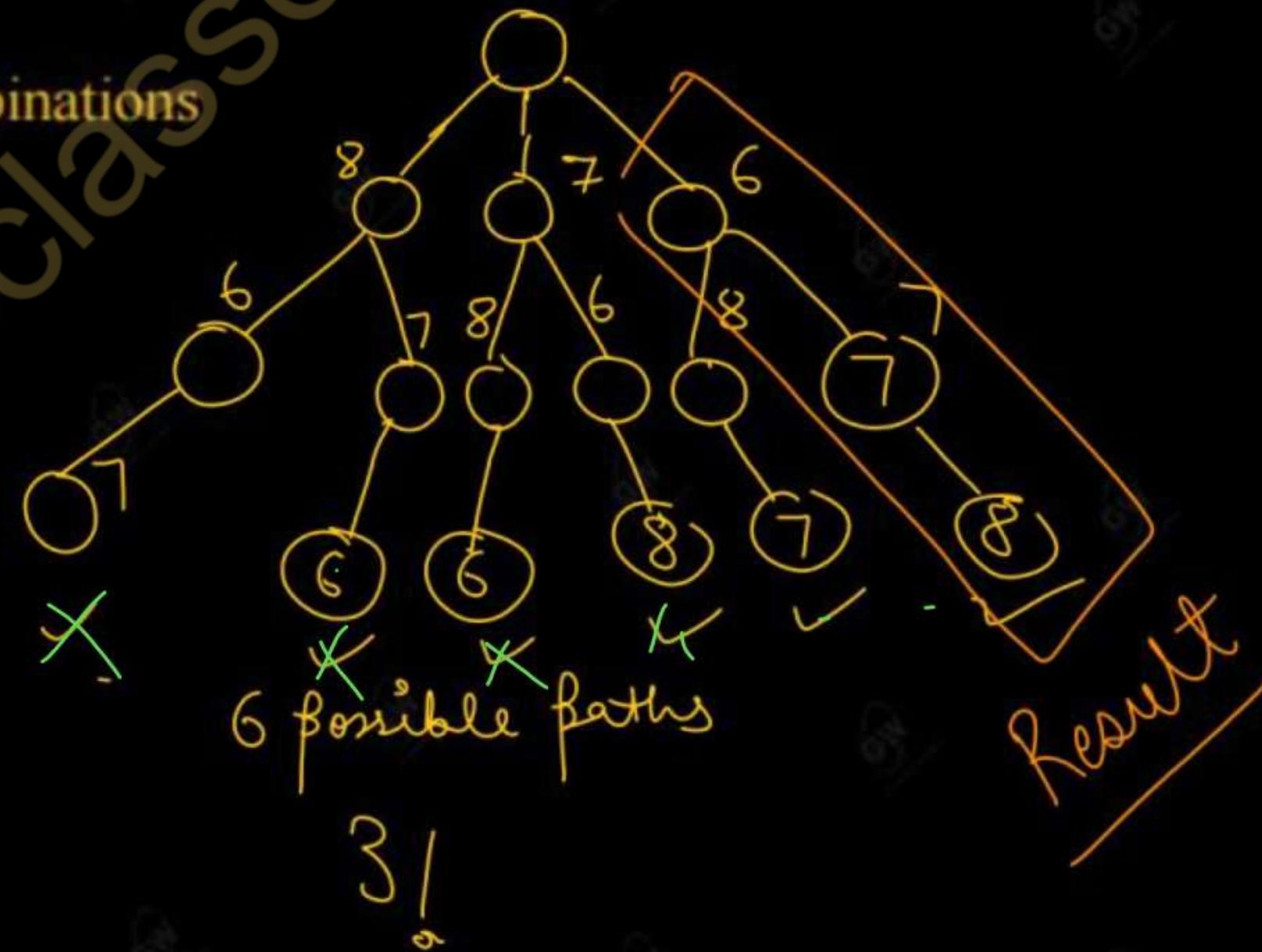
Applying brute force method: we will try all combinations

No. :- 8, 7, 6

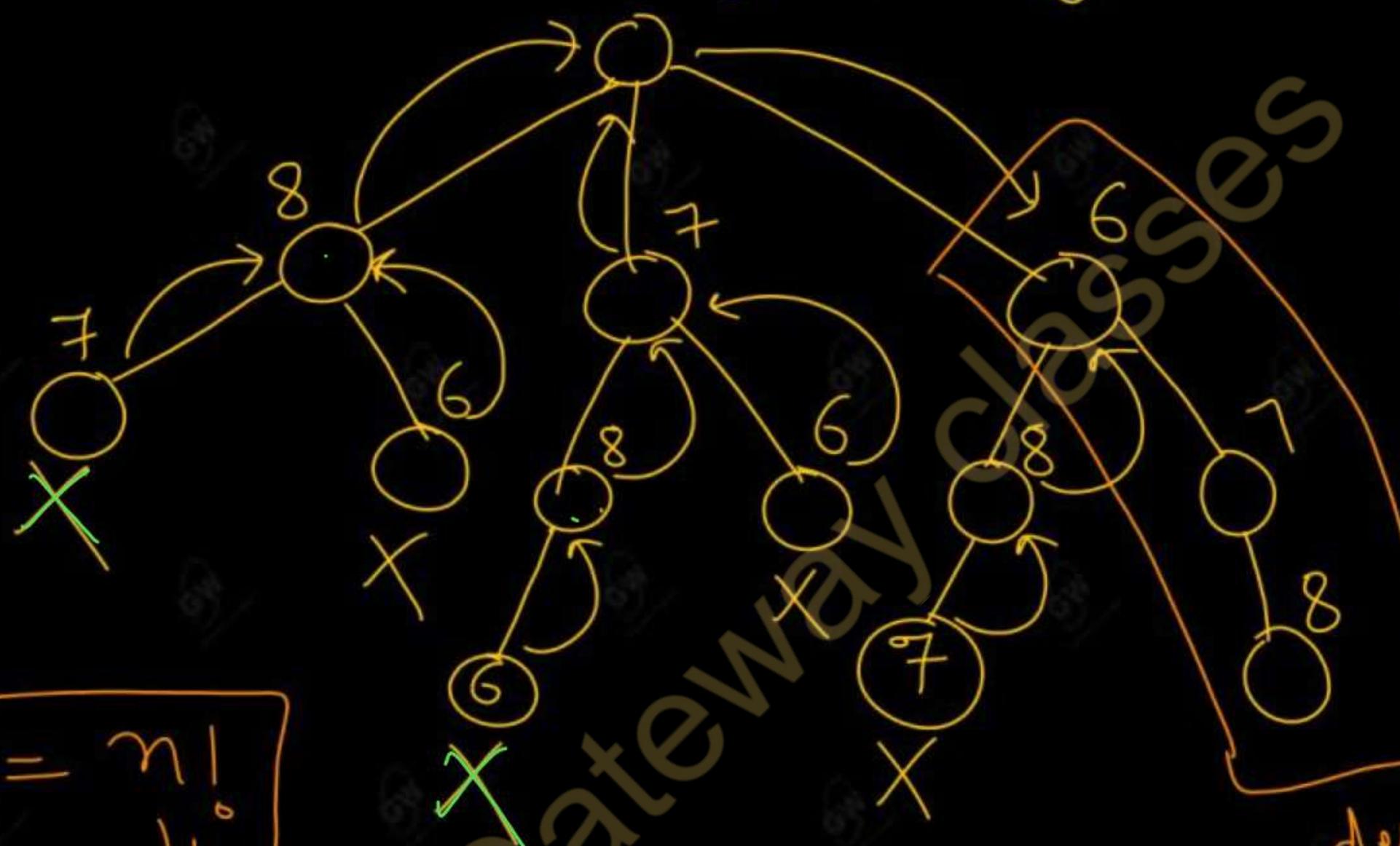
for n numbers

$$\boxed{T(n) = O(n^n)}$$

$$\begin{array}{c} \overbrace{8, 7, 6}^3 \times \overbrace{2}^2 \times \overbrace{1}^1 \\ 3! \end{array}$$



Sort 8, 7, 6 into Ascending order using Backtracking



state space
Tree

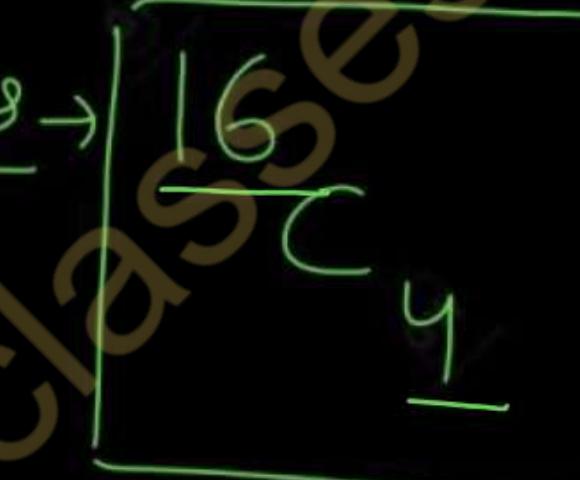
$$T(n) = n!$$

$\Theta(m^n)$

N queen's Problem

Place N queens in $N \times N$ chessboard so that no two queens can attack i.e. no two of them are in the same row, column or diagonal. Solutions are expressed in n - tuple.

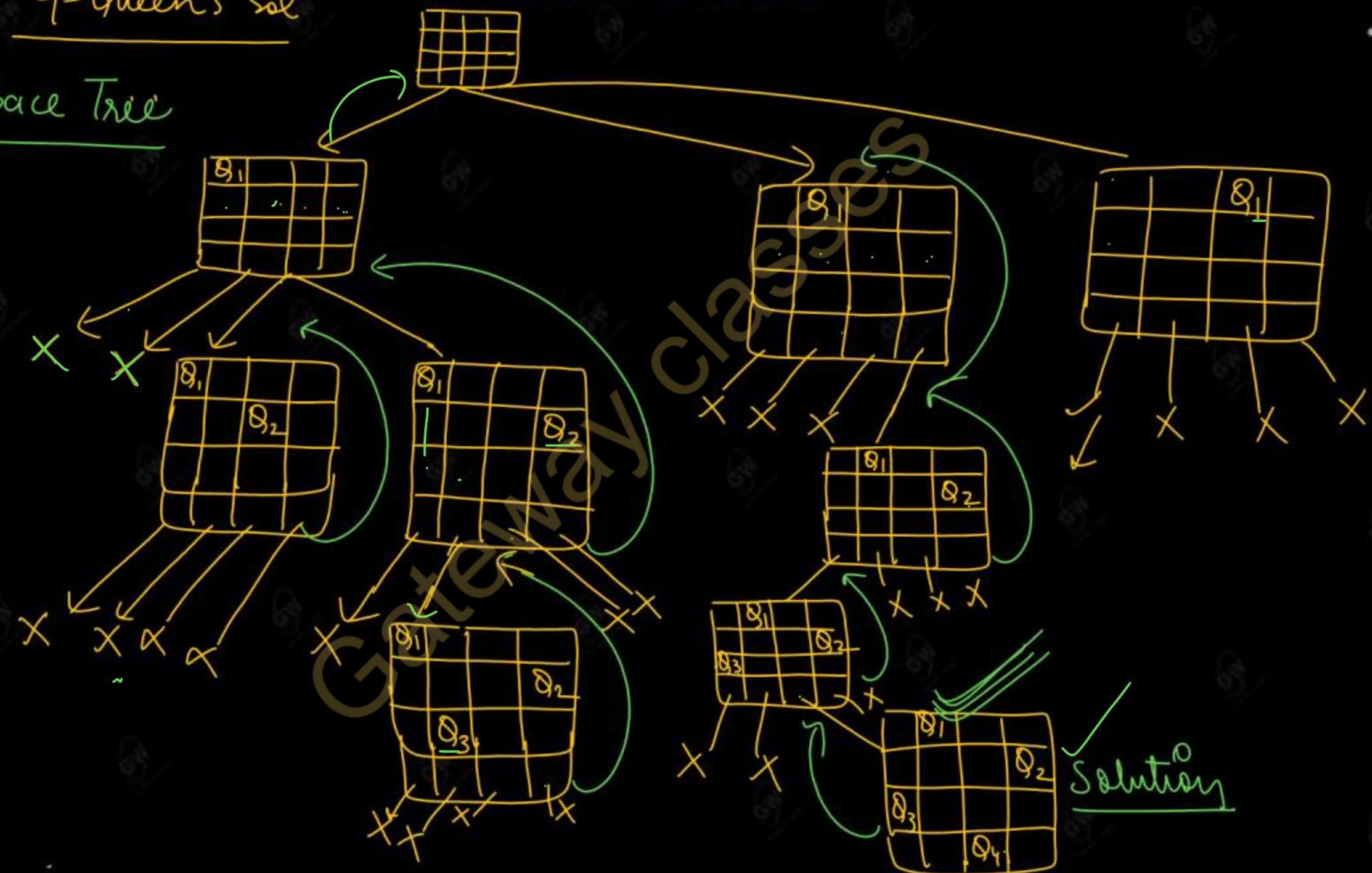
	1	2	3	4
1	Q ₁			
2			Q ₂	
3				
4				

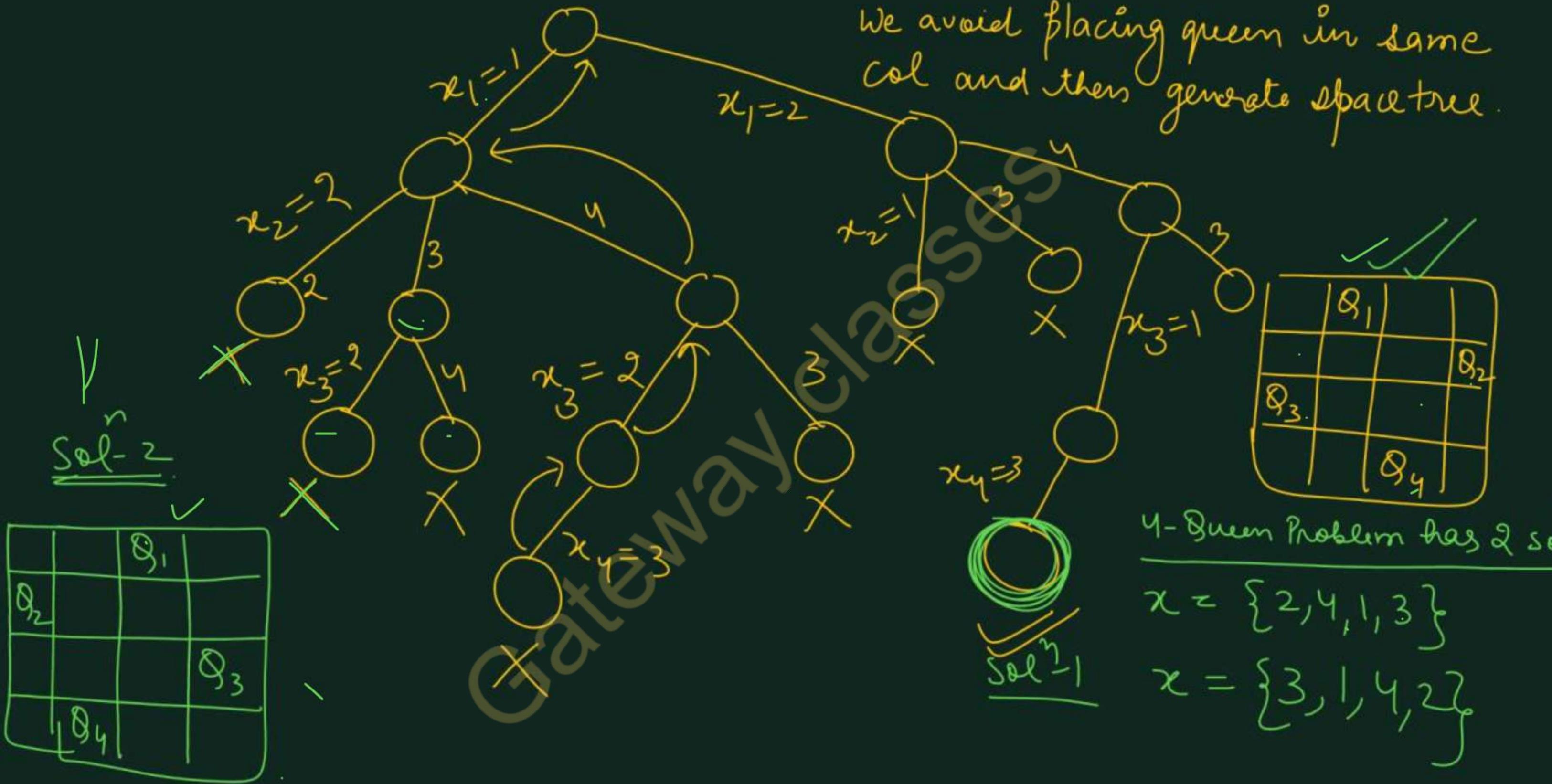
4-Queens →  → possible combinations

Note { → 4-Queen's problem has 2 solutions.
→ 8-Queen's problem 92 solutions.

4-Queen's solⁿstate Space Tree

N - Queen's Problem





(8x8) 8-Queens

	1	2	3	4	5	6	7	8
1								Q_1
2								Q_2
3								Q_3
4								Q_4
5								Q_5
6								Q_6
7								Q_7
8								Q_8

Solⁿ

$$X = \{4, 6, 8, 2, 7, 1, 3, 5\}$$

N - Queen's Problem

case-1 $(3, 1), (4, 2), (5, 3), (6, 4)$

Cell-1 $\rightarrow (i, j)$, Cell-2 $\rightarrow (k, l)$

$$(i-j) = (k-l) \Rightarrow j-l = i-k \quad \textcircled{1}$$

case-2 $(5, 8), (6, 7), (7, 6), (8, 5)$

$$(i+j) = (k+l) \Rightarrow j-l = k-i \quad \textcircled{2}$$

$$\text{Abs}(j-l) = \text{Abs}(i-k) \rightarrow \text{condition for same diagonal}$$

Algorithm Place (K, i)

// Returns true if Q is placed at kth row and ith column otherwise returns false

// x [] is a global array.

for j = 1 to (K-1) do // check with all the already placed queen for a

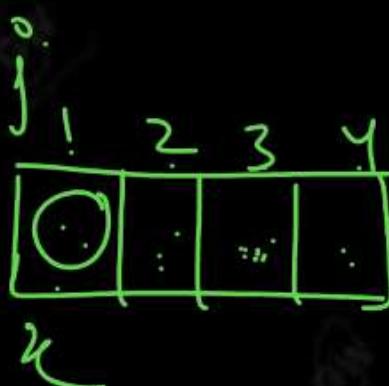
{

if ((x[j] = i) // in same column

Or

(Abs (x[j] - i) = Abs (j-k))) // in same diagonal then

Return false



} Return True

}



Algorithm NQueens (K, n) \rightarrow no. of queens
// using backtracking, two procedure prints all combinations of solutions
{

for $i = 1$ to n do // For all 4-Queens
{

if (Place (K, i)) then
{

for 4-Queen's
problem

first call is

\Rightarrow NQueens (1, 4)

$x[k] = i$ // Insert the col no. into solution

if ($K == n$) then // If you are placing
last queen

Print ($x[1:n]$)

$j \Rightarrow$ new no.

$x[j] \Rightarrow$ col no

$j \rightarrow$	1	2	3	4
x	2	1	4	

}

}

1. Discuss N queen's problem. Solve 4 queen's problem using backtracking and draw state space tree. (AKTU 2020-21, 2021-22, 2022-23)
2. Explain backtracking. (AKTU 2021-22, 2022-23)



AKTU

B.Tech 5th Sem



CS IT & CS Allied

DAA : Design & Analysis Of Algorithm

Unit-4 : Lecture-7

Today's Target

- Solving Sum of Subset problem using backtracking
- AKTU PYQs



By Dr. Nidhi Parashar Ma'am

- M.Tech Gold Medalist
- Net Qualified

1. What is the sum of subsets problem? Let $w=\{5,7,10,12,15,18,20\}$ and $m=35$. Find all possible subsets of w that sum to m using recursive backtracking algorithm for it. Draw the portion of the state-space tree that is generated. (AKTU 2022-23)
2. What is the sum of subsets problem? Let $S=\{1,3,4,5\}$ and $X=8$. Find all possible subsets of S using backtracking. (AKTU 2021-22)
3. What is the sum of subsets problem? Let $n=6$, $m=30$ $w[1:6]=\{5,10,12,13,15,18\}$. Find all possible subsets using backtracking. Draw state space tree (AKTU 2023-24)
4. Solve sum of subsets problem using backtracking where $n=4$, $m=18$ $w[4]=\{5,10,8,13\}$ (AKTU 2018-19)

Sum of subsets Problem

Problem Statement :-

The sum of subset problem finds a subset of a given set of n positive integers whose sum is exact equal ^{+to} is a given positive integer m .

$$W = \{a_1, a_2, \dots, \dots, \dots, a_n\}$$

Some instances of this problem may have no solutions.

For convenience, sort set of elements in according order. So, we assume that

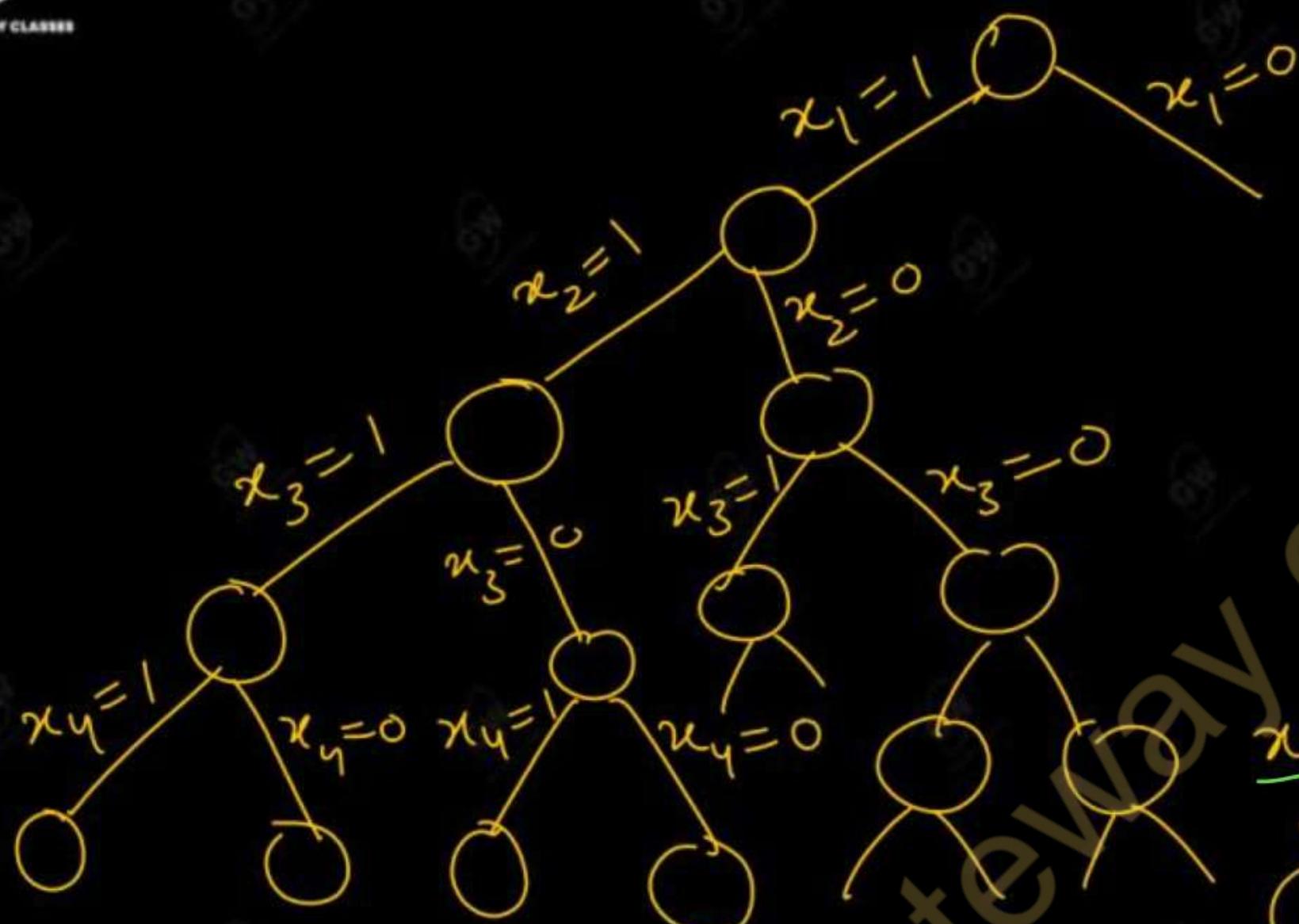
$$a_1 \leq a_2 \leq a_3 \dots \dots \dots \leq a_n$$

For example for $W[1:5] = \{1, 2, 5, 6, 8\}$ and $m=9$, $n=5$ and there are two solutions:-

(1) {1, 2, 6}

(2) {1, 8}

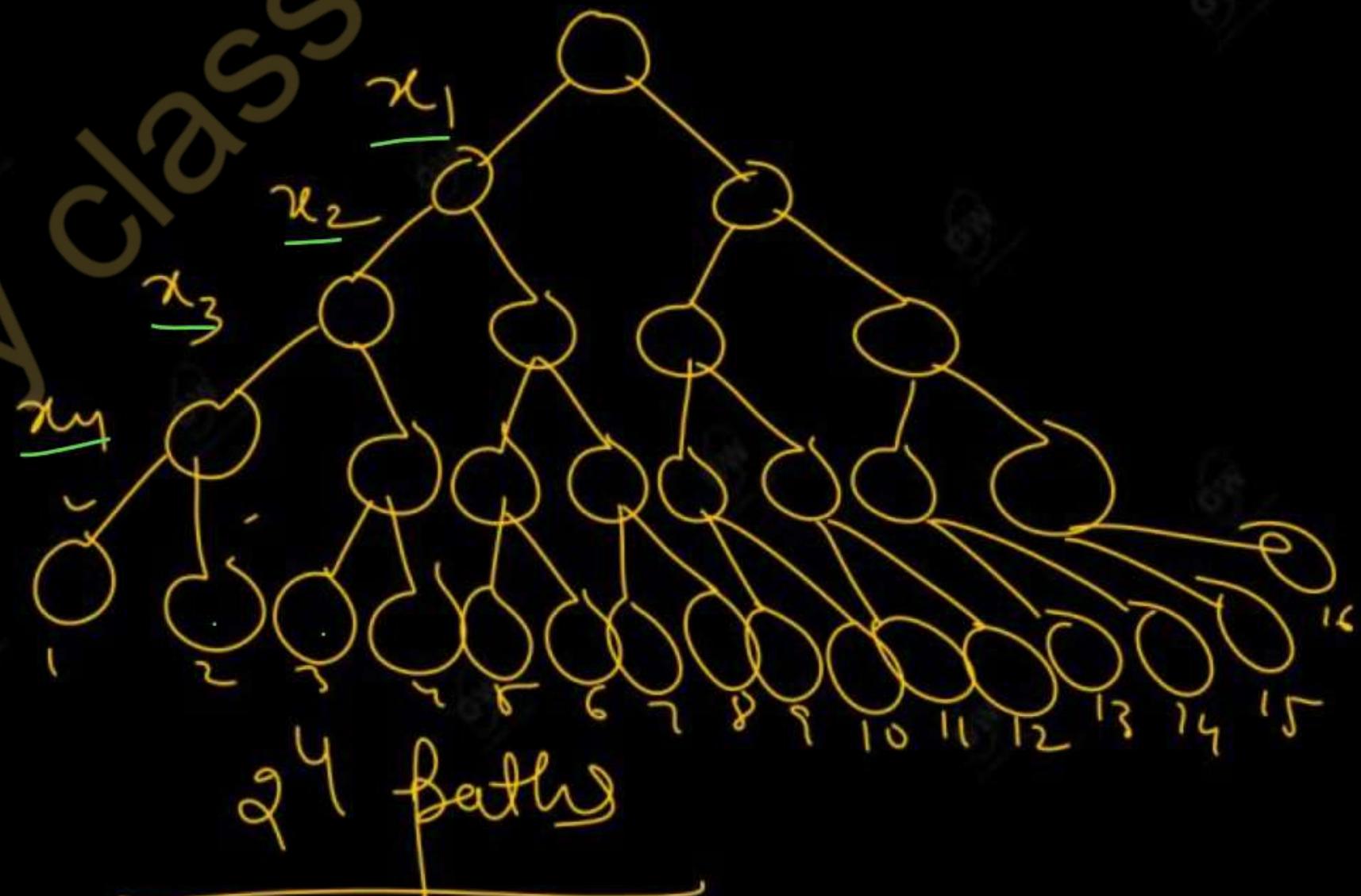
Brute force method



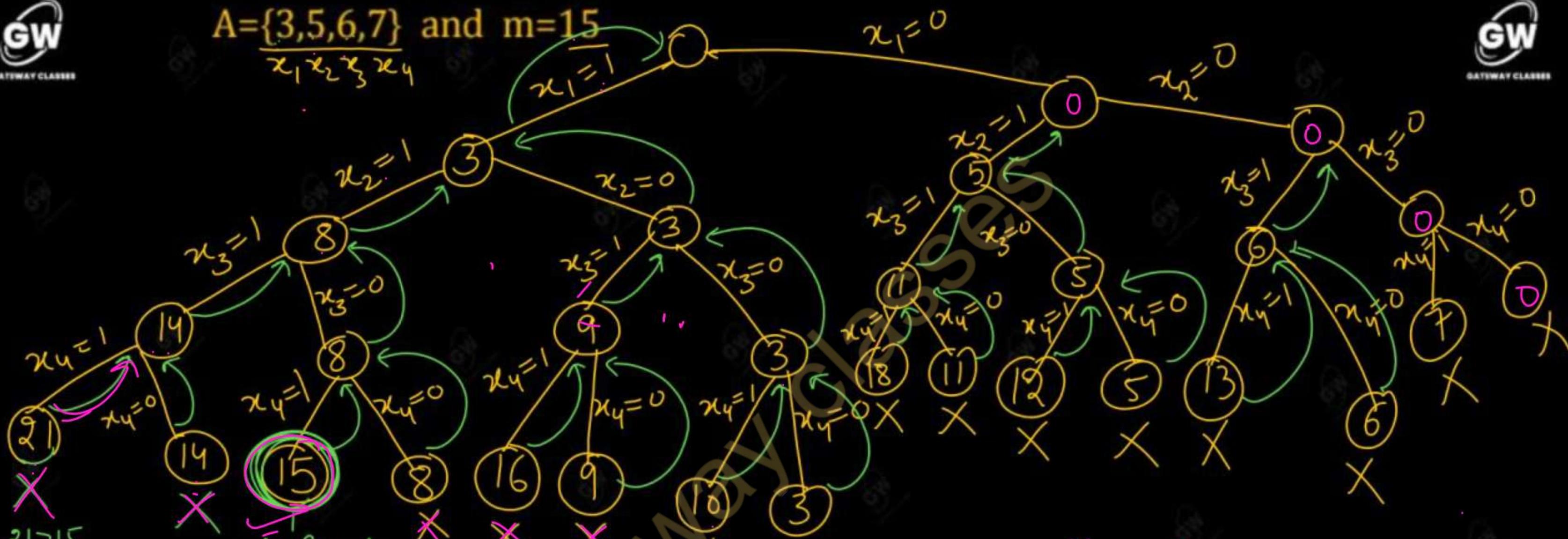
$$T(n) = O(2^n)$$

$$\omega[1:4] = \{n_1, n_2, n_3, n_4\}$$

GATEWAY classes



$A = \{3, 5, 6, 7\}$ and $m = 15$



desired
solution

Solution

$x[i] \xrightarrow{\text{not included}} 1$
 \downarrow
 $x[i] \xrightarrow{\text{Included}} 0$

$$X = \begin{bmatrix} 1 & 1 & 0 & 1 \end{bmatrix}$$

➤ We record the value of S, the sum of these numbers in the node.

If S is equal to 'm' then we have solution to the problem.

➤ We can either report this result and stop or, if all solutions need to be find, continue by backtracking to the mode's parent

➤ If S is not equal to M, we can terminate the node as a non-promising node. Following are the ✓ bounding conditions for a node to be a promising node.

for Kth number

S_p (sum of all the numbers added before Kth no.)

(i) $\boxed{\sum_{i=1}^{K-1} w_i x_i} + w_k \leq m$

↳ value of the current Number

S

or

(ii) $\boxed{\sum_{i=1}^{K-1} w_i x_i} + \boxed{\sum_{i=k}^n w_i} \geq m$

$$w[1:4] = \{ \textcolor{red}{5}, \textcolor{blue}{8}, \textcolor{brown}{10}, \textcolor{green}{13} \}_{x_1=1}$$



Two Solutions

$$\left. \begin{array}{l} 1) X = \boxed{1|0|0|1} \\ 2) X = \boxed{0|1|1|0} \end{array} \right\} \checkmark$$

Rules for writing Pseudocode

Assume that $w[1] \leq M$ and $\sum_{i=1}^n w_i \geq M$

First no. $\leq M$

Initially $S = 0$, $K = 1$ and $r = \sum_{i=1}^n w_i$

no being considered

Total sum of all numbers $\geq M$

At k th element $S = \sum_{i=1}^{K-1} w_i x_i$ and $r = \sum_{i=K}^n w_i$

Algorithm SumofSubsets (S, K, r)

{ //Algorithm for generating left child and evaluate

{
 x[K]=1; // Include the number
 if (S + w[K] = m) then
 Write (x[1...K]);
 else if (S + w[K] + w[K + 1] ≤ m) then
 SumofSubsets (S + w[K], K + 1, r - w[K])

//For generating right child and evaluate (Exclude x [K])

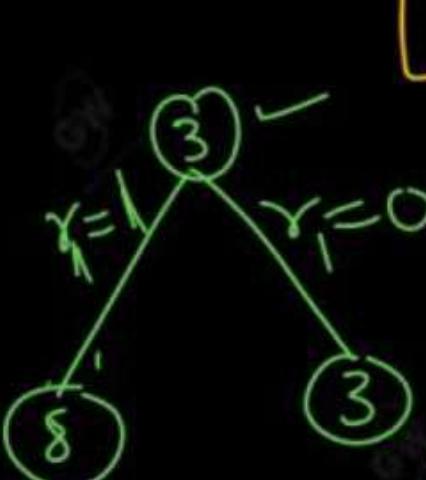
if ((S + r - w[K] ≥ m) and (S + w[K + 1] ≤ m)) then

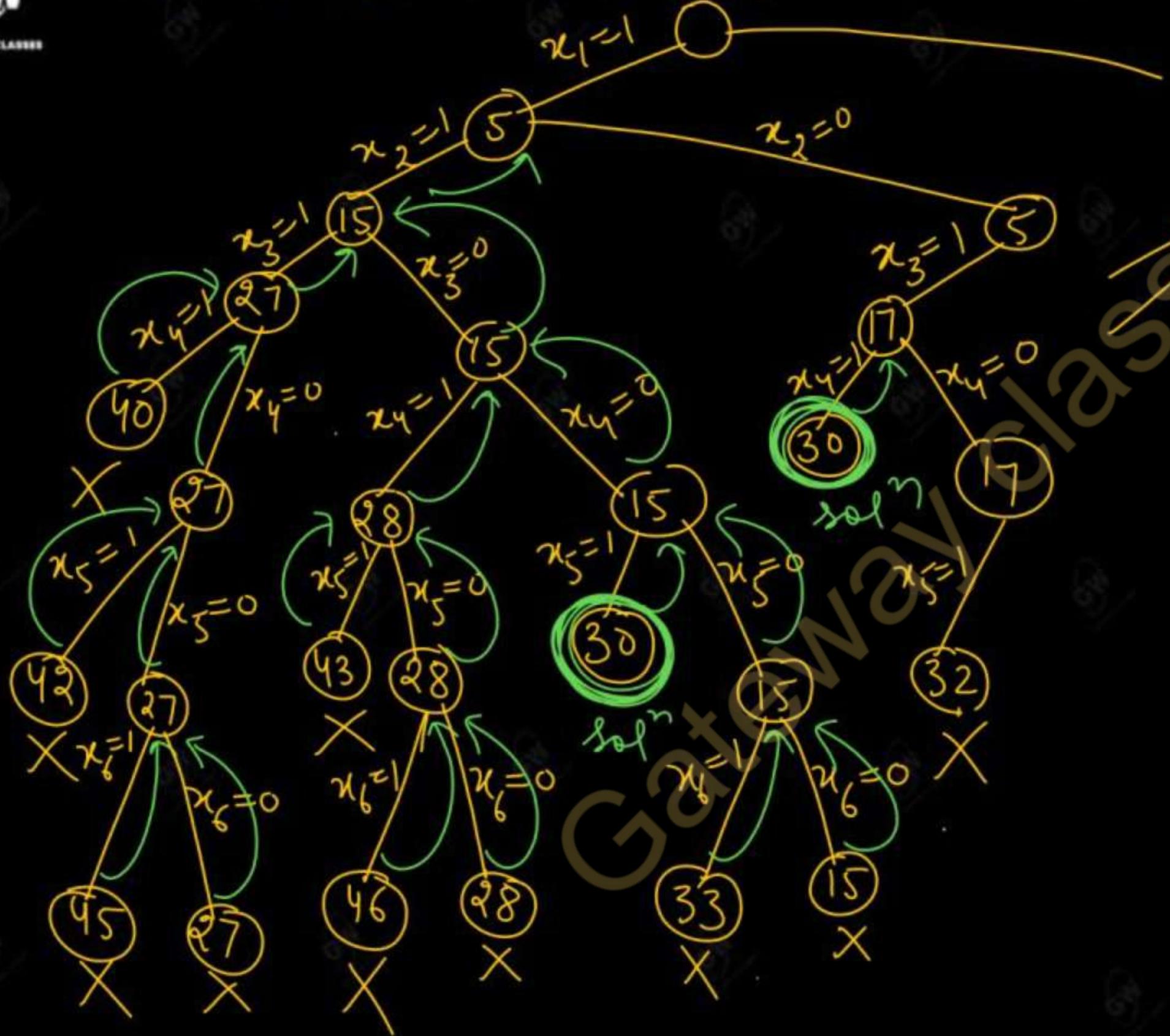
x[K] = 0 // Exclude the no.

 SumofSubsets (S, K + 1, r - w[K]);

}

}





Complete state space tree in similar way

$$\pi = \boxed{x_1 \quad x_2 \quad x_3 \quad x_4 \quad x_5 \quad x_6}$$

$$\pi = \boxed{1 \quad 1 \quad 0 \quad 0 \quad 1 \quad 0}$$

$$\pi = \boxed{0 \quad 0 \quad 1 \quad 0 \quad 0 \quad 1}$$

1. What is the sum of subsets problem? Let $w=\{5,7,10,12,15,18,20\}$ and $m=35$. Find all possible subsets of w that sum to m using recursive backtracking algorithm for it. Draw the portion of the state-space tree that is generated. (AKTU 2022-23)
2. What is the sum of subsets problem? Let $S=\{1,3,4,5\}$ and $X=8$. Find all possible subsets of S using backtracking. (AKTU 2021-22)
3. What is the sum of subsets problem? Let $n=6$, $m=30$ $w[1:6]=\{5,10,12,13,15,18\}$. Find all possible subsets using backtracking. Draw state space tree (AKTU 2023-24)
4. Solve sum of subsets problem using backtracking where $n=4$, $m=18$ $w[4]=\{5,10,8,13\}$ (AKTU 2018-19)



AKTU

B.Tech 5th Sem



CS IT & CS Allied

DAA : Design & Analysis Of Algorithm

Unit-4 : Lecture-8

Today's Target

- Solving Hamiltonian cycle problem using backtracking
- AKTU PYQs

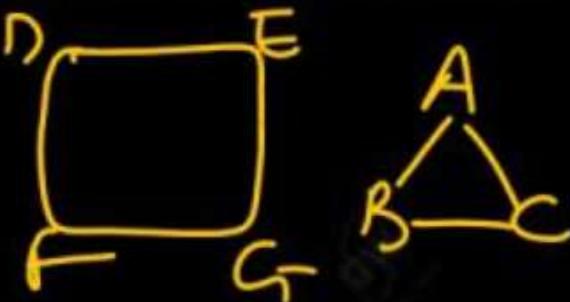


By Dr. Nidhi Parashar Ma'am

- M.Tech Gold Medalist
- Net Qualified

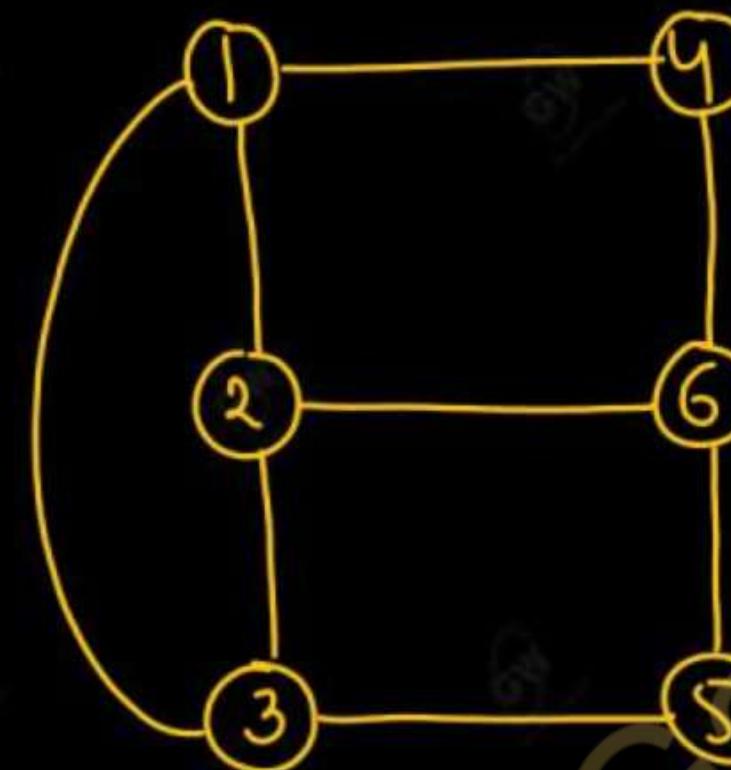
Hamiltonian Cycle

- In a given graph we start from some starting vertex and visit all remaining vertices of the graph exactly once and return back to the same starting vertex. The cycle formed is called Hamiltonian cycle.
- The problem is to check if Hamiltonian cycle is possible in a graph, and if possible, then what is a cycle, if multiple Hamiltonian cycles are present then we have to find all those cycles.
- The given graph may be directed /undirected.
- The graph must be connected (It should not have multiple correspondents).
- This is a NP – complete problem. Means it's exponential time taking problem.



Formal Definition

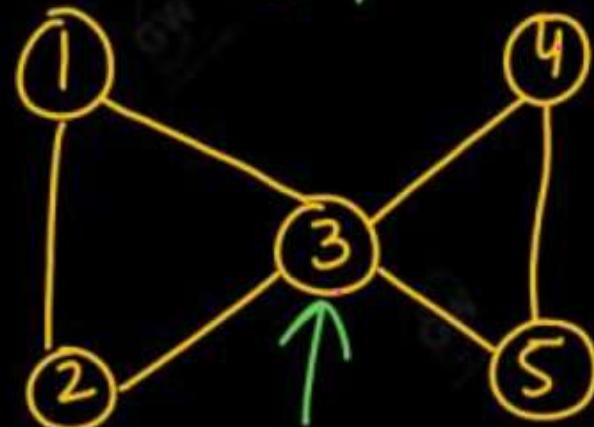
Hamiltonian cycle begins at some vertex $V \in G$ and vertices of G are visited in the order v_1, v_2, \dots, v_{n+1} then the edges (v_i, v_{i+1}) are in E , $1 \leq i \leq n$, and v_i are distinct except for v_1 and v_{n+1} which are equal.



- 1 → 2 → 3 → 5 → 6 → 4 → 1
- 1 → 4 → 6 → 5 → 3 → 2 → 1
- 2 → 1 → 4 → 6 → 5 → 3 → 2
- 3 → 5 → 2 → 6 → 4 → 1 → 3

same

(A)



\times Articulation points

For Complete Graph :-

(for n-nodes)

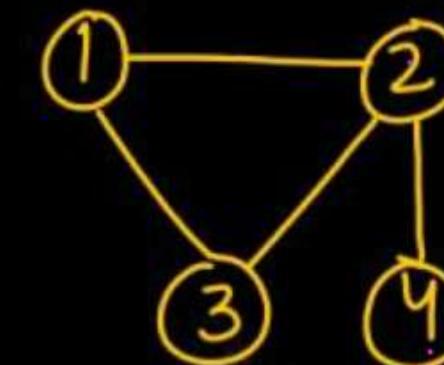
(B)



$1 \rightarrow 2 \rightarrow 5 \rightarrow 4 \rightarrow 3 \rightarrow 1$
 $1 \rightarrow 3 \rightarrow 4 \rightarrow 5 \rightarrow 2 \rightarrow 1$

2 hamiltonian cycles are present ✓

(C)

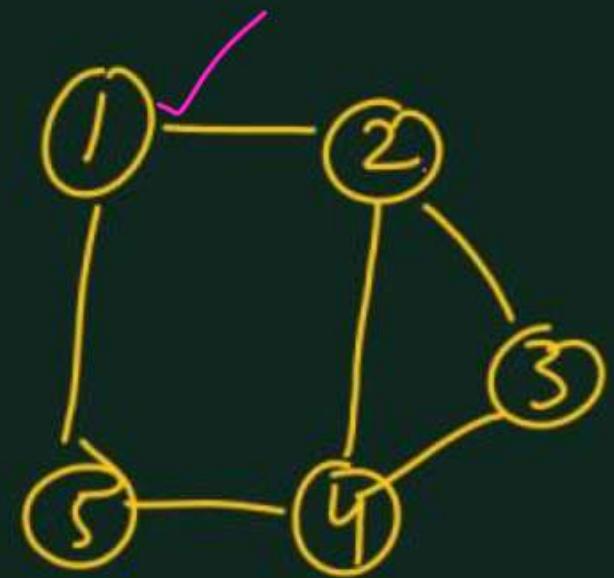


Pendant node

\times

No. of hamiltonian cycles = $\frac{(n-1)!}{2}$ (undirected graph)
 $= (n-1)!$ (directed graph)

Using Brute-force Approach

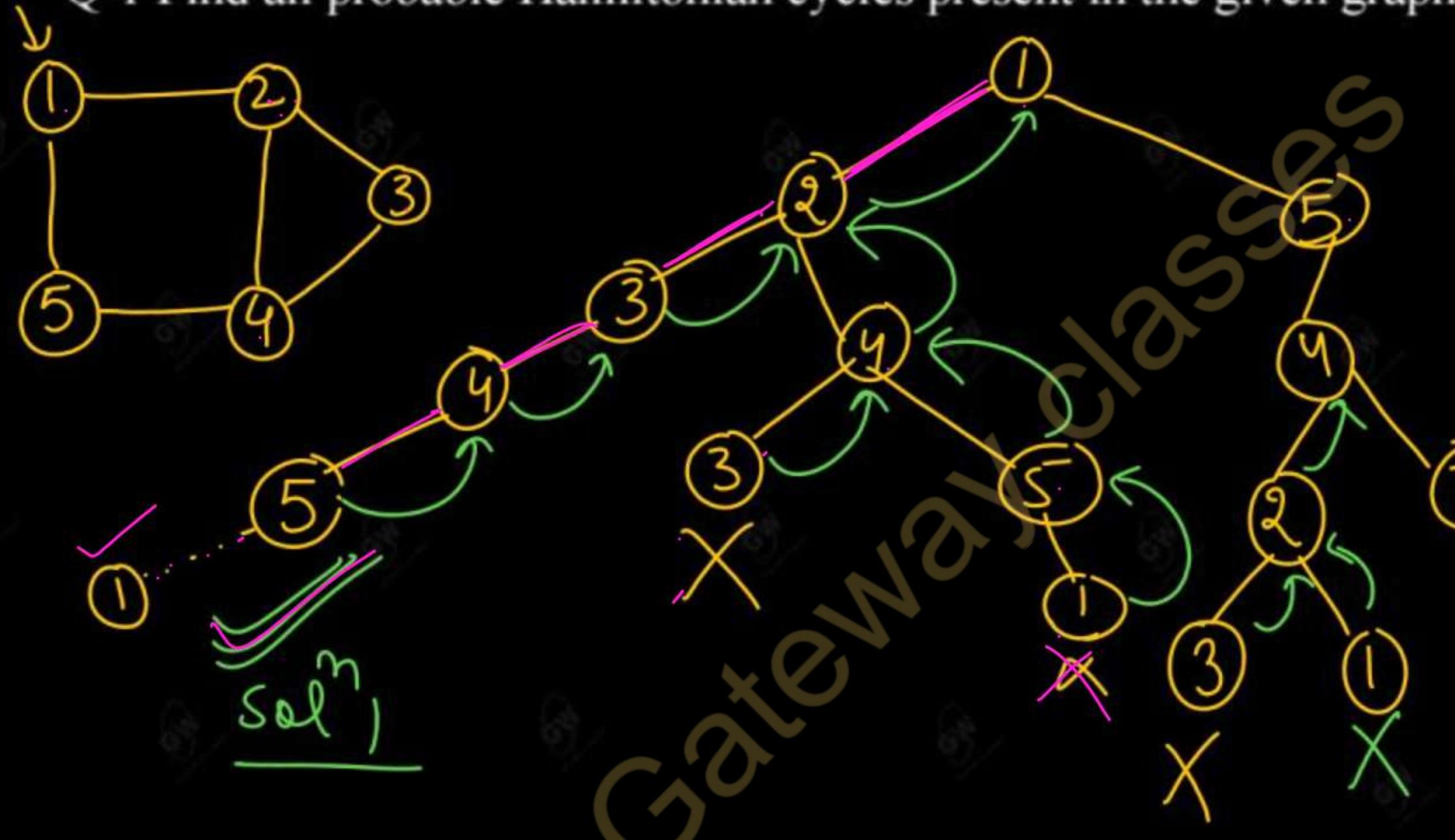


for n -nodes - $(n-1)! \approx n!$

$$T(n) = O(n^n)$$

Solving Hamiltonian cycle problem using backtracking

Q-1 Find all probable Hamiltonian cycles present in the given graph using backtracking



2 solutions

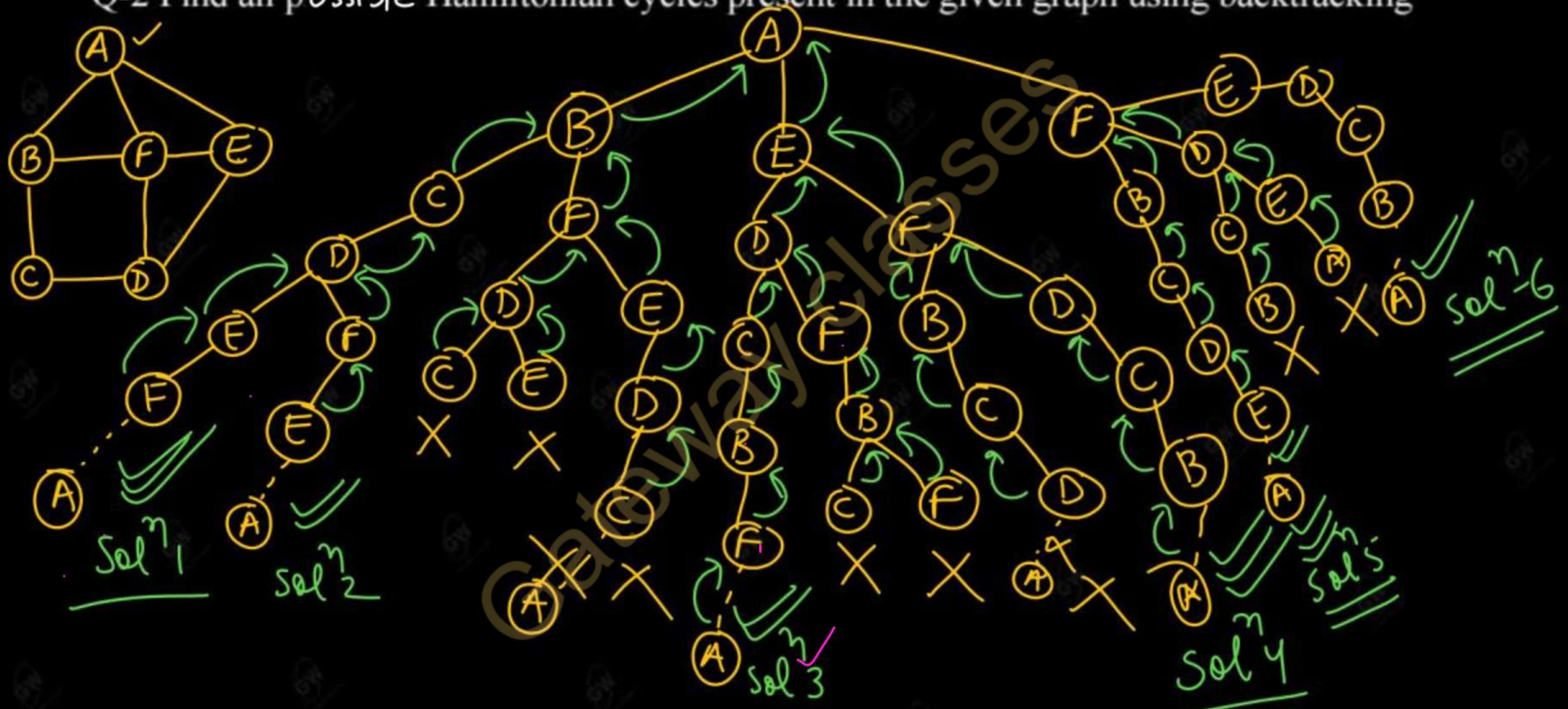
$$\pi = \boxed{1 | 2 | 3 | 4 | 5}$$

$$\pi = \boxed{1 | 5 | 4 | 3 | 2}$$

sol²

Solving Hamiltonian cycle problem using backtracking

Q-2 Find all possible Hamiltonian cycles present in the given graph using backtracking



Solutions

$$1) \quad x = \boxed{A | B | C | D | E | F}$$

$$2) \quad x = \boxed{A | B | C | D | F | E}$$

$$3) \quad x = \boxed{A | E | D | C | B | F}$$

$$4) \quad x = \boxed{A | E | F | D | C | B}$$

$$5) \quad x = \boxed{A | F | B | C | D | E}$$

$$6) \quad x = \boxed{A | F | E | B | C | D}$$

Backtracking: Algorithm for finding Hamiltonian cycle

Algorithm Hamiltonian (k) // finding all Hamiltonian cycles

{// graph G is stored as an adjacency matrix G [1 : n, 1:n]. All cycles begin at node 1.

Repeat

{

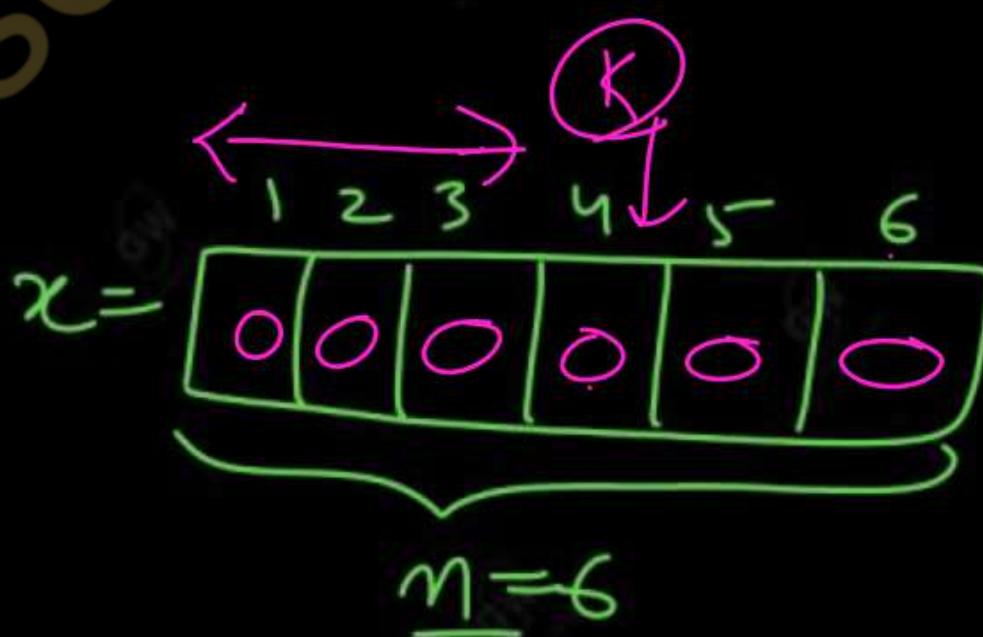
Next value (k)

if ($x[k] == 0$) then return // All vertices are explored

else Hamiltonian (k+1)

} until (false)

}



Note:

- $x[1:K - 1]$ is a path of $(k-1)$ distinct vertices.
- if $x[K] = 0$ means no vertex has yet been Assigned to $x [k]$
- After execution, $x [k]$ is assigned to the next highest numbered Vertex which doesn't already appear in $x[1:K - 1]$ and connected by an edge to $x[K - 1]$.

l — k₁ Kth $x [k]$

Algorithm Next value (K)

{

repeat

{

Initially $x[k] = 0$

$x[K] = (x[K]+1) \bmod (n+1)$ // Next vertex

If ($x[K] == 0$) then return // All vertices are explored.

If (($G[x[k-1], x[k]] \neq 0$) then // There is an edge b/w K^{th} and $k-1^{th}$ vertex

{

for $j=1$ to $k-1$ do

if ($x[j] == x[k]$) then break // condition for repeated

if ($j == k$) then this is the last vertex

if ($(k < n)$ or (($k == n$) and ($G[x[n], x[1]] \neq 0$))

Still some
vertices are remaining

then return

There is an edge between the
 n^{th} vertex and 1^{st} vertex ✓

} until (false)

}

Applications:-

- (1) May be used to model and optimize travel results for public transportsations and logistics.
- (2) Can be used to solve network and scheduling problems in computer science .
- (3) Solving travelling salesman problem.
- (4) Robotics
- (5) Urban planning.

- (1) Explain the method of finding Hamiltonian cycles in a graph using backtracking method. Explain with example. (AKTU 2021-22)



AKTU

B.Tech 5th Sem



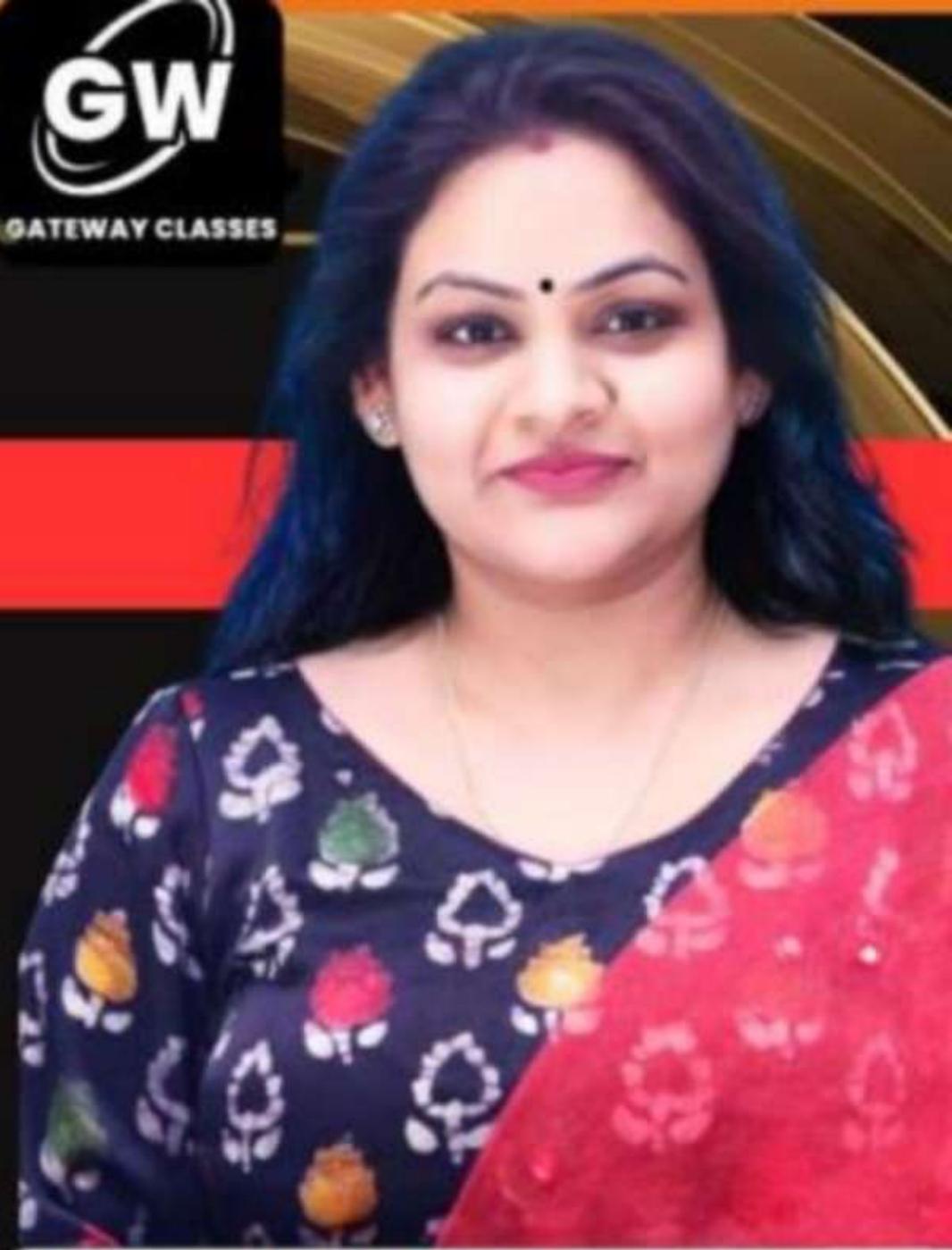
CS IT & CS Allied

DAA : Design & Analysis Of Algorithm

Unit-4 : Lecture-9

Today's Target

- Solving graph coloring problem using backtracking
- AKTU PYQs



By Dr. Nidhi Parashar Ma'am

- M.Tech Gold Medalist
- Net Qualified

- (1) Write applications of graph coloring. (AKTU 2022-23, 2021-22)
- (2) Define the term “Graph coloring”. (AKTU 2023-24)

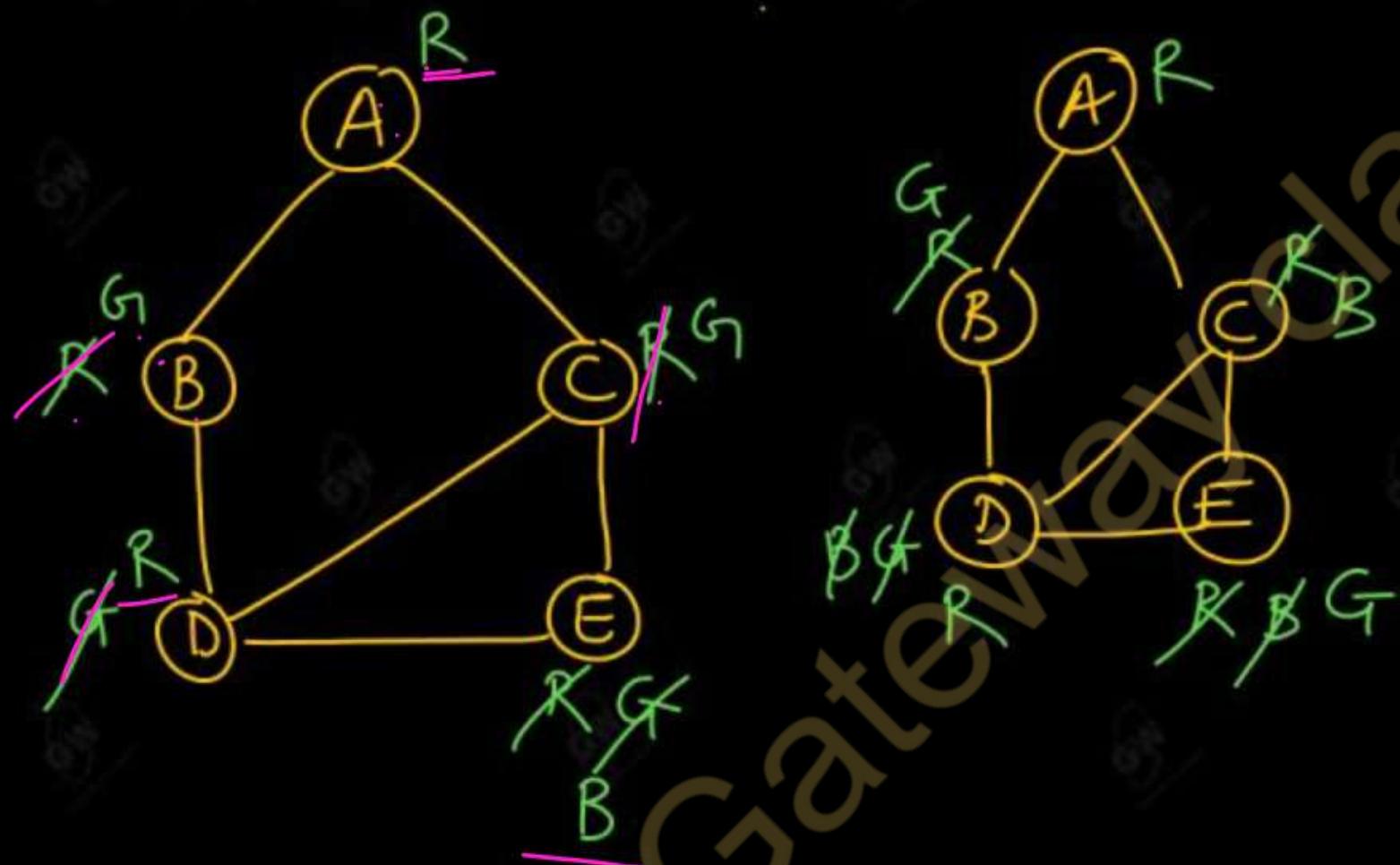
Graph Coloring

(m)

Problem – A graph is given and some colors are given. We have to color the vertices of graph such that no two adjacent vertices must have the same color.

Color = { R, G, B }

colors = { R, G, B }



Solutions

A B C D E

R G G R B

R G B R G

1),

2),

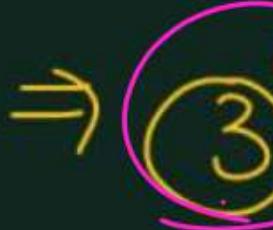
3),

;

Brute Force Approach

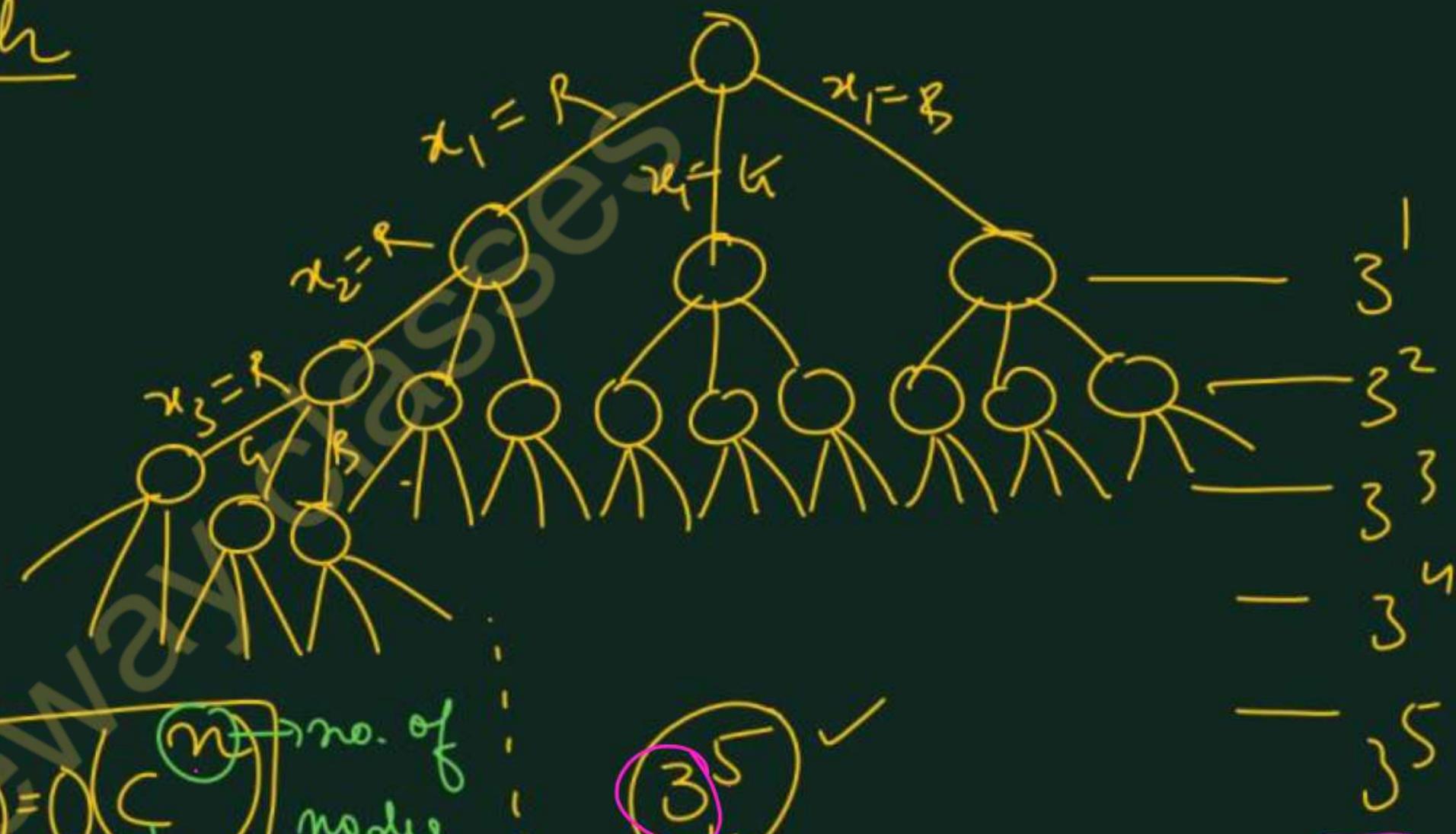
$$\frac{3}{\text{---}} \quad \frac{3}{\text{---}} \quad \frac{3}{\text{---}} \quad \frac{3}{\text{---}} \quad \underline{\underline{3}}$$

$$3 \times 3 \times 3 \times 3 \times 3$$

\Rightarrow  $\xrightarrow{\text{no. of nodes}}$
 $\xrightarrow{\text{no. of colors}}$

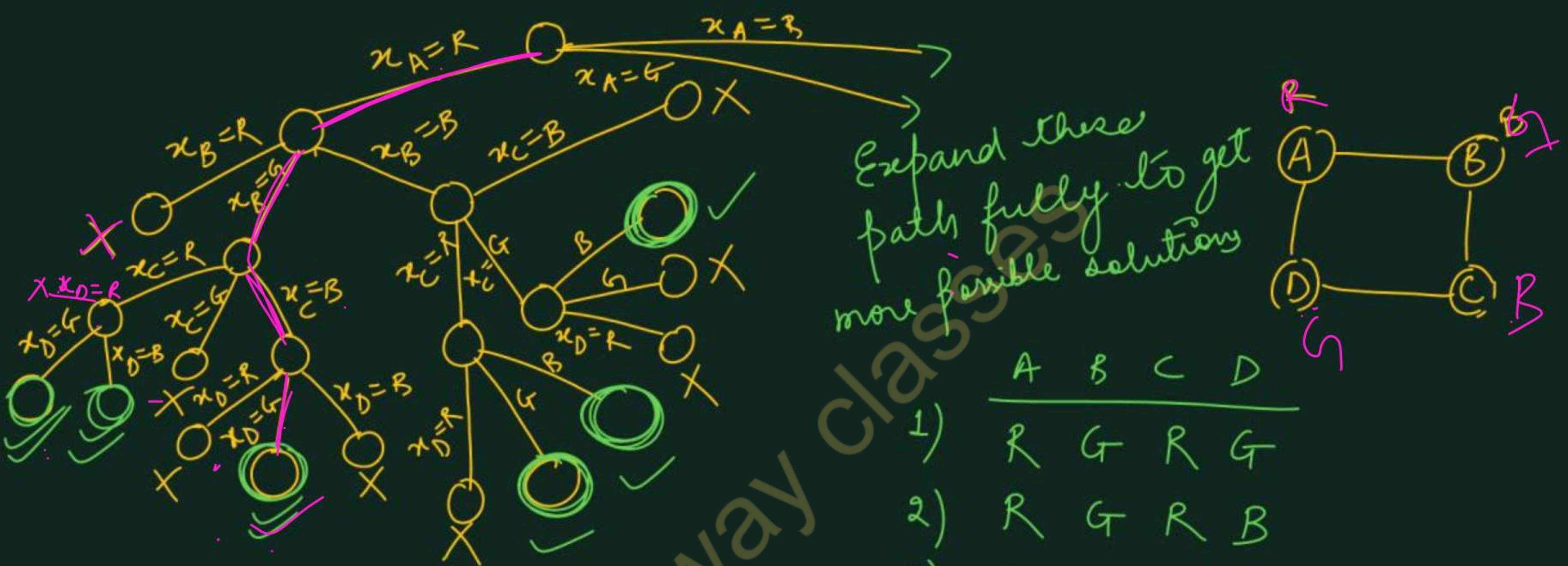
$$T(n) = O(c^n)$$

$\xrightarrow{\text{no. of nodes}}$
 $\xrightarrow{\text{no. of colors}}$





\checkmark



expand these
path fully to get
more possible solutions

- ! R
- 1) $\frac{A \quad B \quad C \quad D}{R \quad G \quad R \quad G}$
 - 2) $\frac{R \quad G \quad R \quad B}{R \quad G \quad R \quad B}$
 - 3) $\frac{R \quad G \quad B \quad G}{R \quad G \quad B \quad G}$
 - 4) $\frac{R \quad B \quad R \quad G}{R \quad B \quad R \quad G}$
 - 5) $\frac{R \quad B \quad R \quad B}{R \quad B \quad R \quad B}$
 - 6) $\frac{R \quad B \quad G \quad B}{R \quad B \quad G \quad B}$
and many more

Pseudo code

Algorithm mColoring (k)

{

Repeat

{

 Next value (K)

 If (x[K] == 0) then return

 If (K == n) then write (x [1:n])

 else

mColoring (k+1)

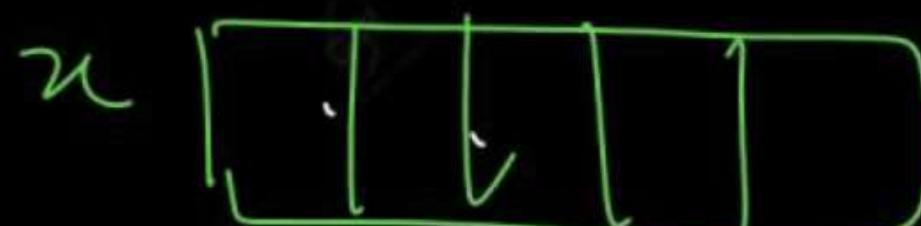
 } until (false)

}

Algorithm Next value (K) //Generating the next color

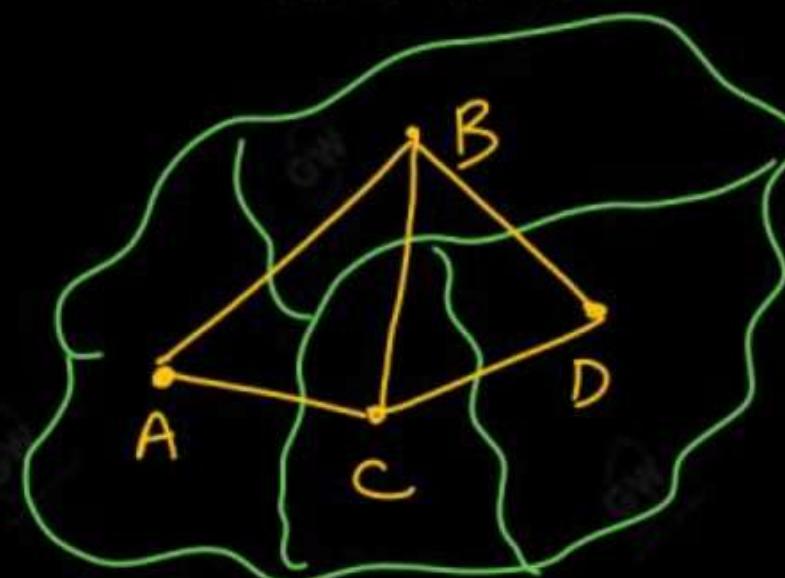
{
repeat ^{D initially}
{
 $X[K] = (x[K]+1) \text{ mod } (m+1)$ // Next highest node color
 If ($x[K] == 0$) then return // color exhausted ^{no more colors are remaining}
 For j = 1 to ^{No. of nodes} n do
 {
 If ($G[K, j] \neq 0$) and ($x[K] == x[j]$) then break. // adjacent ^{nodes} with some color.
 }
 If ($j == (n+1)$) then return // new color found otherwise find other
} until (false)
}

$$\text{No. of colors} = m$$



Applications- (Imp)

- (1) If we need to color a map of any nations so that all regions are visibly distinct, we can repeat color and using minimum colors we want to color the map. One way is that convert map into a graph and then apply graph coloring also.



- (2) In wireless communications for assigning frequencies to different devices so that no two adjacent devices use the same frequency.

- 3) Setting time schedules.

- (1) Write applications of graph coloring. (AKTU 2022-23, 2021-22)
- (2) Define the term “Graph coloring”. (AKTU 2023-24)



AKTU

B.Tech 5th Sem



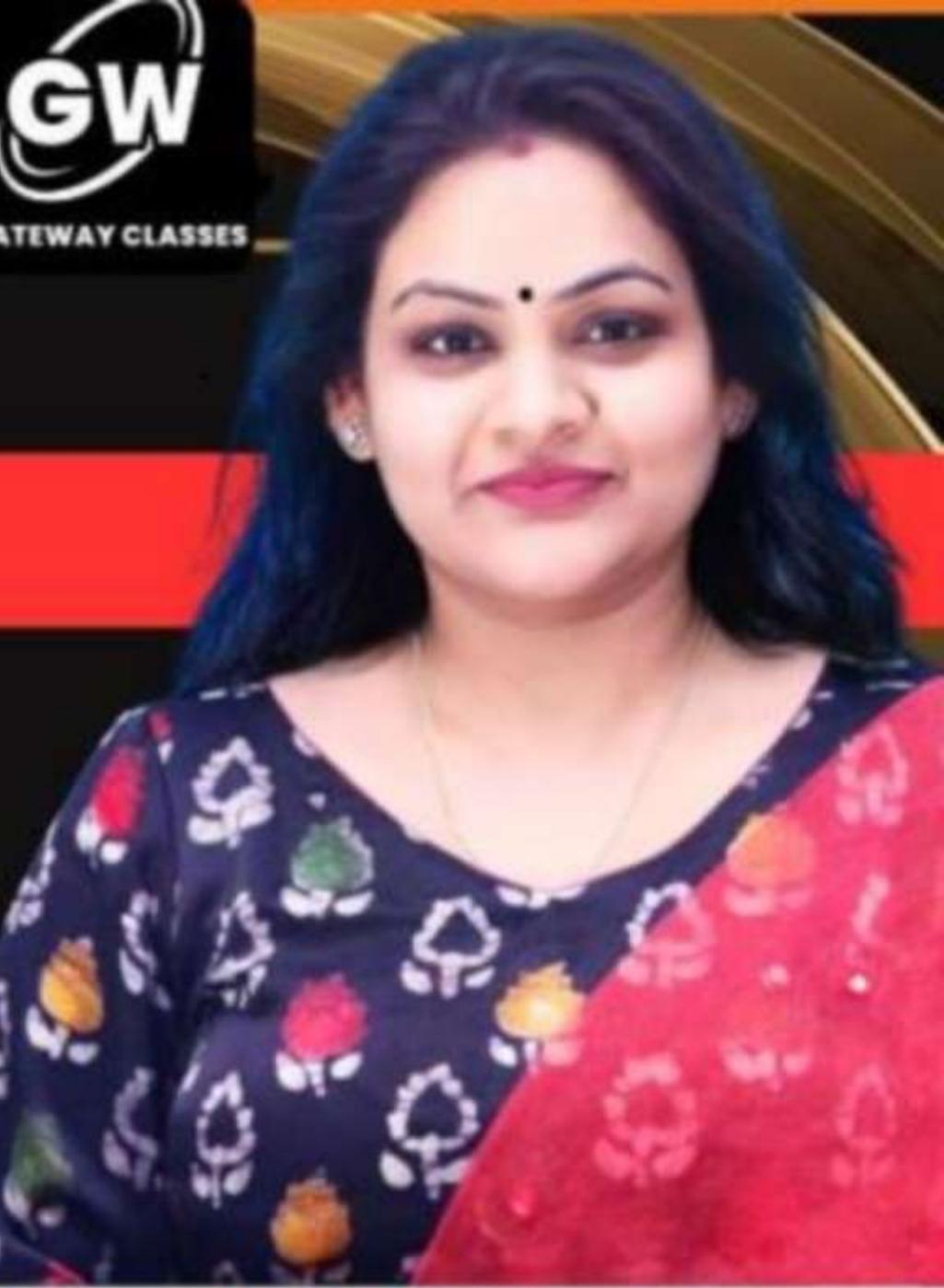
CS IT & CS Allied

DAA : Design & Analysis Of Algorithm

Unit-4 : Lecture-10

Today's Target

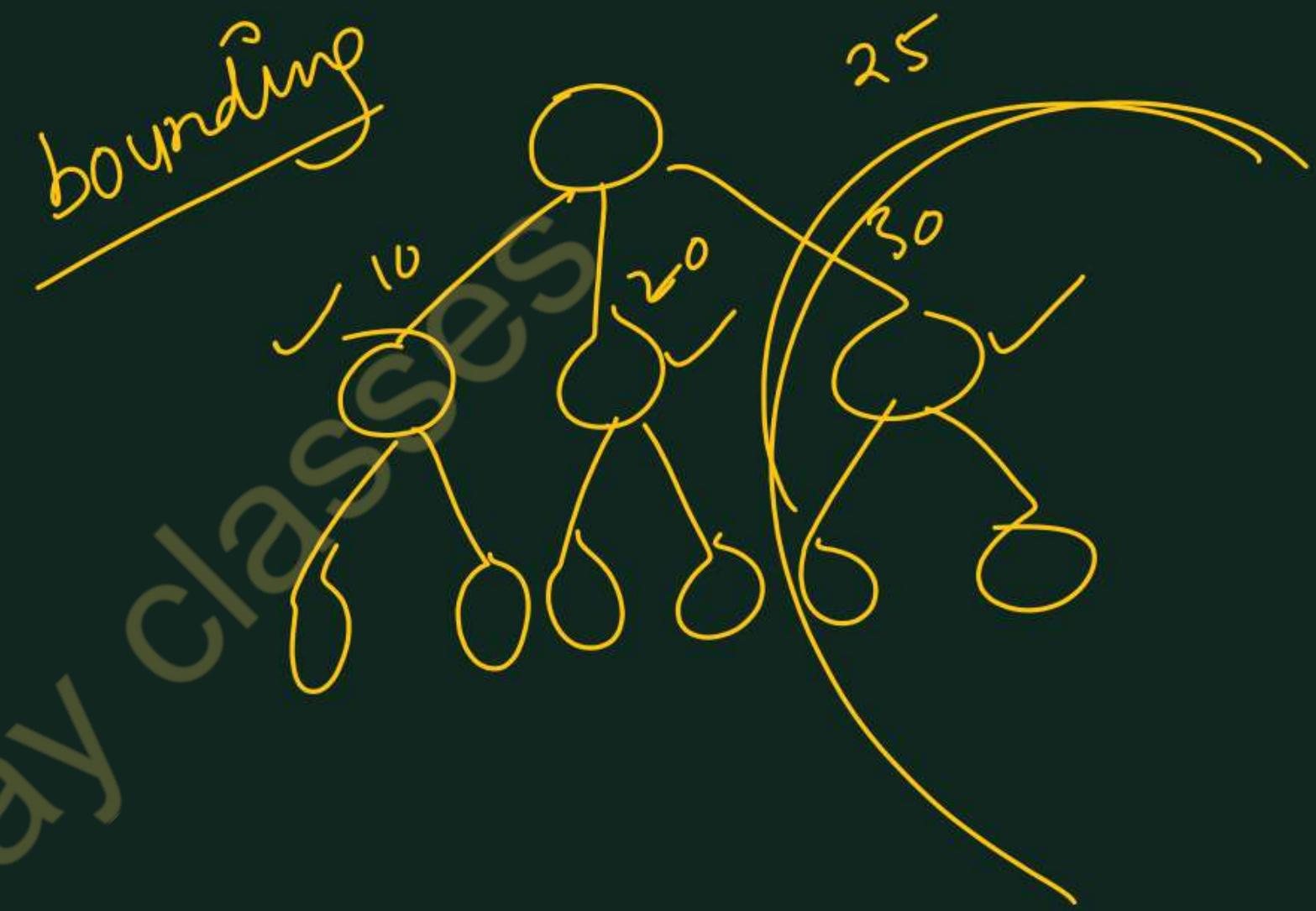
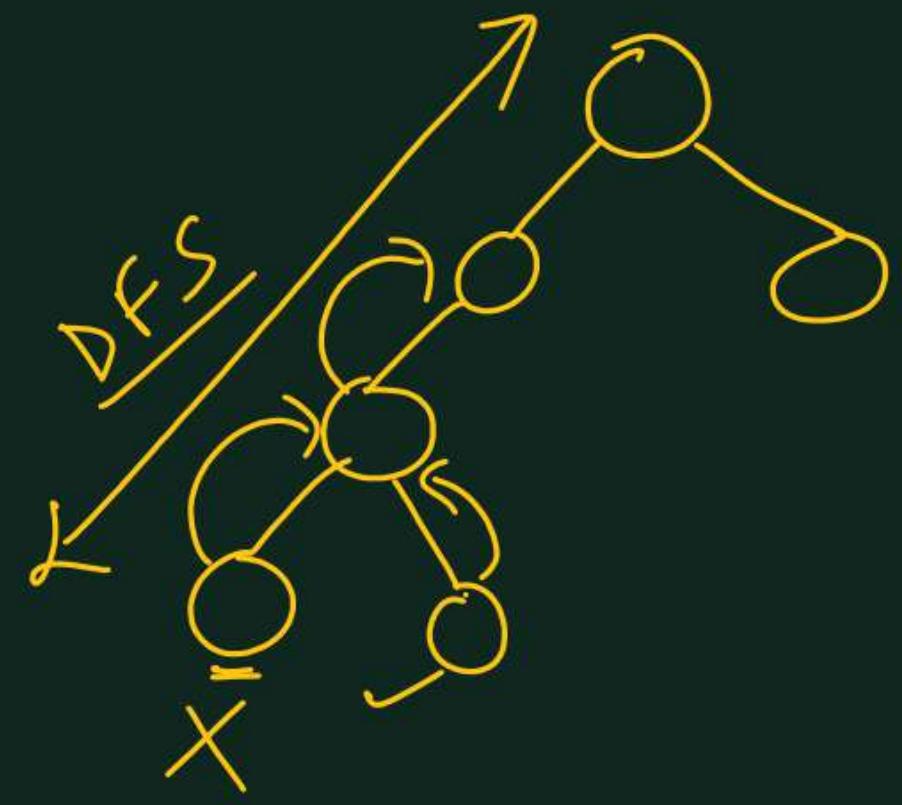
- Introduction to branch and bound ✓
- Difference between backtracking and branch and bound ✓
- Traveling salesman problem using branch and bound
- AKTU PYQs



By Dr. Nidhi Parashar Ma'am

- M.Tech Gold Medalist
- Net Qualified

- (1) Follows BFS (Breadth first search)
- (2) the idea is to cut off a ~~branch~~^{branch} of the problem's state – space tree as soon as we can deduce that it can not lead to a solution.
- (3) Similar to back tracking as it uses a state space tree. But branch and bound is used to solve optimization problems.
- (4) Compared to backtracking branch and bound requires two additional steps –
 - (a) for every node of a state – space tree, a bound on the best value of the objective function.
 - (b) Value of the best solution seen so far.



Branch and Bound:

- Branching: Original problem is partitioned into smaller subproblems(branches).
- Bounding: Compare the computed optimal solution with the best known solution.
- Elimination: Identify nodes which do not lead to the best solution and eliminate them.
- Selection: Exploration strategy used mainly is breadth first search, ,least cost search etc..
- Example: Job sequencing, Travelling salesman problem

FIFO

Three types of nodes		
Live node	E - node	Dead node
A node that has been generated but its children are not yet generated.	Live node whose children are currently being explored.	Generated node that is not to be expanded any further.

~~Advantages of Branch and Bound Algorithm~~

- It can reduce the time complexity by avoiding unnecessary exploration of the state space tree.
- It has different search techniques(DFS,BFS, least cost search) that can be used for different types of problems.

~~Disadvantages of Branch and Bound Algorithm~~

- In the worst case scenario, it may search for all the combinations to produce solutions.
- It can be time consuming if the state space tree is too large.

Parameter	Backtracking	Branch and Bound
Approach	Write definitions	Write definitions
Traversal	traverses the state space tree by <u>DFS</u> <u>(Depth First Search)</u> manner.	traverse the tree by <u>BFS</u> (breadth first search).
Function	Involves feasibility function.	Involves a bounding function.
Problems	used for solving Decision Problem. (YES/NO)	used for solving Optimization Problem.
Applications	Useful in solving <u>N-Queen Problem</u> , <u>Sum of subset</u> , <u>Hamiltonian cycle problem</u> , <u>graph coloring problem</u>	Useful in solving <u>Knapsack Problem</u> , <u>Travelling Salesman Problem</u> .
Next move	Next move from current state can lead to bad choice.	Next move is always towards better solution.
Solution	On successful search of solution in state space tree, search stops.	Entire state space tree is search in order to find optimal solution.

Solution of Traveling Salesman using Branch and Bound.

Step-1 Reduce matrix row-wise and col-wise

(AKTU-2016-17)

∞	20	30	10	11
15	∞	16	4	2
3	5	∞	2	4
19	6	18	∞	3
16	4	7	16	∞

∞	10	20	0	1
13	∞	14	2	0
1	3	∞	0	2
16	3	15	∞	0
12	0	3	12	∞

1	0	3	0	0
4	0	3	0	4

reduction cost = $21 + 4 = 25$

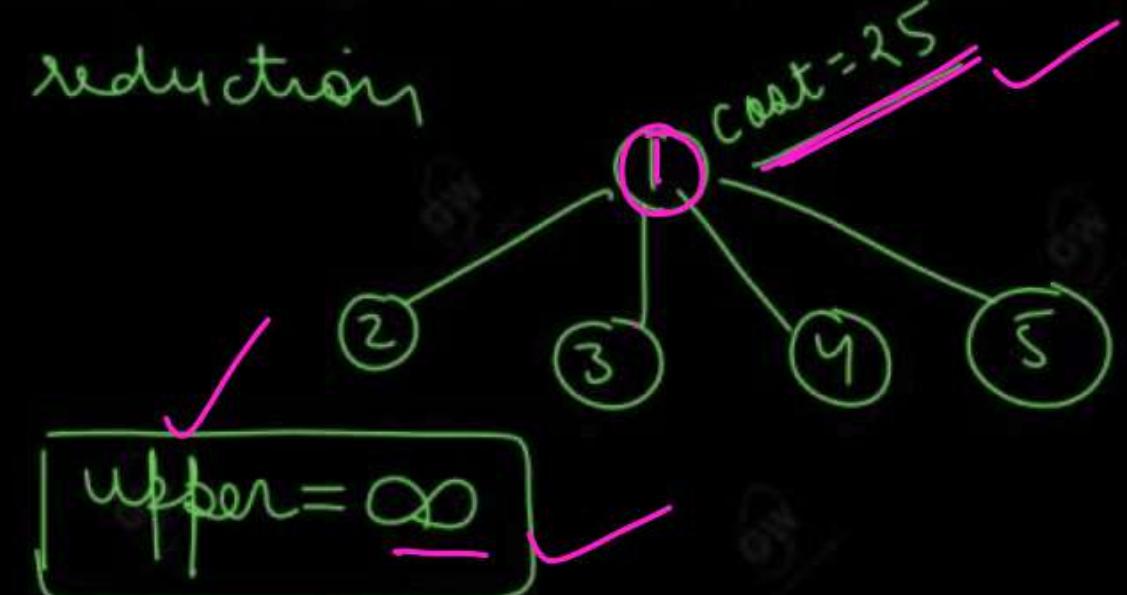
∞	10	17	0	1
12	∞	11	2	0
0	3	∞	0	2
15	3	∞	0	2
11	0	12	∞	0

This cost matrix

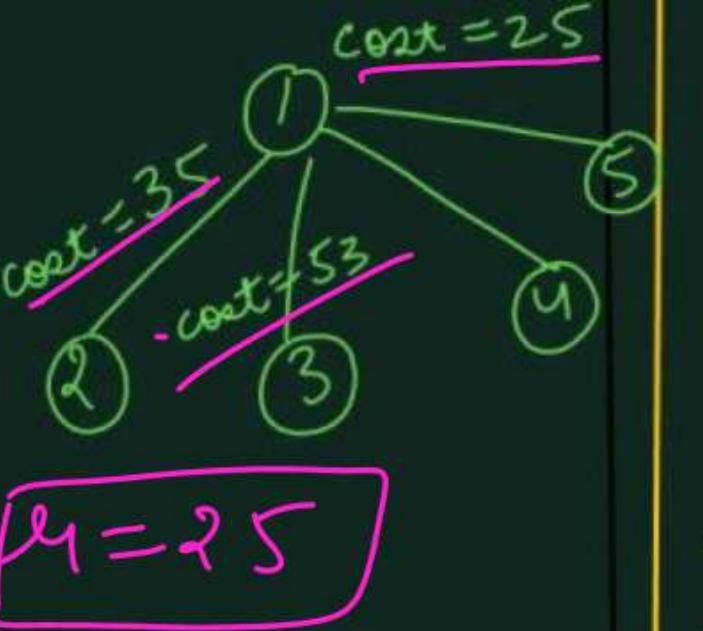
will be used to prepare
cost matrix for
 $\begin{matrix} 1 \rightarrow 2 \\ 1 \rightarrow 3 \\ 1 \rightarrow 4 \\ 1 \rightarrow 5 \end{matrix}$

1	2	3	4	5
∞	20	30	10	11
15	∞	16	4	2
3	5	∞	2	4
19	6	18	∞	3
16	4	7	16	∞

col-wise
reduction



	1	2	3	4	5
1	∞	10	17	0	1
2	12	∞	11	2	0
3	0	3	∞	0	2
4	15	3	12	∞	0
5	11	0	0	12	∞



To calculate $\text{cost}(1, 2) \rightarrow$ Make first row and 2nd col $\Rightarrow \infty$, and we don't need to return to node 1 from node 2, so we have to make the entry for $(2 \rightarrow 1)$ as ∞ .

	1	2	3	4	5
1	∞	∞	∞	∞	∞
2	∞	∞	11	2	0
3	0	∞	∞	0	2
4	15	∞	12	∞	0
5	11	0	0	12	∞

\Rightarrow This matrix is already

reduced.

$\hat{r}_1 = 0$

So $\text{cost}(1, 2) = \text{cost}(1, 2) + r_1 + \hat{r}_1 \rightarrow$ reduction of reductions cost of reference Matrix

$\text{cost}(1, 2) = \frac{10 + 25 + 0}{-} = 35$

To calculate $\text{cost}(1, 3) \rightarrow$ set 1st row and 3rd col $\Rightarrow \infty$ and $(3 \rightarrow 1)$ entry $\Rightarrow \infty$

	1	2	3	4	5
1	∞	∞	∞	∞	∞
2	12	∞	∞	2	0
3	0	3	∞	0	2
4	15	3	∞	∞	0
5	11	0	∞	12	∞

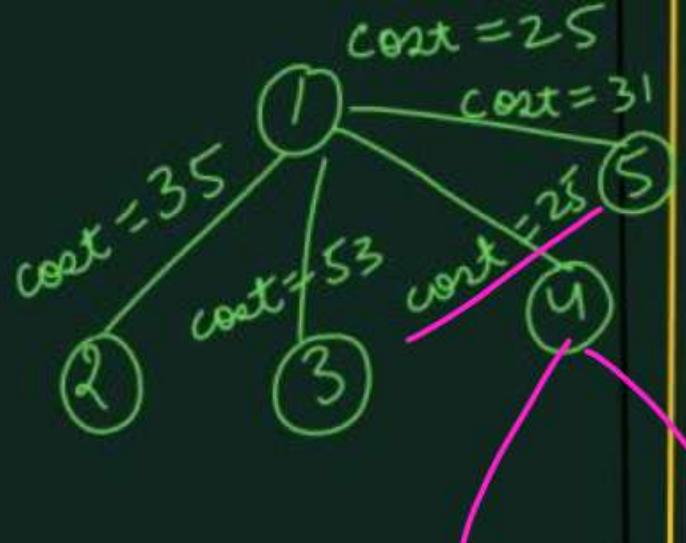
Rowwise reduced

∞	∞	∞	∞	∞
1	∞	2	0	
∞	3	0	2	
4	3	∞	0	
0	0	12	∞	

$\hat{r}_1 = 11$

$\text{cost}(1, 3) = \text{cost}(1, 3) + r_1 + \hat{r}_1$
 $= 17 + 25 + 11 \Rightarrow 53$

	1	2	3	4	5
1	∞	10	17	<u>0</u>	1
2	12	∞	11	2	0
3	0	3	∞	0	2
4	15	3	12	∞	0
5	11	0	0	12	∞



For $\text{cost}(1, 4) \Rightarrow [1^{\text{st}} \text{ row}] \Rightarrow \infty$

~~1 2 3 4 5~~ ($4 \rightarrow 1$) entry $\Rightarrow \infty$

	1	2	3	4	5
1	∞	∞	∞	∞	∞
2	12	∞	11	∞	0
3	0	3	∞	0	2
4	∞	3	12	∞	0
5	11	0	0	∞	∞

$$\text{cost}(1, 4) = 0 + 25 + 0 = 25$$

$\text{cost}(1, 5) \Rightarrow [1^{\text{st}} \text{ row}] \Rightarrow \infty$

($5 \rightarrow 1$) entry $\Rightarrow \infty$

	1	2	3	4	5
1	∞	∞	∞	∞	∞
2	12	∞	11	2	0
3	0	3	∞	0	∞
4	15	3	12	∞	∞
5	∞	0	0	12	∞

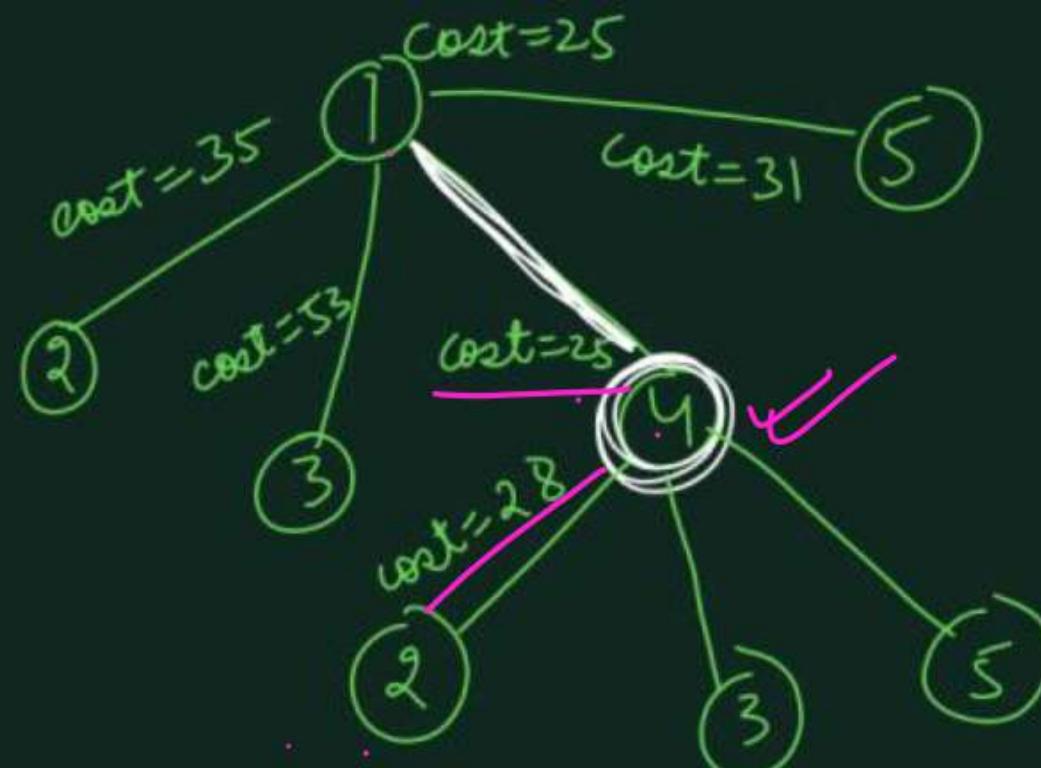
$$\hat{r}_1 = 5$$

$\hat{s}_1 = 5$

$$\text{cost}(1, 5) = \text{cost}(1, 5) + \hat{r}_1 + \hat{s}_1$$

$$= 1 + 25 + 5 = 31 \quad \checkmark$$

It is col-wise
already reduced.



$\text{cost}(1,4)$ matrix will be used to calculate the $\text{cost}(4,2)$, $\text{cost}(4,3)$ and $\text{cost}(4,5)$

	1	2	3	4	5
1	∞	∞	∞	∞	∞
2	12	∞	11	∞	0
3	0	3	∞	∞	2
4	∞	3	12	∞	0
5	11	0	0	∞	∞

$r_1 = 25$

$\text{cost}(4,2)$ \Rightarrow { 4th row } $\Rightarrow \infty$

$\text{cost}(4,2)$ \Rightarrow { 2nd col } $\Rightarrow \infty$

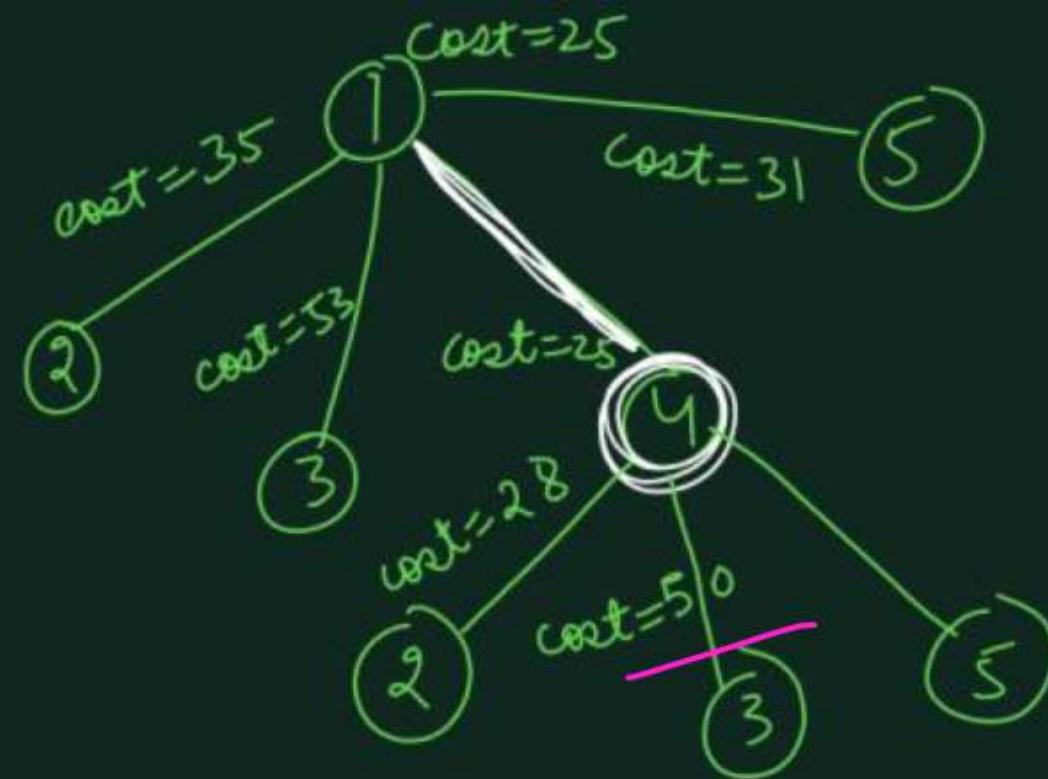
$(2 \rightarrow 1) \Rightarrow \infty$

$\begin{bmatrix} 1 & 2 & 3 & 4 & 5 \\ 1 & \infty & \infty & \infty & \infty \\ 2 & \infty & \infty & 11 & \infty & 0 \\ 3 & 0 & \infty & \infty & \infty & 2 \\ 4 & \infty & \infty & \infty & \infty & \infty \\ 5 & 11 & 0 & 0 & \infty & \infty \end{bmatrix}$

\Rightarrow already reduced

$\hat{r}_1 = 0$

$$\begin{aligned}
 \text{cost}(4,2) &= \text{cost}(4,2) + r_1 + \hat{r}_1 \\
 &= 3 + 25 + 0 \\
 &= 28
 \end{aligned}$$



$\text{cost}(1,4)$ matrix will be used to calculate the $\text{cost}(4,2)$, $\text{cost}(4,3)$ and $\text{cost}(4,5)$

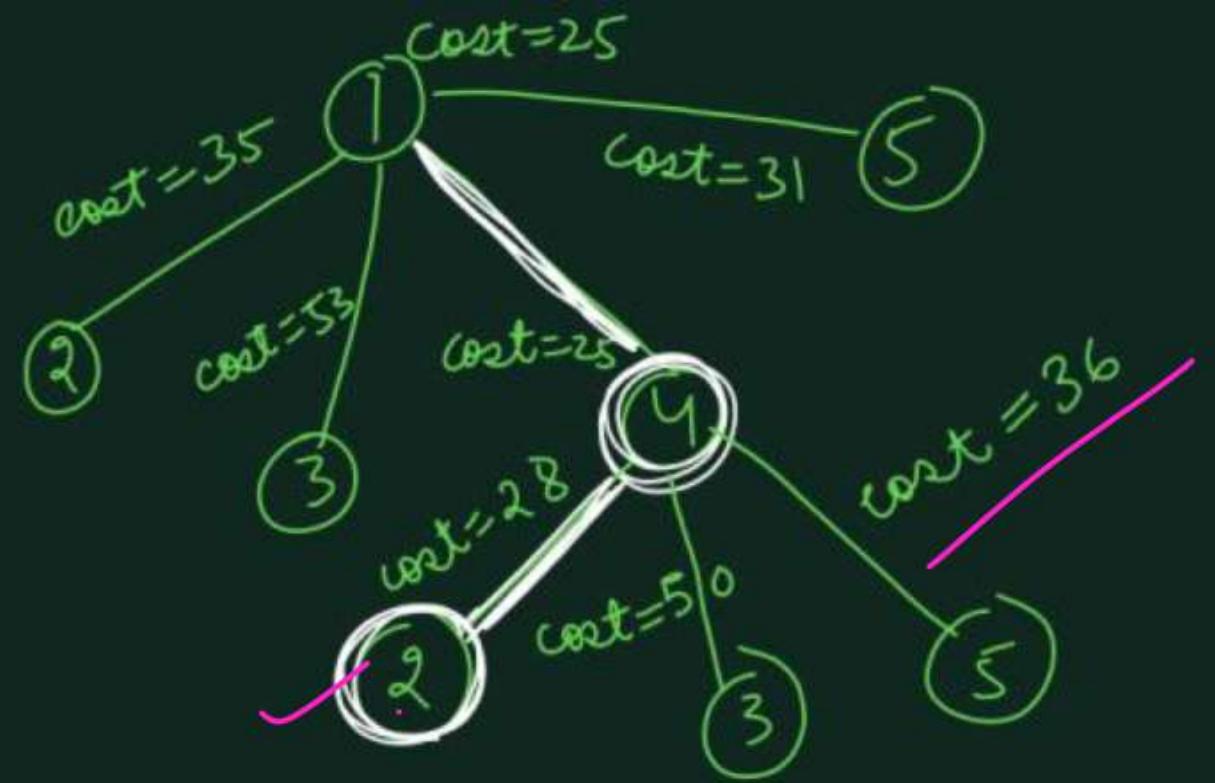
$$\begin{matrix}
 & 1 & 2 & 3 & 4 & 5 \\
 \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{matrix} & \left[\begin{array}{cccccc} \infty & \infty & \infty & \infty & \infty \\ 12 & \infty & 11 & \infty & 0 \\ 0 & 3 & \infty & \infty & 2 \\ \infty & 3 & 12 & \infty & 0 \\ 11 & 0 & 0 & \infty & \infty \end{array} \right] & \xrightarrow{\text{R1} = 25}
 \end{matrix}$$

$\text{cost}(4,3) \Rightarrow$ 4th row } $\Rightarrow \infty$
 3rd col } $\Rightarrow \infty$
 $(3 \rightarrow 1)$ entry $\Rightarrow \infty$

$$\begin{matrix}
 & 1 & 2 & 3 & 4 & 5 \\
 \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{matrix} & \left[\begin{array}{ccccc} \infty & \infty & \infty & \infty & \infty \\ 12 & \infty & \infty & \infty & 0 \\ \infty & 1 & \infty & \infty & 0 \\ \infty & \infty & \infty & \infty & \infty \\ 11 & 0 & \infty & \infty & 0 \end{array} \right] & \xrightarrow{\text{R2} = 11} \\
 & 1 & 2 & 3 & 4 & 5 \\
 \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{matrix} & \left[\begin{array}{ccccc} \infty & \infty & \infty & \infty & \infty \\ 12 & \infty & \infty & \infty & 0 \\ \infty & 1 & \infty & \infty & 0 \\ \infty & \infty & \infty & \infty & \infty \\ 11 & 0 & 0 & 0 & 0 \end{array} \right] & \xrightarrow{\text{R3} = 11}
 \end{matrix}$$

$$\begin{aligned}
 \text{cost}(4,3) &= \underline{12 + 25 + 13} \\
 &= \boxed{50}
 \end{aligned}$$

$$\begin{matrix}
 & 1 & 2 & 3 & 4 & 5 \\
 \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{matrix} & \left[\begin{array}{ccccc} \infty & \infty & \infty & \infty & \infty \\ 1 & \infty & \infty & \infty & 0 \\ \infty & 1 & \infty & \infty & 0 \\ \infty & \infty & \infty & \infty & \infty \\ 0 & 0 & \infty & \infty & 0 \end{array} \right]
 \end{matrix}$$



$\text{cost}(1,4)$ matrix will be used to calculate the $\text{cost}(4,2)$, $\text{cost}(4,3)$ and $\text{cost}(4,5)$

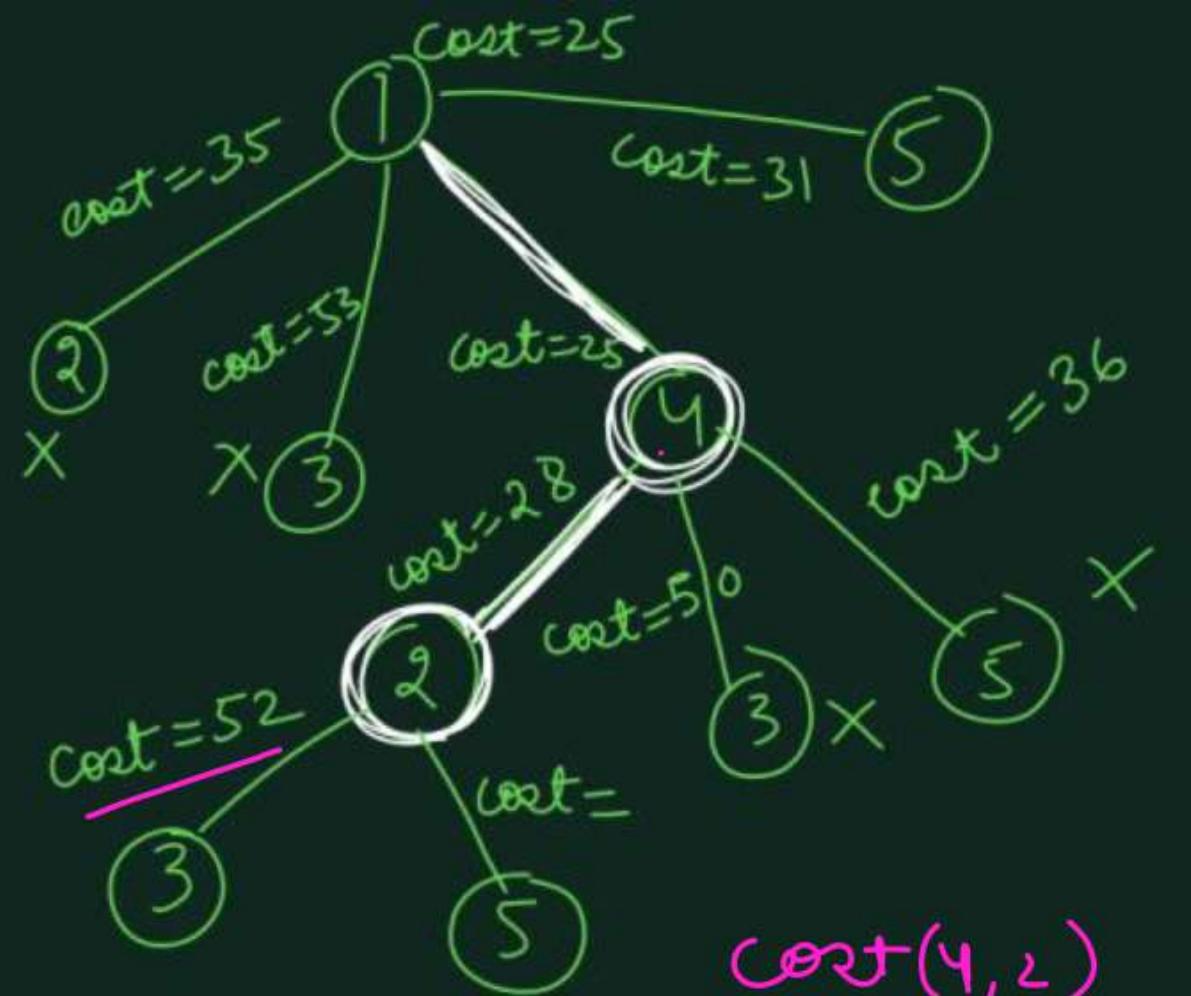
$$\begin{matrix}
 & 1 & 2 & 3 & 4 & 5 \\
 \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{matrix} &
 \left[\begin{array}{ccccc}
 \infty & \infty & \infty & \infty & \infty \\
 12 & \infty & 11 & \infty & 0 \\
 0 & 3 & \infty & \infty & 2 \\
 \infty & 3 & 12 & \infty & 0 \\
 11 & 0 & 0 & \infty & \infty
 \end{array} \right] &
 \xrightarrow{\text{r}_1 = 25}
 \end{matrix}$$

$\text{cost}(4,5) \Rightarrow \left. \begin{matrix} 4^{\text{th}} \text{ row} \\ 5^{\text{th}} \text{ col} \end{matrix} \right\} \Rightarrow \infty$
 $(5 \rightarrow 1)$ entry $\Rightarrow \infty$

$$\begin{matrix}
 & 1 & 2 & 3 & 4 & 5 \\
 \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{matrix} &
 \left[\begin{array}{ccccc}
 \infty & \infty & \infty & \infty & \infty \\
 12 & \infty & 11 & \infty & 0 \\
 0 & 3 & \infty & \infty & 2 \\
 \infty & \infty & \infty & \infty & 0 \\
 11 & 0 & 0 & \infty & \infty
 \end{array} \right] &
 \xrightarrow{\text{R}_4 \rightarrow 0} \left[\begin{array}{ccccc}
 \infty & \infty & \infty & \infty & \infty \\
 12 & \infty & 11 & \infty & 0 \\
 0 & 3 & \infty & \infty & 2 \\
 0 & 0 & 0 & \infty & 0 \\
 0 & 0 & 0 & 0 & 0
 \end{array} \right] \xrightarrow{\text{R}_1 \rightarrow 0} \left[\begin{array}{ccccc}
 0 & 0 & 0 & 0 & 0 \\
 12 & \infty & 11 & \infty & 0 \\
 0 & 3 & \infty & \infty & 2 \\
 0 & 0 & 0 & \infty & 0 \\
 0 & 0 & 0 & 0 & 0
 \end{array} \right] \xrightarrow{\text{R}_1 \rightarrow 0} \left[\begin{array}{ccccc}
 0 & 0 & 0 & 0 & 0 \\
 0 & 0 & 0 & 0 & 0 \\
 0 & 0 & 0 & 0 & 0 \\
 0 & 0 & 0 & 0 & 0 \\
 11 & 0 & 0 & 0 & 0
 \end{array} \right]
 \end{matrix}$$

$\boxed{\lambda = 11}$

$$\begin{aligned}
 \text{cost}(4,5) &= 0 + 25 + 11 \\
 &= \boxed{36}
 \end{aligned}$$



Matrix

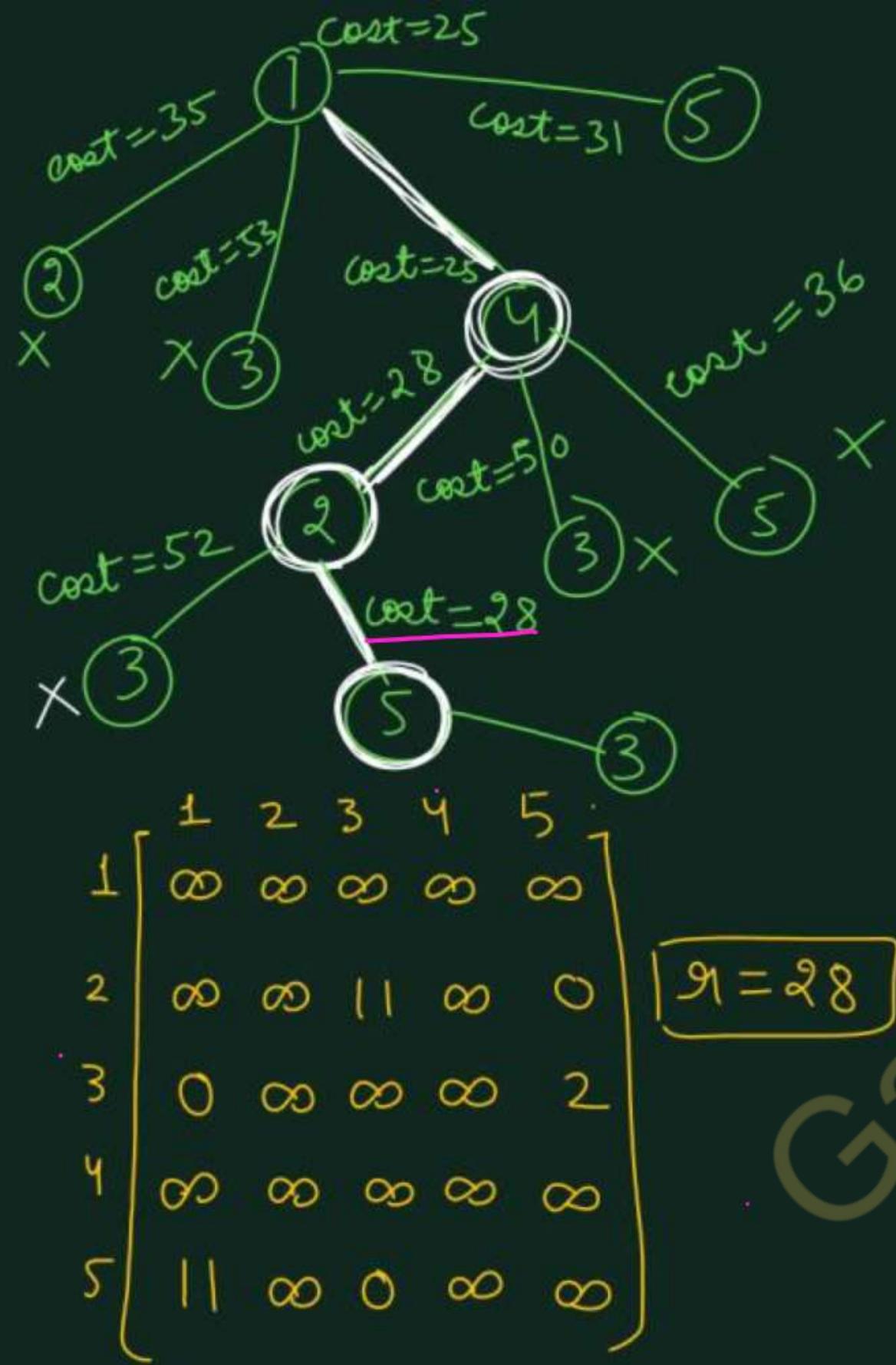
	1	2	3	4	5
1	∞	∞	∞	∞	∞
2	∞	∞	11	∞	0
3	0	∞	∞	∞	2
4	∞	∞	∞	∞	∞
5	11	∞	0	∞	∞

$\boxed{r_1 = 28}$

$\text{Cost}(2,3) \Rightarrow \left. \begin{array}{l} 2^{\text{nd}} \text{ row} \\ 3^{\text{rd}} \text{ col} \end{array} \right\} \Rightarrow \infty$
 $(3 \rightarrow 1) \text{ entry} \Rightarrow \infty$

$$\begin{matrix} & 1 & 2 & 3 & 4 & 5 \\ 1 & \cancel{\infty} & \cancel{\infty} & \cancel{\infty} & \cancel{\infty} & \cancel{\infty} \\ 2 & \cancel{\infty} & \cancel{\infty} & \cancel{\infty} & \cancel{\infty} & \cancel{\infty} \\ 3 & \cancel{\infty} & \cancel{\infty} & \cancel{\infty} & \cancel{\infty} & 2 \\ 4 & \cancel{\infty} & \cancel{\infty} & \cancel{\infty} & \cancel{\infty} & \cancel{\infty} \\ 5 & 11 & \cancel{\infty} & 0 & \cancel{\infty} & \cancel{\infty} \end{matrix} \xrightarrow{\hat{r}_1 = 13} \begin{matrix} & 1 & 2 & 3 & 4 & 5 \\ 1 & \cancel{\infty} & \cancel{\infty} & \cancel{\infty} & \cancel{\infty} & \cancel{\infty} \\ 2 & \cancel{\infty} & \cancel{\infty} & \cancel{\infty} & \cancel{\infty} & \cancel{\infty} \\ 3 & \cancel{\infty} & \cancel{\infty} & \cancel{\infty} & \cancel{\infty} & 0 \\ 4 & \cancel{\infty} & \cancel{\infty} & \cancel{\infty} & \cancel{\infty} & \cancel{\infty} \\ 5 & 0 & \cancel{\infty} & 0 & \cancel{\infty} & \cancel{\infty} \end{matrix}$$

$$\begin{aligned}
 \text{Cost}(2,3) &= \text{cost}(2,3) + \hat{r}_1 + \hat{r}_1 \\
 &= 11 + 28 + 13 \\
 &= \boxed{52}
 \end{aligned}$$

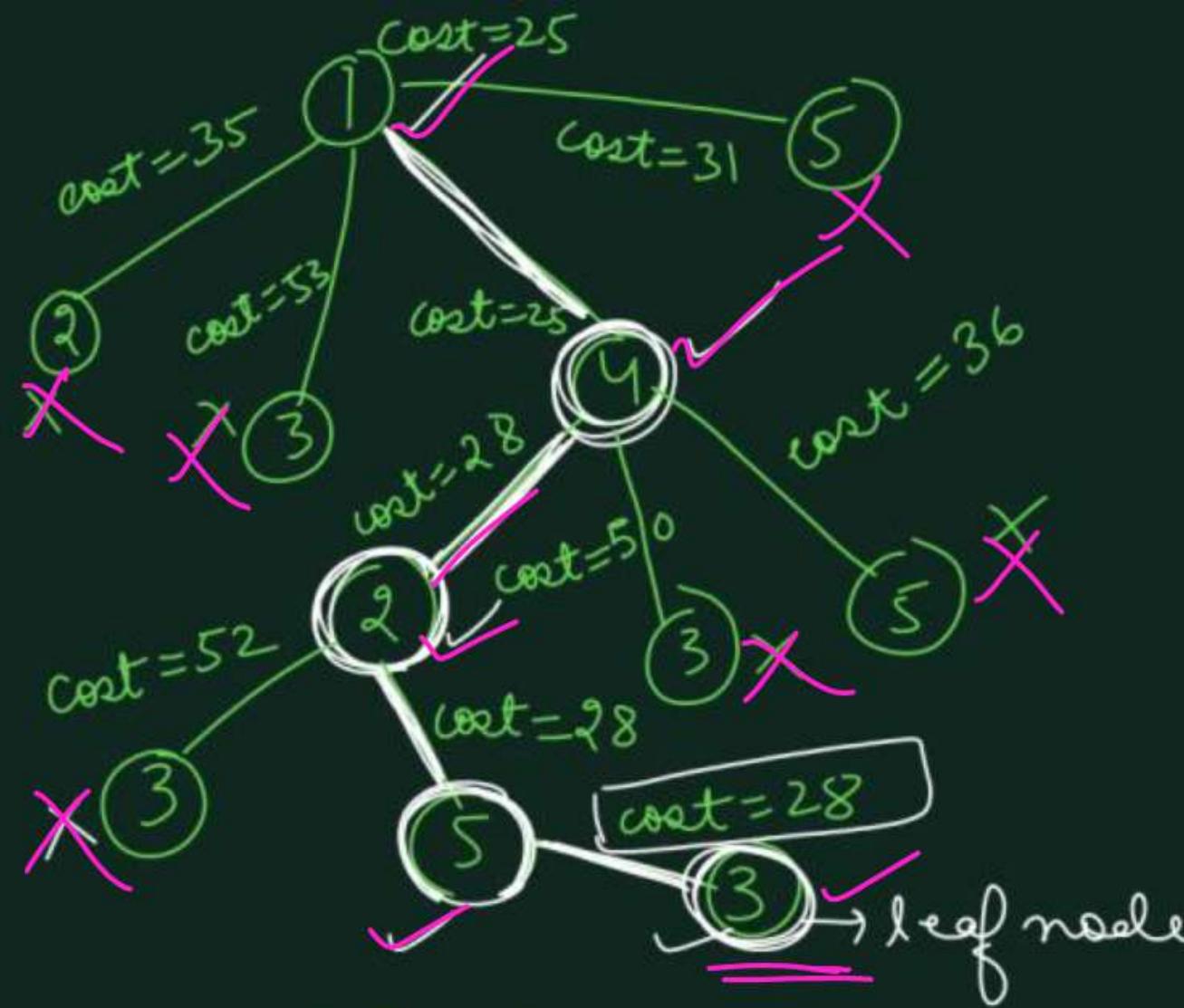


$\text{Cost}(2, 5) \Rightarrow \left. \begin{array}{l} 2^{\text{nd}} \text{ row} \\ 5^{\text{th}} \text{ col} \end{array} \right\} \rightarrow \infty$
 $(5 \rightarrow 1) \text{ entry} \rightarrow \infty$

$\begin{matrix} & 1 & 2 & 3 & 4 & 5 \\ 1 & \infty & \infty & \infty & \infty & \infty \\ 2 & \infty & \infty & \infty & \infty & \infty \\ 3 & 0 & \infty & \infty & \infty & 0 \\ 4 & \infty & \infty & \infty & \infty & \infty \\ 5 & \infty & \infty & 0 & \infty & \infty \end{matrix}$

$\hat{s}_1 = 0$

$\text{Cost}(2, 5) = 0 + 28 + 0 \Rightarrow \boxed{28}$



$\text{cost}(5, 3)$

$\left. \begin{matrix} 5^{\text{th}} \text{ row} \\ 3^{\text{rd}} \text{ col} \end{matrix} \right) \Rightarrow \infty$

$(3 \rightarrow 1) \Rightarrow \infty$

$$\frac{\text{cost}(2 \rightarrow 5)}{=} \boxed{r = 28}$$

	1	2	3	4	5
1	∞	∞	∞	∞	∞
2	∞	∞	∞	∞	∞
3	0	∞	∞	0	0
4	∞	∞	0	0	∞
5	∞	0	0	∞	∞

$$\frac{\text{cost}(5, 3)}{=}$$

	1	2	3	4	5
1	∞	∞	∞	∞	∞
2	∞	∞	∞	∞	∞
3	∞	∞	∞	∞	∞
4	∞	∞	∞	∞	∞
5	∞	∞	∞	∞	∞

: $\text{cost}(5, 3) = 0 + 28 + 0$

$= 28$

$\boxed{r = 0}$

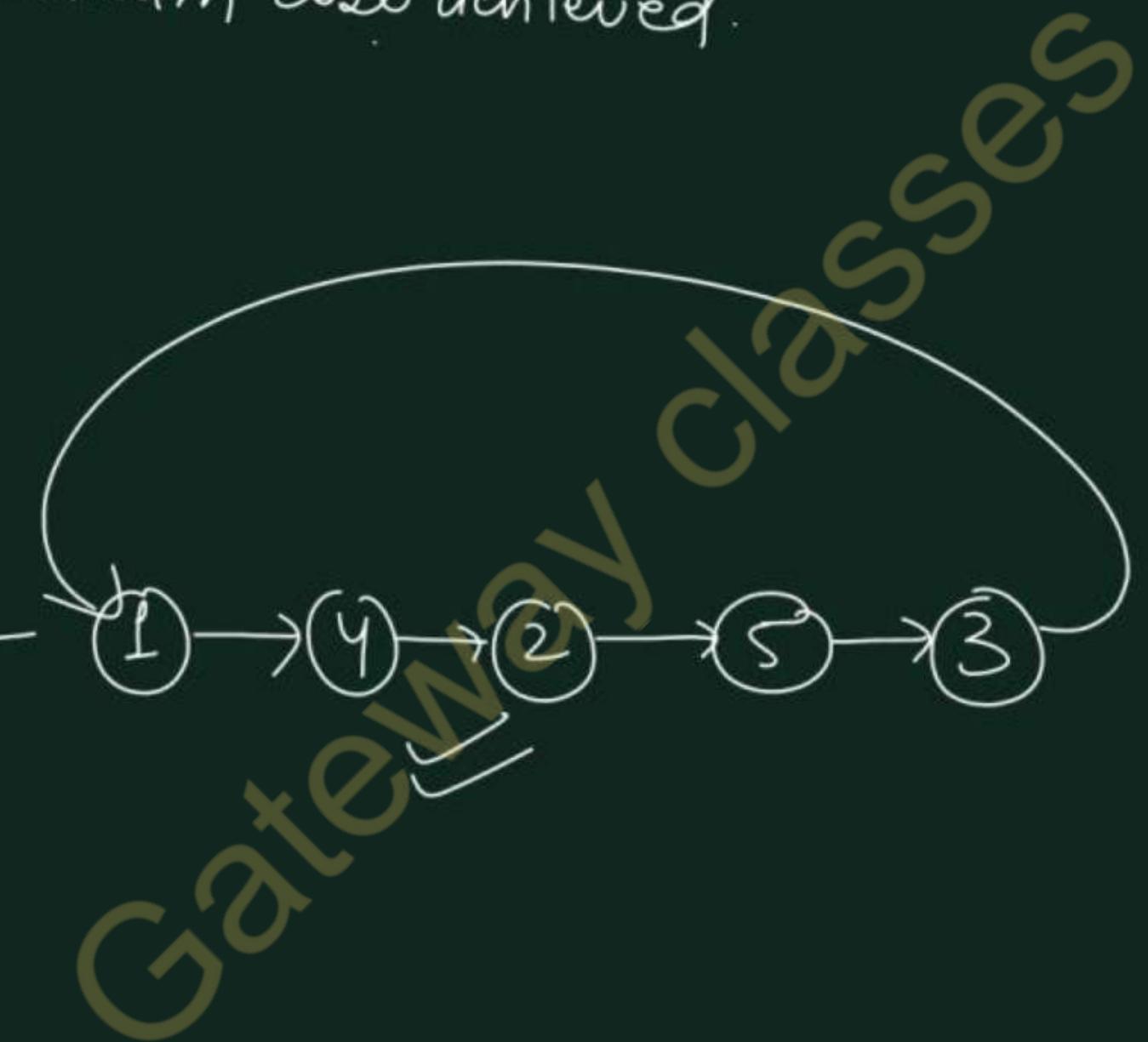
cost of leaf node = 28

So this is the minimum cost achieved.

upper = \emptyset

↓
upper = 28

closed path



1. Differentiate Backtracking and Branch and Bound Techniques. (AKTU 2020-21, 2022-23)
2. Explain Branch and bound in brief. (AKTU 2021-22)
3. Apply Branch and Bound technique to solve traveling salesman problem for the graph whose cost matrix given below: (AKTU 2023-24)

Cost matrix =

∞	17	13	22	18
13	∞	16	24	19
15	18	∞	16	28
19	13	15	∞	21
28	24	19	18	∞

Thaunk
you

Gax wa classes