AKTU

B.Tech 5th Sem

GATEWAY CLASSES

CS IT & CS Allied

DAA : Design & Analysis Of Algorithm

ONE SHOT Revision

(Crash Course)

By Dr. Nidhi Parashar Ma'am

- M.Tech Gold Medalist
- Net Qualified

Unit-4 : Dynamic Programming, Backtracking and Branch-and-Bound

AKTU

B.Tech 5th Sem

GW GATEWAY CLASSES

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.

① ②

Formal **Procedure :-**

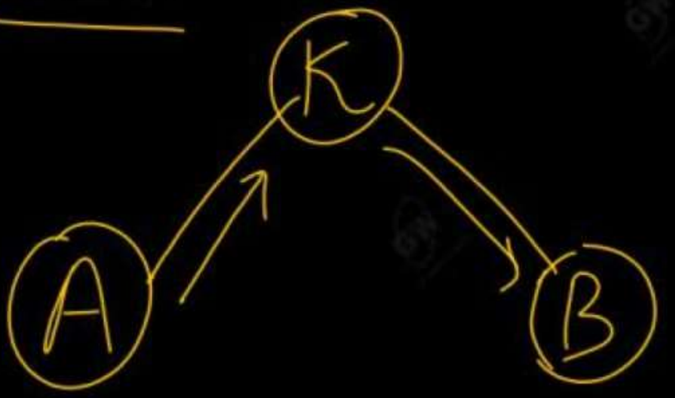
(1) Define problem variables, objective function and set up the constraints.

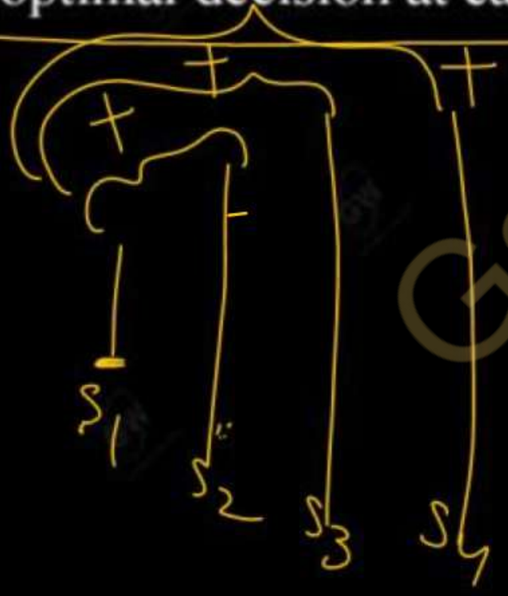
(2) Determine stages of the problem.

(3) Develop the recursive method. → Iteration / Recursion

(4) Make tabular representation

(5) Find optimal decision at each stage and then overall optimal





Principle of optimality (Imp)

Suppose that in solving a problem, we have to make a sequence of decisions D1, D2, ..., D_n

- If this sequence is optimal, then the last k decisions, $1 <= k <= n$ must be optimal.

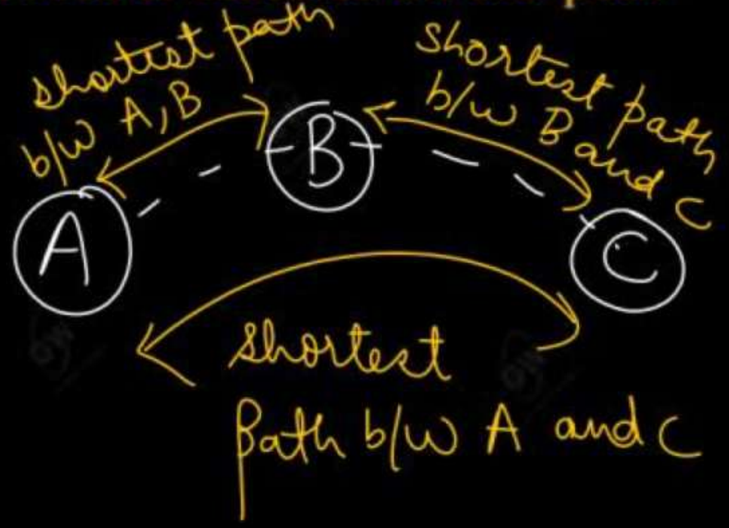
Ex-1 ➢ In the shortest path problem, If i, i_1, i_2,..., j is a shortest path from i to j, then i_1, i_2,..., j must be a shortest path from i to j

Ex-2 ➢ To Search for best path: Dynamic programming principle(Principle of Optimality) says……

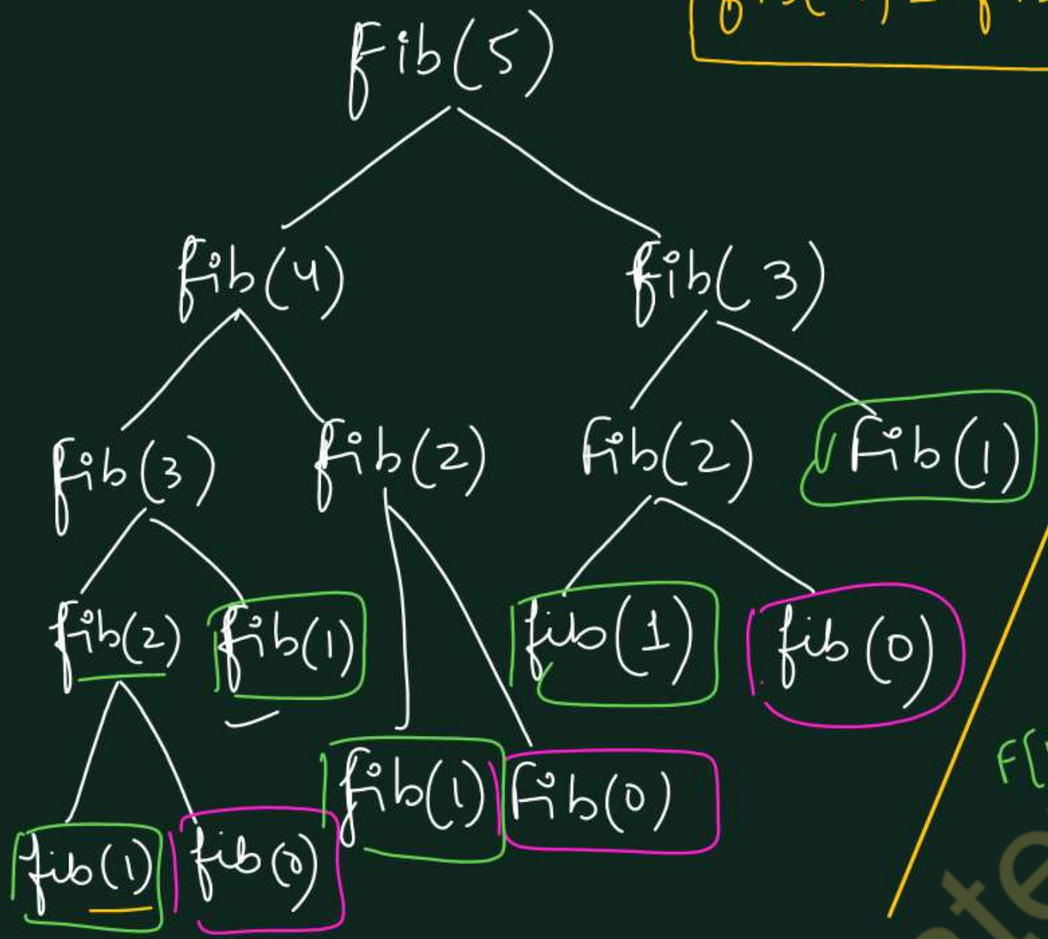
"If B is on the shortest path from A to C, then the path from A to B that lies on the path from A to C is the shortest path from A to B"

Ex-1

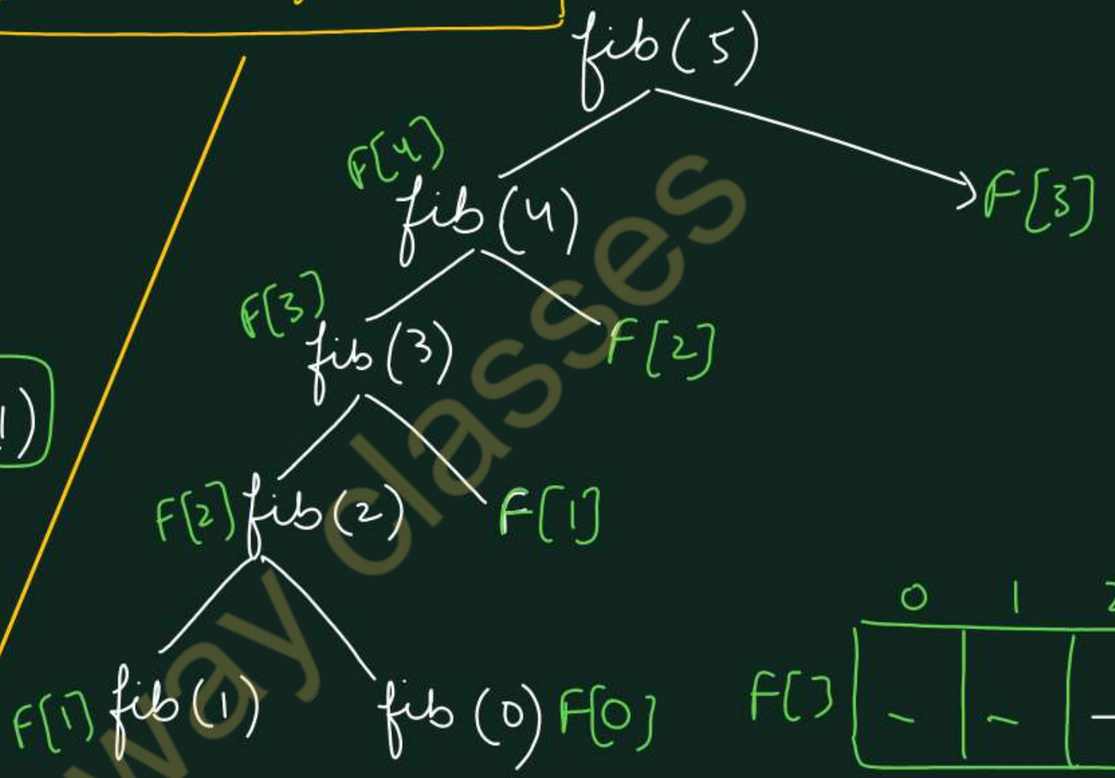
Ex-2

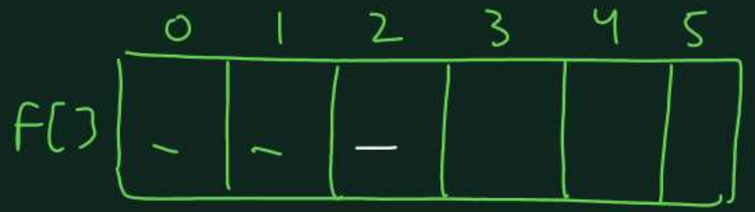


$$fib(n) = fib(n-1) + fib(n-2)$$



divide and conquer
(Recursive solution)





dynamic programming

Approach

(1) Break the main 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~~ Profits			
	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

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

$$\text{Subjected to, } \quad x_1 + x_2 + x_3 + x_4 \leq 6,$$

$$\text{Where } x_1, x_2, x_3, x_4 \geq 0$$

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

store 2

Store 2		
x_2	$f_2(x_2)$	
0	0	0
1	2	2
2	4	4
3	6	6
4	8	8
5	9	9
6	10	10

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

0	4	6	7	7	7	7	
2	6	8	9	9	9		
4	8	10	11	11			
6	10	12	13				
8	12	14					
9	13						
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 + store2		0+0	1+0	2+0 1+1	2+1 1+2	2+2 1+3	2+3 1+4	2+4
Store 3		Max. of $[f_1(x_1) + f_2(x_2)] + f_3(x_3)$						
x_3	$f_3(x_3)$							
0	0	0	4	6	8	10	12	14
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						

Stage – 4 : (store 1 + store 2 + store 3) + 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 + store2+ store3 (optimal solution of stage 3)	0+0+0	0+0+1	1+0+1	2+0+1 1+1+1 1+0+2	2+1+1 1+2+1 2+0+2 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)

AKTU PYQs

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

GW GATEWAY CLASSES

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.

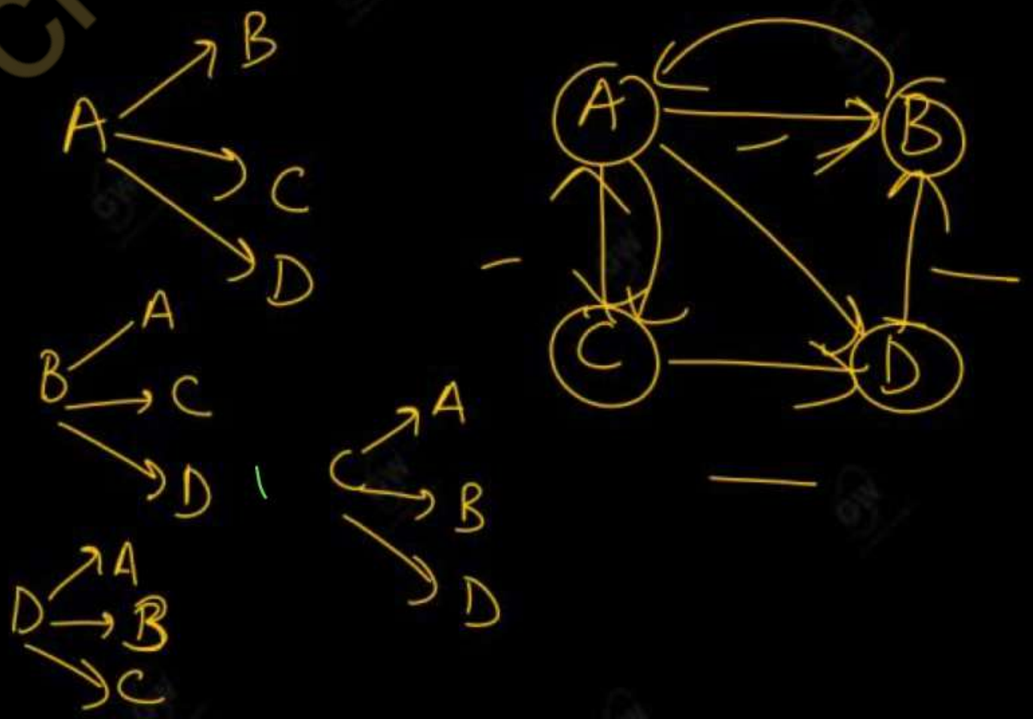
→ Also called as Floyd's Algo.

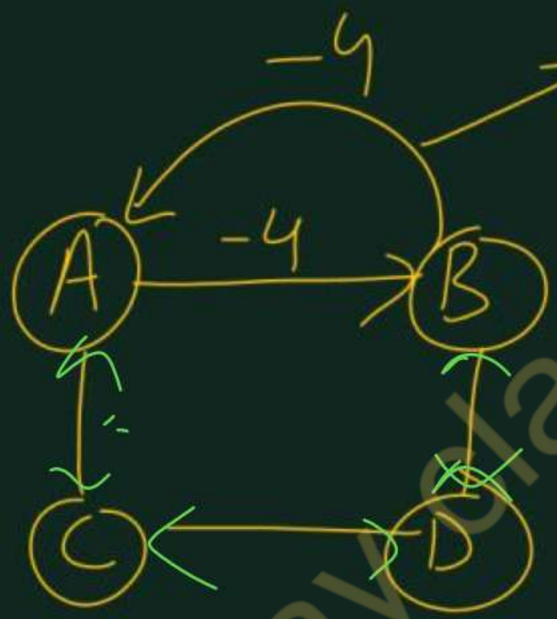
→ It is all - pair shortest path algorithm used to find the shortest path or shortest distance between every pair of Vertices in the given graph.

→ 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.





negative cost cycle

⇓

warshall's algo does not work.

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

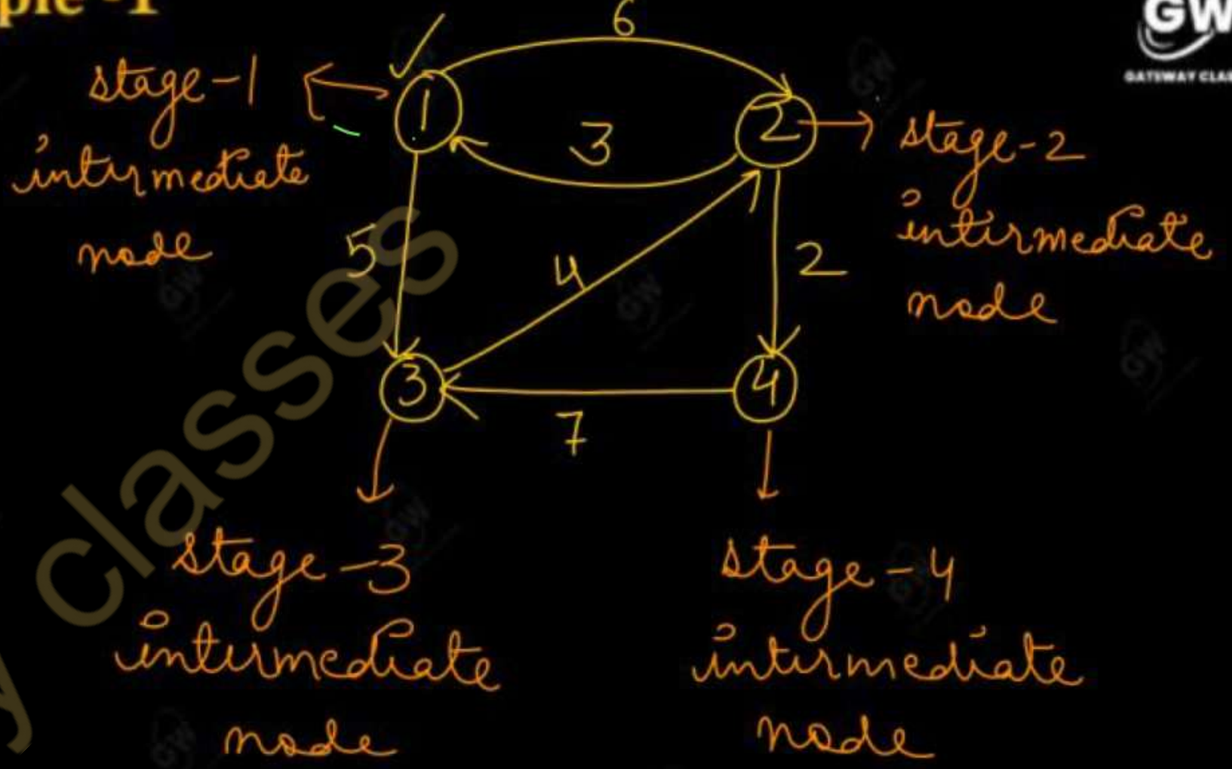
```
Floyd-Warshall(w)
n = no. of rows in w
D^0 = w
for k = 1 to n      // for n number of nodes (one vertex chosen as intermediate vertex at a time
    for i = 1 to n  // rows
        for j = 1 to n  // cols
            D^K(i,j) = min{D^{k-1}[i,k] + D^{k-1}[k,j], D^{K-1}(i,j)}
return D^n
```

n^3

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

Example -1

$$W = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{bmatrix} 0 & 6 & 5 & \infty \\ 3 & 0 & \infty & 2 \\ \infty & 4 & 0 & \infty \\ \infty & \infty & 7 & 0 \end{bmatrix} \end{array}$$



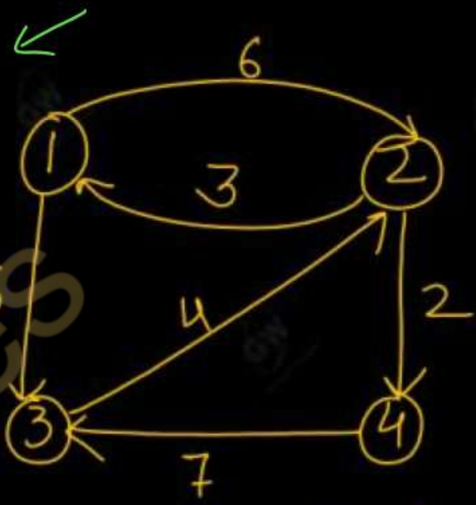
intermediate node

D^0 =

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

Example -1

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



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

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

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

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

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

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

$D^0 =$

	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

optimal solution of stage 1 ⟸

Example -1

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

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

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

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

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

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

$D^1 =$

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

$D^2 =$

	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

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

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

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

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

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

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

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

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

optimal solution till stage-3

Example -1

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

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

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

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

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

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

$D^3 =$

	1	2	3	4
1	0	6	5	8
2	3	0	8	2
3	7	4	0	6
4	14	11	7	0

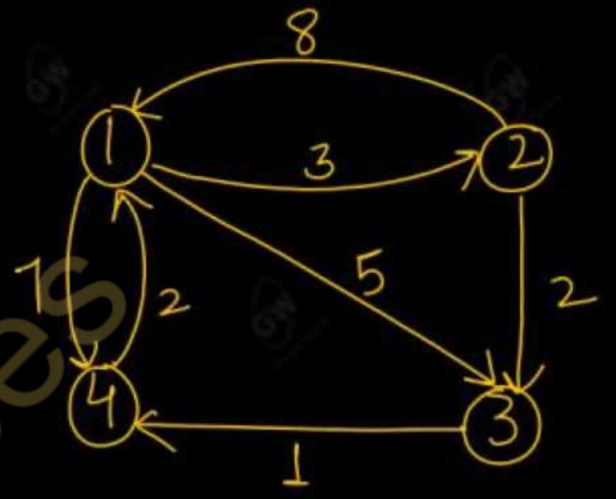
$D^4 =$

	1	2	3	4
1	0	6	5	8
2	3	0	8	2
3	7	4	0	6
4	14	11	7	0

Final solution

Example -2

DPP



Solution —

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

$$D^1 = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{bmatrix} 0 & 3 & \infty & 7 \\ 8 & 0 & 2 & 15 \\ 5 & 8 & 0 & 1 \\ 2 & 5 & \infty & 0 \end{bmatrix} \end{array}$$

$$D^2 = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{bmatrix} 0 & 3 & 5 & 7 \\ 8 & 0 & 2 & 15 \\ 5 & 8 & 0 & 1 \\ 2 & 5 & 7 & 0 \end{bmatrix} \end{array}$$

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

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

Application-

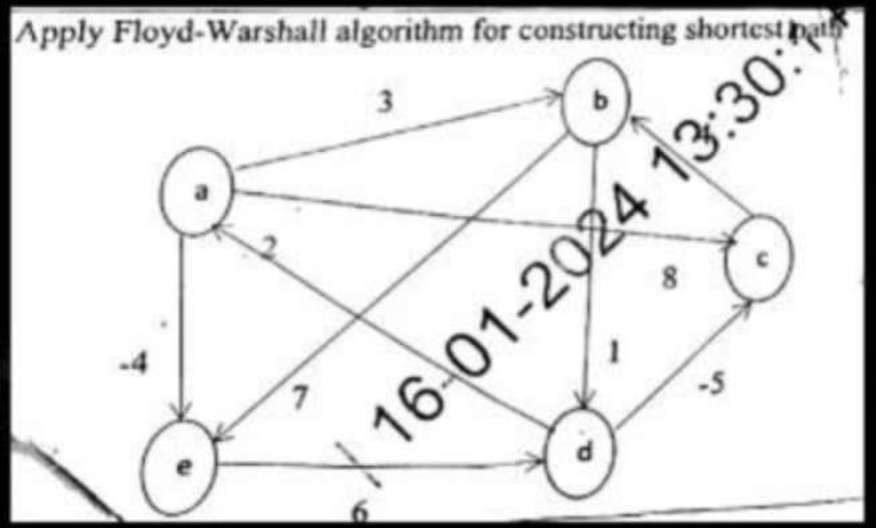
1) In networking devices

2) To calculate transitive closure of directed graphs.

3) In routing data packets

AKTU PYQs

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

GW GATEWAY CLASSES

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 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 ⟹ Maximize $\left(\sum_{1 \le i \le n} p_i x_i \right)$ → No. of objects

Subjected to

constraint ⟹ $\sum_{1 \le i \le n} w_i x_i \le m$ → Knapsack Maximum capacity

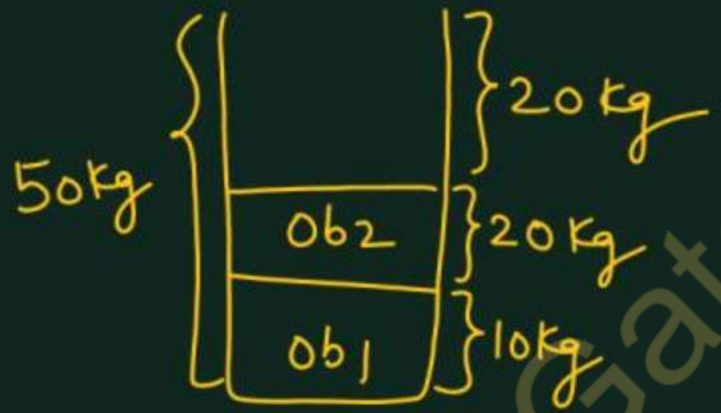
and, $x_i = \begin{cases} 0 \\ 1 \end{cases}$ $1 \le i \le n$

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

Ex-1

$M = 50kg$

objects	:	1	2	3
P	:	50	80	90
w	:	10	20	30
P/w	:	5	4	3



Profit = 50 + 80 = 130

It should have been = 170

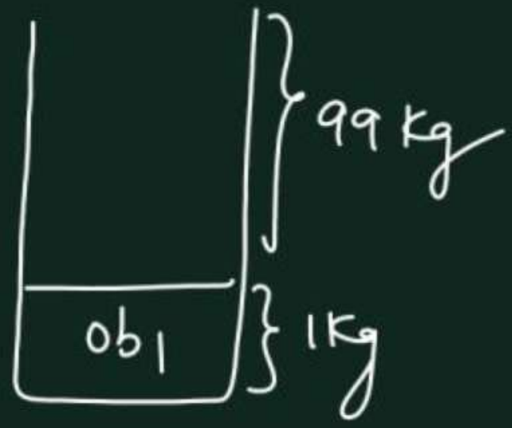
Ex-2

$M = 100$

objects	:	1	2
P	:	2	100
w	:	1	100
P/w	:	2	1

$P = 2$

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:

$$V[n, w] = P_n + V[n - 1, w - w_n]$$ where ` $P_n$ ` is profit of the n^{th} object and w_n is its weight.

  - if the object is not included:

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

available capacity

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

→ weight of n^{th} object

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

n^{th} object

Profit without considering this object

Note ⇒ 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

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	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							
7	5	n=4	0							

$V[2,3] =$

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

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

$= 4$

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

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

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

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

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

objects	1	2	3	4
p	1	4	5	7
w	1	3	4	5

Table Dimensions-

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

(n+1) rows, (M+1) cols

Remaining bag capacity

P	W_n	n	w=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							

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.

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

$= \max[5, 5+0] \Rightarrow 5$

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

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	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	1	1	4	5	7	8	9

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

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

$V[3,6] = \max[V[2,6], 5+V[2,2]]$

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

$V[3,7] = \max[V[2,7], 5+V[2,3]]$

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

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

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

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

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

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

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

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

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

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

x_1, x_2, x_3, x_4

Remaining bag capacity

P	W	n	w=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	1	1	4	5	7	8	9

Max Profit = 9

Remaining Profit = 9 − $P_{\text{third object}}$

= 9 − 5 ⇒ (4)

Remaining Profit 4 − 4 ⇒ 0

Ans -

Objects 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 w_n ≤ w                    // if object_n under consideration (weight) is less than/equal to remaining bag capacity.
        if P_n + V[n-1, w-w_n] > V[n-1, w]
            V[n, w] = P_n + V[n-1, w-w_n]
        else
            V[n, w] = V[n-1, w]
    else
        V[n, w] = V[n-1, w]
return V[N,M]
```

if object weight is greater than remaining capacity → else

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).

Ob :	1	2	3	4
P :	1	2	5	6
w :	2	3	4	5

M=8

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

$= \max\{1, 2+0\}$

$= 2$

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

$= \max\{1, 2+1\}$

$= 3$

P	w	m
0	0	0
1	2	1
2	3	2
5	4	3
6	5	4

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

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

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

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

Max = 8

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

x_1, x_2, x_3, x_4

First arrange weights in ↑ order.

$V[4,6] = \max[V[3,6], 1+V[3,0]]$

$= \max[4, 1+0]$

$\Rightarrow (4)$

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

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

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)

P	W	n	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
11	2	1	0	0	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11
21	11	2	0	0	11	11	11	11	11	11	11	11	11	21	21	32	32	32	32	32	32	32	32
33	15	3	0	0	11	11	11	11	11	11	11	11	11	21	21	32	32	33	33	44	44	44	44
31	22	4	0	0	11	11	11	11	11	11	11	11	11	21	21	32	32	33	33	44	44	44	44

$x = \{ \underset{x_1}{1} \quad \underset{x_2}{0} \quad \underset{x_3}{0} \quad \underset{x_4}{1} \}$

MAX PROFIT = 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)

AKTU

B.Tech 5th Sem

GW GATEWAY CLASSES

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

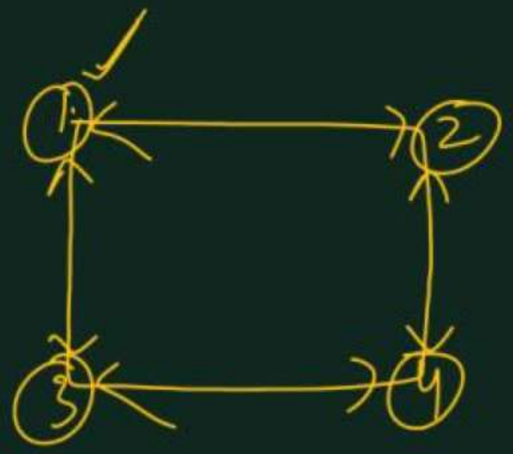
Traveling Salesman Problem

(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.

Brute-Force Approach



1 → 2 → 3 → 4 → 1

1 → 2 → 4 → 3 → 1

1 → 3 → 2 → 4 → 1

1 → 3 → 4 → 2 → 1

1 → 4 → 2 → 3 → 1

1 → 4 → 3 → 2 → 1

$(3!)$

If 'n' nodes are present in the graph.

$(n-1)!$

$\Rightarrow \approx n!$ = $O(n^n)$

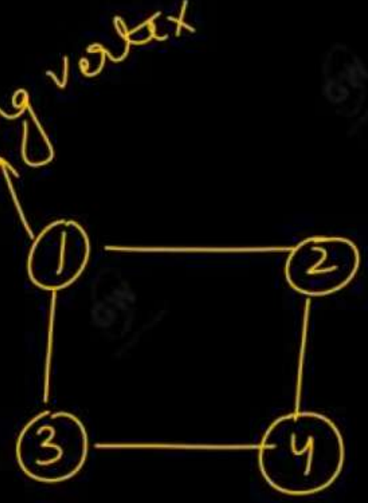
Recursive Equation :-

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

starting vertex

set of remaining Vertices

$$= Min \begin{cases} C(1,2) + TSP(2, \{3,4\}) \\ C(1,3) + TSP(3, \{2,4\}) \\ C(1,4) + TSP(4, \{2,3\}) \end{cases}$$



i, S − K

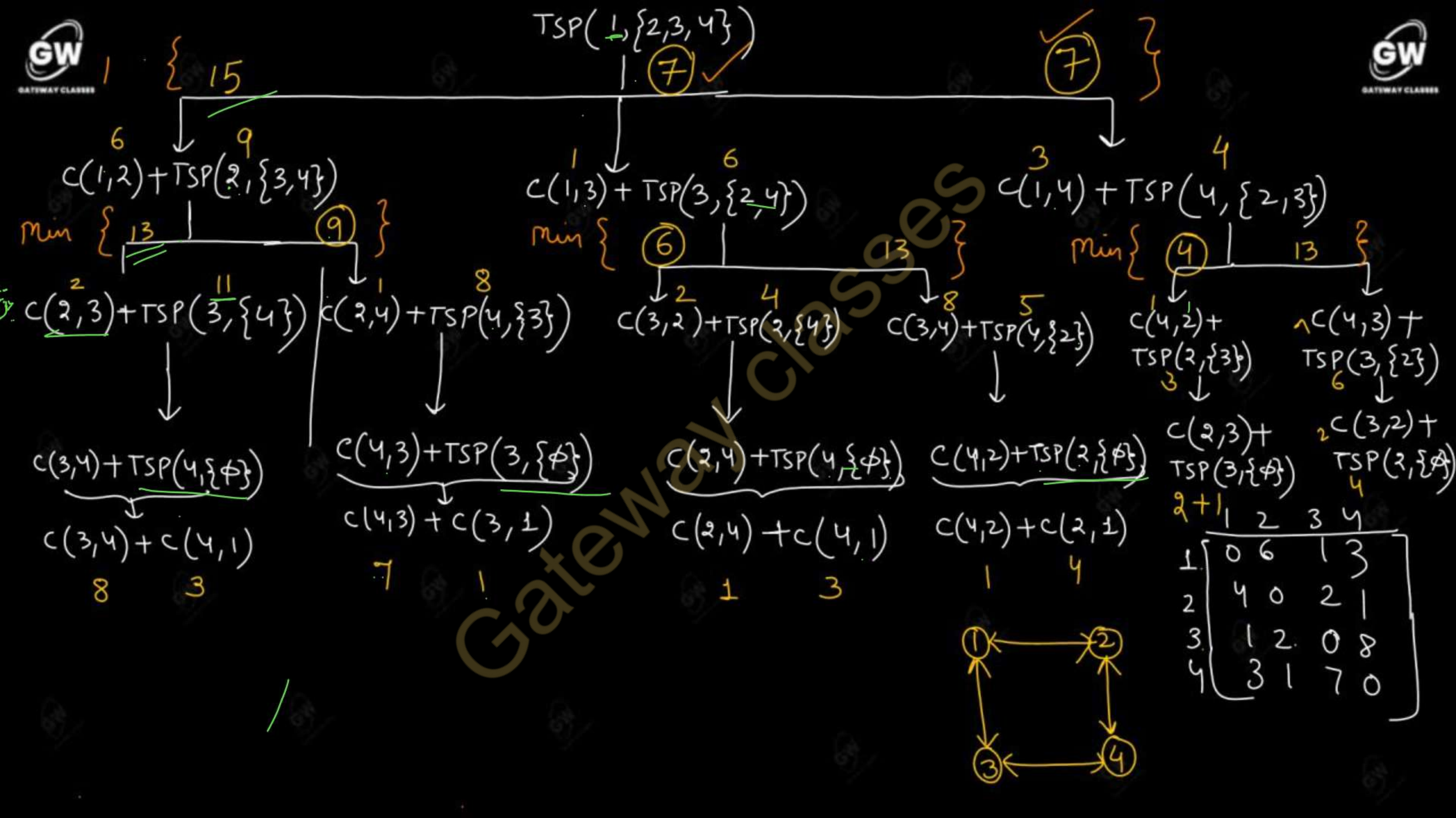
i → K

General Recursive Equation

chosen next vertex

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

TSP(i, ϕ)



There are two optimal solutions with cost = (7)

$sol^n 1 \Rightarrow$ $1 \to 3 \to 2 \to 4 \to 1$

$sol^n 2 \Rightarrow$ $1 \to 4 \to 2 \to 3 \to 1$

Time Complexity -

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

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

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 ⇒ 15 ✓

But TSP(4, {ϕ}), TSP(3, {ϕ}), TSP(2, {ϕ}) called twice

Distinct function calls

$= 15 - 3 = (12)$

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

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

$= 3.4 \Rightarrow 12$

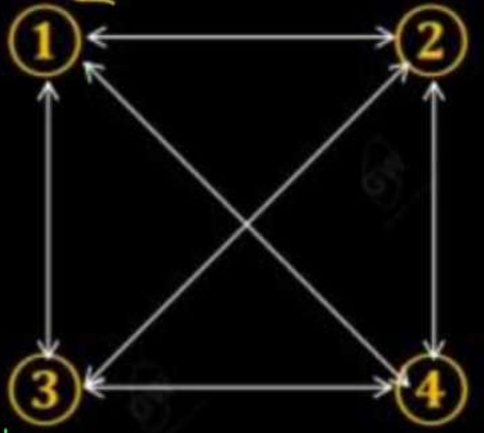
$$TSP(1,\{2,3,4\}) = \min\begin{cases} c(1,2) + TSP(2,\{3,4\}) \\ c(1,3) + TSP(3,\{2,4\}) \\ c(1,4) + TSP(4,\{2,3\}) \end{cases}$$

$$TSP(2,\{3,4\}) = \min\begin{cases} c(2,3) + TSP(3,\{4\}) \\ c(2,4) + TSP(4,\{3\}) \end{cases}$$

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

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

Q-2



1 → 2 → 4 → 3 → 1

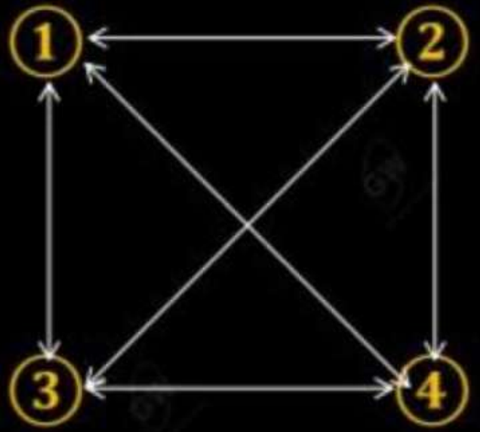
	1	2	3	4
1	0	10	15	20
2	4	0	9	10
3	6	13	0	12
4	8	8	9	0

$$TSP(2,\{3,4\}) = \min\{9+20,\ 10+15\}$$
$$= 25$$

$$TSP(3,\{2,4\}) = \min\begin{cases} C(3,2) + TSP(2,\{4\}) \\ C(3,4) + TSP(4,\{2\}) \end{cases}$$
$$= \min\begin{cases} 41 \ (C(3,2)=13,\ TSP(2,\{4\})=28) \\ 25 \ (C(3,4)=12,\ TSP(4,\{2\})=13) \end{cases}$$

$$TSP(2,\{4\}) = C(2,4) + TSP(4,\{\phi\})$$
$$= C(2,4) + C(4,1)$$
$$= 10 + 8 = 28$$

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



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

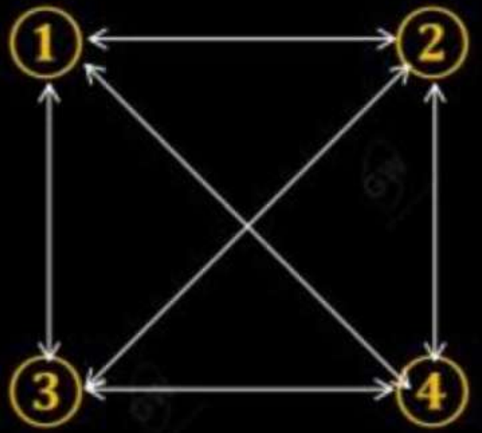
$$T(3,\{2,4\}) = \min\{13+28, 12+13\}$$
$$= \min\{41, 25\} = 25$$

$$TSP(4,\{2,3\}) = \min\begin{cases} c(4,2) + TSP(2,\{3\}) \\ c(4,3) + TSP(3,\{2\}) \end{cases}$$

(8 for $c(4,2)$, 15 for $TSP(2,\{3\})$, 9 for $c(4,3)$, 18 for $TSP(3,\{2\})$)

$$TSP(2,\{3\}) = c(2,3) + TSP(3,\{\phi\})$$
$$= c(2,3) + c(3,1)$$
$$= 9 + 6 = 15$$

$$TSP(3,\{2\}) = c(3,2) + TSP(2,\{\phi\})$$
$$= c(3,2) + c(2,1)$$
$$= 13 + 5 = 18$$



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

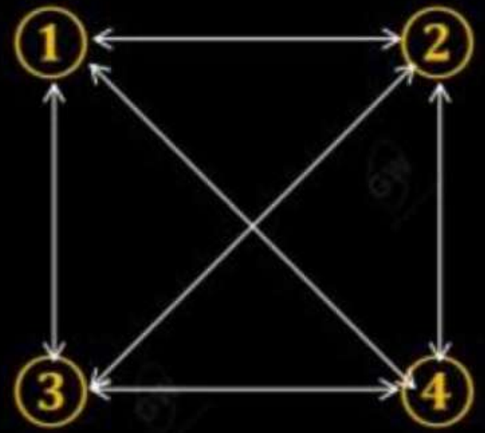
$$TSP(4,\{2,3\}) = \min\{\ 8+15,\ 9+18\}$$
$$= \min\{23, 27\} = 23$$

$$TSP(1,\{2,3,4\}) = \min\{\ 10+25,\ 15+25,\ 20+23\}$$
$$= \min\{35, 40, 43\}$$

Min cost = 35 ⟵ Ans

Optimal Route ⟶ $1 \to 2 \to 4 \to 3 \to 1$

$10+10+9+6$ ✓



	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	2	3	4
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

GW GATEWAY CLASSES

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 - $r_1 - r_2 - r_3 - r_4$
$0.9 \quad 0.9 \quad 0.9 \quad 0.9$

Chances that D_1 works perfectly

suppose for D_1, $r_i = 0.9$

System Reliability = Πr_i

$= (0.9)^4 = 0.6561$

$= 65\%$ Reliable

We have to develop a system with Max. Reliability

Reliability of working perfectly = 65%

Chances that system may fail = 35%.

Maximization 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
Suppose	0.9	0.9	0.9	0.9

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 is parallel.

Stages	S_1	S_2	S_3	S_4
	D_1	D_2	D_3	D_4
	D_1	D_2	D_3	D_4
	D_1	D_2	D_3	D_4

General Formula for Reliability calculations

$$\left(1-(1-r_i)^{n_i}\right)$$

n_i → Total copies considered for a particular device i.

Let's take device D_1

Reliability of each device = 0.9

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

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

$(0.1)^3 = .001$

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

$= 0.999 \Rightarrow 99.9\%$ Reliable system

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.

C = 105

Example

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

Total Reliability = πr_i

$0.9 \times 0.8 \times 0.5$

$= 0.36$

$= 36\%$ Reliable

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

$$\sum C_i = 30 + 15 + 20 = 65$$

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

Remaining amount = $[C - \sum C_i] = (105 - 65) = 40\ \$$

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 = $\left\lfloor \frac{40}{30} \right\rfloor = 1 + 1 \Rightarrow 2 \Rightarrow$ 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 $U_i = \left\lfloor \frac{C - \sum C_i}{C_i} \right\rfloor + 1$

We start with a set of order pair.

(R, C) — R: Reliability, C: cost

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

$C = 105$

I stage — S^0 = no device is included

$S^0 = \{(1, 0)\}$ → No cost

1 → 100% Reliable

// There is no system included at all

for D_1

D_1 = 1 copy

device no. (1) — no. of copies

$S_1^1 = \{(0.9 \times 1, 0+30)\} = \{(0.9, 30)\}$

$S_2^1 = \{(.99 \times 1, 0+60)\} = \{(0.99, 60)\}$

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

1 2

$D_1 = 1$ copy

for 2 copies of D_1

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

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

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

$= .99$

cost = 60

For D_2

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

$S_1^2 = \{(0.72, 45)(0.79, 75)\}$

$S_2^2 = \{(0.86, 60)(0.95, 90)\}$ ✗

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

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

$C = 105$

For 2 copies of D_2

$1-(1-0.8)^2$

$= 1-(0.2)^2$

$= 1-.04$

$= 0.96$ — Cost = 30

For D_2 → ② $S^1 = \{(0.9, 30)\ (0.99, 60)\}$

$S_1^2 = \{(0.72, 45)\ (0.79, 75)\}$

$S_2^2 = \{(0.86, 60)\ (0.95, 90)\}$ ✗

$S_3^2 = \{(0.89, 75)\ (0.98, 105)\}$ ✗ → Insufficient amount to purchase even a single copy of D_3.

$S^2 = \{(0.72, 45)\ (0.79, 75)\ (0.86, 60)\ (0.89, 75)\}$ ✗

If Reliability ↑ then cost should also ↑ 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_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

For 2 copies of D_2

$1-(1-0.8)^2$
$=1-(0.2)^2$
$=1-.04$
$= 0.96$ Cost = 30

For 3 copies of D_2

$1-(1-0.8)^3$
$=1-(0.2)^3 = 1-.008$
$= 0.992$

C = 45

For D_3 → 2

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

$S_1^3 = \{(0.36, 65)\ (0.43, 80)\ (0.44, 95)\}$

$S_2^3 = \{(0.54, 85)\ (0.64, 100)\ (0.66, 75+40)\}$ > 105

$S_3^3 = \{(0.62, 105)\ (0.74, 120)\ (0.77, 75+60)\}$ > 105

$S^3 = \{(0.36, 65)\ (0.43, 80)\ (0.44, 95)\ (0.54, 85)\ (0.62, 105)\ (0.64, 100)\}$

$S^3 = \{(0.36, 65)\ (0.43, 80)\ (0.54, 85)\ (0.64, 100)\}$ → $D_3 = 2$ copies

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

Reliability for 2 copies of D_3

$1-(1-0.5)^2$

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

$= 0.75$, $C = 40$

For 3 copies of D_3

$1-(1-0.5)^3$

$= 1-(0.5)^3 = 0.875$

$C = 60$

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

Final Solution –

$D_1 \Rightarrow$ 1 copy

$D_2 \Rightarrow$ 2 copies

$D_3 \Rightarrow$ 2 copies

Max Reliability = 0.64

$\therefore$ 64% Reliable system

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

$\pi r_i = 36\%$

improved to

AKTU

B.Tech 5th Sem

GW GATEWAY CLASSES

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

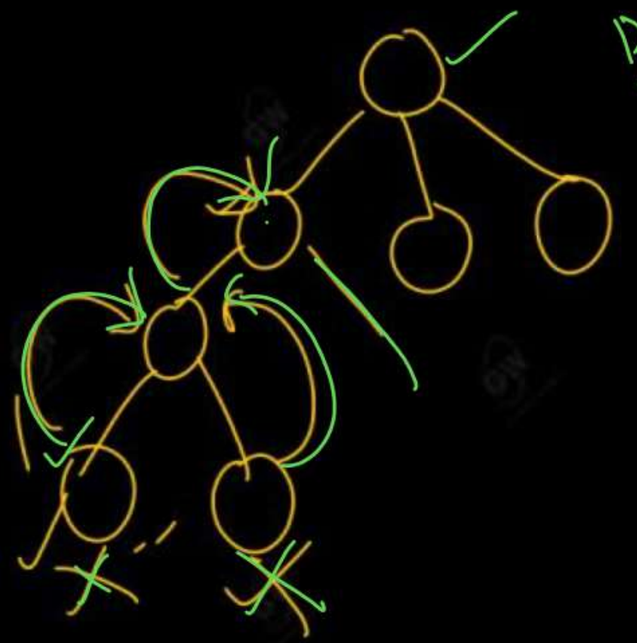
- AKTU PYQs

By Dr. Nidhi Parashar Ma'am

- M.Tech Gold Medalist
- Net Qualified

Backtracking (AKTU)

- 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 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.



Backtracking(AKTU 2023-24):

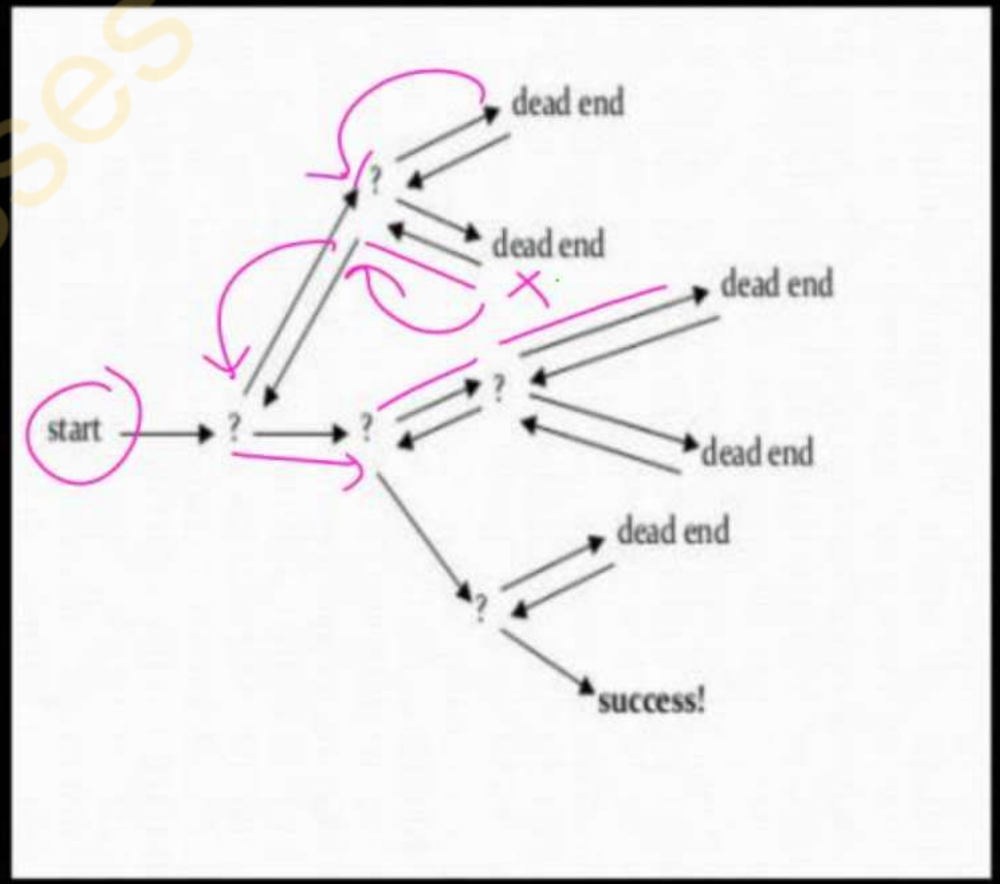
Pseudo-code for backtracking problems:

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!



State Space Tree (AKTU)

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 number 8 6 7
Applying brute force method: we will try all combinations

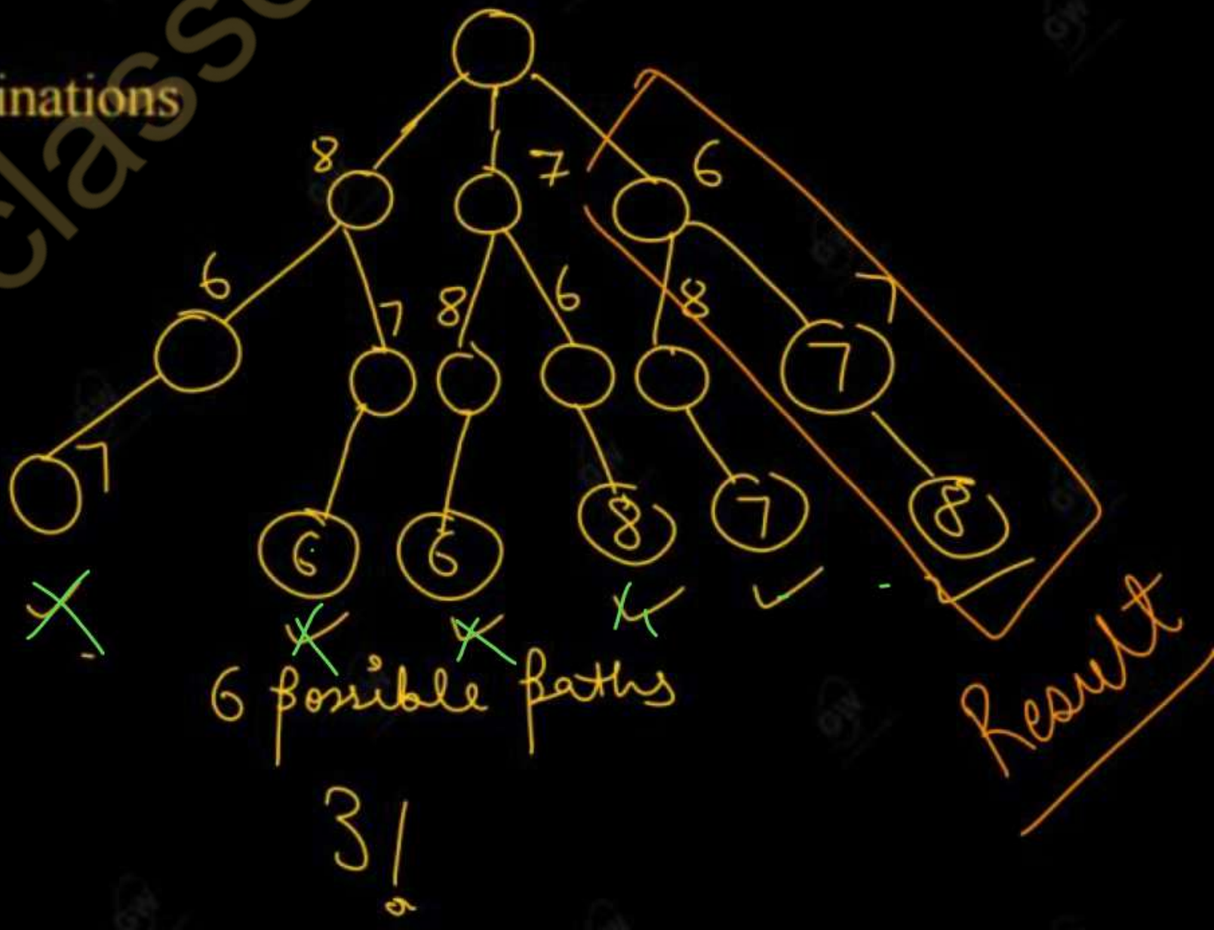
No. :- 8, 7, 6

8,7,6 _ _ _
3 × 2 × 1
3!

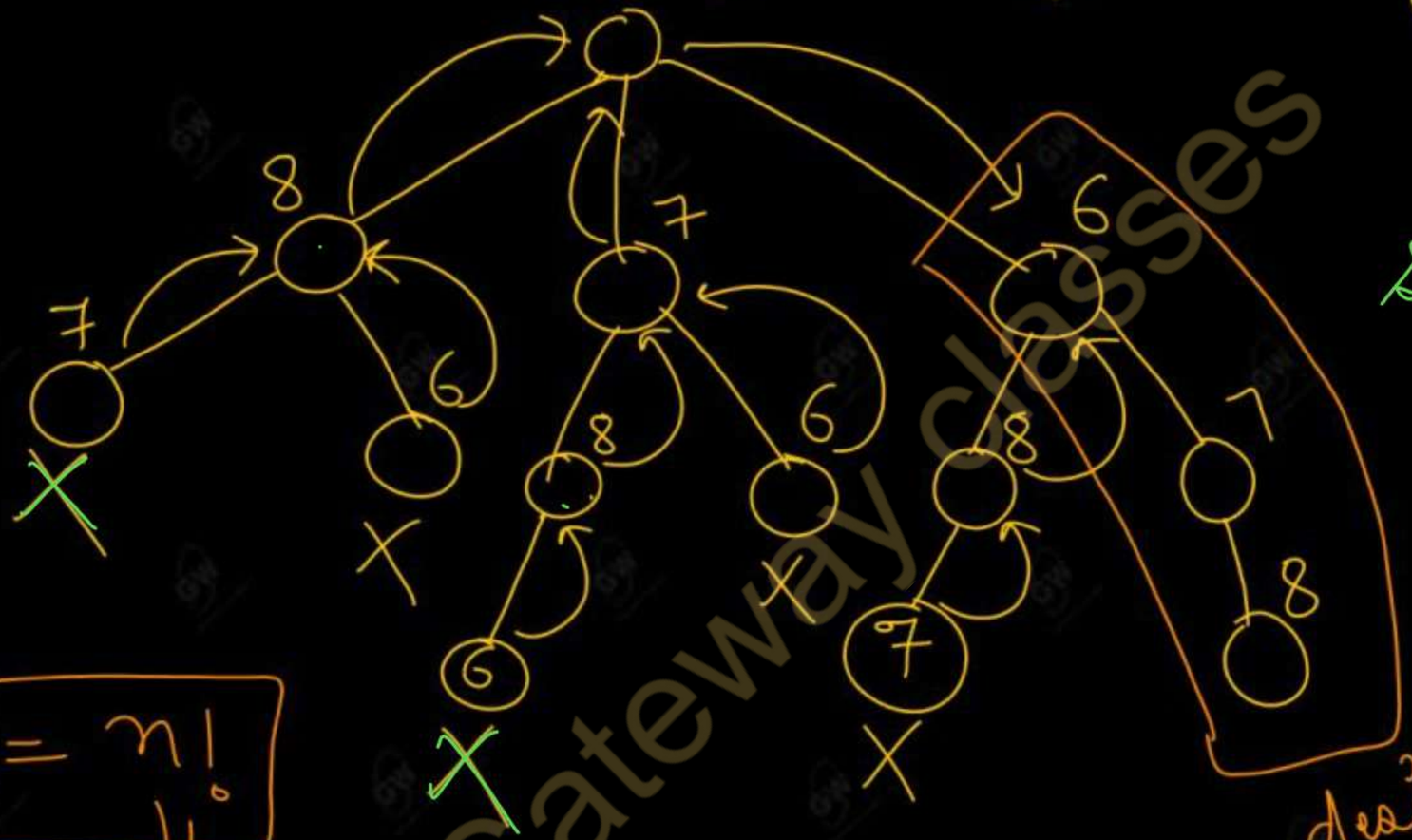
for n numbers

(n!)

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



Sort 8, 7, 6 into Ascending order using Backtracking



State space Tree

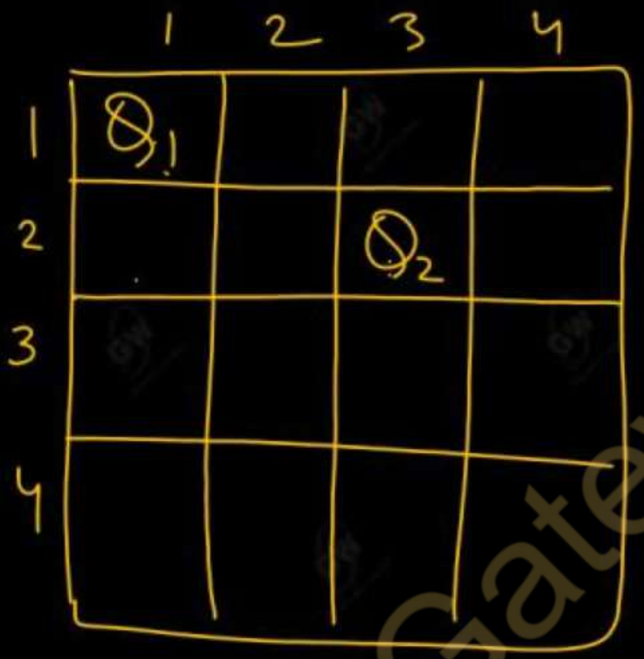
desired solution

$T(n) = n!$

$O(n^n)$

N queen's Problem

Place N queens in N× N chessboard so that no two queens can attack i.e. no two of them are an the same row, column or diagonal. solutions are expressed in n − tuple.



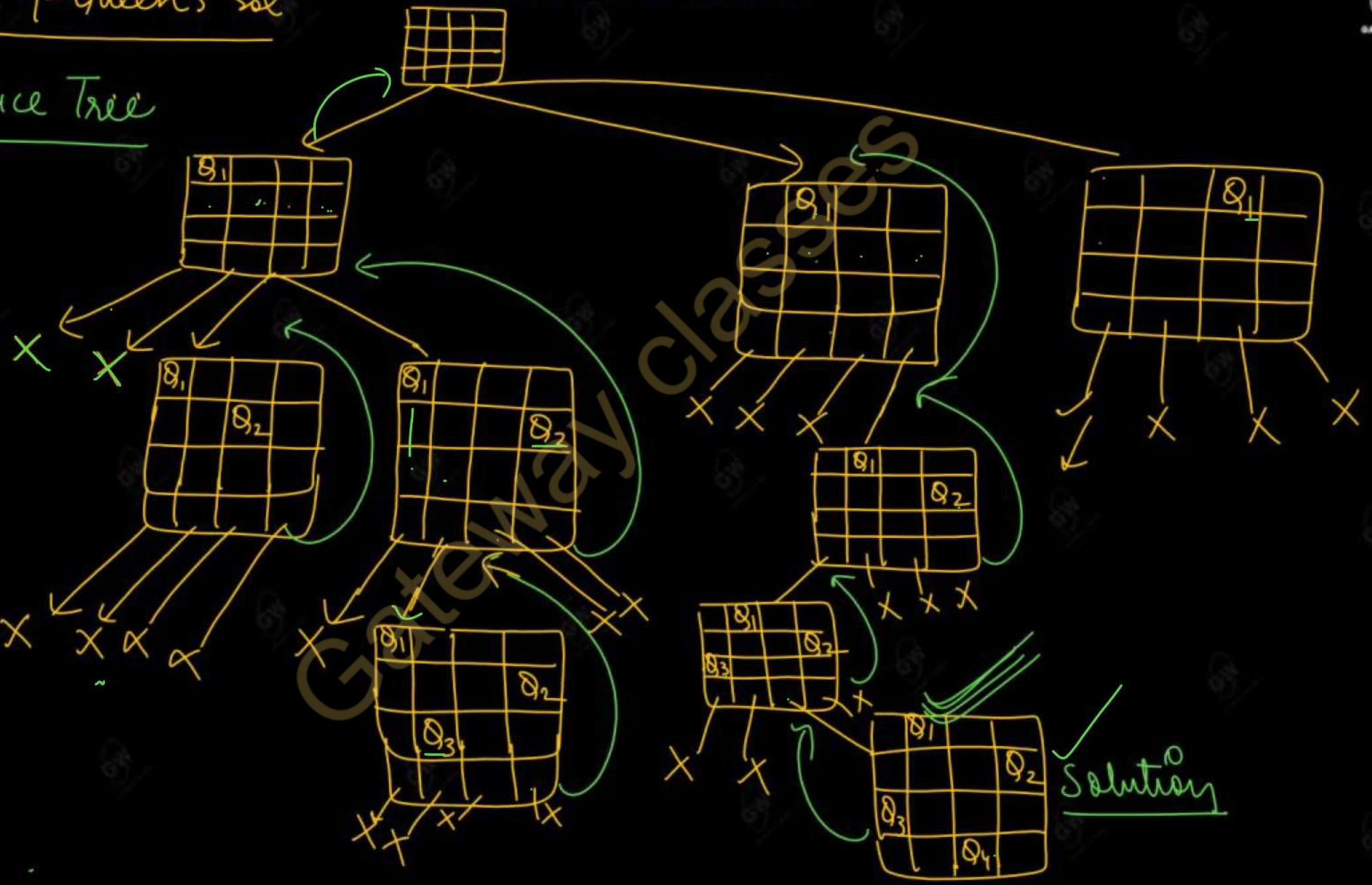
4-Queens → $^{16}C_4$ ⇒ possible combinations

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

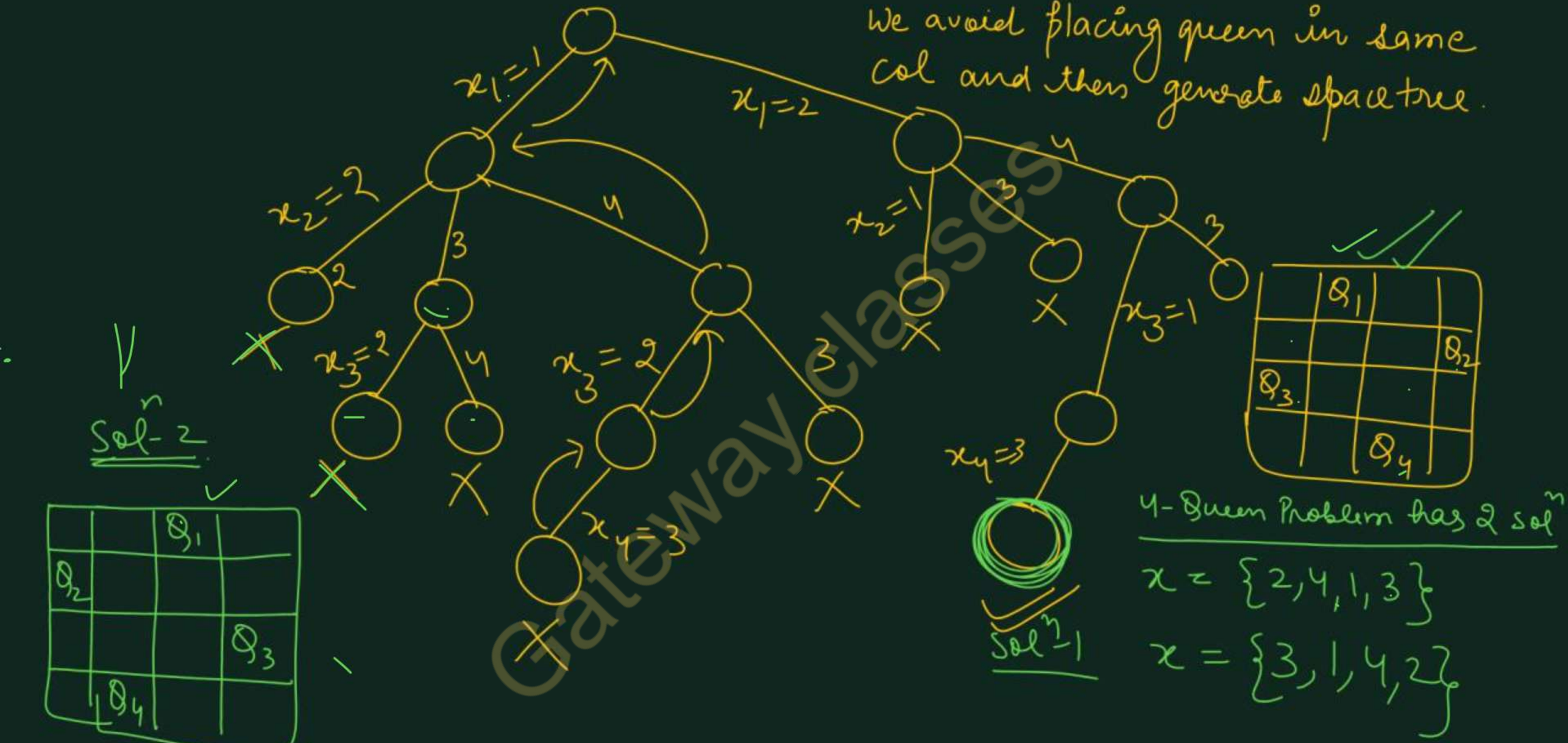
N – Queen's Problem

4-Queen's soln

state space Tree



We avoid placing queen in same col and then generate space tree.



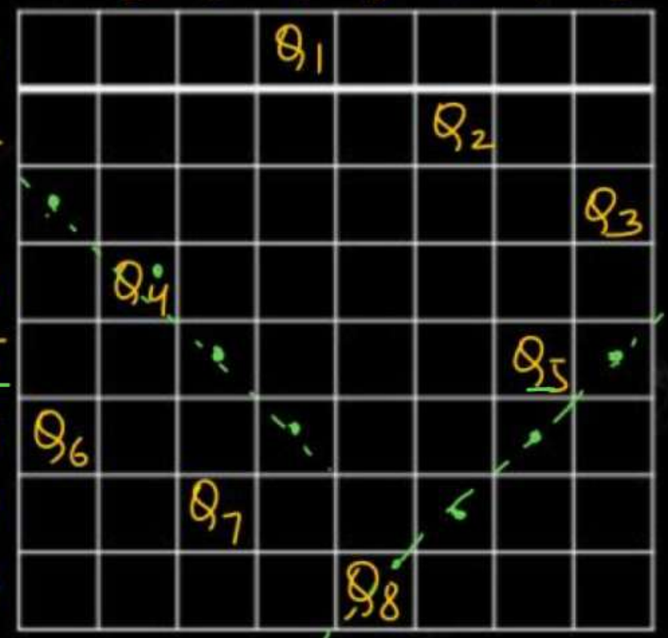
4-Queen Problem has 2 sol^n

$x = \{2, 4, 1, 3\}$

$x = \{3, 1, 4, 2\}$

N – Queen's Problem

(8 × 8) 8-Queens



Soln

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

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

Cell-1 → (i, j) Cell-2 → (K, l)

$(i-j) = (K-l) \Rightarrow \boxed{j-l = i-K}$ — ①

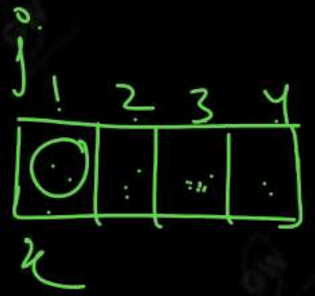
case-2 (5,8) (6,7) (7,6) (8,5)

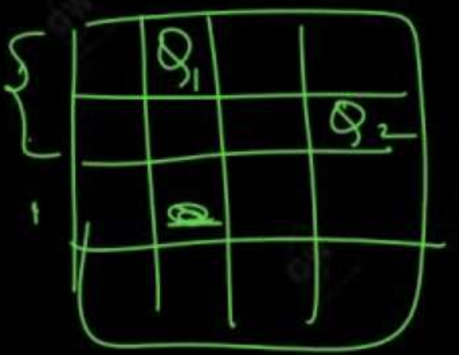
$(i+j) = (K+l) \Rightarrow \boxed{j-l = K-i}$ — ②

$\boxed{Abs(j-l) = Abs(i-K)}$ ⇒ condition for same diagonal

Pseudo code

```
Algorithm Place (K, i)
// Returns true if Q is placed at k^th row and i^th column otherwise returns false
// x [ ] is a global array.
        for j = 1 to (K-1) do // check with all the already placed queens for a possible attack
        {
                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)        → no. of queens
// using backtracking, two procedure prints all combinations of solutions
                                   → Kth Queen
{
    for i = 1 to n do   // For all 4-Queens
    {
        if (Place (K, i) ) then
        {
            x [k] = i   // Insert the col no into solution
            if (K== n) then   // If you are placing last queen
                Print (x [1: n] )
            else
                NQueens (K+1, n)
        }
    }
}
```

for 4-Queen's problem

First call is

⇒ NQueens (1, 4)

j ⇒ row no.

x[j] ⇒ col no

j →	1	2	3	4
x	2	1	4	

AKTU PYQs

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

GATEWAY CLASSES

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

AKTU PYQs

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 is a given positive integer m .

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

Some instances of this problem may have no solutions.

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

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

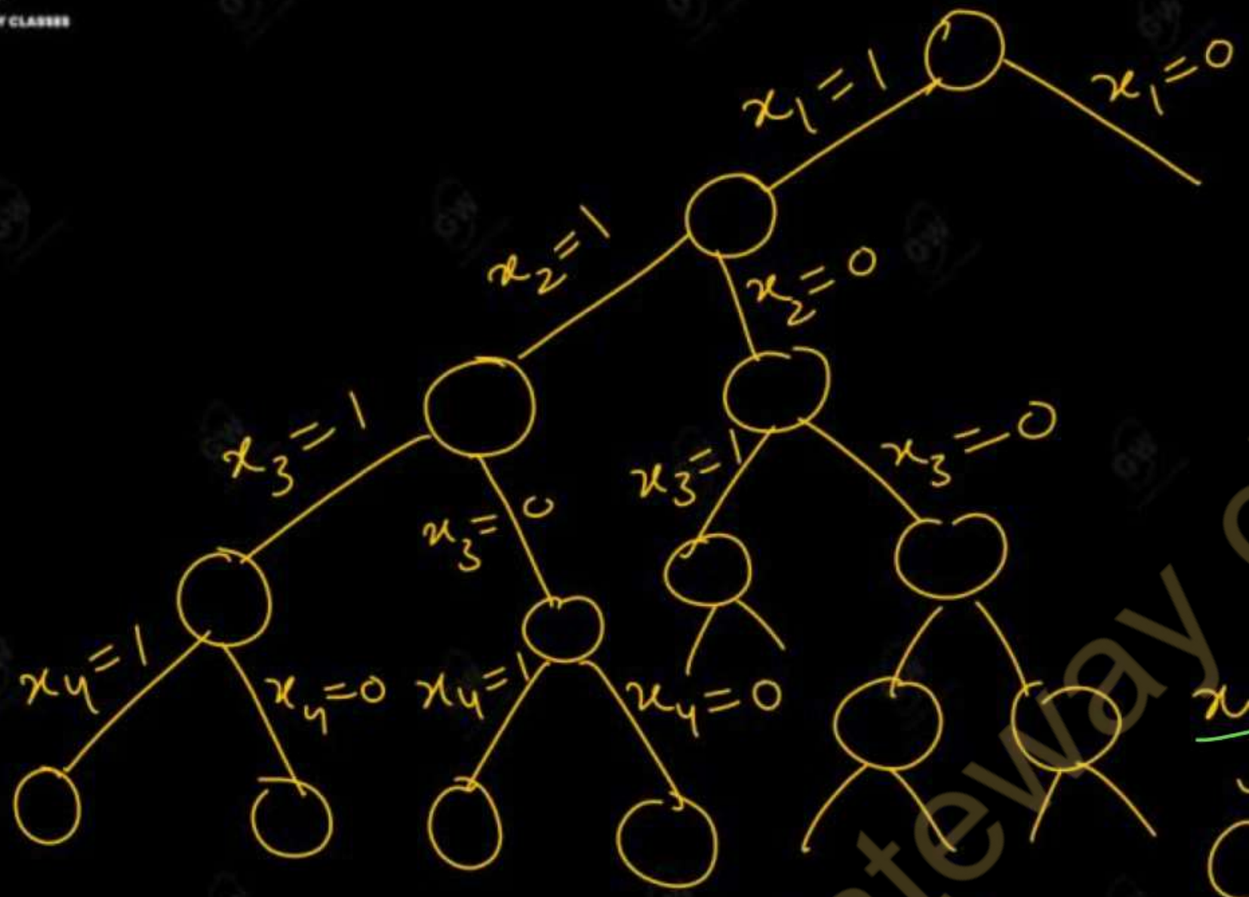
For example for $W[1:5] = \{1, 2, 5, 6, 8\}$ and m= 9, $n = 5$ and there are two solutions:-

(1) $\{1, 2, 6\}$

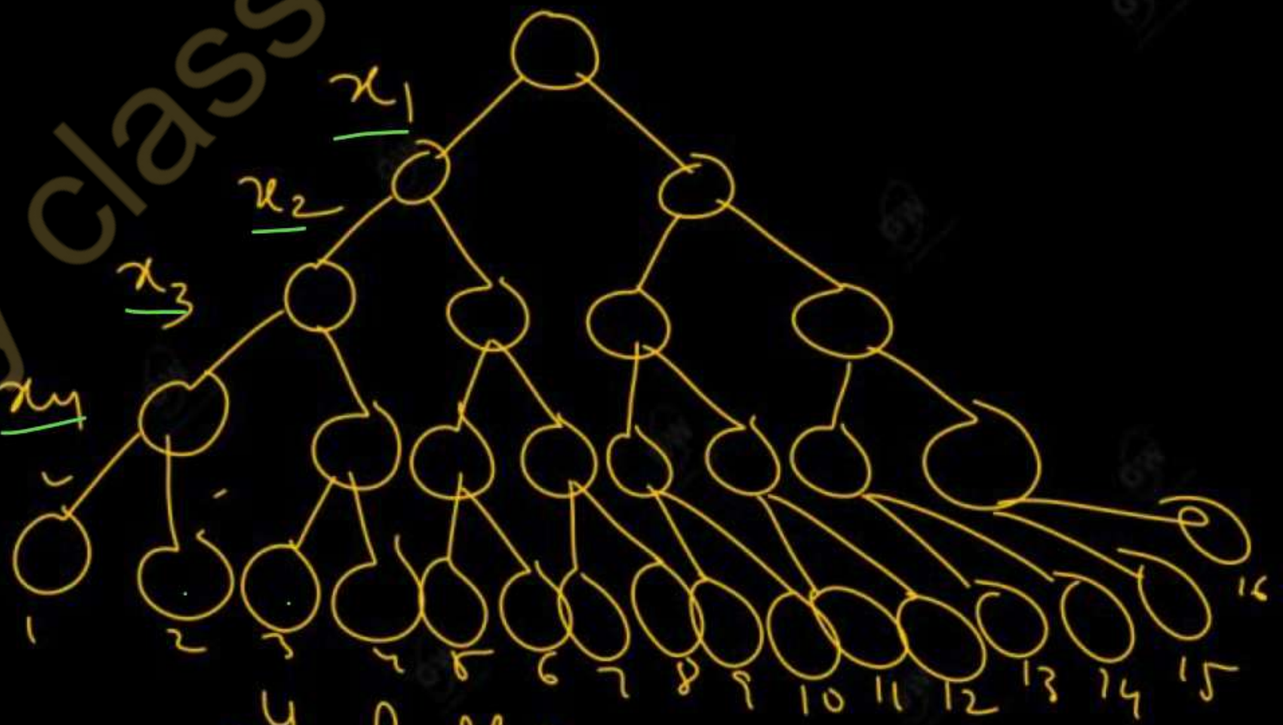
(2) $\{1, 8\}$

Brute force method

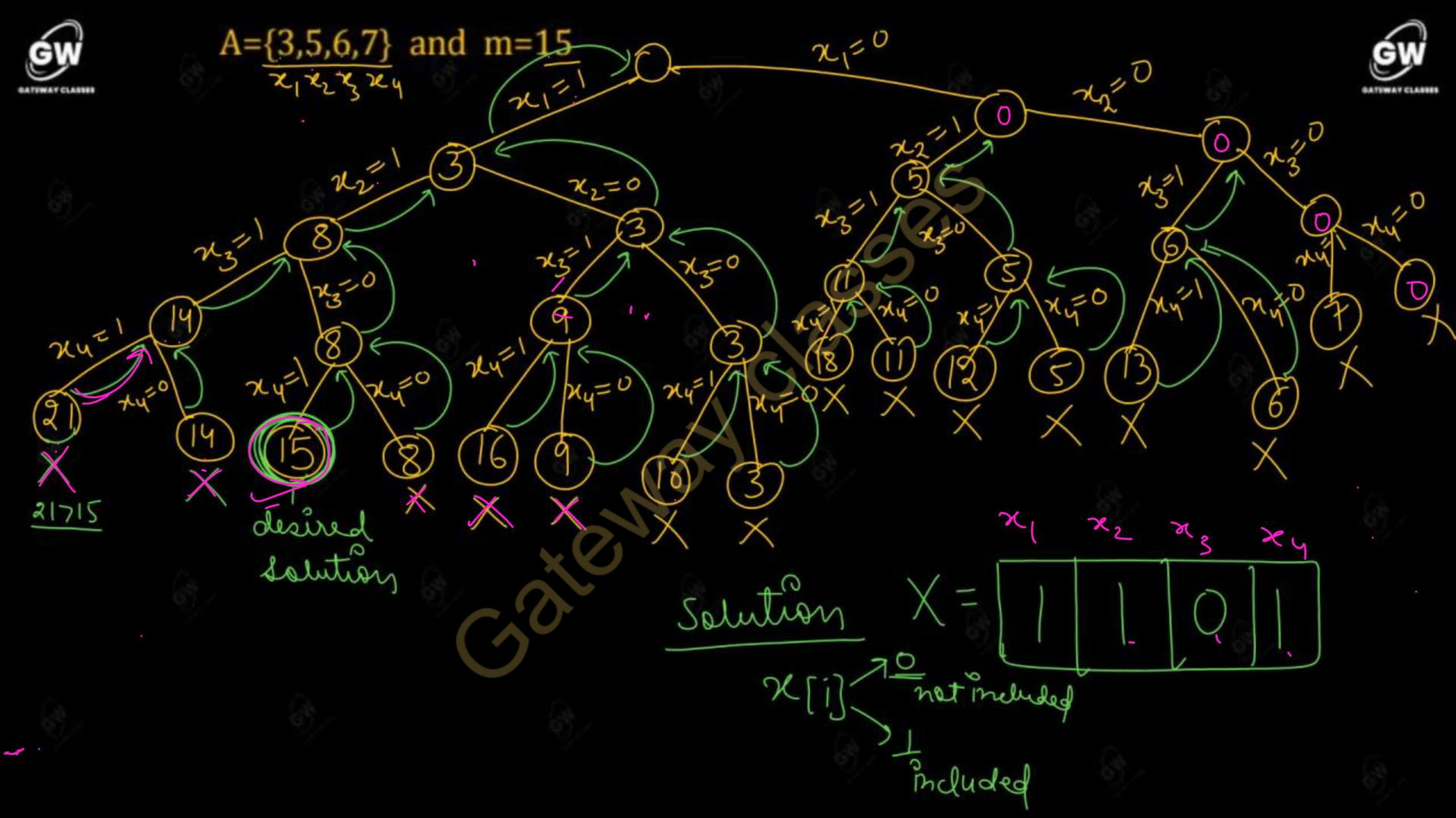
$w[1:4] = \{ w_1\ w_2\ w_3\ w_4 \}$



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



24 paths


A={3,5,6,7} and m=15
x_1 x_2 x_3 x_4
$x_1=1$
$x_1=0$
$x_2=1$
$x_2=0$
$x_3=1$
$x_3=0$
$x_4=1$
$x_4=0$
21715
desired solution
Solution X =
x_1 x_2 x_3 x_4
1 1 0 1
x[i] = 0 not included
x[i] = 1 included
Gatewayclasses

- 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 K^{th} number

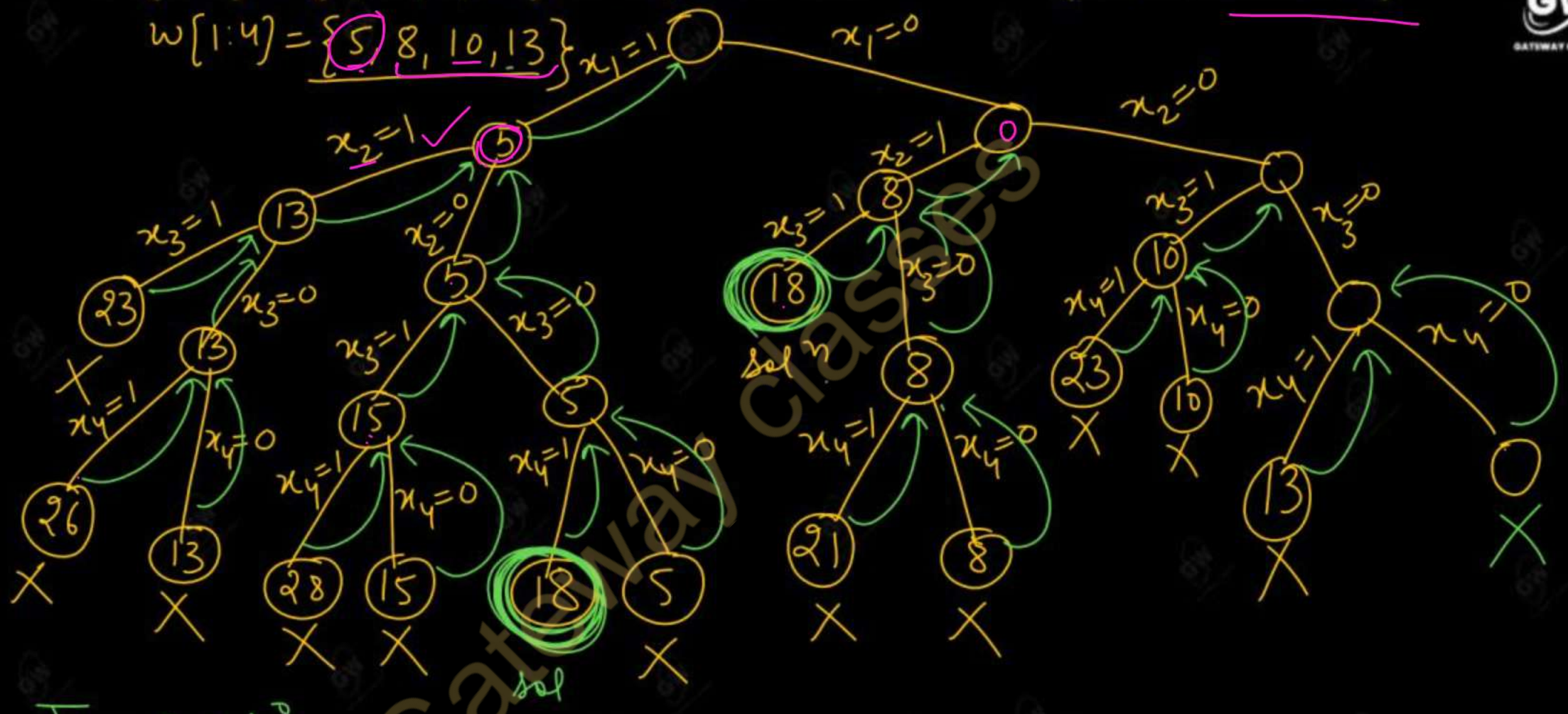
S (sum of all the numbers added befor K^{th} no.

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

↳ value of the current Number

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

$W\{1:4\} = \{5, 10, 8, 13\}\ n = 4,\ m = 18$ (AKTU 2018–19)



Two solutions

1) X = | 1 | 0 | 0 | 1 |

2) X = | 0 | 1 | 1 | 0 |

Rules for writing Pseudocode

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

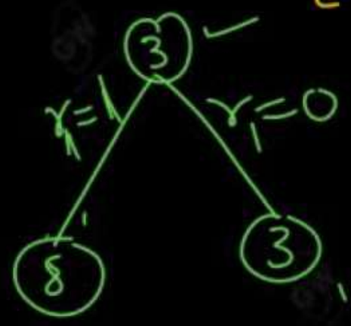
First no. $\leq M$

Total sum of all numbers $\geq M$

Initially $S = 0$, $K = 1$ (no. being considered) and $r = \sum_{i=1}^{n} w_i$

At kth 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]);
    }
}
```



$W\{1:6\} = \{5,10,12,13,15,18\}, n = 6, \quad m = 30$ (AKTU–2021–22)

Complete state space tree in similar way

	x_1	x_2	x_3	x_4	x_5	x_6
$x =$	1	1	0	0	1	0
$x =$	1	0	1	1	0	0
$x =$	0	0	1	0	0	1

AKTU PYQs

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

GW GATEWAY CLASSES

CS IT & CS Allied

DAA : Design & Analysis Of Algorithm

Unit-4 : Lecture-8

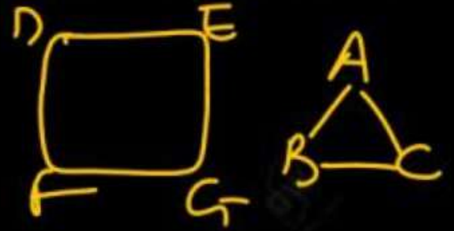
Today's Target

- Solving Hamiltonian cycle problem using backtracking
- AKTU PYQs

By Dr. Nidhi Parashar Ma'am
- M.Tech Gold Medalist
- Net Qualified

Hamiltonian Cycle

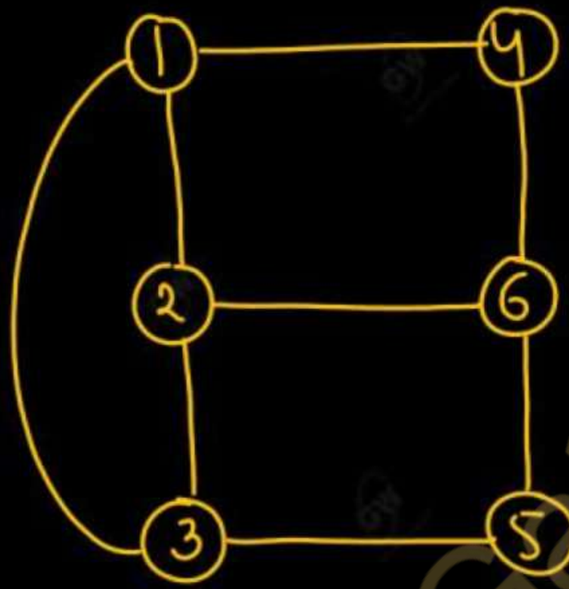
- In a given graph we start from some starting vertex and visit all remaining vertices of the graph exactly once and return back to the same starting vertex. The cycle formed is called Hamiltonian cycle.
- The problem is to check if Hamiltonian cycle is possible in a graph, and if possible, then what is a cycle, if multiple Hamiltonian cycles are present then we have to find all those cycles. $T(n) = O(n^n)$



- 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∈G and vertices of G are visited in the order v_1, v_2 ..... 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.



(n)

same

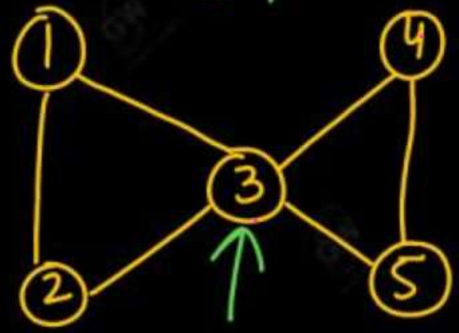
$1 \to 2 \to 3 \to 5 \to 6 \to 4 \to 1$

$1 \to 4 \to 6 \to 5 \to 3 \to 2 \to 1$

$2 \to 1 \to 4 \to 6 \to 5 \to 3 \to 2$

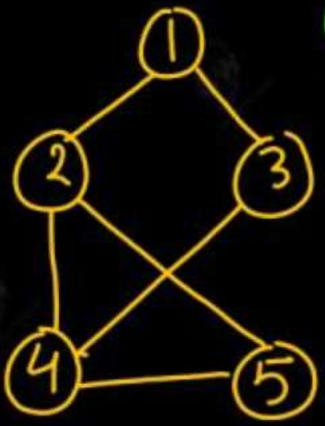
~~$3 \to 5 \to 2 \to 6 \to 4 \to 1 \to 3$~~

(A)



Articulation points

(B)

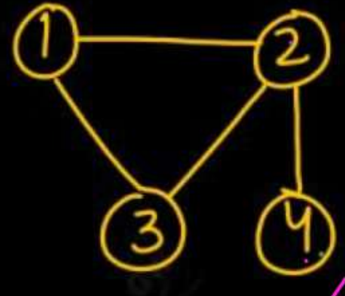


$1 \to 2 \to 5 \to 4 \to 3 \to 1$

$1 \to 3 \to 4 \to 5 \to 2 \to 1$

2 hamiltonian cycles are present

(C)



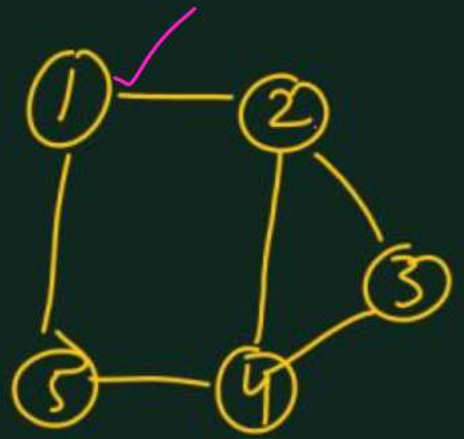
pendant node

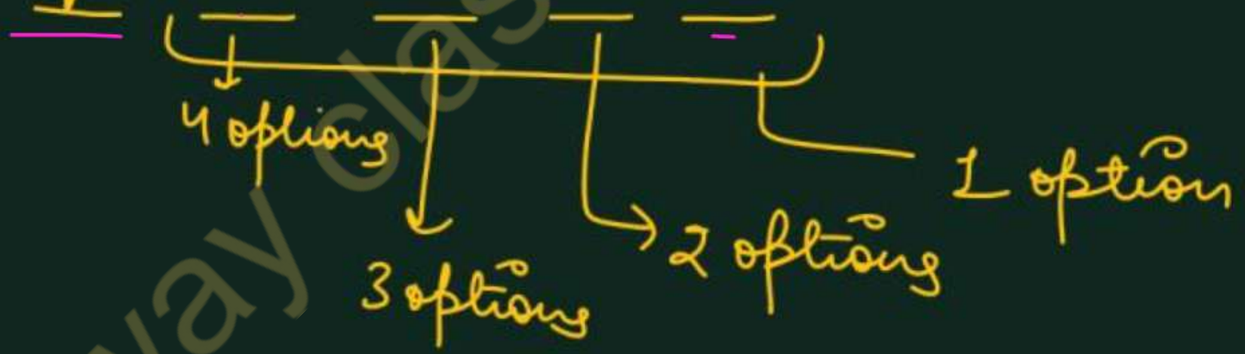
For Complete Graph :-
(for n-nodes)

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

$= (n-1)!$ (directed graph)

Using Brute-force Approach





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

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

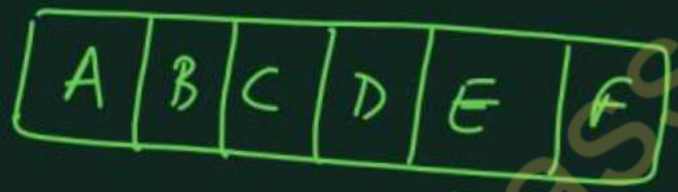
Solving Hamiltonian cycle problem using backtracking

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

Solving Hamiltonian cycle problem using backtracking

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

Solutions

1) $x =$ 

2) $x =$ 

3) $x =$ 

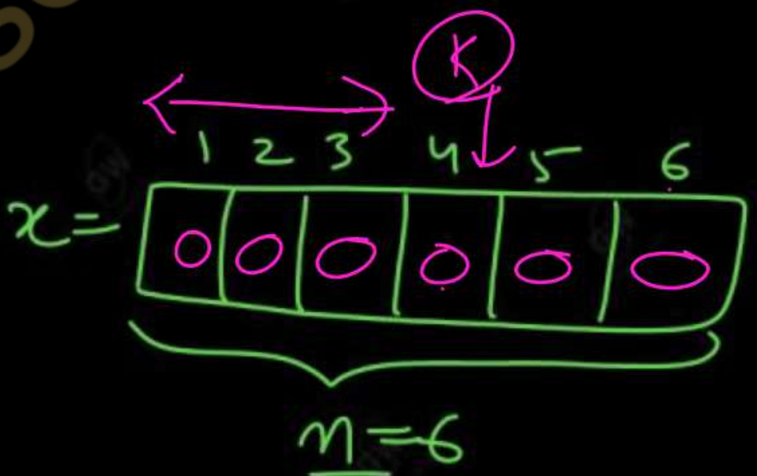
4) $x =$ 

5) $x =$ 

6) $x =$ 

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
        If (k= =n) then write (x[1……n])
        else Hamiltonian (k+1)
    } untill (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 $x[K-1]$.

1 — K-1 K^{th} $x[k]$

```
Algorithm Next value (K)
{
    repeat
    {
        X[K] = (x [K]+1) mod  (n+1) // Next vertex
        If (x [K] = =0 ) then return
        If ( ( G[x[k-1], x[k]] ≠ 0) then
        {
            for j=1 to k-1 do
                if(x [j] = = x [k] ) then break.
            if (j= = k) then
                if((k < n) or (( k= = n) and ( G[x[n], x[1]] ≠ 0) )
                    then return

        } until (false)

    }
```

Initially x[k]=0

// All vertices are explored.

// There is an edge b/w K^th and K-1^th vertex

// condition for repeated Vertex

this is the last vertex

still some vertices are remaining

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

n = 5

mod 6

Applications:-

(1) May be used to model and optimize travel results for public transportations 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.

AKTU PYQs

GW GATEWAY CLASSES

(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

GW GATEWAY CLASSES

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

AKTU PYQs

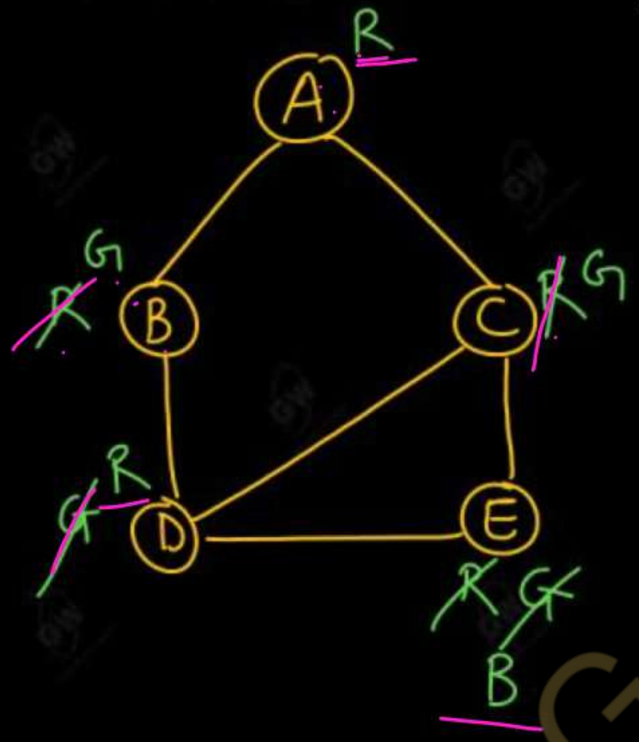
(1) Write applications of graph coloring. (AKTU 2022-23, 2021-22)

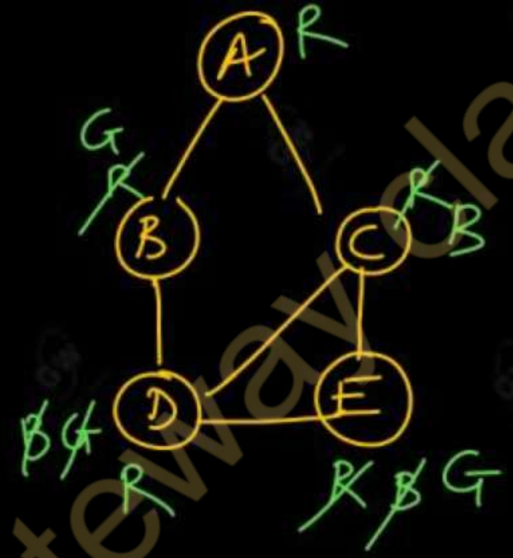
(2) Define the term "Graph coloring". (AKTU 2023-24)

Graph Coloring

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, C, B}





Colors = { R, G, B }

Solutions

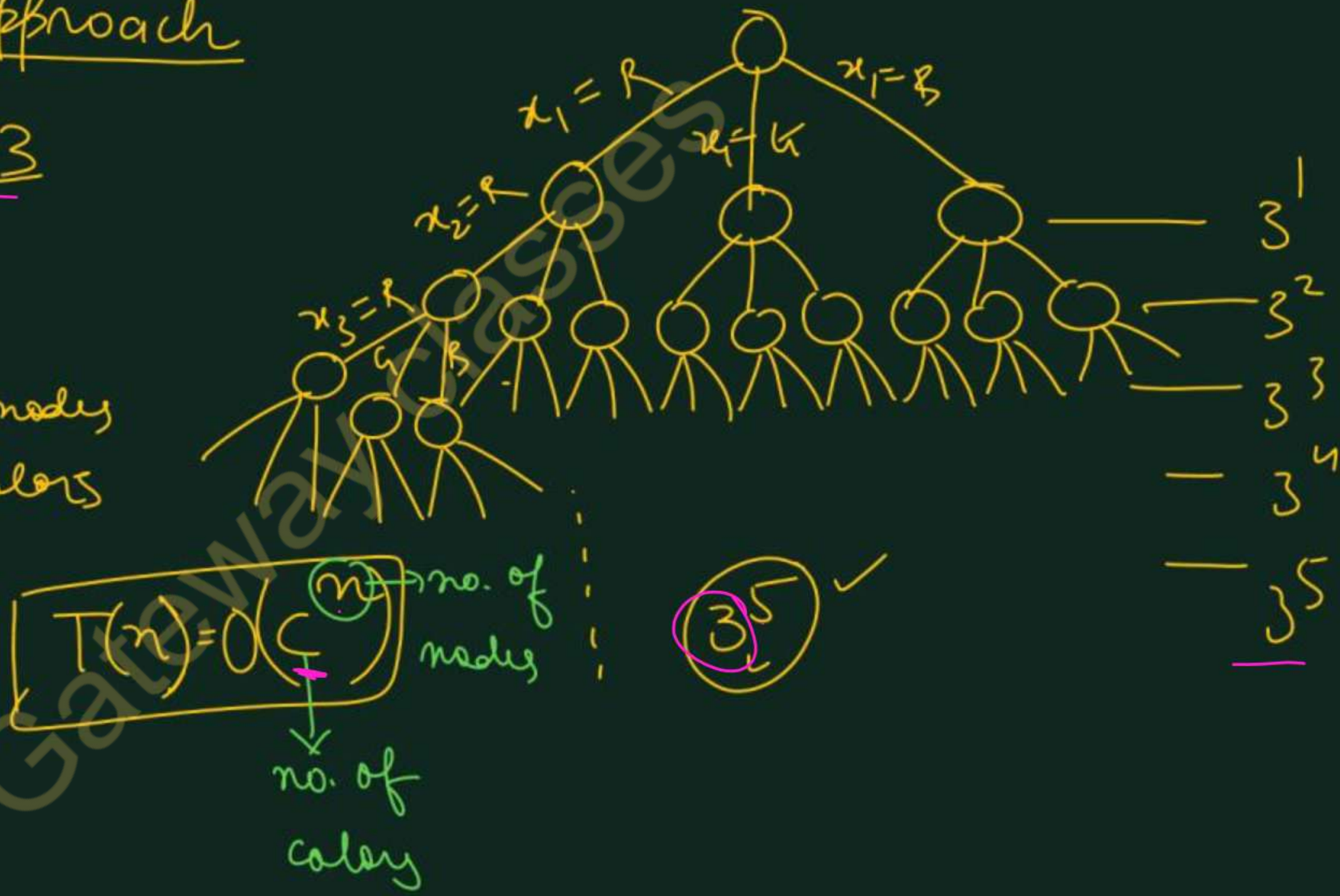
	A	B	C	D	E
1)	R	G	G	R	B
2)	R	G	B	R	G

Brute Force Approach

3 3 3 3 3

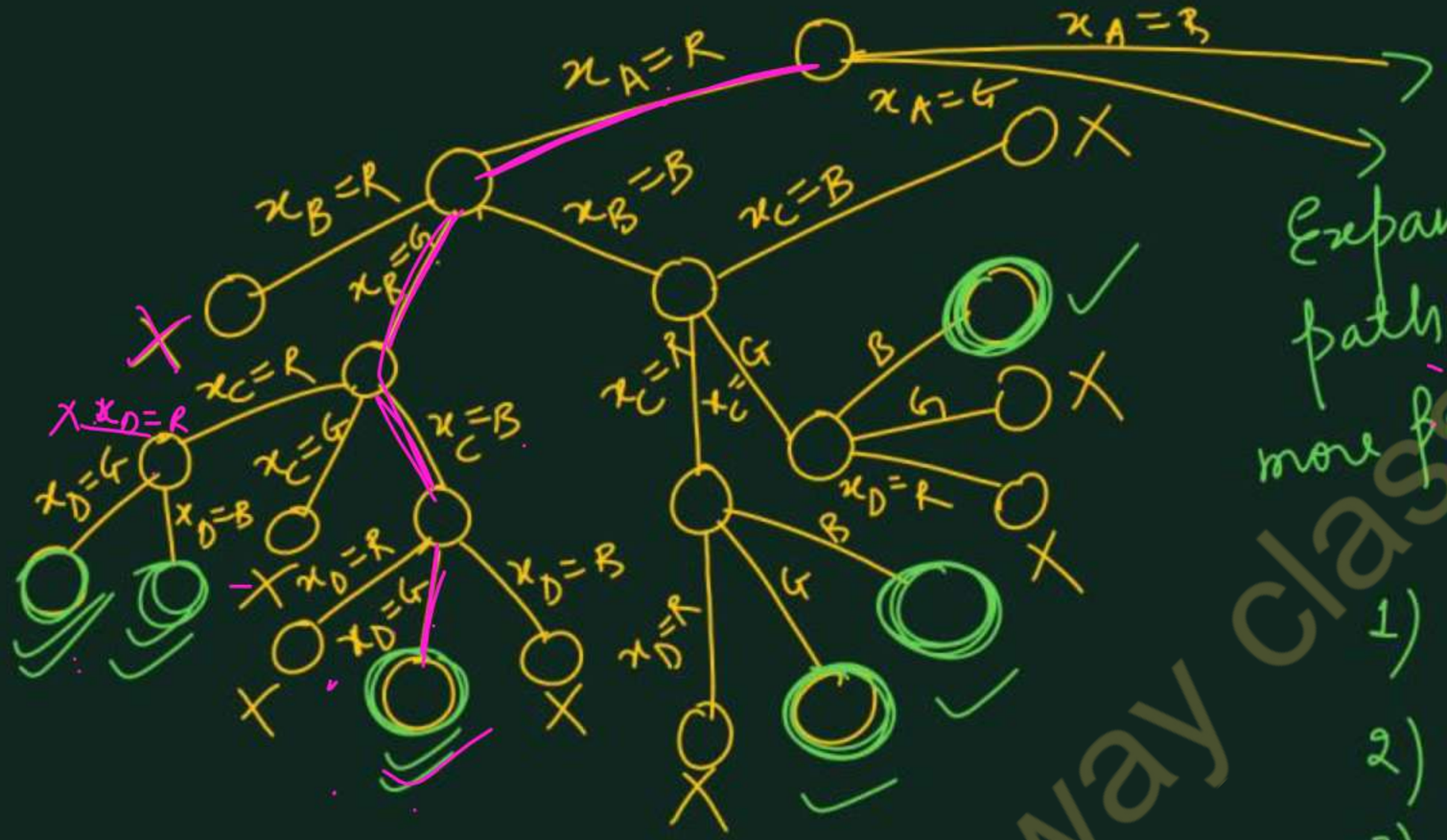
$3 \times 3 \times 3 \times 3 \times 3$

$\Rightarrow 3^5$ → 5: no. of nodes; 3: no. of colors

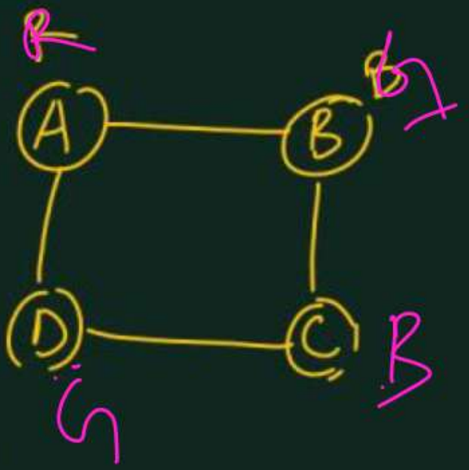


$T(n) = O(c^n)$ → n: no. of nodes; c: no. of colors

3^5 ✓



Expand these path fully to get more feasible solutions



	A	B	C	D
1)	R	G	R	G
2)	R	G	R	B
3)	R	G	B	G
4)	R	B	R	G
5)	R	B	R	B
6)	R	B	G	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
    {
        X[K] = (x [K]+1) mod  (m+1) // Next highest node color
        If (x [K] = =0 ) then return // color exhausted
        For j = 1 to n do
        {
            If ( ( G[K, j]≠0) and (x [K] = = x [j] )) then break.  // adjacent with some color.
        }
        If (j = =(n+1) then return                                  // new color found otherwise find other
    } until (false)
}
```

0 initially

No. of colors = m

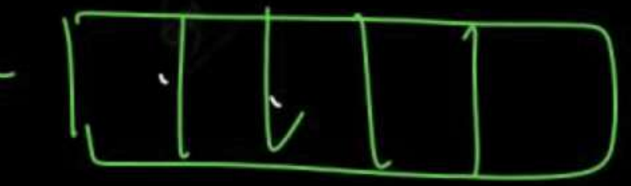
No more colors are remaining

n → No. of nodes

Adjacent node

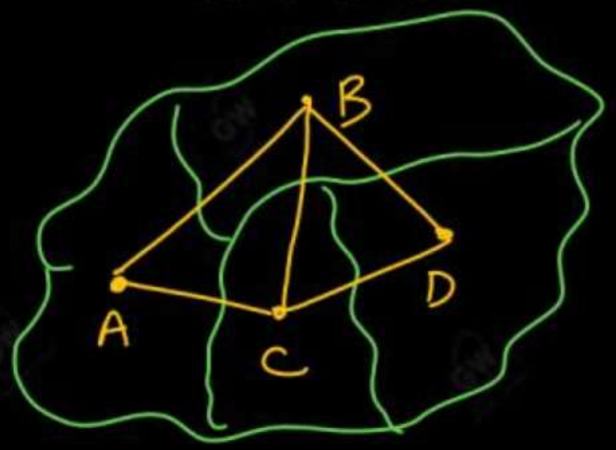
same color

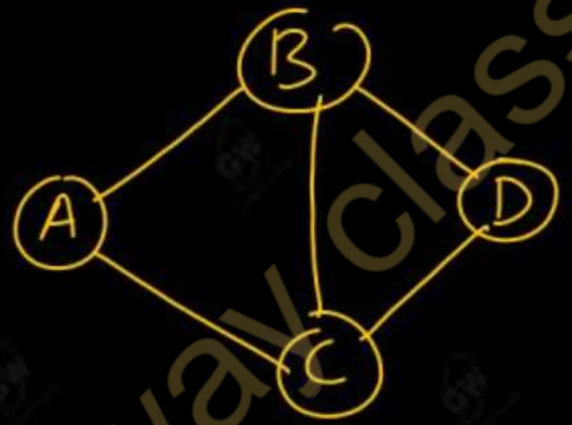
nodes



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.





⟹ Apply graph coloring using backtracking

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

3) Setting time schedules.

AKTU PYQs

(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

GATEWAY CLASSES

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

Branch and Bound

(1) Follows BFS (Breadth first search)

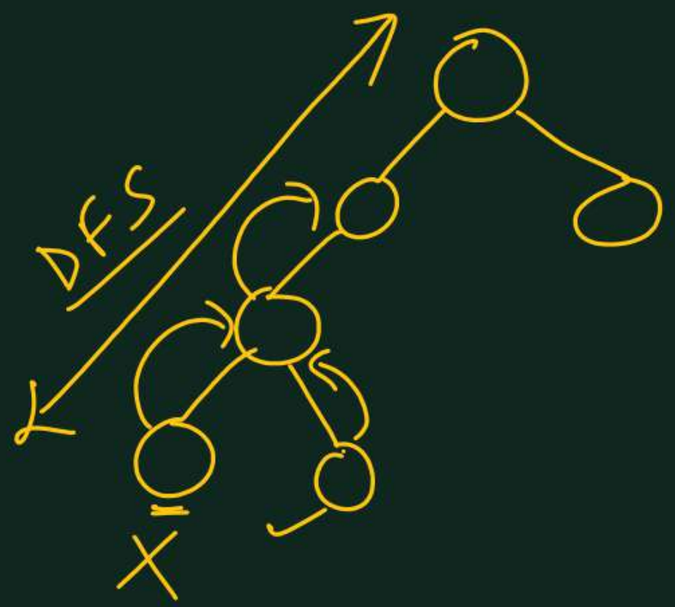
(2) the idea is to cut off a ~~Brach~~ branch of the problem's state – space tree as soon as we can deduce that it can not lead to a solution.

(3) Similar to back tracking as it uses a state space tree. But branch and bound is used to solve optimization problems.

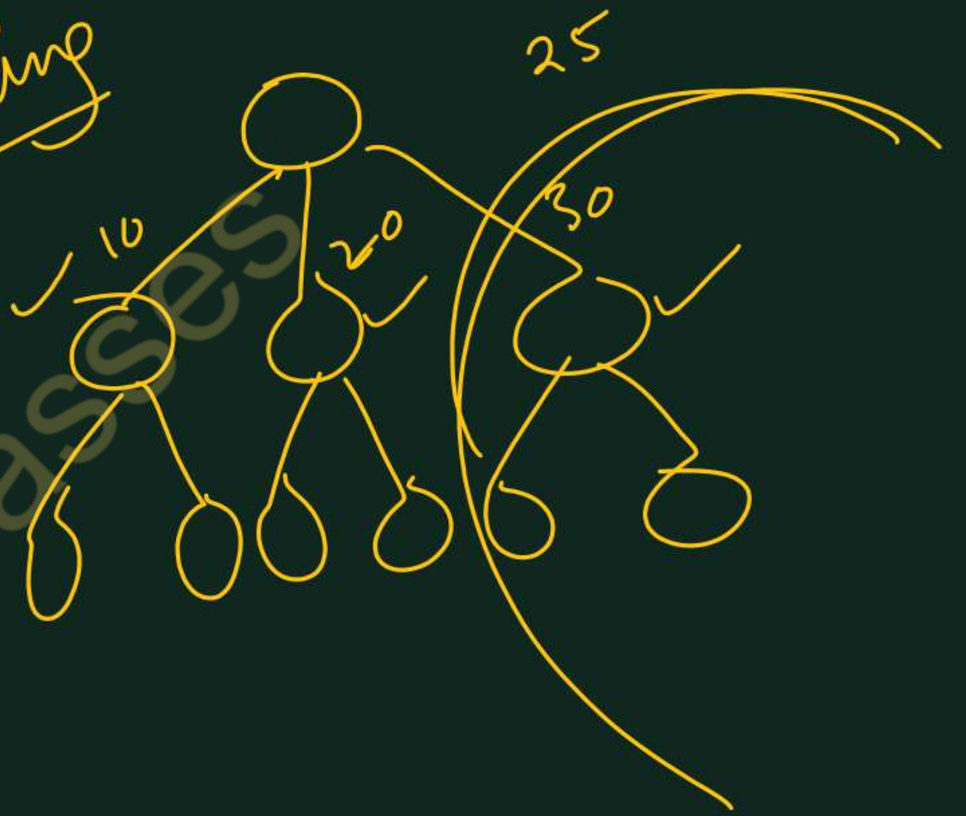
(4) Compared to backtracking branch and bound requires two additional steps –

(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.



bounding



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.

Difference between Backtracking and Branch and Bound.(AKTU 2020-21, 2022-23)

Parameter	Backtracking	Branch and Bound
Approach	Write definitions	Write definitions
Traversal	traverses the state space tree by DFS (Depth First Search) manner.	traverse the tree by BFS (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 N-Queen Problem, Sum of subset, Hamiltonian cycle problem, graph coloring problem	Useful in solving Knapsack Problem, Travelling Salesman Problem.
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.

(AKTU-2016-17)

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

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

$$\begin{bmatrix} \infty & 20 & 30 & 10 & 11 \\ 15 & \infty & 16 & 4 & 2 \\ 3 & 5 & \infty & 2 & 4 \\ 19 & 6 & 18 & \infty & 3 \\ 16 & 4 & 7 & 16 & \infty \end{bmatrix} \begin{matrix} 10 \\ 2 \\ 2 \\ 3 \\ 4 \end{matrix}$$

row-wise Reduct^n →

$$\begin{bmatrix} \infty & 10 & 20 & 0 & 1 \\ 13 & \infty & 14 & 2 & 0 \\ 1 & 3 & \infty & 0 & 2 \\ 16 & 3 & 15 & \infty & 0 \\ 12 & 0 & 3 & 12 & \infty \end{bmatrix}$$

21

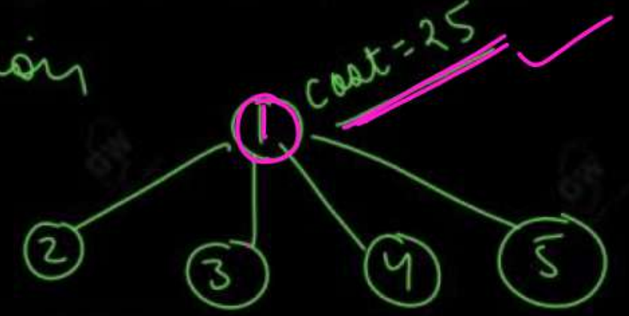
1 0 3 0 0 = 4

col-wise reduction

reduction cost = 21+4 = 25

$$\begin{bmatrix} \infty & 10 & 17 & 0 & 1 \\ 12 & \infty & 11 & 2 & 0 \\ 0 & 3 & \infty & 0 & 2 \\ 15 & 3 & 12 & \infty & 0 \\ 11 & 0 & 0 & 12 & \infty \end{bmatrix}$$

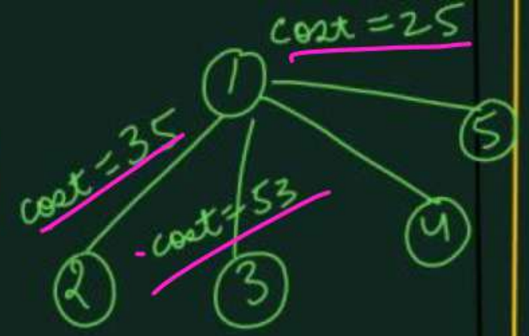
This cost matrix will be used to prepare cost matrix for 1→2, 1→3, 1→4, 1→5



upper = ∞

node (1)

	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	∞



$r = 25$

To calculate cost (1, 2) → Make first row and 2nd col ⇒ ∞, and we don't need to return to node 1 from node 2, so we have to make the entry for (2 → 1) as ∞.

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

⇒ This matrix is already reduced.

$\hat{r} = 0$

So cost (1,2) = cost (1,2) + r + $\hat{r}$ → reduction cost of (1 → 2) matrix

↓ reductions cost of reference Matrix

cost (1,2) = 10 + 25 + 0 = 35

To calculate cost (1, 3) → set 1st row and 3rd col ⇒ ∞ and (3 → 1) entry ⇒ ∞

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

⇒ Row wise reduced ⟹

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

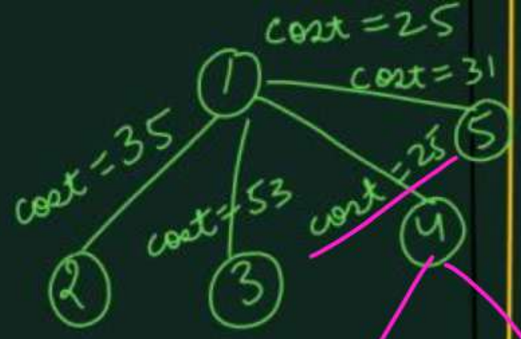
$\hat{r} = 11$

cost (1,3) = cost (1,3) + r + $\hat{r}$

= 17 + 25 + 11 ⇒ 53

node (1)

	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	∞



For cost (1,4) ⇒ 1st row, 4th col ⇒ ∞

(4→1) entry ⇒ ∞

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

⇒ already row-wise and col-wise reduced.

$\hat{r}_4 = 0$

cost (1,4) = 0 + 25 + 0 = 25

cost (1,5) ⇒ 1st row, 5th col ⇒ ∞

(5→1) entry ⇒ ∞

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

⇒

∞	∞	∞	∞	∞
10	∞	9	0	∞
0	3	∞	0	∞
12	0	9	∞	∞
∞	0	0	12	∞

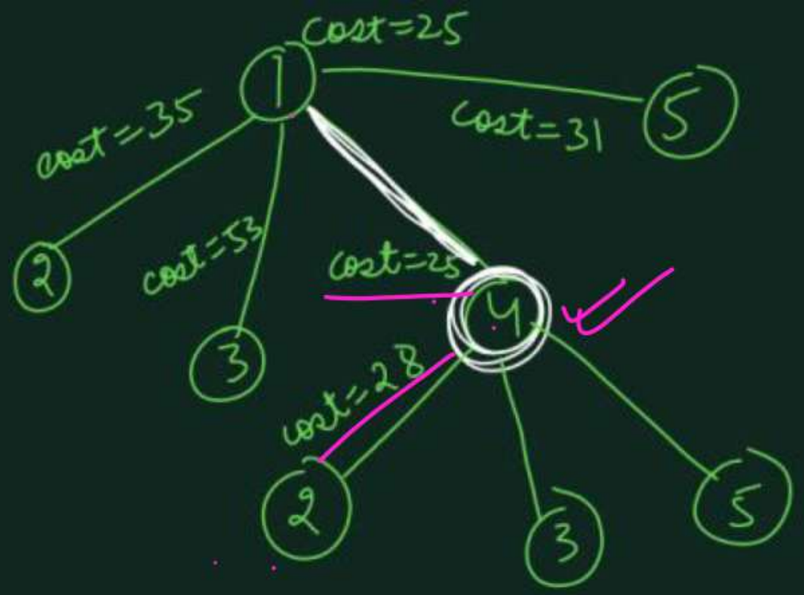
It is col-wise already reduced.

$\hat{r}_4 = 5$

5

cost (1,5) = cost (1,5) + r + r̂

= 1 + 25 + 5 = 31 ✓



cost (1,4) matrix will be used to calculate the cost (4,2), cost (4,3) and 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	∞	∞

$r = 25$

cost (4,2) $\Rightarrow$ 4th row, 2nd col $\Rightarrow \infty$

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

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

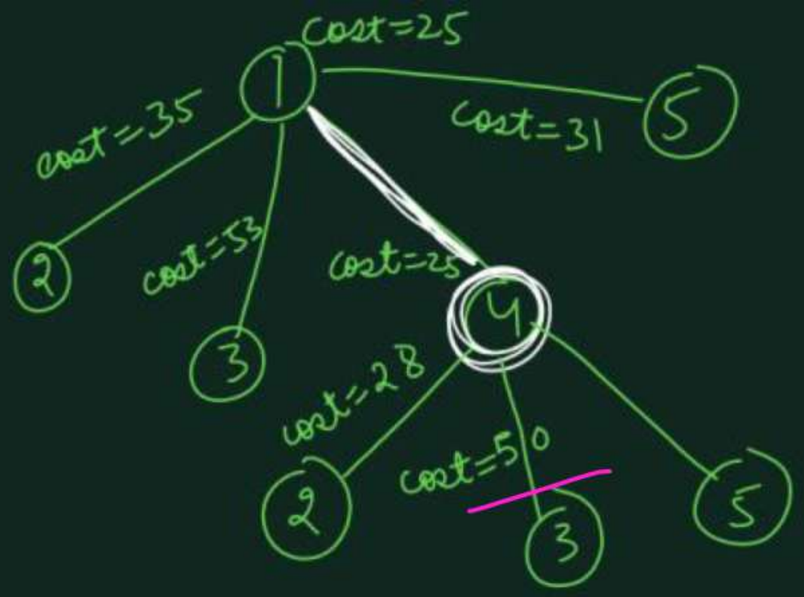
$\Rightarrow$ already Reduced.

$\hat{r} = 0$

$$\text{cost}(4,2) = \text{cost}(4,2) + r + \hat{r}$$

$$= 3 + 25 + 0$$

$$= 28$$



cost (1,4) matrix will be used to calculate the cost (4,2), cost (4,3) and 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	∞	∞

$\mathcal{A}_1 = 25$

cost (4,3) $\Rightarrow$ 4th row, 3rd col $\Rightarrow \infty$

(3→1) entry $\Rightarrow \infty$

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

$\Rightarrow$

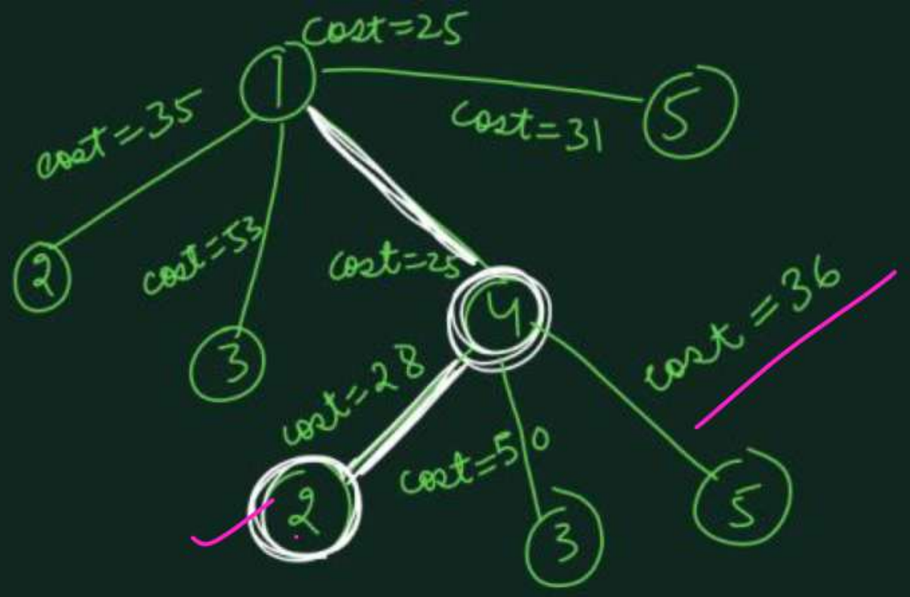
∞	∞	∞	∞	∞	
12	∞	∞	∞	0	
∞	1	∞	∞	0	
∞	∞	∞	∞	∞	
11	0	∞	∞	0	
11	0	0	0	0	$\Rightarrow$ (11)

$\mathcal{A} = 2 + 11 = 13$ ✓ (2)

cost (4,3) = 12 + 25 + 13

= (50) ✓

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



cost(1,4) matrix will be used to calculate the cost(4,2), cost(4,3) and 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	∞	∞

$\lambda = 25$

cost(4,5) ⇒ 4th row, 5th col } ⇒ ∞

(5→1) entry ⇒ ∞

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

⇒

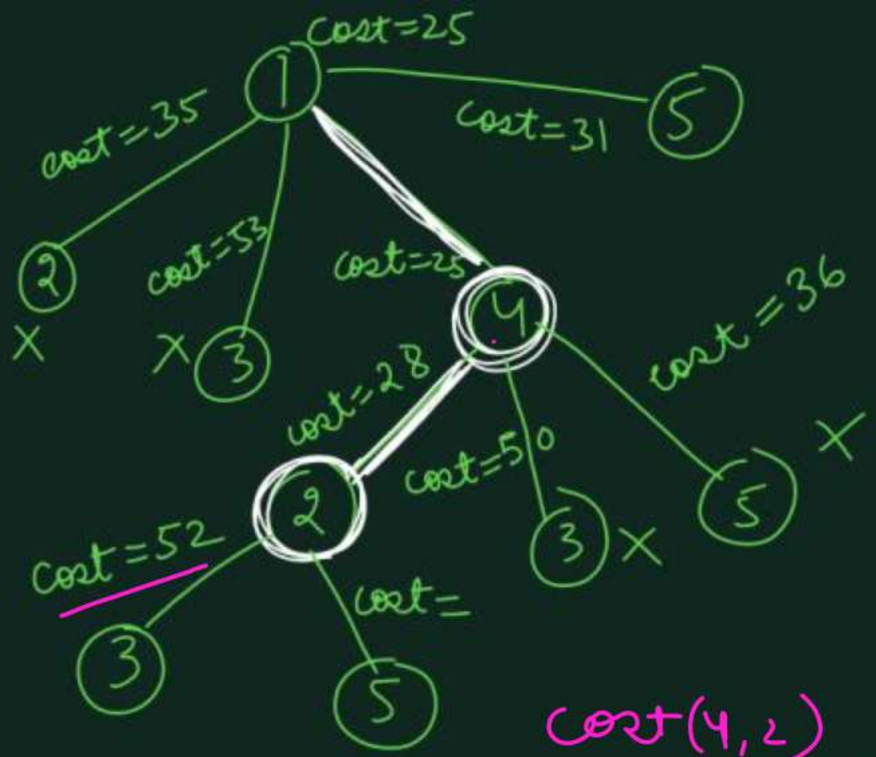
∞	∞	∞	∞	∞
1	∞	0	∞	∞
0	3	∞	∞	∞
∞	∞	∞	∞	∞
∞	0	0	∞	∞

11

$\lambda = 11$

cost(4,5) = 0 + 25 + 11

= 36



cost(4,2) Matrix

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

$r_1 = 28$

Cost(2,3) ⇒ 2nd row, 3rd col } ⇒ ∞

(3→1) entry ⇒ ∞

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

⇒

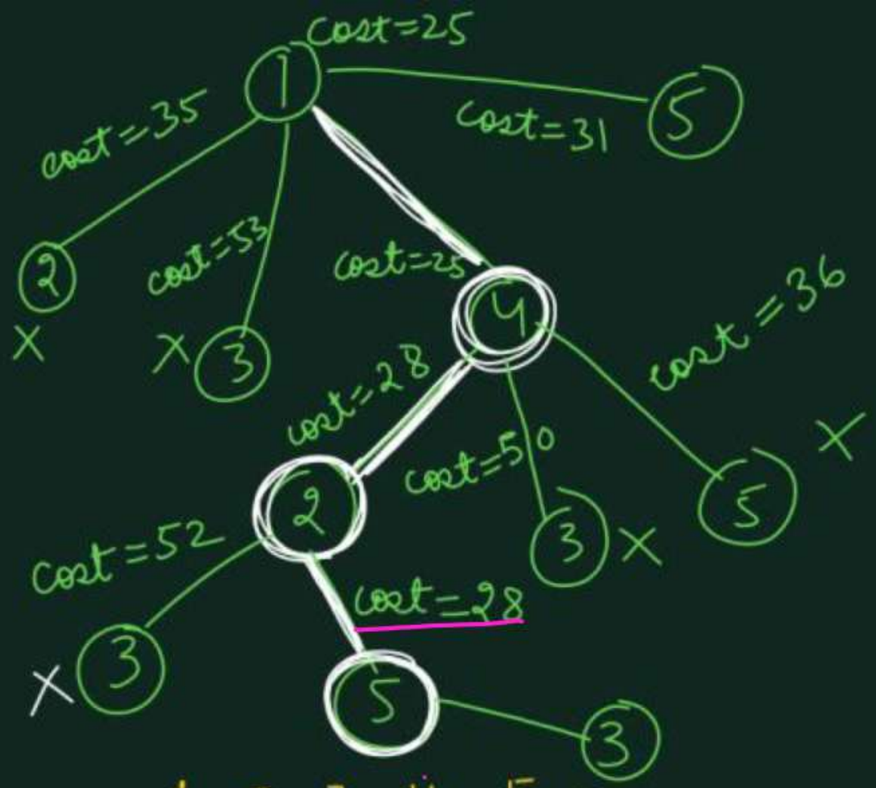
∞	∞	∞	∞	∞
∞	∞	∞	∞	∞
∞	∞	∞	∞	0
∞	∞	∞	∞	∞
0	∞	∞	∞	∞

$\hat{r} = 13$ ✓

$cost(2,3) = cost(2,3) + r_1 + \hat{r}$

$= 11 + 28 + 13$

$= 52$ ✓



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

$r_1 = 28$

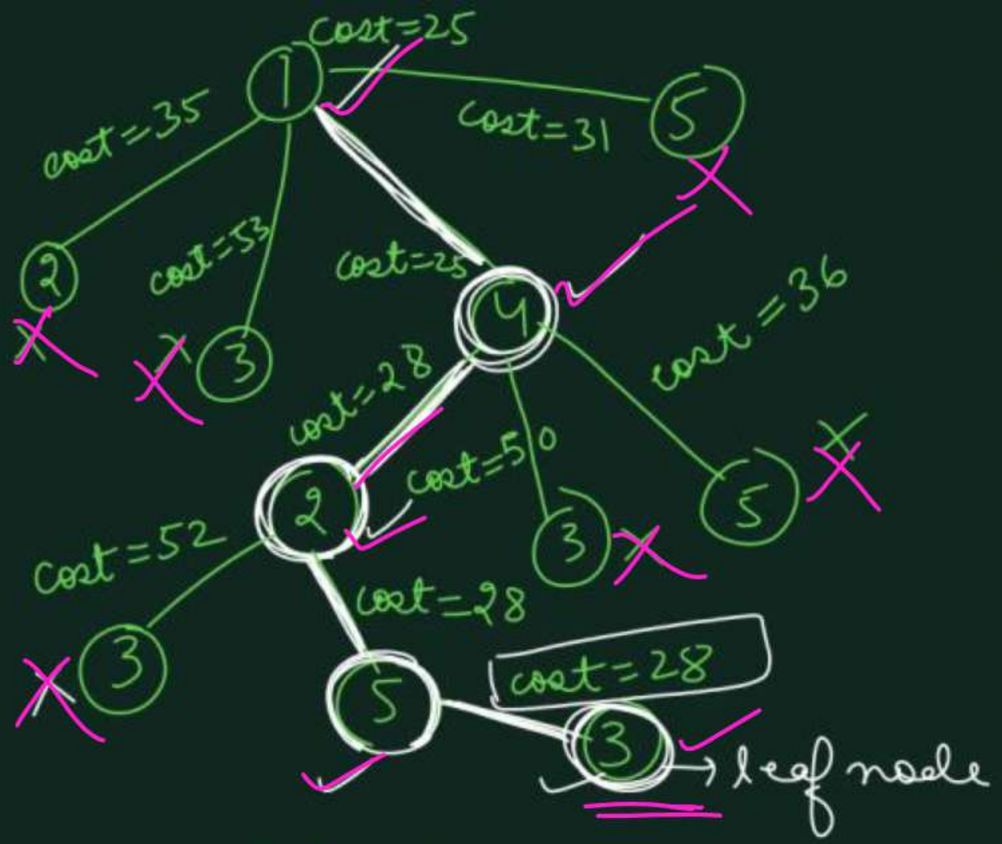
Cost(2,5) ⇒ 2nd row, 5th col } ⇒ ∞

(5→1) entry ⇒ ∞

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

$\hat{r} = 0$ ✓

cost(2,5) = 0 + 28 + 0 ⇒ 28



cost (5,3) { 5th row ⇒ ∞
3rd col ⇒ ∞
(3→1) ⇒ ∞

cost (2→5)

//

A = 28

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

cost (5,3) = 0 + 28 + 0
= 28

cost (5,3)

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

λ = 0

cost of leaf node = 28

So this is the minimum cost achieved.

upper = ∞

upper = 28

Closed path :— 1 → 4 → 2 → 5 → 3

AKTU PYQs

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	∞