Date: __/__/__ Sheet 5 Page No. : ____

1) What is backtracking?

Backtracking is a problem-solving algorithmic technique that involves finding a soln incrementally by trying different options & undoing them if they lead to dead end.

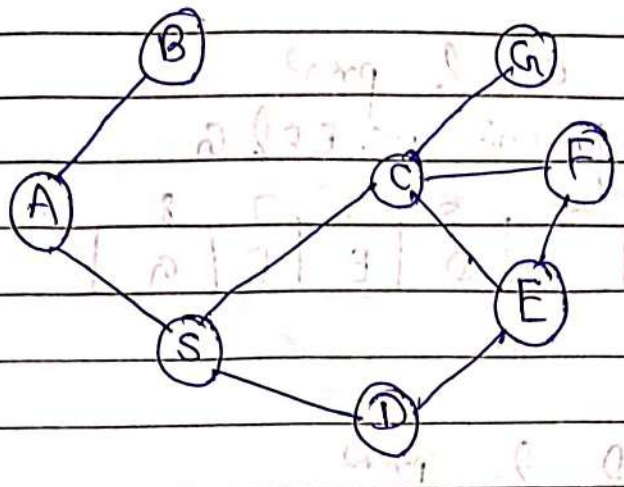
Commonly used in situations where we need to explore multiple possibilities to solve a problem.
eg searching path in maze
solving sudoku puzzles.

When dead end is reached, the algorithm backtracks to previous decision point and explores a different path until a soln is found or all possibilities have been exhausted.

Types of Backtracking problems :-

1. Decision problems
2. optimization problems
3. Enumeration problems

Date: __/__/__ Page No.: ____



BFS

Let here starting vertex (A).

Step 1: Insert starting vertex A in queue.

1	2	3	4	5	6	7	8
A							

Step 2: Remove A & print it
insert its adjacent vertex i.e. B, S

1	2	3	4	5	6	7	8	Print
	B	S						A

Step 3: Remove B & print it
insert its adjacent: None

1	2	3	4	5	6	7	8	Print
		S						A, B

Step 4: Remove S & print it
insert its adjacent. i.e. C & D

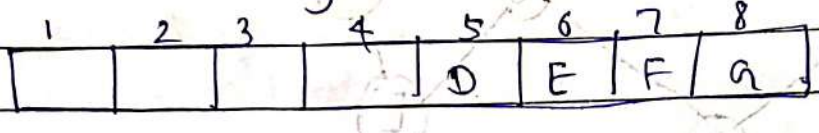
1	2	3	4	5	6	7	8	Print
		C	D					A, B, S

Date: __/__/__ Page No.: ____

Step 5

Remove C & print inner adjacent i.e. E, F & G

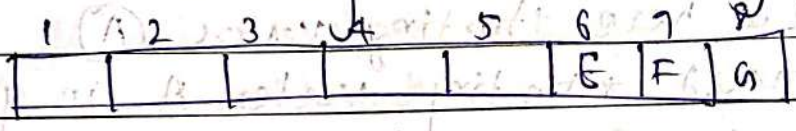
Print: A, B, S, C



Step 6

Remove D & print inner adjacent i.e. already present

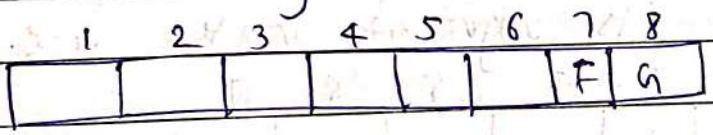
Print: A, B, S, C, D



Step 7

Remove E & print inner adj.

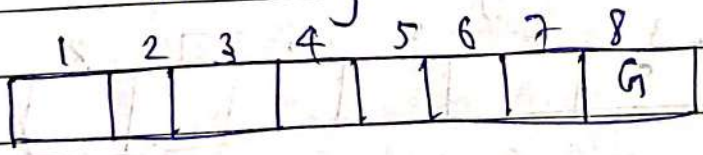
Print: A, B, S, C, D, E



Step 8

Remove F & print inner adj

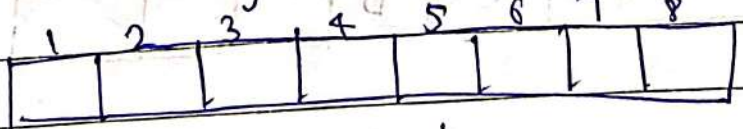
Print: A, B, S, C, D, E, F



Step 9

Remove G & print inner adj

Print: A, B, S, C, D, E, F, G



Here, q were is empty.

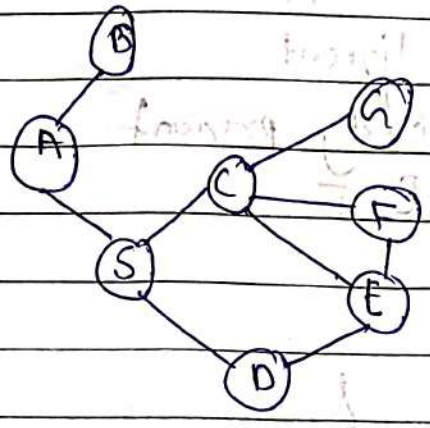
So, search is complete

A → B → S → C → D → E

G ← F

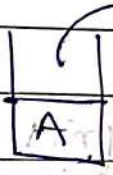
Date: __/__/__ Page No.: ____

DFS



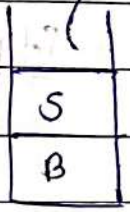
Stack ld, Here initial vertex is (A)

Step 1: Insert A in stack



Step 2: ~~insert its adj~~

pop topmost element (A) & insert its adjacent in stack
print poped element



~~Step~~

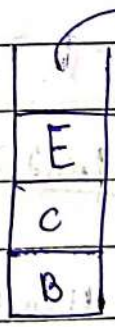
print
A

Step 3: pop topmost element (S) & insert its adj

print A, S

Step 4: pop (D) & insert adj

print A, S, D



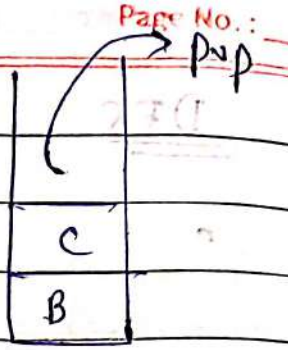
Step 5: pop (E) & insert adj

print A, S, D, E

Date: __/__/__ Page No.: ____

Step 6: pop (F) & insert adjacent i.e. already present

print A, S, D, E, F

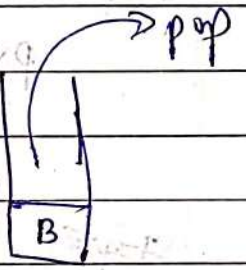


Step 7: pop (C) & insert it's adjacent ~~i.e. C~~

print A, S, D, E, F, C

Step 8: pop (G) & insert adj. i.e. none

print A, S, D, E, F, C, G



Step 9: Pop (B) & insert adj i.e. none

Here stack is empty

empty stack

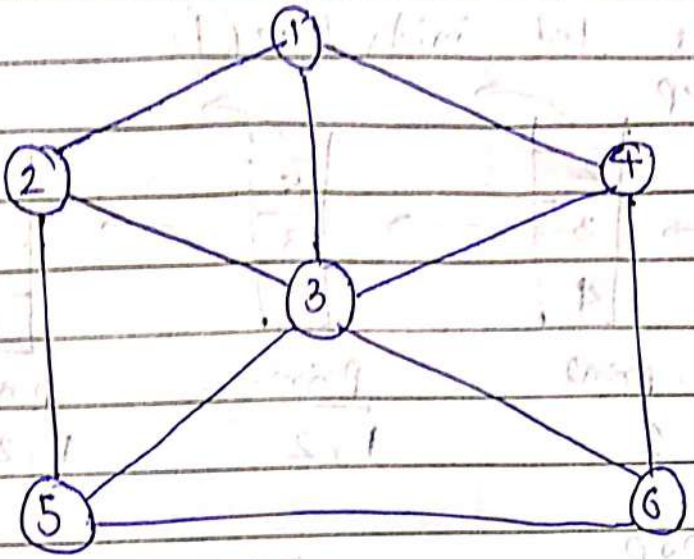
print: A, S, D, E, F, C, G, B

Here traversal is completed

A → S → D → E → F → C → G → B

Date: __/__/__ Page No. : ____

Q 2)



BFS Led, Here initial ①

Direct krta

S1

1	2	3	4	5	6
1					

S2

1	2	3	4	5	6
	2	3	4		

print 1

S3

1	2	3	4	5	6
		3	4	5	

print 1, 2

1 → 2 → 3 → 4
↓
6 ← 5

S4

1	2	3	4	5	6
			4	5	6

print 1, 2, 3

S5

1	2	3	4	5	6
				5	6

print 1, 2, 3, 4

S6

1	2	3	4	5	6
					6

print 1, 2, 3, 4, 5

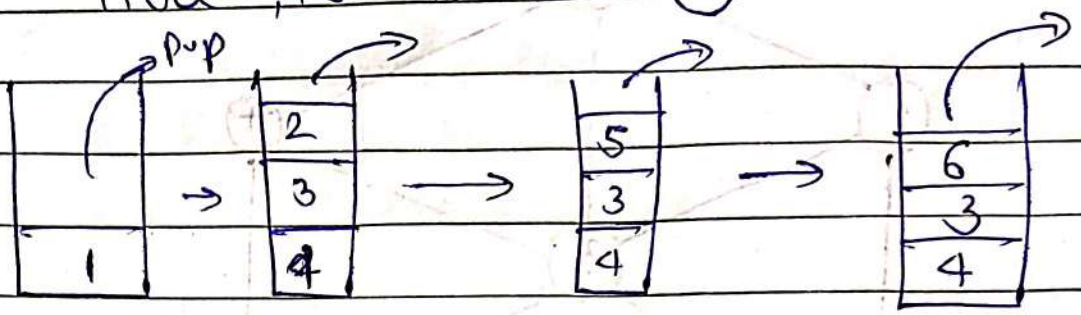
S7

1	2	3	4	5	6

empty

print 1, 2, 3, 4, 5, 6

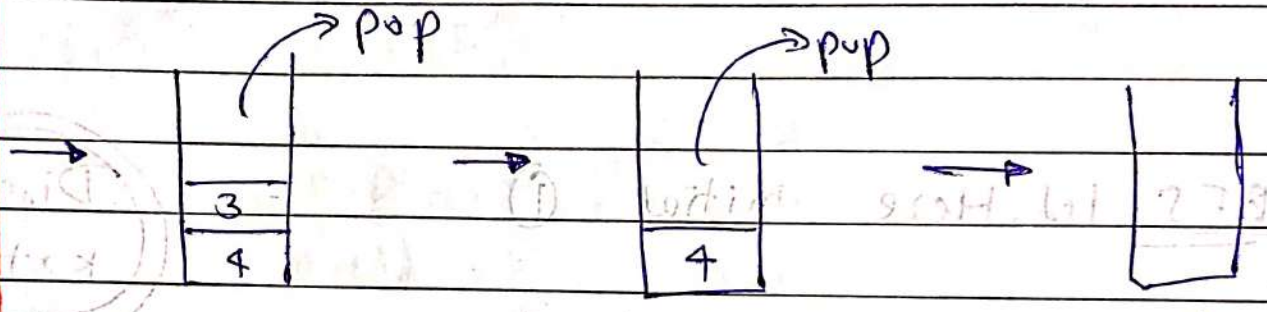
DFS Here, let initial is (1)



print
1

print
1, 2

print
1, 2, 5



print
1, 2, 5, 6

print
1, 2, 5, 6, 3

empty

print
1, 2, 5, 6, 3, 4

Traversal is complete

1 → 2 → 5 → 6 → 3 → 4

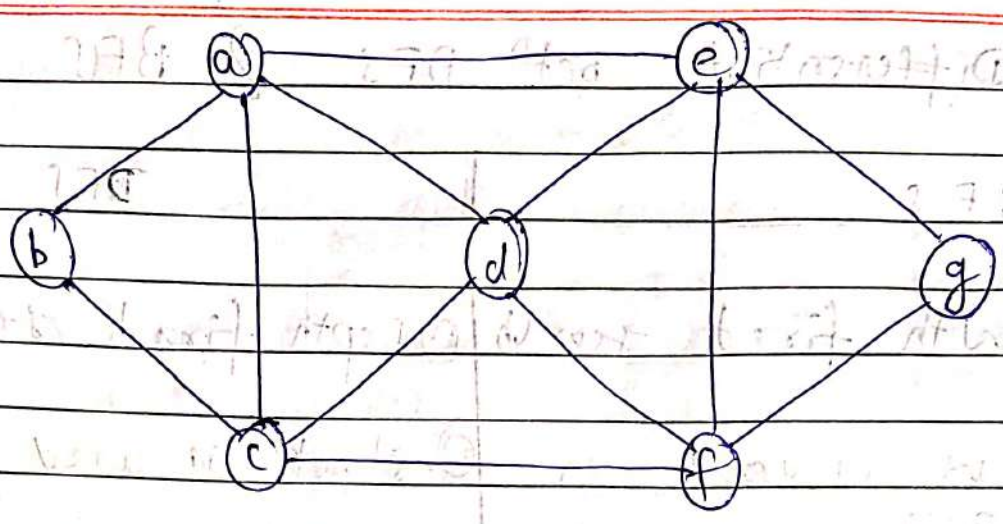
Date: __/__/__

Page No. : ____

3. Differentiate betn DFS & BFS

BFS	DFS
① Breadth first search	① Depth first search
② queue is used in BFS	② stack is used
③ for finding shortest path.	
③ level by level	③ with depth, through nodes as far as possible with no unvisited nearby nodes.
④ Concept of FIFO	④ concept of LIFO
⑤ more suitable for searching vertices closer to given src.	⑤ more suitable when there are soln away from src.
⑥ not suitable for decision making trees, because it consider all neighbours first	⑥ more suitable for decisions, game & puzzle problems.
⑦ Time complexity O(V+E)	⑦ O(V+E).
⑧ Space complexity is high.	⑧ lesser complexity.
⑨ Speed is slow	⑨ Speed is fast
⑩ require more memory	⑩ require less memory.
⑪ optimal for finding shortest path.	⑪ not optimal for finding shortest path.

Date: __/__/__ Page No.: ____



BFS Let (a) in initial.

1	2	3	4	5	6	7	Print
a							
	b	c	d	e			a
		c	d	e			a, b
			d	e	f		a, b, c
				e	f		a, b, c, d
					f	g	a, b, c, d, e
						g	a, b, c, d, e, f
							a, b, c, d, e, f, g

empty queue

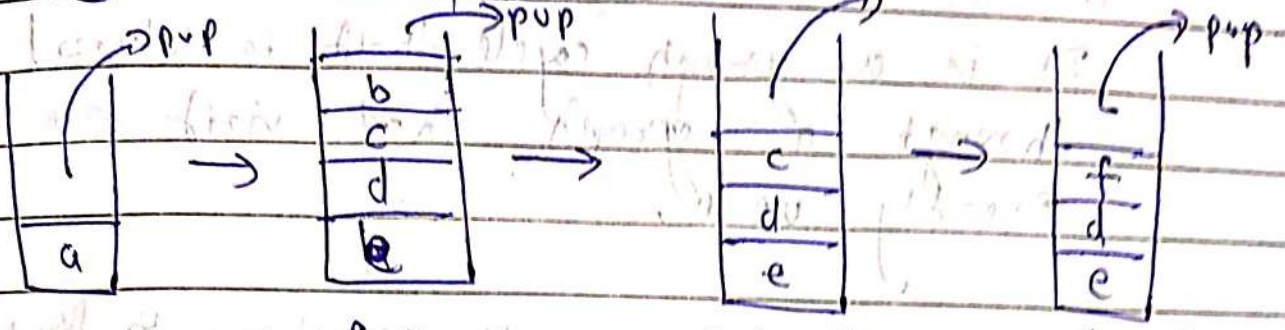
a → b → c → d → e → f → g

Date: __/__/__

Page No.: ____

DFS

Let (a) is initial



print a

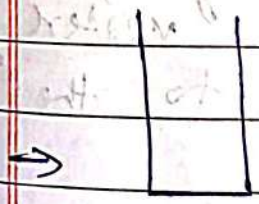
print a, b

print a, b, c

print a, b, c, f

print a, b, c, f, g

print a, b, c, f, g, d



empty stack

print a, b, c, f, g, d, e

Here traversal is completed.

a → b → c → f → g → d → e

Date: __/__/__ Page No.: _____

Q4. What is Hamiltonian cycle?

It is also called Hamilton circuit.
It is a graph cycle that is closed loop through a graph that visit each node exactly once.

In hamiltonian cycle, we have to find the all possible tour.

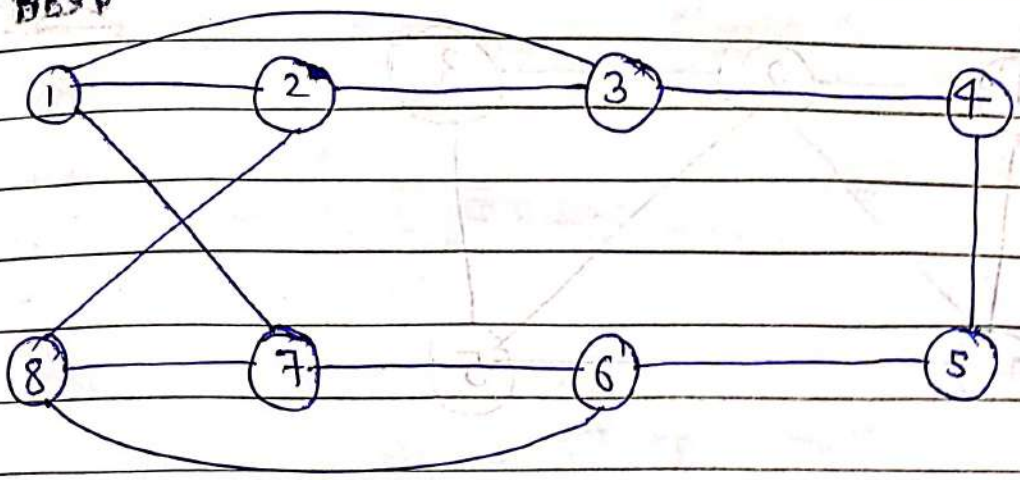
The graph may be directed or undirected, it must be connected. If it is not connected then cycle is not formed.
This is np-hard problem.

Hamiltonian cycle or circuit in a graph G is a cycle that visits every vertex of G exactly once and returns to the starting vertex.

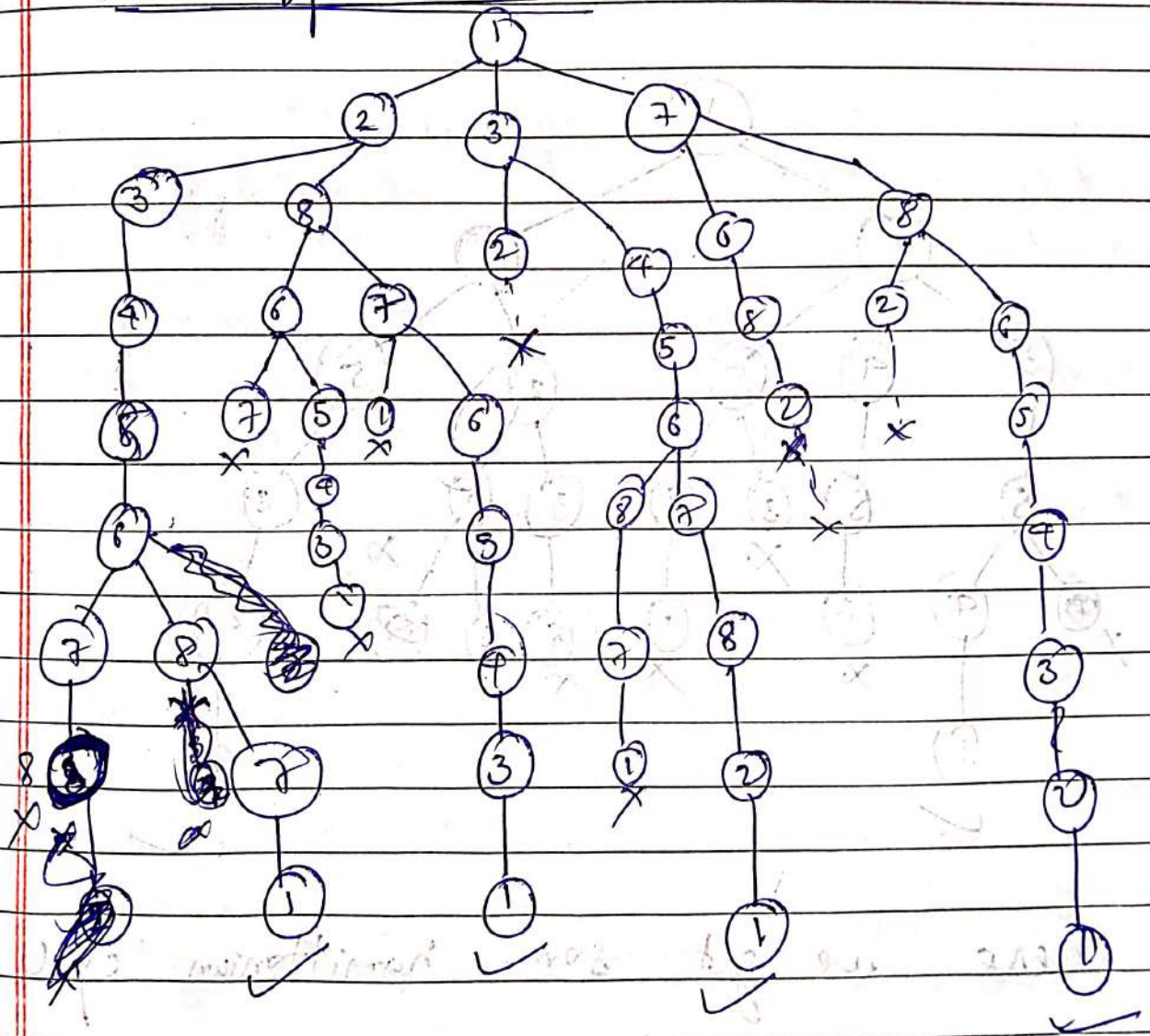
Use of hamiltonian Cycle:

- finding optimal route in transportation networks
- Circuit design
- graph theory research
- in computer graphics
- mapping genomes

DBSV



State space tree



Here we get 1 form of path, Hamiltonian cycle!

1 — 2 — 3 — 4 — 5 — 6 — 8 — 7 — 1

1 — 2 — 8 — 7 — 6 — 5 — 4 — 3 — 1

1 — 3 — 4 — 5 — 6 — 7 — 8 — 2 — 1

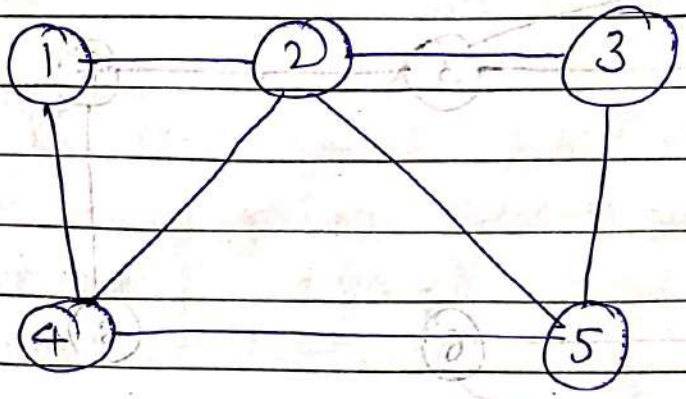
1 — 7 — 8 — 6 — 5 — 4 — 3 — 2 — 1

Q6 & Q7
bhi thoda sa karna
he hai
Veda

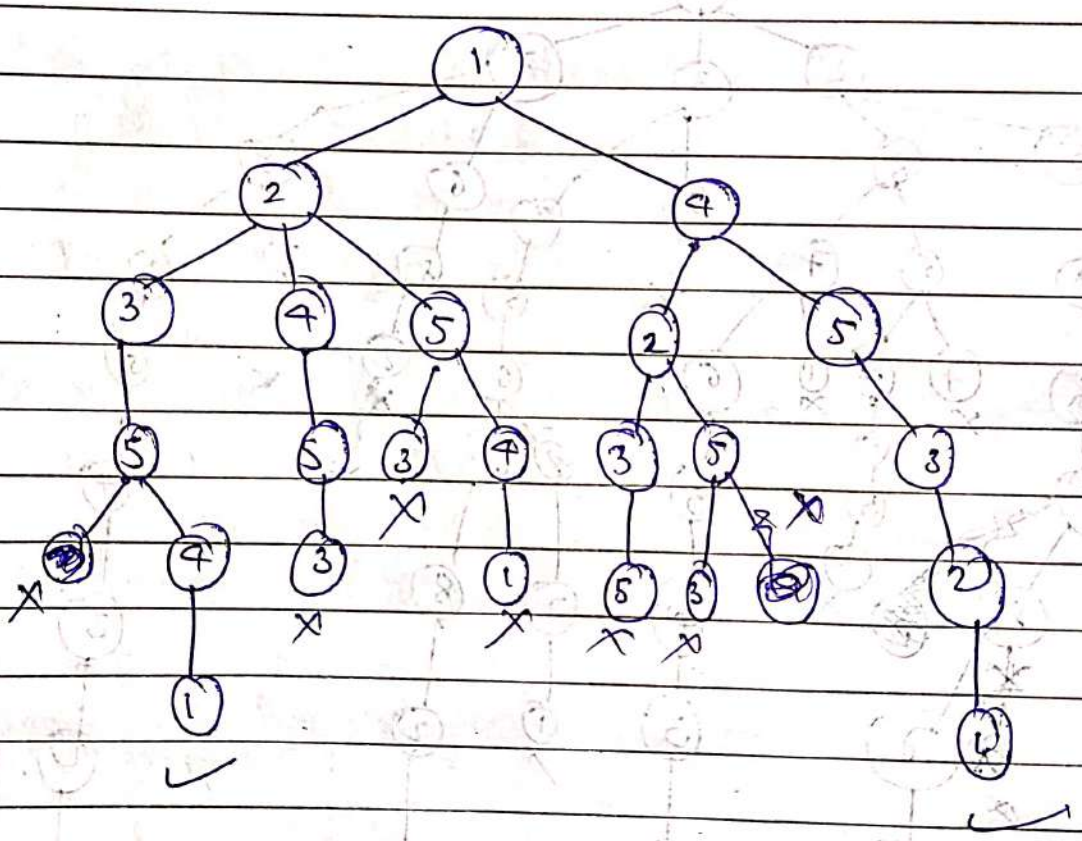
Date: __/__/__

Page No.:

Q5.



State space tree



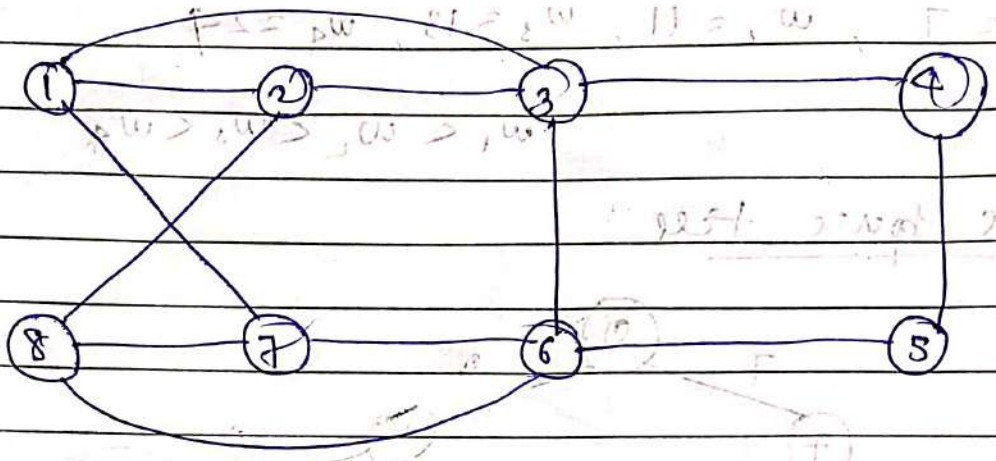
Here we get some Hamiltonian cycle

1 — 2 — 3 — 5 — 4 — 1 ✓

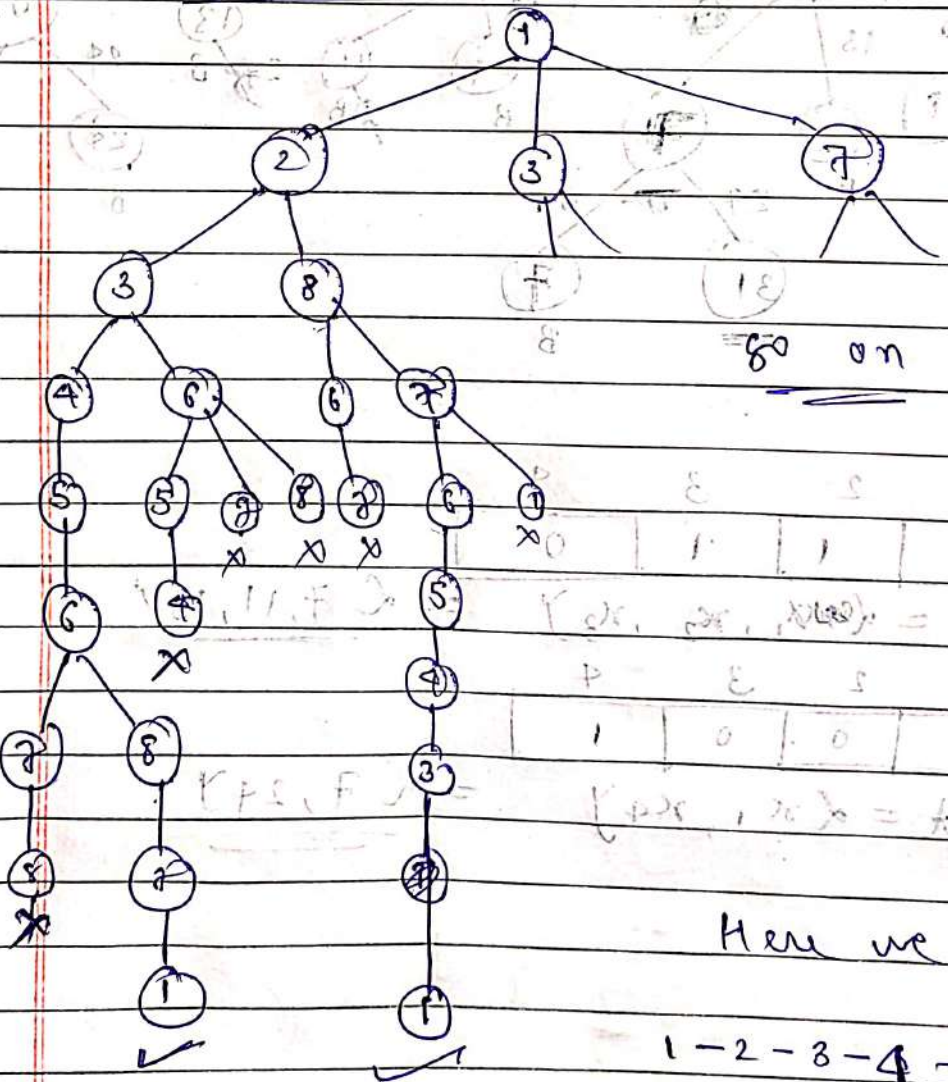
1 — 4 — 5 — 3 — 2 — 1 ✓

Date: __/__/__ Page No.:

Q6) Hamiltonian



State space tree



so on

Here we got some paths:

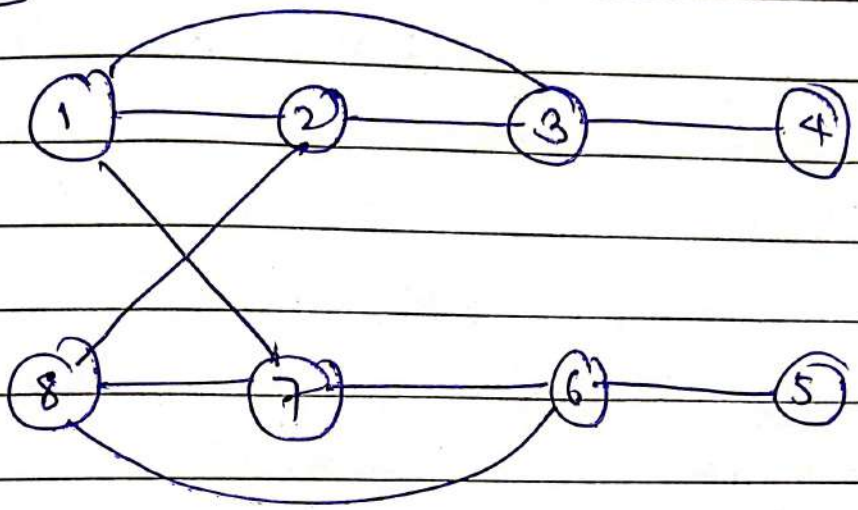
1 - 2 - 3 - 4 - 5 - 6 - 8 - 7 - 1

1 - 2 - 8 - 7 - 6 - 5 - 4 - 3 - 1

Date: __/__/__

Page No. :

Q7



ye nahi hogy hamiltonian cycle

because woo connected nahi har.

Date: __/__/__

Page No. : ____

Q8. What is 8-queen problem? Explain implicit & explicit constraints associated with this problem. Give at least two soln for problem.

⇒ Backtracking approach for 8-queen problem
- it is a recursive approach for solving any problem where we must search among all possible soln's following some constraints.

Given an 8-queen problem, we must place 8 queens on the 8×8 chessboard such that none of them attack one another.

Here no two are in same row, column, or diagonal.

More generally, n queens problem places n queens on nxn chessboard.

Here, we suppose that the queen "i" is to be placed in first row, only but we can say that 1 queen is placed in first row only but can have columns from 1, 2, ---n so that they satisfy all explicit & implicit constraints.

All soln to 8-queen problem are represented as 8-tuples $(x_1, x_2, x_3, \ldots, x_8)$ where x_i is the column on which queen "i" is to be placed.

Date: __/__/__ Page No.: ___

Explicit constraints:

The explicit constraints using this formula are $S_i = \{1, 2, 3, \ldots, n\}$ $1, 8$

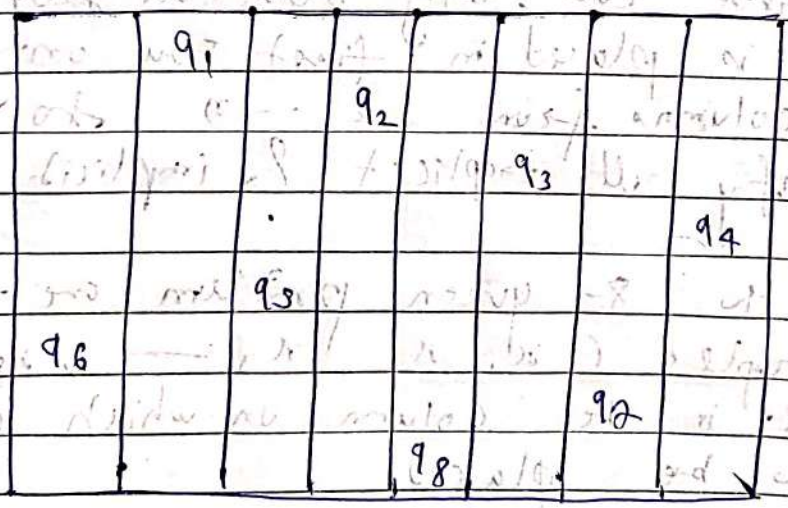
where $1 \leq I \leq 8$

Therefore, solution space consists of 8×8 8 tuples.

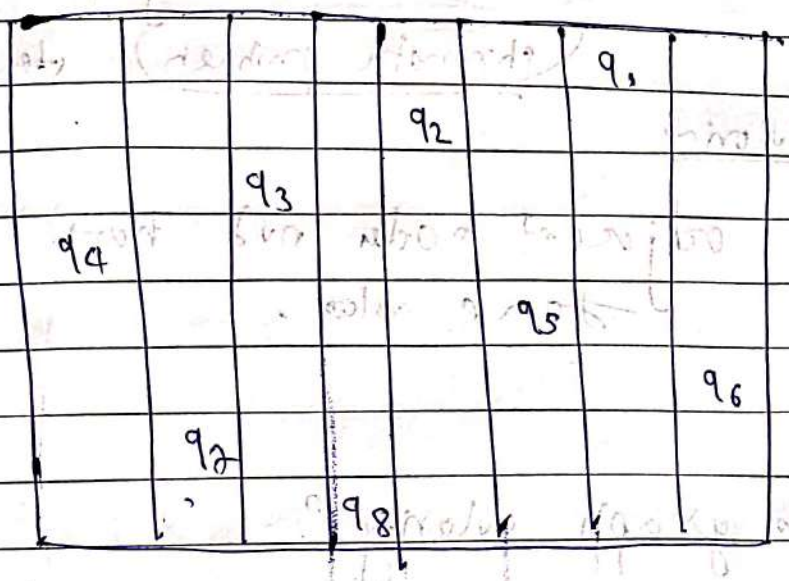
Implicit constraints:

No two x_i's can be same i.e. two queens cannot be in same row, same or column or in same diagonal so each x_i should be different.

8 queen problem soln



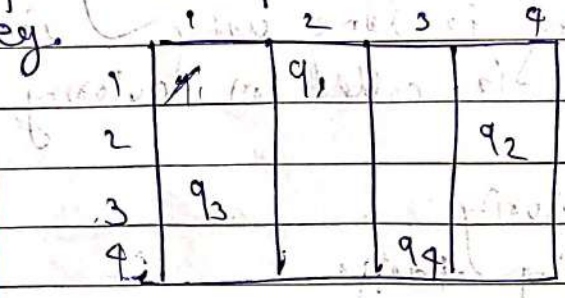
Date: __/__/__ Page No. : ____



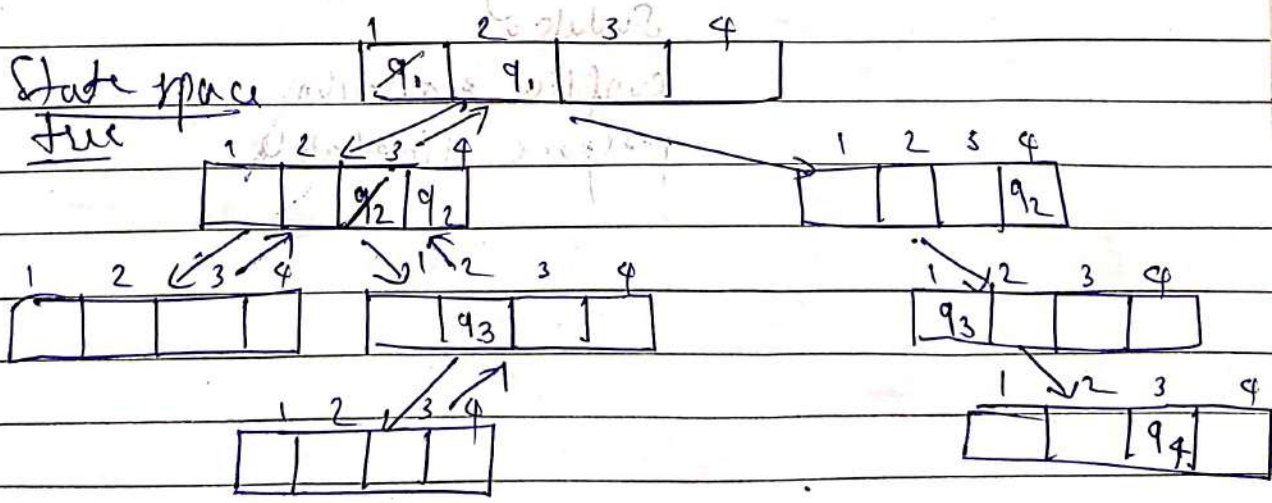
q9 Similar as Q8
Only 8-queen ke place par n-queen.

for 4-queen

eg.



q1 State space tree

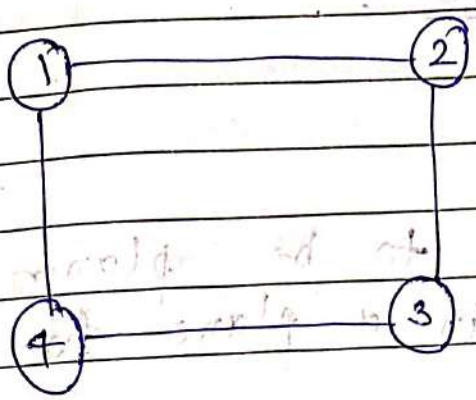


Q10) What is graph coloring?

Graph coloring refers to the problem of coloring vertices of a graph in such a way that no two adjacent vertices have same color.

This is also called the vertex coloring problem. If coloring is done using atmost m colors, it is called m coloring

Application — map coloring
scheduling tasks
Sudoku
conflict resolution
prepare timetable

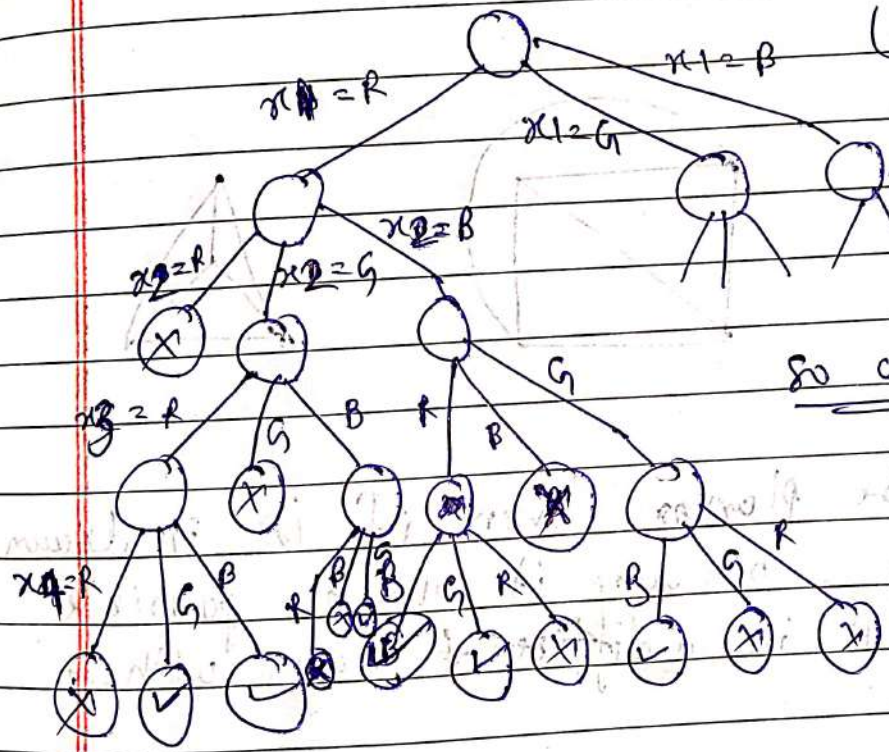


Let
Color = { R, G, B }

⇒ Sol^n

(i) R G R G
(ii) R G R B
(iii) R G B G
(iv) R B R G
(v) R B R B
(vi) R B G B
:

State space tree



So on. —

Q11) isme kitne color or konse diye hee nahi ?

Date: __/__/__ Page No.:

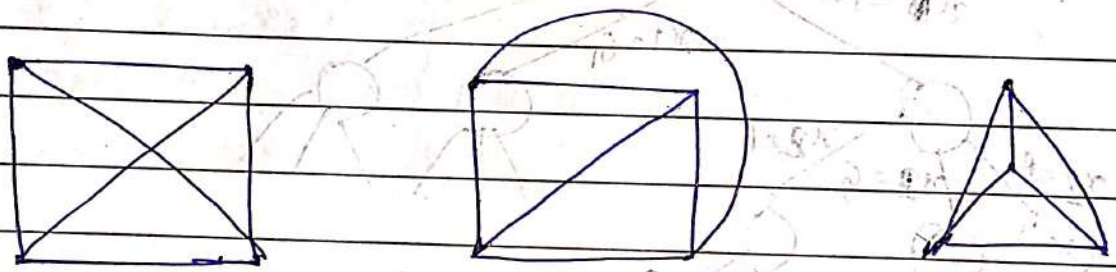
(Q12) Similar as que 10.

Planar Graph

A graph is said to be planar if it can be drawn in a plane so that no edge cross.

(or) It can be drawn on a plane without any edges crossing.

eg. K4



A graph may be planar even if it is drawn with crossing, because it may be possible to draw it in a different way without crossings.

Date: __/__/__ Page No.: ____

q16) Apply backtracking algo. Solve instance of sum of subset problem. $S = \{1, 3, 4, 5\}$ & $m = 10$. find all possible subset of S that generates sum = m & draw solution space tree for same.

Sol^n ⇒

This problem will be solved using space state tree with help of backtracking technique.

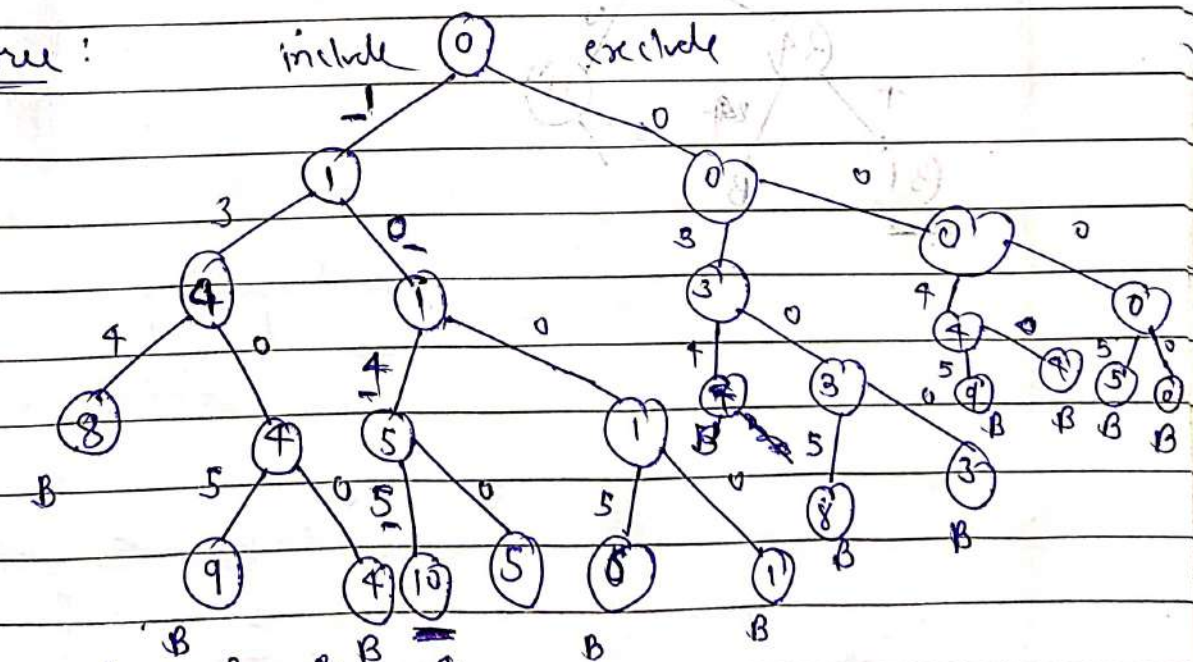
It involves determining whether or not a subset from list of integers can sum to a target value.

Bounding fun^c: $\sum_{i=1}^{k} w_i x_i + w_{k+1} \leq m$

$$\sum_{i=1}^{k} w_i x_i + \sum_{i=k+1}^{n} w_i > m$$

Let, $w_1 = 1$, $w_2 = 3$, $w_3 = 4$, $w_4 = 5$, $m = 10$

State Space tree:



	1	2	3	4
x	1	0	1	1

subset = $\{x_1, x_3, x_4\}$ = $\{1, 4, 5\}$

Date: __/__/__ Page No.: ____

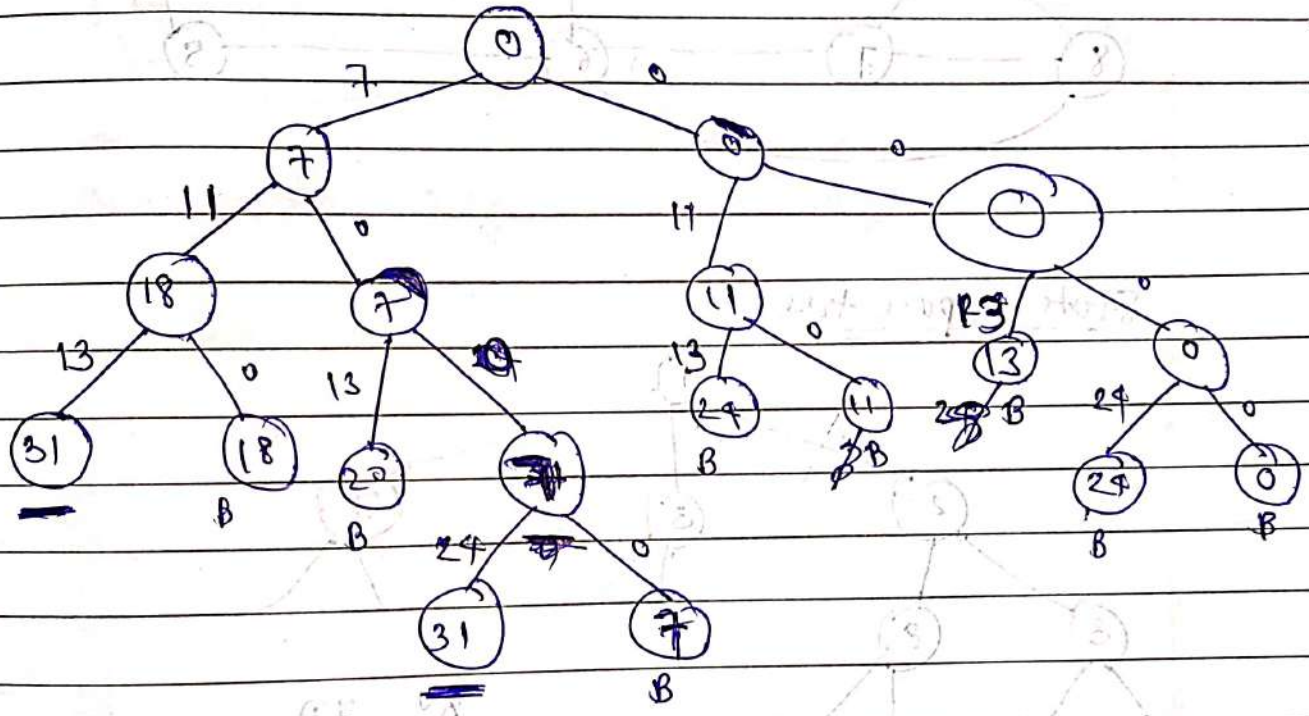
15 $w = \{11, 13, 24, 7\}$

$m = 31$

$w_1 = 7$, $w_2 = 11$, $w_3 = 13$, $w_4 = 24$

$w_1 < w_2 < w_3 < w_4$

State space tree :



	1	2	3	4
X	1	1	1	0

subset = $\{x_1, x_2, x_3\}$ = $\{7, 11, 13\}$

	1	2	3	4
X	1	0	0	1

subset = $\{x_1, x_4\}$ = $\{7, 24\}$