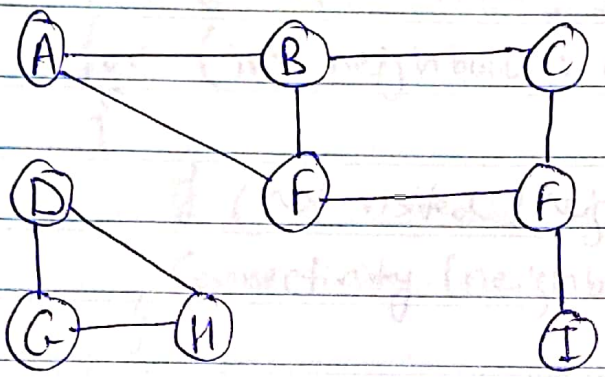
DAA ASSIGNMENT-4

Submitted by -
BHUVAN CHADHA
B00815783

1. We are given an undirected graph



The connectivity of a graph can be determined by Depth first search (DFS) technique.
In DFS, the graph is traversed by starting from any node, then visiting any neighbouring node, exploring that node first and then come back to the starting node to explore other neighbouring nodes one by one.
If after the whole graph is traversed, there are any nodes remaining and not reachable, then the graph is not connected.

In the given graph, DGH subgraph cannot be traversed from IFEABC subgraph as there is no connecting node.

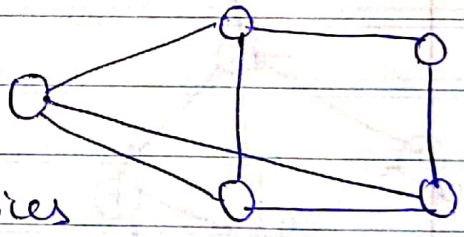
The algorithm for checking the Connectivity of a graph can be written as—

```
Algorithm Graph (node)
{
    visit [node] = true;
    for (int neighbour : adj [node])
    {
        if (not visited [neighbour])
        Connectivity [neighbour];
    }
}

Boolean check_conn()
{
    Connectivity(0);
    flag = true;
    for (int i=0; i<count; i++)
        if (not visited [i])
        {
            flag = false;
            break;
        }
    return flag
}
```

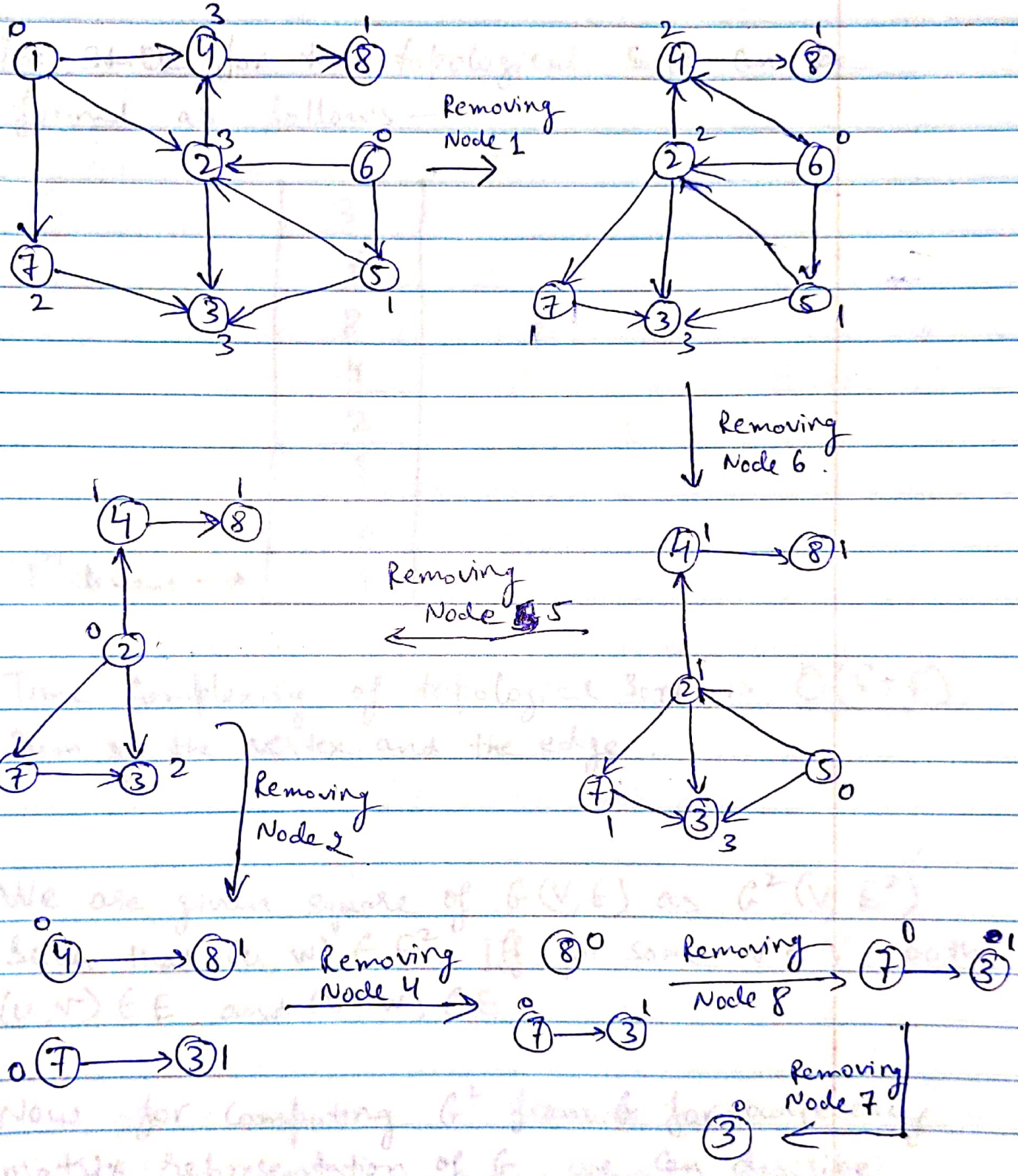
for an undirected graph to be a tree, there should be no cycles and the graph should be connected.
In the given example, the graph is non-connected Hence, it is not a tree. Nodes D & H Cannot be visited from I, E, A, B, C.

2. We are given an undirected graph with 5 vertices, each of degree 3.
A degree of a vertex is the number of edges associated with it. Now, let E be an edge from a vertex, then the sum of the edges be 2E. Also, Sum of the degrees of all the vertices is $3 \times 5 = 15$.



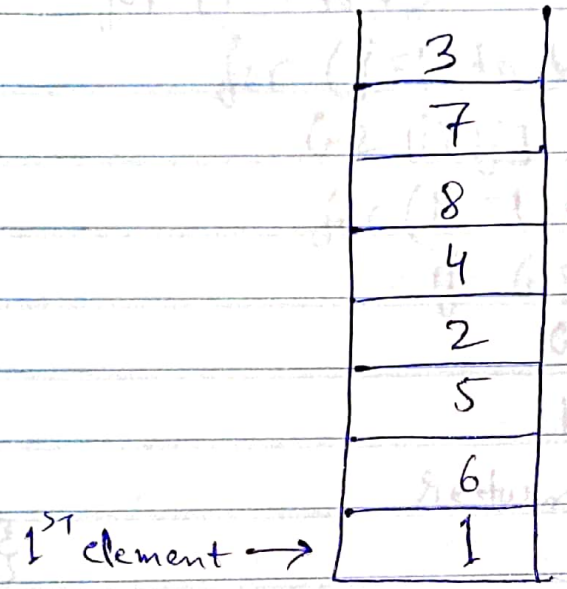
So, from this, number of edges $E = 15/2$, which is a decimal value and is not possible.
Hence, the undirected graph with 5 vertices each of degree 3 Cannot exist.

4. Topological Sorting on the given graph is as follows -
We start from the node ①.
Degrees of each node will be updated in every step and mentioned along with the node.



Hence, the sequence is stored in a Stack.
The order in which nodes were removed/deleted is → 1, 6, 5, 2, 4, 8, 7, 3.

The stack for the topological sort can be formed as follows –



Time complexity of topological sort is $O(V+E)$, sum of the vertex and the edge.

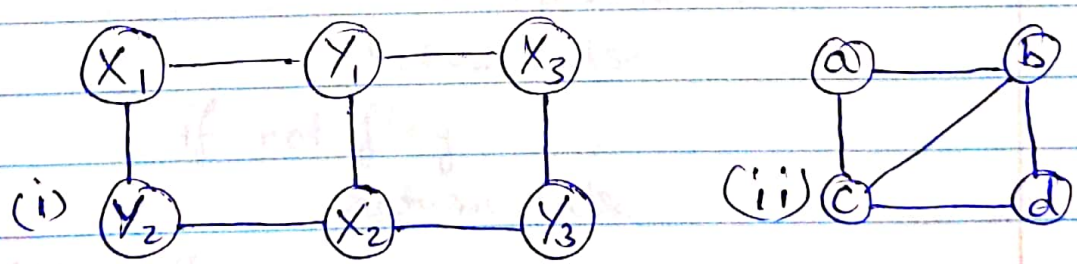
5. We are given square of $G(V, E)$ as $G^2(V, E^2)$ such that $(u, w) \in E^2$, iff for some $v \in V$, both $(u, v) \in E$ and $(v, w) \in E$.

Now, for computing G^2 from G for adjecency matrix representation of G, we can describe an algorithm using three for loops for matrix multiplication.

```
Matrix G-square (V)
{
    for (i=1 to V)
        for (j=1 to V)
            G2 [i] [j] = 0
            for (k=1 to V)
                if G [i] [k] = =1 and G [k] [j] ==1
                    G2 [i] [j] ==1
                    break
                return G2
}
```

Time complexity of this algorithm is $O(V^3)$

6. We are given two following graphs.



(a) As per the definition of Bipartite graph, (i) is a bipartite graph as it can be coloured with 2 colours with no adjecent node having same colour. Graph (ii) is not a bipartite graph as it it will always have two connecting nodes having same colour.

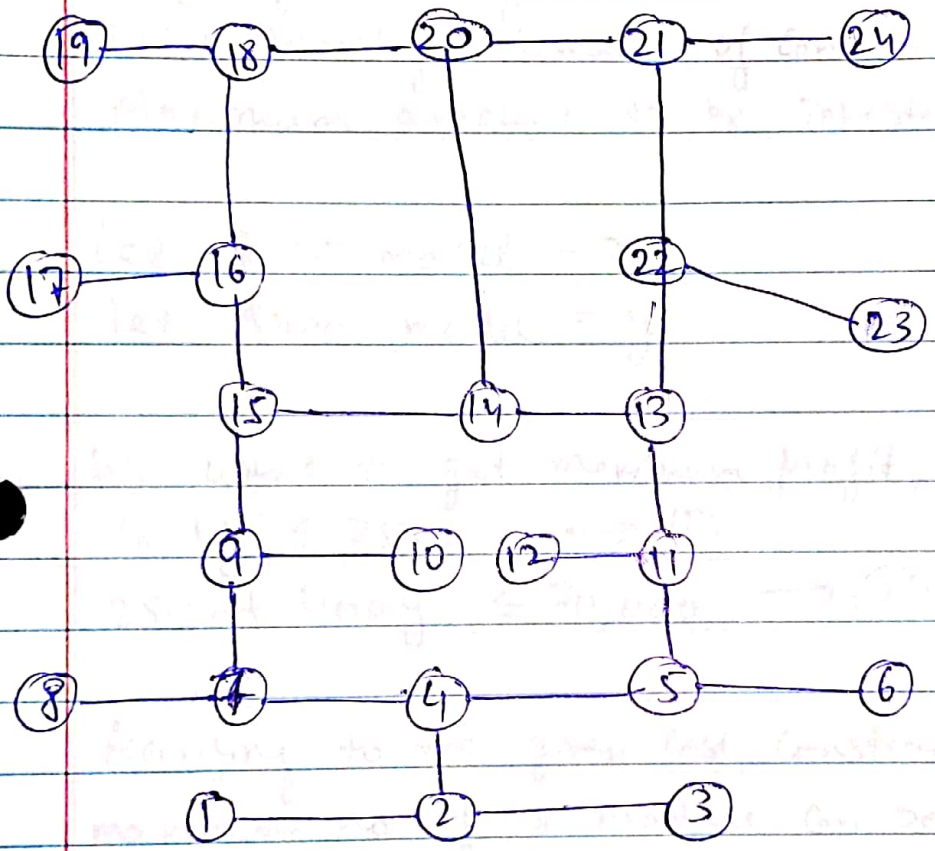
(b) DFS algorithm for checking whether a graph is bipartite or not —

```
Bipartite DFS ( G, colour, node, node_col)
{
    if (colour [node] is not -1 and colour [node] is
        not equal to node_col)
        return false;
    colour [node] = node_col
    flag = True.
    for (i = 0 to V)
            if G[node] [i]:
                if (colour [i] == -1)
                    flag = Bipartite_DFS (G, colour, i, 1-node_col)
                if (colour[i] is not equal to -1 and colour[i]
                    is not equal to 1 - node_col)
                    return false
            if not flag
                return false
    return True
}
```

(C) BFS algorithm for checking whether a graph is bipartite or note is as follows —

```
Bipartite BFS (G, colour[], visit[], V)
{
    flag 0
    Colour [0 to n-1] = 0
    visit [0 to n-1] = 0
    for (i=0 to V)
        if visit[V] = 0
            visit[V] = 1
            Colour[V] = 1    // (1st colour to a node)
        Enquene [V]
            while (quene is not empty)
            do {
                for (each vertex in V) do
                {
                    if visit [vertex] = 0
                        visit [vertex] = 1
                        Colour [vertex] = 2  // (2nd colour to node)
                        Enquene [vertex]
                    if (visit[vertex] == 1 and colour[vertex] ==
                                                    colour[V])
                        flag = -1
                }
            }
        dequene [V]
}
```

7. We are given a maze, constructing a graph with node numbering starting from the entrance point and ending at the exit point of the maze.



Nodes can be arranged in any order. Any node can be explored first.

(b) Depth First Search (DFS) is more efficient for passing through the maze because there are more possibilities of finding a path if we ~~traverse~~ explore each junction (node) than just ~~a~~ visiting each nearby neighbouring node once and then returning back to the previous junction.

When DFS traverses the next vertex (node), it is connected to the current node by an edge, whereas it is not the case in BFS.

8. We are given costs of two models of computers - $250 and $400
Profit yeilded by $250 model = $45
Profit yeilded by $400 model = $50
Total monthly demand of computers = 250 units
Maximum amount to be invested = $70,000

Let $250 model = x
Let $400 model = y

We want to get maximum profit, $P = 45x + 50y$.
$x + y \leq 250 \rightarrow (1)$
$250x + 400y \leq 70,000 \rightarrow (2)$

According to the given cost constraint in eqn (2), maximum no. of x models can be 250 and maximum no. of y models can be 175.

Hence, we have different values of x and y which can be substituted to get maximum Profit, P.

(1) $(x=0, y=0)$ - $P = 45(0) + 50(0) \Rightarrow 0$
(2) $(x=0, y=175)$ - $P = 45(0) + 50(175) \Rightarrow 8750$
(3) $(x=250, y=0)$ - $P = 45(250) + 50(0) \Rightarrow 11,250$
(4) $(x=200, y=50)$ - $P = 45(200) + 50(50) \Rightarrow 11,500$
Hence, $250 model = 200 and $400 model = 50 to get max P.

9. Maximize $P = 20x_1 + 10x_2 + 15x_3$ → (1)

Subject to:

$$3x_1 + 2x_2 + 5x_3 \leq 55$$
$$2x_1 + x_2 + x_3 \leq 26$$
$$x_1 + x_2 + 3x_3 \leq 30$$
$$5x_1 + 2x_2 + 4x_3 \leq 57$$
$$x_1, x_2, x_3 \geq 0.$$

Since there are 4 constraint equations, we introduce 4 slack variables and re-writing the equations -

$$3x_1 + 2x_2 + 5x_3 + S1 = 55$$
$$2x_1 + x_2 + x_3 + S2 = 26$$
$$x_1 + x_2 + 3x_3 + S3 = 30$$
$$5x_1 + 2x_2 + 4x_3 + S4 = 57$$

Also, from eq^n (1), we can adjust as -

$$-20x_1 - 10x_2 - 15x_3 + P = 0.$$

Now, forming the initial iterative table from the above equations.

x_1	x_2	x_3	S1	S2	S3	S4	P	Constant
3	2	5	1	0	0	0	0	55
2	1	1	0	1	0	0	0	26
1	1	3	0	0	1	0	0	30
5	2	4	0	0	0	1	0	57
-20	-10	-15	0	0	0	0	1	0

Now, from the table, the most negative indicator in the last row is -20. So, x_1 is the pivot column

1st ratio = $55/3 \Rightarrow 18.33$
2nd ratio = $26/2 \Rightarrow 13$
3rd ratio = $30/1 \Rightarrow 30$
4th ratio = $57/5 \Rightarrow 11.4$

Lowest ratio is $57/5$. So, 5 is the pivot element. We will make 5 equal to 1 and all other elements equal to 0 in the first column.

Using row operation,

$1/5\, R_4 \longrightarrow R_4$
$R_3 - R_4 \longrightarrow R_3$
$R_2 - 2R_4 \longrightarrow R_2$
$R_1 - 3R_4 \longrightarrow R_1$
$R_5 + 20R_4 \longrightarrow R_5$

Updating the iterative table using above operations

x_1	x_2	x_3	S1	S2	S3	S4	P	Constant
0	4/5	13/5	1	0	0	-3/5	0	104/5
0	1/5	-3/5	0	1	0	-2/5	0	16/5
0	3/5	4/5	0	0	1	-1/5	0	93/5
1	2/5	4/5	0	0	0	1/5	20	57/5
0	-2	1	0	0	0	4		228

We still have negative entries in the last row.
So, selecting -2 from x_2 column

1st ratio $= \frac{104}{5} \div \frac{4}{5} \Rightarrow 26$

2nd ratio $= \frac{16}{5} \div \frac{1}{5} \Rightarrow 16$

3rd ratio $= \frac{93}{5} \div \frac{3}{5} \Rightarrow 31$

Fourth ratio $= \frac{57}{5} \div \frac{2}{5} \Rightarrow 28.5$

Lowest ratio is 16, So, 1/5 is the pivot element.
We will make 1/5 equal to 1 and rest all other elements of x_2 column equal to 0.

Again, using row operations,

$5R_2 \rightarrow R_2$

$4/5\,R_2 - R_1 \rightarrow R_1$

$3/5\,R_2 - R_3 \rightarrow R_3$

$2/5\,R_2 - R_4 \rightarrow R_4$

$2R_2 + 5R_5$

Updating the table using above row operations,

x_1	x_2	x_3	S1	S2	S3	S4	P	Constant
0	0	5	1	-4	0	1	0	8
0	0	-3	0	+5	0	-2	10	16
0	0	4	0	-3	0	1	0	9
1	1	2	0	-2	1	1	20	5
0	0	-5	0	10	0	0		260

We still have a negative integer in the last row, So, selecting -5 from x_3 column

1ST ratio $\Rightarrow$ $8/5 = 1.6$

2nd ratio $\Rightarrow$ $-16/3 = -$

3rd ratio $\Rightarrow$ $9/4 = 2.25$

4th ratio $\Rightarrow$ $5/2 \Rightarrow 2.5$

Lowest ratio is $8/5$. So, 5 is the pivot element. We need to make 5 equal to 1 and all other elements in x_3 column equal to 0.

Using row operations –

$R_1/5 \rightarrow R_1$

$R_2 + 3R_1 \rightarrow R_2$

$R_3 - 4R_1 \rightarrow R_3$

$R_4 - 2R_1 \rightarrow R_4$

Updating the table using above operations –

x_1	x_2	x_3	S1	S2	S3	S4	P	Constant
0	0	1	1/5	-4/5	0	1/5	15	8/5
0	1	0	3/5	13/5	0	-7/5	10	104/5
0	0	0	-4/5	1/5	1	1/5	0	13/5
1	0	0	-2/5	-2/5	0	3/5	20	9/5
0	0	0	1	6	0	1		268

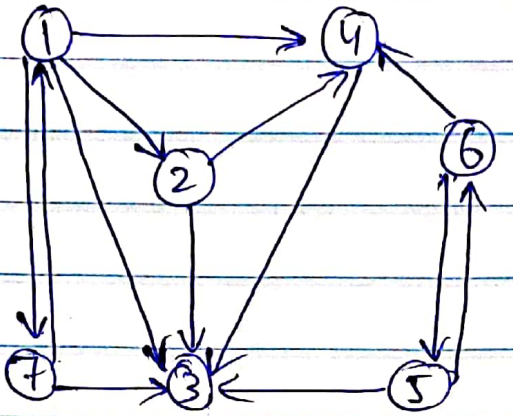
Now, last row has no negative values. Hence, we have reach to the optimal solution.
S1, S2, S4 are not unit columns, which makes them non-basic, i.e., S1, S2, S4 = 0.

$\therefore x_1 = 9/5$, $x_2 = 104/5$, $x_3 = 8/5$

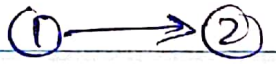
Using these value, we can maximize our eqn ① .

$P = 268$

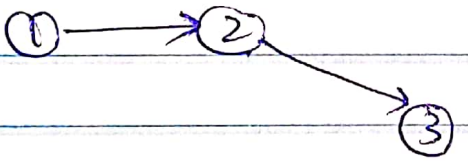
3. We are given the following graph.



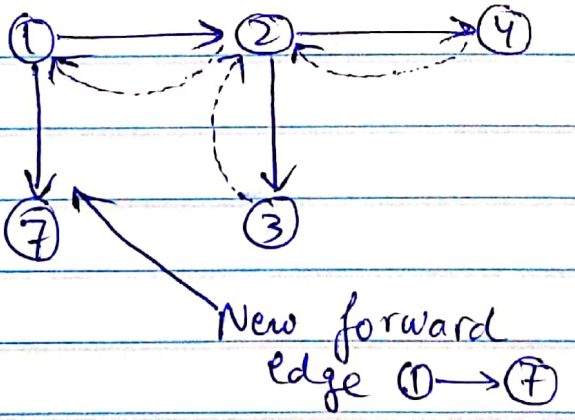
Applying Depth First Search,



→ New Edge.



→ New forward Edge.



→ Back edge is denoted by dotted lines.

Hence,
The final DFS sequence is — 1, 2, 3, 4, 7
Nodes 5 and 6 cannot be visited.