Chapter-5 (10)
Graph Traversal and Search Technique

1) Define graphs, complete graphs, bi-partite graphs, planar graphs and trees.

i) Graphs

→ A graph can be defined as group of vertices and edges that are used to connect these vertices.

→ Also, a graph G can be defined as $G = \{V, E\}$ where V = set of vertices and E = set of edges.

Types ① Directed graph
② Undirected graph

ii) Complete Graph

→ The graph in which every vertex is connected with all other nodes.

iii) Bi-partite Graph

→ A graph is said to be bi-partite graph if its vertices are divided into two parts such that the vertices of first part are connected to the vertices of second part but the vertices of same part are never connected.

For example:

let $P_1 = \{V_1, V_2, V_3\}$

$P_2 = \{V_4, V_5\}$

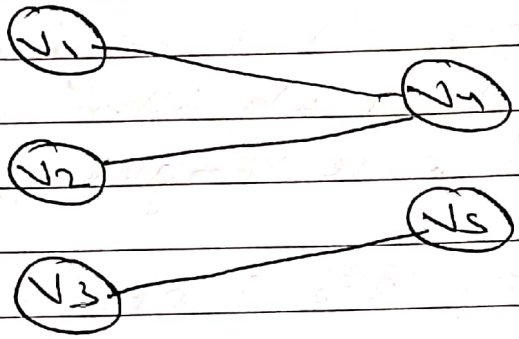
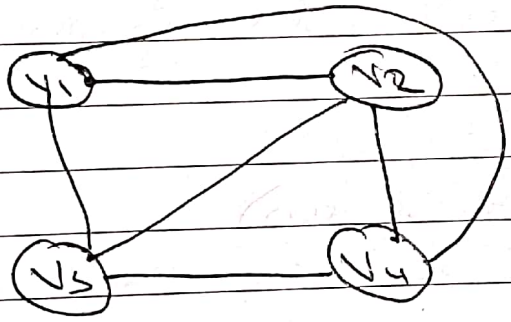


Fig:- Bipartite graph

iv) Planar graph

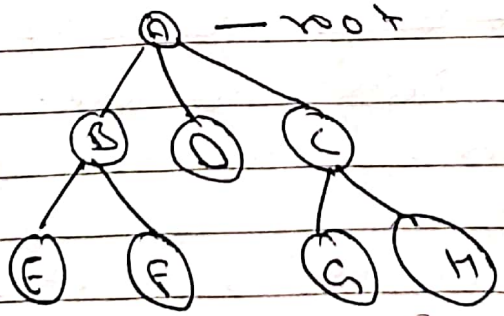
→ A graph is said to be planar if it can be drawn in a plane so that no edge cross.



v) Tree

→ A tree is a non-linear and a hierarchical data structure consisting of a collection of nodes such that each node of the tree stores a value.

Eg.



IIII 2) Explain the graph traversal technique of BFS and DFS with the help of algorithm Analyze their time and space complexity.

or

(write an algorithm for BFS & DFS and discuss their complexity)

Ans: Traversal means visiting all the nodes of a graph.

→ BFS and DFS are recursive algorithm for searching all the vertices of a graph or tree data structure.

→ BFS (Breadth first search)

→ BFS is a graph traversal algorithm that starts traversing the graph from root node and explores all the neighbouring nodes. Then it selects the nearest node and explores all the unexplored nodes.

→ In BFS, the neighbour nodes are traversed first before the child nodes.

→ ~~It uses queue to keep track of the visited nodes.~~

→ It uses queue data structure.

→ Select next vertex for exploration from queue only.

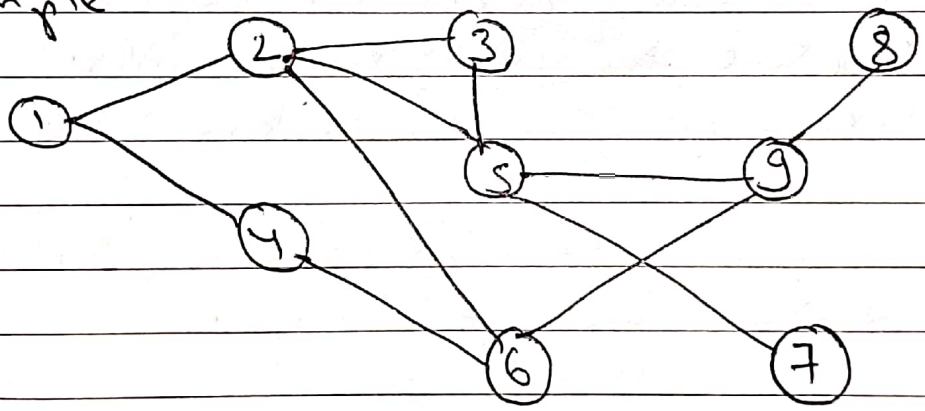
→ Time complexity of BFS is O(V+E) where V is vertices and E is edges.

→ it is like a level order in a binary tree.

Algorithm:

1) Select the root of the graph
2) Push the root vertex into the queue.
3) Pop a vertex from the queue, mark it as visited, and output its value.
4) Visit its unvisited neighbour vertex, push them to the queue, and mark visited.
5) Repeat step 3 and 4 until the queue is empty.
6) When the queue is empty, end the program.

Example



Queue: dequeue

X	Z	Y	X	X	X	X	X	X

Output: 1 2 4 3 5 6 9 7 8

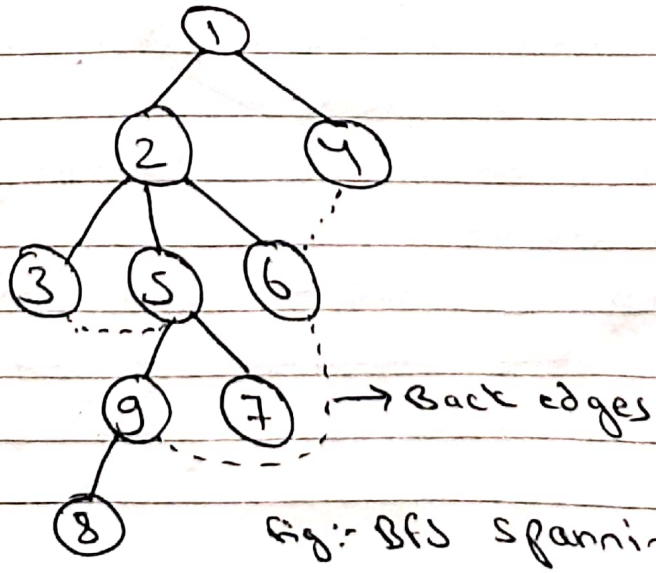


Fig:- BFS spanning Tree

i) DFS (Depth first Search)

→ DFS algorithm starts with the initial node of the graph G, and then goes to deeper and deeper until we find the node which has no children. The algorithm then backtracks from the dead end towards the most recent node that is yet to be explored.

→ The data structure which is being used in DFS is stack.

→ In DFS, the child nodes are explored/traversed first before the neighbour nodes.

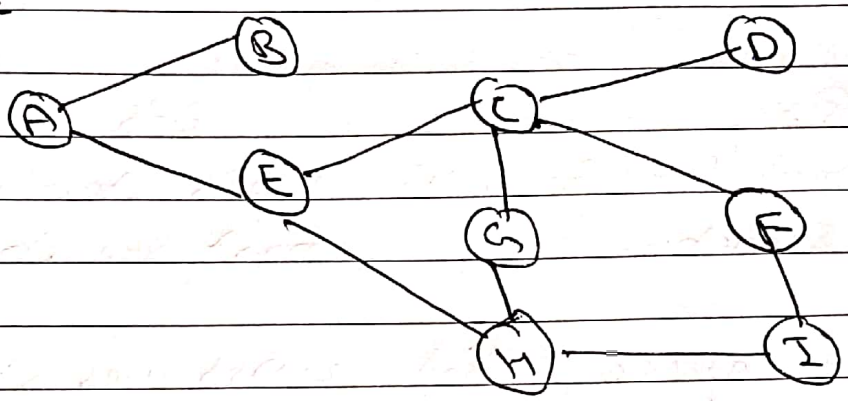
→ It is like a pre-order traversal in tree.

→ Time complexity of DFS is also $O(V+E)$ where V is vertices and E is edges.

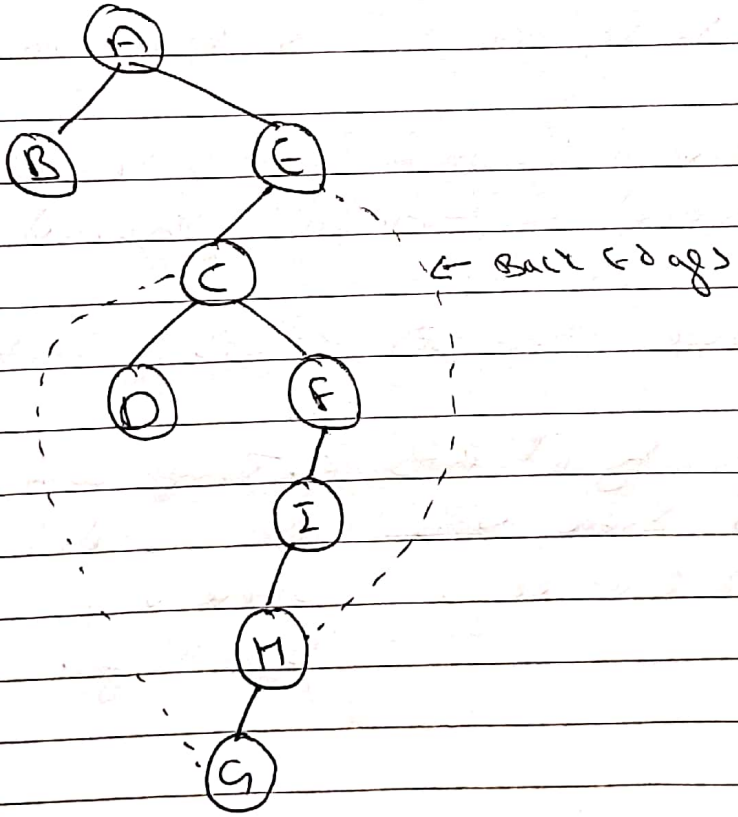
Algorithm:

1) Select the root of the graph
2) Insert root vertex into the stack.
3) Visit its child vertex, mark it as visited. Display it. Push it in a stack.
4) If no adjacent vertex is found, popup a vertex from the stack.
5) Repeat step 2, 3 and 4 until the stack is empty.
6) When the stack is empty, end the program.

Example



Output: A B E C D F I H G



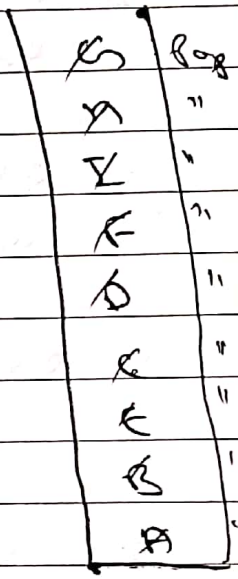


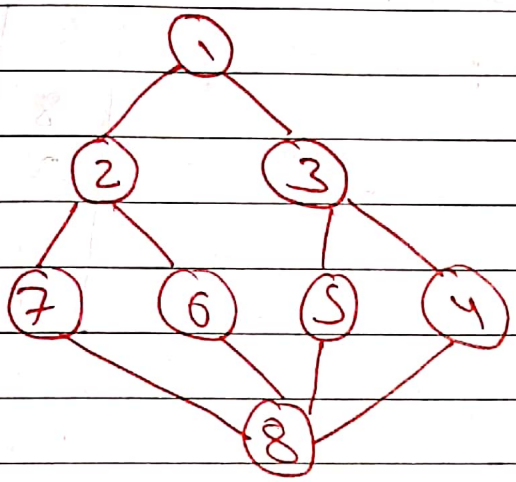
Fig: DFS Spanning Tree

DFS vs BFS

	BFS	DFS
1.	BFS stands for Breadth first search.	1. DFS stands for Depth first search.
2.	BFS uses queue to find the shortest path.	2. DFS uses stack to find the shortest path.
3.	BFS is better when target is closer to source.	3. DFS is better when target is far from source.
4.	BFS traverses a graph in a breadthward motion.	4. DFS traverses graph in a depthward motion.
5.	BFS is slower than DFS	5. DFS is faster than BFS.
6.	Time complexity of BFS = O(V+E) where V is vertices and E is edges.	6. Time complexity of DFS is also O(V+E) where V is vertices and E is edges.

3) Generate Spanning tree for graph below using ~~both~~ BFS and DFS

i)

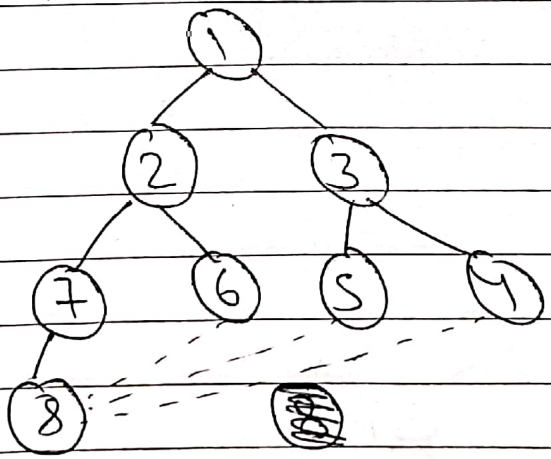


BFS: Dequeue

1	2	3	7	6	5	4	8

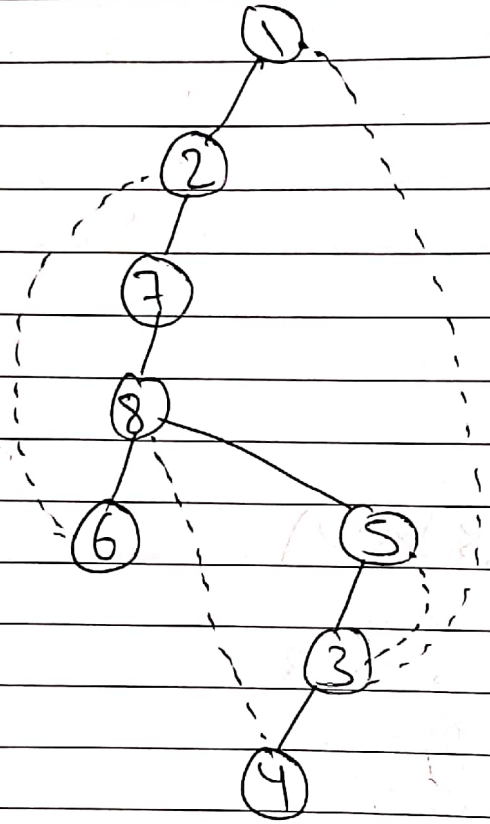
Queue:

Output: 1 2 3 7 6 5 4 8



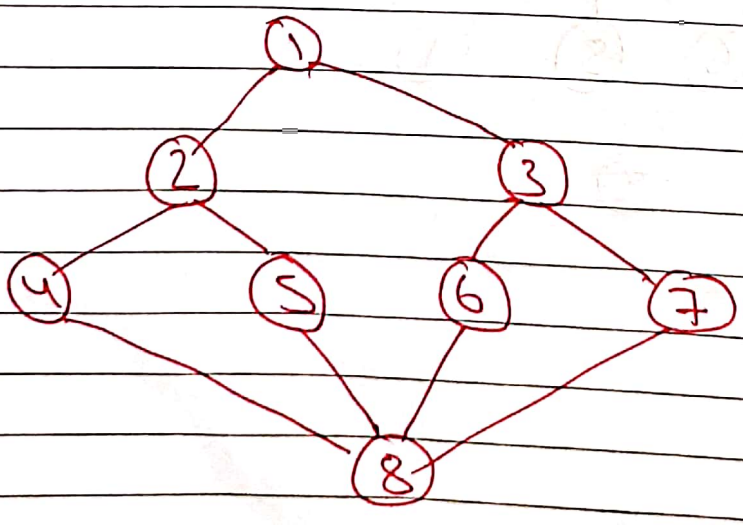
DFS:

Output: 1 2 7 8 6 5 3 4



1	Pop
3	Pop
5	Pop
6	Pop
8	Pop
7	Pop
2	Pop
1	Pop

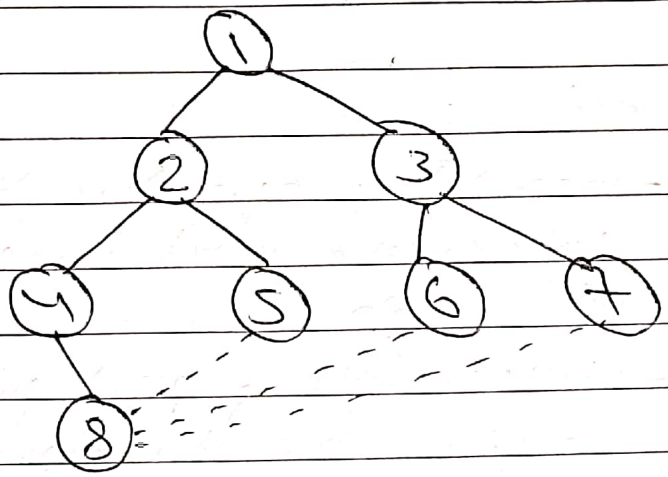
ii)



BFS Deque

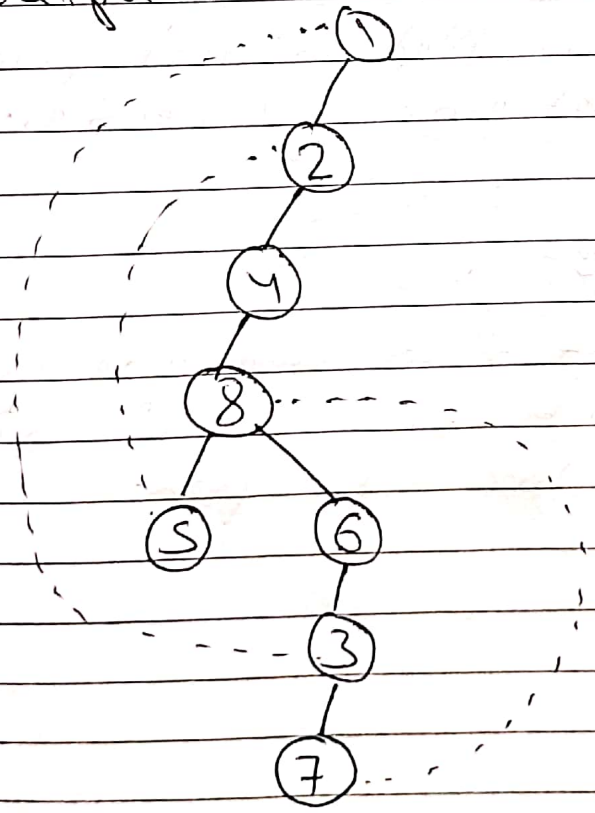
Queue: 1 2 3 4 5 6 7 8

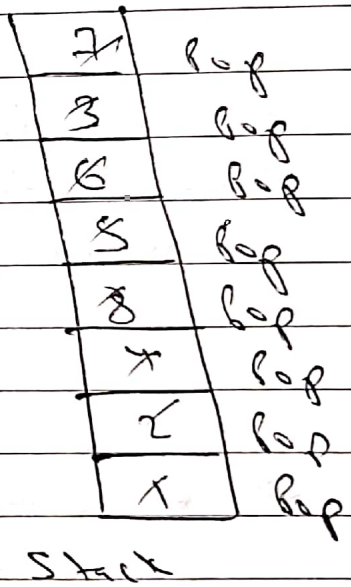
Output: 1 2 3 4 5 6 7 8



DFS

Output: 1 2 4 8 5 6 3 7





Stack

Extra information

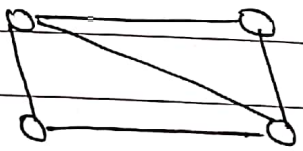
→ Applications of DFS

i) undirected graph
- connected components, articulation point, bridges, biconnected components

ii) directed graph
- cyclic/acyclic graph, topological sort, strongly connected components.

→ connected components

→ An undirected graph is called connected graph if there is a path between any 2 vertices.



→ A connected component of an undirected graph is a sub-graph in which each pair of nodes is connected with each other via a path.

11 3) Define articulation point. Write the algorithm for finding articulation point in the graph with suitable example.

Ans: Articulation point is any vertex of a graph whose removal results in disconnected graph.

-> In road networks or computer networks we donot want any ~~partition~~ articulation point because failure of that articulation point will break that network into multiple components.

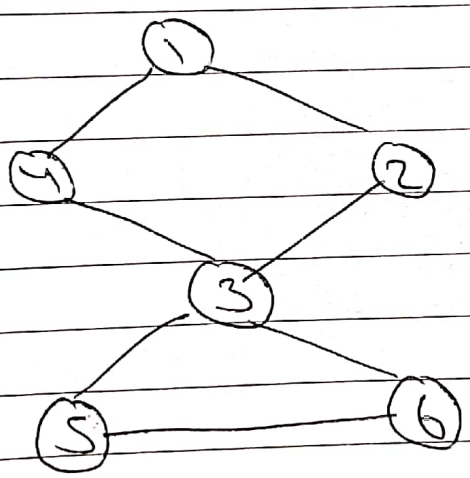


Fig: Graph

Here vertex 3 is an articulation point. It's removal will make this graph a disconnected graph.

-> Connect vertex 4 & 5 to remove the articulation point.

3) Algorithm to find articulation point in a graph

Example

find the articulation point in the given graph

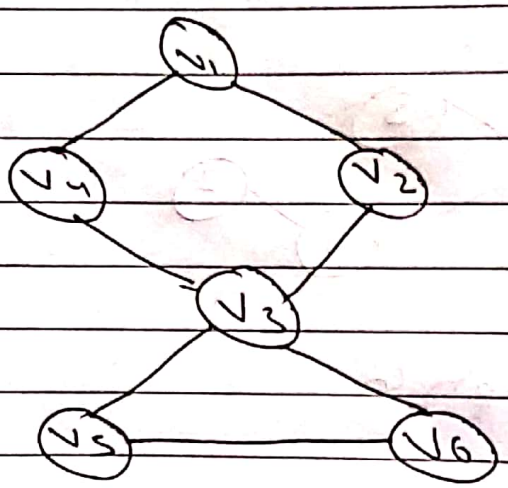
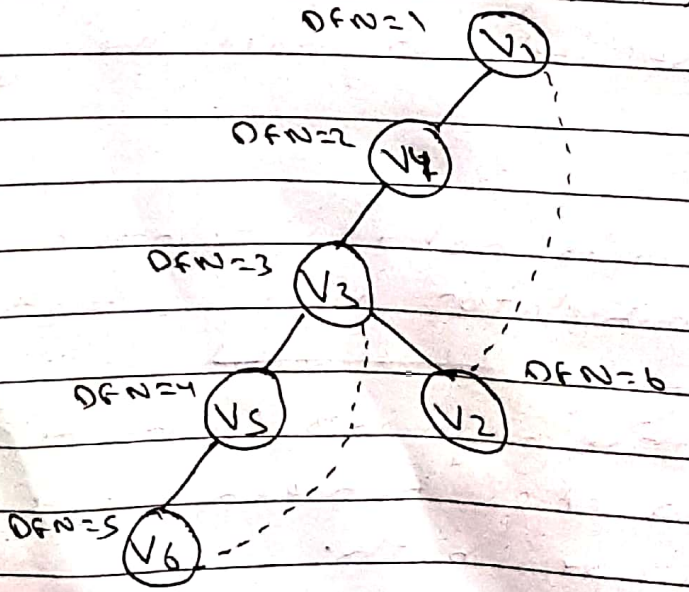


fig graph G

Step1: Perform the DFS of the given graph & compute the Depth first Number (DFN)



Step2: Calculate the lowest Depth first Number (LDFN) of each node in the DFS tree

Vertex	V_1	V_2	V_3	V_4	V_5	V_6
DFN	1	6	3	2	4	5
LDFN	1	1	1	1	3	3

Step3: finding the articulation point.

If u is a parent & v is child

If $LDFN(v) \geq DFN(u)$

then 'u' is a articulation point.

Note: This condition doesnot satisfy for root edge.

checking $u = V_3$ & $v = V_5$

$LDFN(V_5) = 3$

$DFN(V_3) = 3$

$3 \geq 3$ (true)

∴ The condition is true when V_3 is a parent node and V_5 is a child. So V_3 is a articulation point

11.4) Bi-connected components

→ A graph is biconnected if it contains no articulation point.

Construction of bi-connected graph

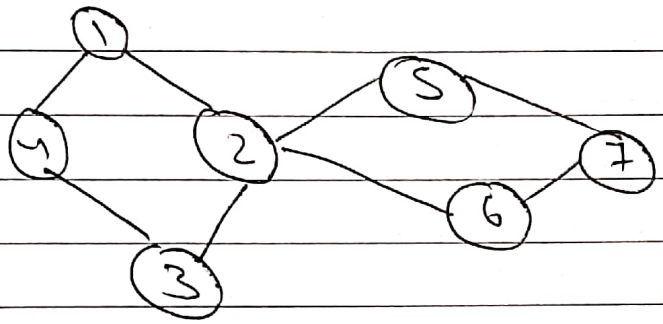


Fig: Graph G

→ Check the given graph whether it is biconnected or not.

→ if the given graph is not biconnected then identify all the articulation points.

→ if the articulation point exists, construct a set of edges whose inclusion makes the graph connected

→ The given graph G is not a bi-connected graph. It includes a articulation point i.e vertex 2

→ ~~To make this graph biconnected~~

→ To transform this graph into biconnected graph, we can add new edges between vertex 1 & 5, or 3 and 6 or both.