Ans 1

BFS	DFS
• It stands for Breadth first search	• It stands for Depth first search
• It uses Queue data Structure	• It uses stack data Str.
• Suitable for searching vertices closer to source.	• Suitable for searching vertices away from source.
• It considers all neighbours first and thus not suitable for decision making trees used in games & puzzles.	• More suitable for game & puzzle problem.
• Siblings are visited before children.	• Childreons are visited before parent.
• no cocept of backtracking	• It is a recursive algorithm that uses backtracking.
• It requires more memory	• It requires less memory.

Applications :-

- BFS → Bipartite graph and shortest path, p2p networking, Crawlers in search engine & GPS Navigation System.
- DFS → acyclic graph, topological order, scheduling problems, sudoko puzzle

Ans 2 :-

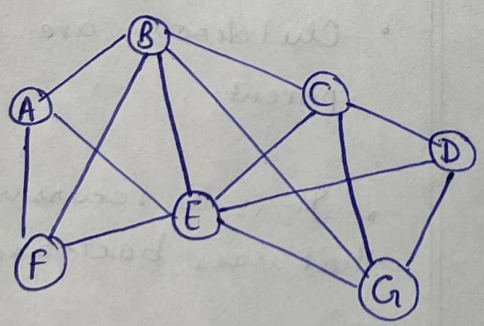
for implementing BFS we need a queue data structure for finding shortest path b/w any node - We use queue because things don't have to be processed immediately. BFS searches for node levelwise. For this queue is better to use in BFS.

for implementing DFS we need a Stack data str. as it traverses in depthward motion.

Ans 3 :-

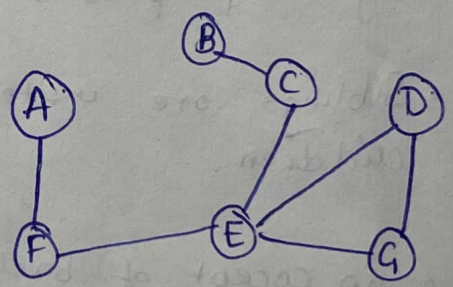
Dense graph is a graph in which no. of edges is close to maximal no. of edges

Sparse graph is a graph in which no. of edges is very less.



Dense graph

(many edges b/w nodes)



Sparse graph

(few edges b/w nodes)

Ans 4:- For detecting a cycle in a graph using BFS :-

Steps Involved are :-

1). Compute in-degree (no. of incoming edges) for each of vertex present in graph

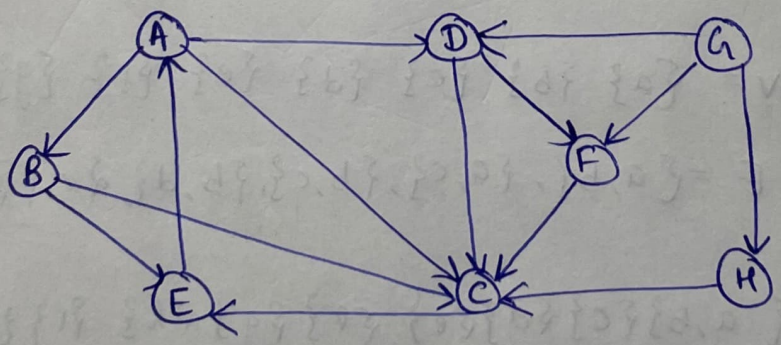
2). Pick all vertices with in degree as 0 and add them in queue.

3). Remove a vertex from queue and then
- inc. count of visited nodes by 1.
- dec. in degree by 1 for all its neighbouring nodes.

4). Repeat 3 until queue is empty

5). If count of visited node is not eq. to no. of nodes in graph, has cycle otherwise not.

Ans :-

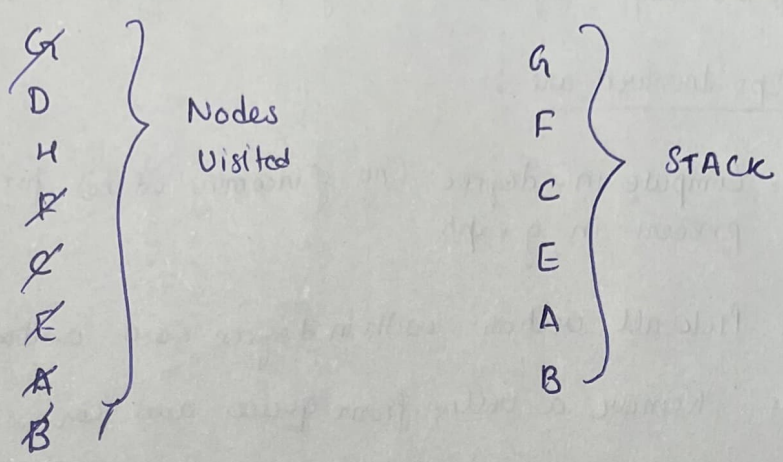


BFS

Child	G	H	D	F	C	E	A	B
Parent		G	G	G	H	C	E	A

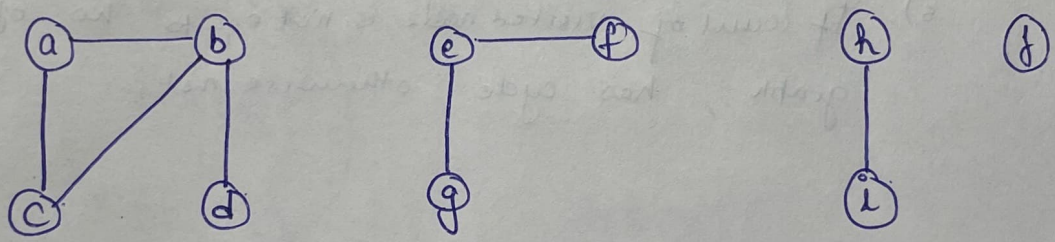
Path :- G → H → C → E → A → B

DFS



Path:- G → F → C → E → A → B

Ans :-



V = {a} {b} {c} {d} {e} {f} {g} {h} {i} {j}

E = {a,b}, {a,c}, {b,c}, {b,d}, {e,f}, {e,g}, {h,i}, {j}

(a,b) {a,b} {c} {d} {e} {f} {g} {h} {i} {j}

(a,c) {a,b,c} {d} {e} {f} {g} {h} {i} {j}

(b,c)

(b,d) {a,b,c,d} {e} {f} {g} {h} {i} {j}

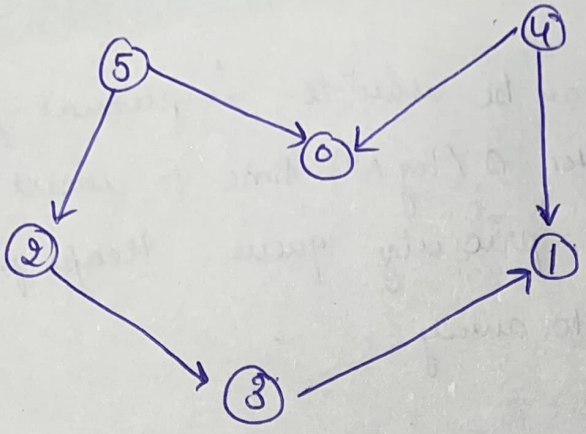
{e,f} {a,b,c,d} {e,f} {g} {h} {i} {j}

{e,g} {a,b,c,d} {e,f,g} {h} {i} {j}

{h,i} {a,b,c,d} {e,f,g} {h,i} {j}

No. of connected components = 3 → Ans

Ans:-



We take source code as 5

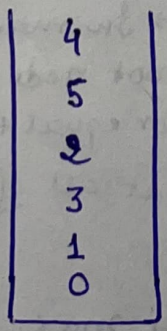
Applying Topological sort

DFS (5)
↓
DFS (0)

DFS (2)
↓
DFS (3)
↓
DFS (1)

DFS (4)
↓
Not possible

DFS



stack

4 → 5 → 2 → 3 → 1 → 0

Ans 9

Yes heap data str. can be used to implement priority queue. It will take $O(\log N)$ time to insert & delete each element in priority queue. Heaps provide Better performance Comparison to array.

The graphs like Dijkstra's Shortest path Algorithm, prims minimum spaning tree uses priority queue.

- Dijkstra's Algorithm :- When graph is stored in form of adjacency list or matrix.
- Prims Algorithm - It is used to store key of nodes and extract min key node at every step.

Ans 10 :-

Min - Heap	Max - Heap
→ In min. heap, key present at root node must be less than or equal to among keys present at all of its children.	→ In max heap key present at root node must be greater than or equal to among keys present at all of its children.
→ min key element → present at root	→ max. key element present at root.
→ It uses Ascending priority.	→ It uses descending priority.