PAGE No.
DATE

Tutorial 5

Q.1) What is difference between DFS & BFS. Write applications of both algorithm

A.1)

DFS	BFS
- Uses stack to find shortest path	- Uses Queue to find shortest path.
- stands for Depth first search	- stands for Breadth first search
- Suitable for searching vertices closes to given source	- suitable where solutions are close to source
- Recursive algorithm - uses backtracking	- NO concept of backtracking
- Requires less Memory	- Requires more Memory.
- Considers not all neighbours first : and then explores all path through this decision	- considers all neighbours first : not suitable for decision making
eg:- acyclic graph, topological order	eg:- Bipartite graph, shortest path etc.

Q.2) Which Data structure is used to implement DFS & BFS & why?

A.2) DFS Algorithm traverses a graph in depthward motion & uses stack to remember to get next vertex to search, when a dead end occurs in any iteration. So basically in order to reach to the deepest node first stack-used

- BFS uses queue to avoid recursion. Queue keeps track of locations remaining to be searched as opposed to needing a potentially large amount of processor stack space for recursive solution.

Q.3) What do you mean by sparse & dense graphs? Which representation of graph is better for both.

A.3)

Dense Graph	Sparse Graph
- A graph in which number of edges is close to maximal n.o of edges	- A graph in which n.o of edges is minimal to n.o of edges.
- Graphs are densly connected.	- Graphs are sparsely connected (eg Trees).
- $G = (V, E)$ in which $\|E\| = O(\|V\|^2)$	- $G = (V, E)$ in which $\|E\| = O(\|V\|)$.
- If a graph is dense we should store it as adjacency matrix.	- If graph-sparse - store it is a list of edges.

Q.4) How to detect cyclic graph using BFS & DFS?

A.4) using BFS:-

(i) compute indegree (n.o of incoming edges) for each vertex present in graph & initialize count of visited nodes as 0.

(ii) Pick all vertices with in-degree as 0, & add to queue. (Enqueue operation)

(iii) Remove vertex from queue.
- Increment count of visited nodes by 1

- Decrease in-degree by 1 for all neighbouring nodes
- If in-degree of a neighbouring node is reduced to zero, add to queue

4) Repeat step 3 until queue's empty.

5)

Using DFS:-

1) Create graph using given number of edges & vertices
2) Create recursive f() that have current index or vertices, visited array & parent node.
3) Mark current node as visited
4) Find unvisited vertices that are adjacent to current node.
5) If adjacent node not parent - already visited - return true

Q.5) What do you mean by disjoint set data structure? Explain 3 operations along with examples, which can be performed on disjoint sets.

A.5) Disjoint-set data structure:- also called union find data structure or merge-find set is data structure that stores a collection of disjoint (non-overlapping) sets. Three operations performed by Disjoint-set data structure:-

i) Making new set containing a new element
ii) Finding representative of set containing given element.
iii) Merging two sets

1) Making New Set:- Make set operation adds a new element into a new set containing only the new element, &

new set is added to data structure.

(ii) finding Set Representatives

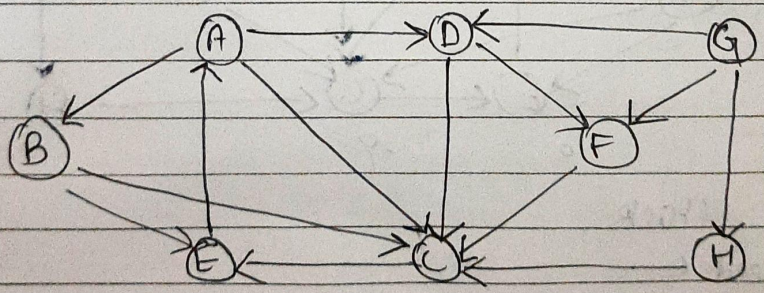
Find operation follows chain of parent pointers from a specified query node x until it reaches a root element. Root element represents set to which x belongs and may be x itself.

(iii) Merging Two Sets :-

Operation union(x,y) replaces set containing x & set containing y with their union.

Q.5) RUN DFS & BFS on graph shown on Pg 45.

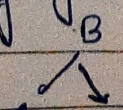
A.5)



Q.6) BFS :- (A) → parent node

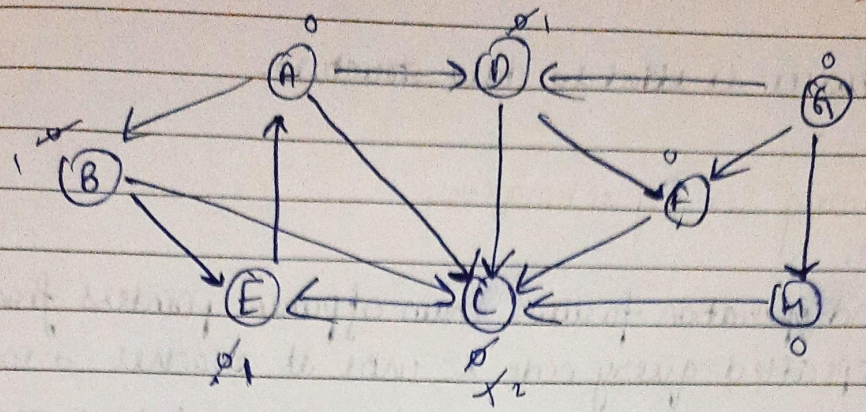
B C D → child node

- taking any one child node, say 'B'



(A, B)

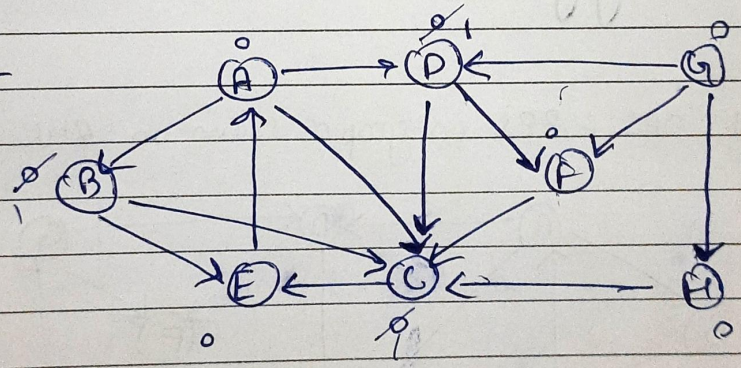
& furthermore . . .



Parent	A	A	A	B	B	D	D	F
Node	B	C	D	E	C	C	F	C

path :- A → B → C.

DFS :-



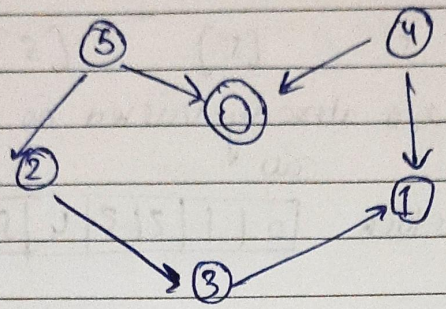
Nodes STACK
Processed ←

A	A
B	A B
E	B E
C	E C
	C

path A → B → C

Q.7.) Apply topological sorting & DFS on graph having vertices from 0 to 5

Ans.)



Adj ent list(G)

0 → 1

1 →

2 → 3

3 → 1

4 → 0, 1

5 → 0, 2

	0	1	2	3	4	5
visited	-	-	-	-	-	-

stack (empty)

step 1: Topological sort(0), visited(0) = true

↓

list empty (no recursion call)

stack [0]

step 2: Topological sort[2], visited[2] = true

" [3] " [3] = true

- 1 - already visited, no more recursion

call

stack [0|1|3|2

Step 5:- "i" [5] "i" [5] = true

2, 0 already visited on recursion

call

stack [0|1|2|3|4|5

print all elements of stack from top → bottom.

Q.8) Can Heap be used to implement priority queue? Name few algorithm where you need to use priority queue?

A.8) Heaps can be used to implement priority queue. It takes O(log N) time to insert & delete each element in priority queue.

Algorithm where priority queue can be applied are Dijkstra, Prim's algorithm.

Q.9) Difference between Max & Min Heap?

- It is used to maximum element in heap
- It's used to access minimum element in heap