Date ___/___/_____ Page No.: _____

Tutorial - 5

Q1 - What's difference between DFS & BFS. Write applications of both algorithm.

Ans1 -

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 where solution are closer to source	→ Suitable for searching vertices closer to given source
→ Recursive algorithm - uses backtracking	→ No concept of backtracking
→ Requires less memory	→ Requires more memory
→ Considers not all neighbours first, ∴ & then explores all path through this decision eg:- acyclic graph, topological order	→ Considers all neighbours first. ∴ not suitable for decision making Eg:- Bipartite graph, shortest path, etc.

Q2 - Which data structure is used to implement DFS & BFS & why?

Ans 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.

Date ___/___/___ Page No.: ___

→ 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

Q3 - What do you mean by sparse & dense graphs? Which representation of graph is better for both.

Dense Graph	Sparse Graph
→ A graph in which no. of edges is close to maximal no. of edges	→ A graph in which no. of edges is minimal to no. of edges
→ Graphs are densely 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 edge

Q4. How to detect cycle graph using BFS & DFS?

Ans 4 - Using BFS :-

i) Compute indegree (no. 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)

Chitra

Date ___/___/_____ Page No.: _____

iii) Remove vertex from queue
- increment count of visited nodes by 1

iv) Decrease in-degree by 1 for all neighbouring nodes

v) If in-degree of a neighbouring node is reduced to zero, add to queue

Repeat step 3 until queue's empty.

Using DFS :-

i) Create graph using given no. of edges & vertices

ii) Create recursive (f) that have current index or vertices, visited array & parent node

iii) Mark current node as visited

iv) Find unvisited vertices that are adjacent to current node

v) If adjacent node not parent - already visited - return true

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

Ans - 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 :-

Chitra

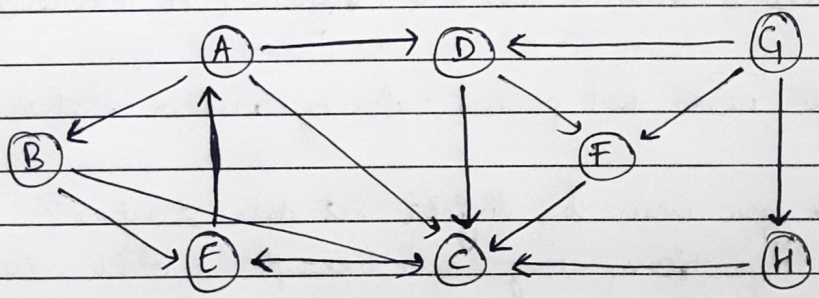
i) Making new set containing a new element
ii) Finding representative of set containing given element
iii) Merging two sets

(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.

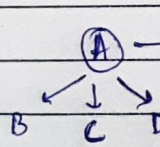
(2) Finding Set Representatives : Find operation follows chain of parent pointers from a specified query node x until it reaches a root element. Root element represents set to which x belongs & may be x itself.

(3) Merging Two Sets : operation union(x,y) replaces set containing x & set containing 'y' with their union.

Q6 - Run DFS & BFS on graph shown on RHS.

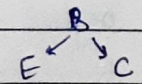


Ans6 - BFS :-



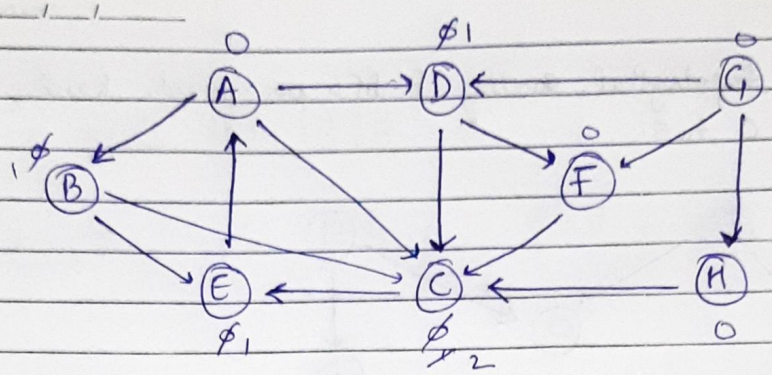
A → parent node
B, C, D → child node

- taking any one child node, say 'B'



& furthermore (A, B)

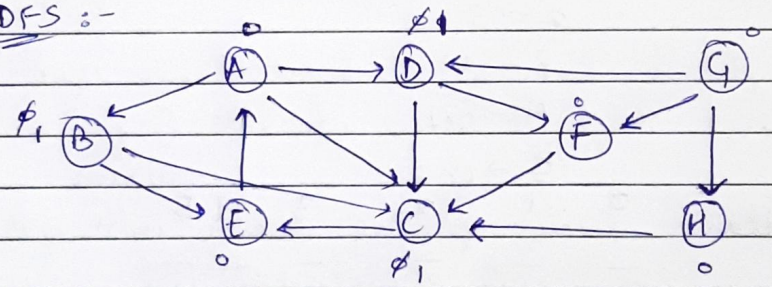
Date ___/___/_____ Page No.: _____



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	AB
E	BE
C	BEC
C	EC

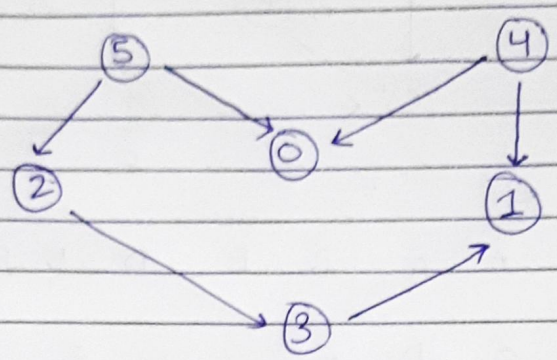
path A → B → C

Chitra

Date ___/___/_____ Page No.: _____

Q7 - Apply topological sorting & DFS on graph having from 0 to 5.

Ans7 -



Adjacent list (G)

0 → 1
1 →
2 → 3
3 → 1
4 → 0, 1
5 → 0, 2

visited	0	1	2	3	4	5
	—	—	—	—	—	—

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

- already visited, no more recursion

Date ___/___/_____ Page No.: _____

call

Stack | 0 | 1 | 3 | 2 |

Step-3 : +1 [5] '' [5] = true

2,0 already visited no recursion call

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

Print all elements of stack from top → bottom

Q8 - Can heap be used to implement priority queue? Name few algorithm where you need to use priority queue?

Ans 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.

Q9 - Difference b/w Max & Min. Heap?

It's used to maximum element in heap

It's used to access minimum element in heap.