Name:- Neeraj Singh Bhandari Sec:- H
Roll No:- 03

Assignment-3
Tutorial - 5

1.

BFS	DFS
(i) BFS stands for Breadth first search	(i) DFS stands for Depth First Search
(ii) BFS uses Queue data structure for finding Shortest path	(ii) DFS uses stack data structure
(iii) BFS can be used to find single source shortest path in an unweighted graph, becaus in BFS we reach a vertix with minimum no. of edges from a source vertex	(iii) In DFS, we might traverse through more edges to reach a destination vertex from a source
(iv) BFS is more suitable for searching verticas which are closer to the given source	(iv) DFS is more suitable when there are sol" away from source.
(v) In BFS there is no concept of backtracking.	(v) DFS algorithm is a recursive algorithm that uses the idea of backtracking.

The DFS or Depth First search is used in different places. Some common uses are.

- If we perform DFS on unweighted graph. then it will create minimum spanning tree for all pair shortest path tree.
- Using DFS we can find path between two given vertices u and v.
- Using DFS, we can find strongly connected coponents of a graph. If there is a path from each vertex to every other vertix, that is strongly connected

Like DFS, the BFS (Breath first search) is also used in different situations, These are like below:-

In peer-to-peer network like bit-torrent, BFS ... nodes

- Using GPS navigation system BFS is used to find neighbouring places
- In networking, when we want to broadcast some packets, we use the BFS algorithm.

Ans 2

Stack is used in DFS. DFS can also be implemented using recursion. In DFS algorithm traverses a graph in a deptward motion and uses a stack to remember to get the next vertex to start a search, when a dead end occurs in any iteration.

When In Breadth first search (BFS) algorithm traverses a graph in a breadthward motion and uses a queue to remember to get the next vertex to start a search, when a dead end occurs in any iteration

Ans 3

Dense graph :- It is a graph in which the number of edges is close to the maximal number of edges.

Sparse graph :- In this type of graph the number of edges is close to the minimal number of edges

If the graph is sparse, we should store it as a list of edges. Alternatively, if the graph is dense, we store it as an adjacency matrix

Ans4

Step

Step

Step

Ans4 BFS for detect cycle

Steps are as follows:-

Step1 Compute in-degree(number of incoming edges) for each of the vertex present in the graph and initialize the count of visited nodes as 0

Step2 Pick all the vertices with in-degree as 0 and add them into a queue.

Step3 Remove a vertex from the queue(Dequeue operation) and then
(i) Increment count of visited nodes by 1.
(ii) Decrease in-degree by 1 for all its neighbouring nodes
(iii) If in-degree of a neighbouring nodes is reduced to zero, then add it to the queue.

Step4 Repeat step3 until the queue is empty.

Step5 If count of visited nodes is not equal to the no. of nodes in the graph has cycle, otherwise not

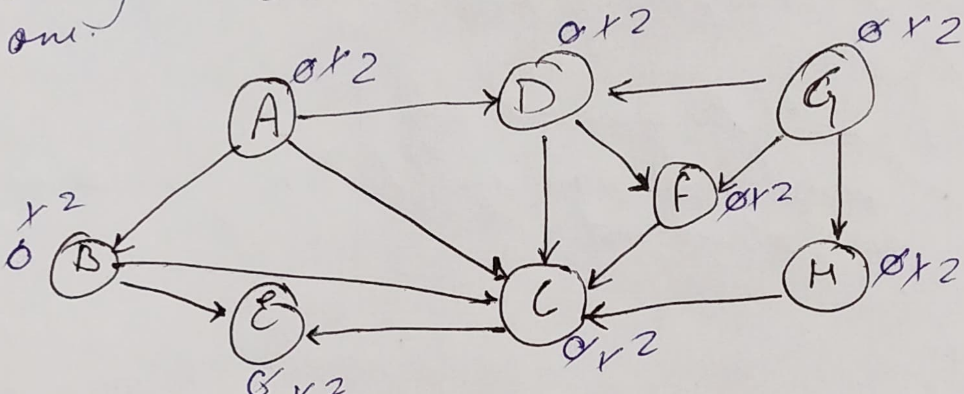
DFS Approach

DFS can be used to detect a cycle in a Graph. DFS for a connected graph produces a tree. There is a cycle in a graph only if there is a back edge present in the graph. A back edge is an edge that is from a node to itself (self loop) or one of its ancestors in the tree produced by DFS. In the following graph, there are 3 back edges, marked with a cross sign. We can observe that these. 3 back edges indicates 3 cycles. present in the graph.

Ans 5 :- The disjoint set can be [illegible] between two [illegible]
there is no common element between
Disjoint sets structure is also called "union-final structure".
So, union, find and makeset operations should be supported anyway.

Make-set (x) :- create a set containing x
Find (x) :- return which set x belongs
Union (x, y) :- merge the sets containing x and containing y into one.

Ans 6



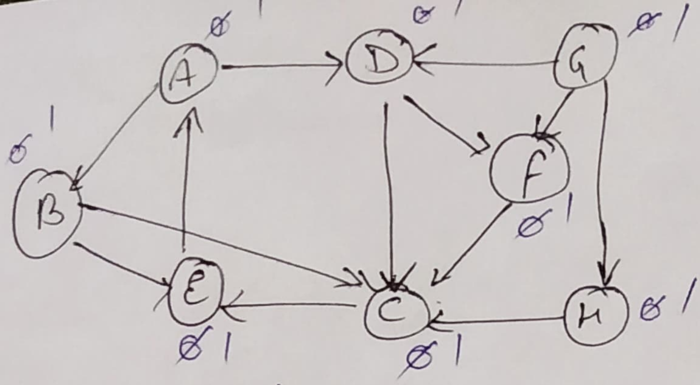
BFS

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

All visited from Source G

Source	Destination	Path
G	A	G→H→C→E→A
G	B	G→H→C→A→B
G	C	G→H→C
G	D	G→D
G	E	G→H→C→E
G	F	G→F
G	G	G→H
G	H	

DFS :-

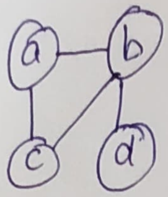


Node Processed	Stack
G	G
D	DFH
C	CFH
E	EFH
A	AFH
B	BFH
~~B~~	FH

Source	Destination	Path
G	A	G→D→C→E→A
G	B	G→D→C→E→A→B
G	C	G→D→C
G	D	G→D
G	E	G→D→C→E
G	F	G→F
G	H	G→H

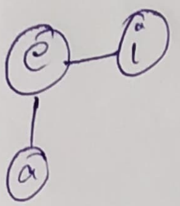
Sol 7

(i)



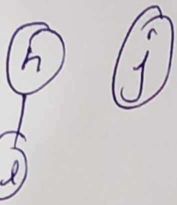
No (V) = 4
No (CC) = 1

(ii)



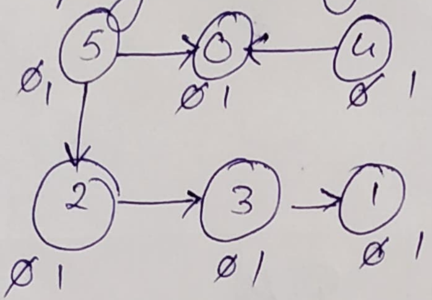
No. of (V) = 3
No. of (CC) = 1

(iii)



No (V) = 3
No (CC) = 2

Sol 8 Topological Sorting



Adjacency list
0 ->
1 ->
2 -> 3
3 -> 1
4 -> 0, 1
5 -> 2, 0

Stack

0	1	3	2	4	5

Topological = 5 4 2 3 1 0

DFS Stack -> | 4 | 0 | 1 | 3 | 2 | 5 |

Head ->

DFS -> 5 -> 2 -> 3 -> 1 -> 0 -> 4

Ans 9 Applications of priority Queue

(i) Dijkstra's algo:- we need to use a priority queue here so that minimal edges can have higher priority

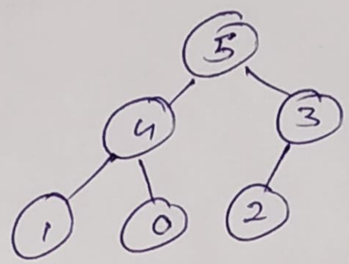
2.7 Load Balancing:- Load Balancing can be done from branches of higher priority to those of lower priority

3.7 Interrupt Handling:- To provide proper numerical priority to more imp interrupt.

4.7 Huffman code:- For data compression in Huffman code

Sol 10:- Max Heap:- Where parent is bigger than both children

Eg:-



Min Heap:- Where parent is smaller than both children

Eg:-