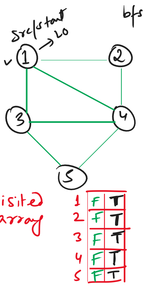
Graph Traversals :→

BFS Traversal :→

Adjacency List:

Node	List of neighbours
→1	2, 3, 4
→2	1, 4
→3	1, 4, 5
→4	1, 2, 3, 5
→5	3, 4

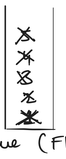


visited array

1	F T
2	F T
3	F T
4	F T
5	F T

bfs O/p: 1 2 3 4 5
v0 v1 v2

Who are your neighbours?

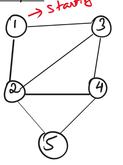


Queue (FIFO)

DFS Traversal in Graphs :→

Adjacency List :→

Node : List of neighbours
1 → 2, 3
2 → 1, 3, 4, 5
3 → 1, 2, 4
4 → 2, 3, 5
5 → 2, 4

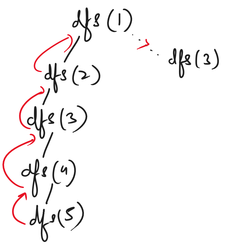


Visited →

1	2	3	4	5
F	F	F	F	F
T	T	T	T	T

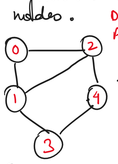
(Recursion) ***

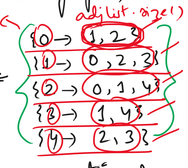
1, 2, 3, 4, 5



Cognizant | IBM | TCS :→ Probable Questions :→

* Given an undirected graph, count the number of edges & nodes.

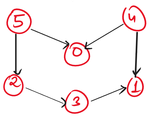




	0	1	2	3	4
0	0	1	1	0	0
1	1	0	1	1	0
2	1	1	0	0	1
3	0	1	0	0	1
4	0	0	1	1	0

* Convert an adjacency matrix of a graph to adjacency list.

Important Graph Algorithms :→ (Topological Sort) (DAG)

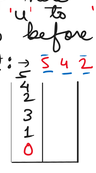


Edges

u	v
5	0, 2
4	0, 1
5	2
2	3
3	1
4	1

Linear ordering of nodes/vertices such that if there is an edge from 'u' to 'v', 'u' always comes before 'v'.

Output :→ 5 4 2 3 1 0



⇒ Component wise traversal:

```
for (i=0; i<n-1; i++) dfs(i)
```

Adj list:
0 →
1 →
2 → 3
3 → 1
4 → 0, 1
5 → 0, 2

T	T	T	T	T	T
F	F	F	F	F	F
0	1	2	3	4	5

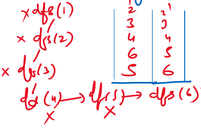
visited Array

dfs(0)
dfs(1)
dfs(2)
dfs(3)
dfs(4)
dfs(5)

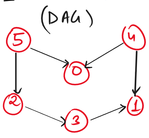
dfs(stack)
⇒ 1 2 3 4 6 5
⇒ 1 2 3 4 5 6

* Which kind of graph is always having linear ordering?

Linear Graph



Topological Sort (BFS) (Kahn's Algorithm) :→ (Indegree)



Edges

u	v
5	0, 2
4	0
5	2
2	3
3	1
4	1

S1: Insert all the nodes with indegree "0" into the queue.

S2: Take out the node from the queue & relax the indegree of adjacent nodes.

queue

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

Indegree Array:

0	0				
2	2	1	1	0	0
0	1	2	3	4	5

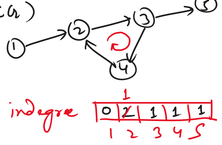
Indegree:
Number of incoming edges to a particular node.

Output: Topo Sort :→ [4 5 0 2 3 1]

if edge from u to v
indegree[v]++

Detect a cycle in a DCG using Kahn's Algo :→

(DCG)



indegree

1				
0	2	1	1	1
1	2	3	4	5

queue

Adj list
1 → 2
2 → 3
3 → 4, 5
4 → 2
5 →

if (topsort.size == N) → no cycle
else size != N → cycle.

Topo Sort: 1 ≠ 5