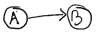
BFS DFS X section

26 February 2025 19:39

Graph Data Structure is a collection of **nodes**. Nodes are connected by **edges**. Edges represent connection between nodes.

Directed graph:  You can go from node A to B, but not B to A. Arrow will be present.

Undirected graph: You can go from B to A and also from B to A. Arrow is absent.

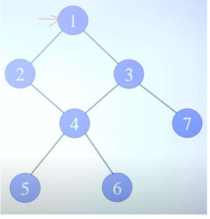
BFS is a graph traversal algorithm that explores all the neighbours of a node before moving on to their neighbours.

DFS is a graph traversal algorithm that explores as far as possible along each branch before backtracking.

Graphs Traversal

To traverse a Graph means to start in one vertex, and go along the edges to visit other vertices until all vertices, or as many as possible, have been visited.

2 techniques: BFS (Breadth first search), DFS (depth first search)

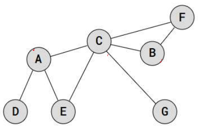


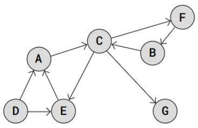
BFS Algorithm

1. **Push the starting node into the queue** and mark it as visited.
2. **While the queue is not empty, repeat:**
   - **Remove** an element (node) from the front of the queue.
   - **Process the node** (if required). Print it
   - **Push all its unvisited neighboring nodes** into the queue and mark them as visited.

Iterative DFS Algorithm (Using a Stack)

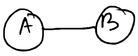
1. **Push start element in stack and print it.**
2. **Repeat till stack is not empty:**
   a. **See the top element in stack.**
   b. **If all its neighbours have been visited, remove the top item from stack.**
   c. **Else push one of its unvisited neighbours and continue the process.**

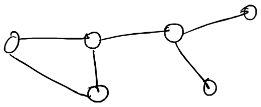




Graph- ① Non Linear Data Structure.
② finite number of nodes/vertices.
③ Nodes— Connections → Edges.

→ Directed

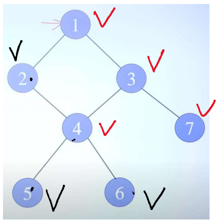
→ Undirected 

Graph Traversal - 

BFS (Breadth first search) DFS (Depth first Search).

→ ① Queue DS.

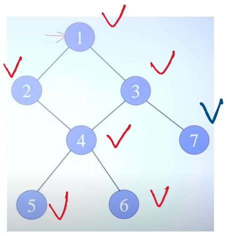
Graph traversal begin from Node 3.



BFS

Queue ~~3~~ ~~7~~ ~~1~~ ~~4~~ 2 ~~5~~ ~~6~~ →

Result:-3 7 1 4 2 5 6



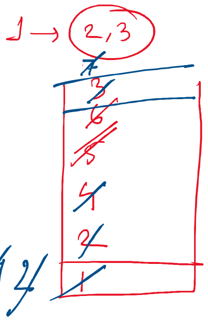
Stack → LIFO.

Start DFS from node 1.

Top = ~~1~~ ~~2~~ ~~4~~ ~~8~~ ~~4~~ ~~6~~ · ~~4~~ ~~3~~

~~1~~ ~~3~~ ~~4~~ ~~2~~

1



Iterative DFS Algorithm (Using a Stack)

1. **Push start element in stack, print it, mark it as visited.**
2. **Repeat ~~till stack~~ is not empty:**
   a. **Find the topmost element in stack.**
   b. **If all its neighbours have been visited, remove the top item from stack.**
   c. **Else push one of its unvisited neighbours, print it, mark it as visited and continue the process.**

Print: 1 2 4 5 6 3 7

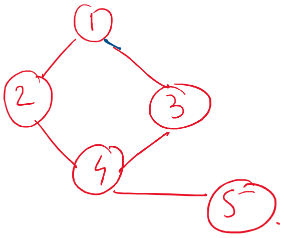
Graph Representation -

(1) Adjacency Matrix (2) Adjacency List.

(1) Adjacency Matrix -

N = no. of nodes = 5.

Adj. Matrix = N × N = 5 × 5,



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

Matrix M.

4,3

M[i][j] =

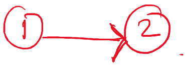
1 → 2

M[1][2].

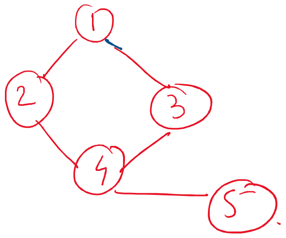
M[i][j] = 0, No edge from nodes i to j.

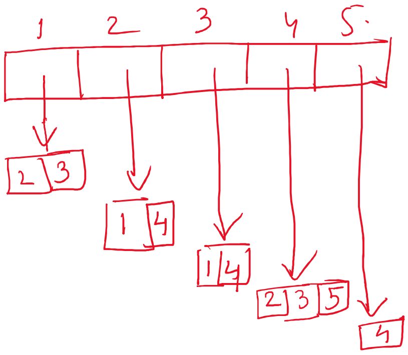
1, Edge from node i to j.

	1	2
1	0	1
2	0	0



(2) Adjacency list → List of List.




1
2
3
4
5
2 | 3
1 | 4
1 | 4
2 | 3 | 5
4