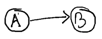
BFS DFS S section

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.

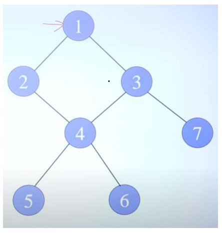
BFS is a graph traversal algorithm that explores all the neighbours of a node before moving on to their neighbours.

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)

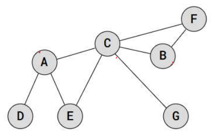


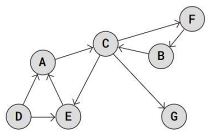
BFS Algorithm

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)

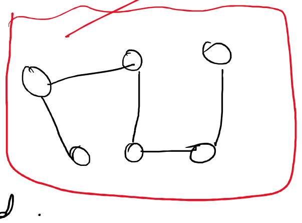
1. **Push start element in stack and print it.**
2. **Repeat till stack is not empty:**
   a. **See the top element in stack.**
   b. **If all its neighbours have been visited, remove the top item from stack.**
   c. **Else push one of its unvisited neighbours and continue the process.**





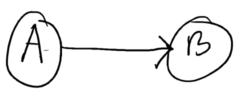
Graph — finite number of vertices (nodes).
Connections ⟶ Edges.

visit every node.



① Directed Graph.
② Undirected

Undirected.

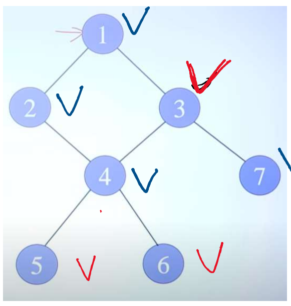
 Directed.

Graph Traversal — 2 techniques —

BFS (Breadth First Search) —

DFS (Depth First Search)

BFS ⟶ Level Order Traversal.



Graph Traversal can start from any node.

Node 3 BFS traversal start.

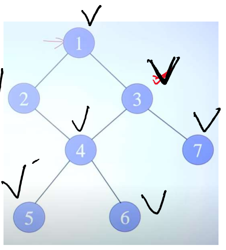
1 7 4

queue (FIFO)

7 1 4
4 1

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

Print - 3 1 4 7 2 5 6



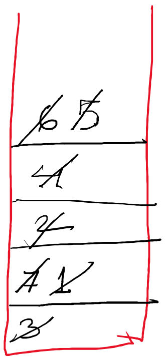
Stack — LIFO.

DFS traversal start from node 3.

1, 4, 6

5,6

top = ~~3~~ ~~7~~
~~3~~ ~~1~~ ~~2~~
~~4~~ ~~6~~ ~~4~~ ~~5~~
~~4~~ ~~2~~ ~~1~~ ~~3~~



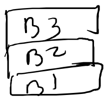
Iterative DFS Algorithm (Using a Stack)

1. **Push start element in stack and print it, mark as visited.**
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, print it, mark as visited and continue the process.**

Print - 3, 7, 1, 2, 4, 6, 5

DFS from node 2

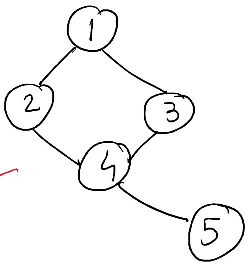
H/W DFS traversal from 1, 2, 4, 5, 6, 7.



Representation of Graph.

① Adjacency Matrix
② Adjacency List.

① Adjacency Matrix. —



N = No. of. nodes = 5.

Adj Matrix dimension = N×N = 5×5.

$$\begin{array}{c|ccccc} & 1 & 2 & 3 & 4 & 5 \\ \hline 1 & 0 & 1 & 1 & 0 & 0 \\ 2 & 1 & 0 & 0 & 1 & 0 \\ 3 & 1 & 0 & 0 & 1 & 0 \\ 4 & 0 & 1 & 1 & 0 & 1 \\ 5 & 0 & 0 & 0 & 1 & 0 \end{array}$$

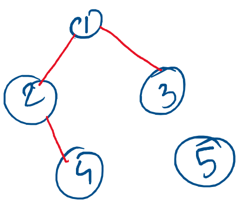
(1,2) 1st row, 2nd col.

4, 3

5, 1

Matrix M.

M[i][j] = 0, no edge from node i to j.
 1, edge from node i to j.



i == j, 0

$$\begin{array}{c|cc} & 1 & 2 \\ \hline 1 & 0 & 1 \\ 2 & 0 & 0 \end{array}$$

$$\begin{array}{c|cc} & 1 & 2 \\ \hline 1 & 0 & 1 \\ 2 & 0 & 0 \end{array}$$

$$\begin{tikzcd} \textcircled{1} \arrow[r] & \textcircled{2} \end{tikzcd}$$