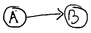
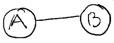
BFS DFS N section

26 February 2025 19:39

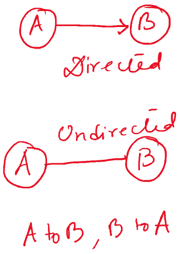
Graph Data Structure is a collection of **nodes**. Nodes are connected by **edges**. Edges represent connection between nodes.

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.

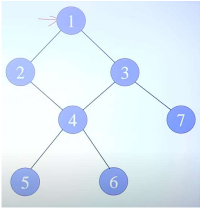
DFS is a graph traversal algorithm that explores as far as possible along each branch before backtracking.



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)

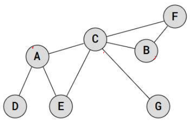


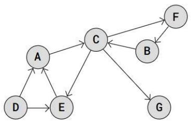
BFS Algorithm

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)

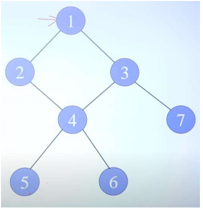
1. **Push start element in stack, mark it visited and print it.**
2. **Repeat till stack is not empty:**
   a. **See the top element in stack.**
   b. **If all its neighbours have been visited, remove the top item from stack.**
   c. **Else push one of its unvisited neighbours, print it, mark it as visited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





BFS Traversal-

Traversal start from node 1.



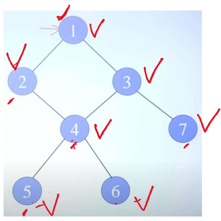
1 2 3 4 7 5 6.

3 1 4 7 2 5 6

3 7 4 1

4 7 1

Start traversal from node 3.

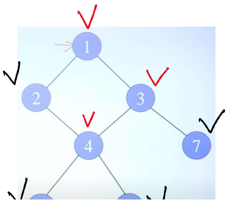


Queue:- ~~3~~ ~~1~~ ~~4~~ ~~7~~ ~~2~~ ~~5~~ ~~6~~

Print:- 3 1 4 7 2 5 6

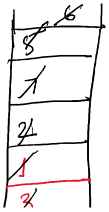
1. **Push the starting node into the queue** and mark it as visited.
2. **While the queue is not empty, repeat:**
   - **Remove** an element (node) from the front of the queue.
   - **Process the node** (if required). Print it
   - **Push all its unvisited neighboring nodes** into the queue and mark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

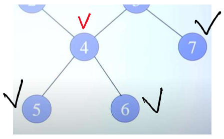
LIFO → top= ~~4~~ ~~3~~, ~~1~~ ~~2~~ ~~1~~ ~~3~~ ~~7~~ ~~3~~ 4 ~~5~~ ~~4~~ ~~6~~ 4



Iterative DFS Algorithm (Using a Stack)

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

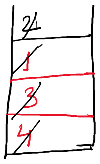




a. See the top element in stack.

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 it as visited and continue the process.



Stack.

DFS traversal start from node 4.

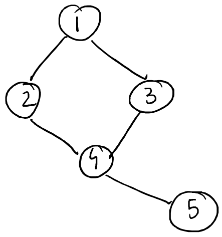
Print:- 4, 3, 1, 2, 7, 5, 6

DFS traversal from node 1.

H/W) Perform BFS from node 5, 7, 2.
Perform DFS from node 1, 7, 3.

Representation of Graph

① Adjacency Matrix
② Adjacency List

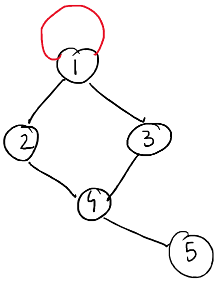


	1	2	3	4	5.
1					
2					
3					
4					
5					

① Adjacency Matrix - N = No of nodes
= 5

Adjacency matrix dimension
N×N = 5×5.

m[1][1] = 1

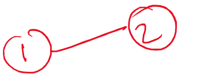


	1	2	3	4	5
1	0	1	1	0	0
2	1	0	0	1	0
3	1	0	0	1	0
4	0	1	1	0	1
5	0	0	0	1	0

Matrix M.

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

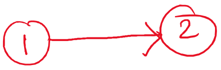
i==j, m[i][j]=0.



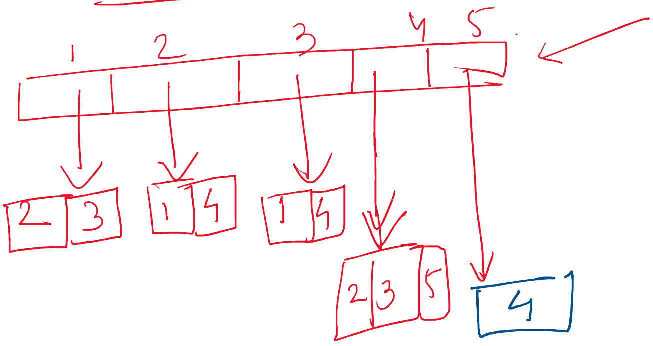
m[1][2]=1
m[2][1]=1

	1	2
1	0	1
2	0	0

Adj Mat.



Adjacency List - list of lists


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