4.2 Traversals – BFS, DFS

4.2.1 Breadth-first traversal (BFS)

Traversing the graph means examining all the nodes and vertices of the graph.

Two standard methods

- Breadth First Search
- Depth First Search.

Breadth First Search (BFS) Algorithm

- Breadth first search is a graph traversal algorithm that starts traversing the graph from root node and explores all the neighbouring nodes.
- Then, it selects the nearest node and explore all the unexplored nodes.
- The algorithm follows the same process for each of the nearest node until it finds the goal.

Algorithm

Steps to implement breadth first search

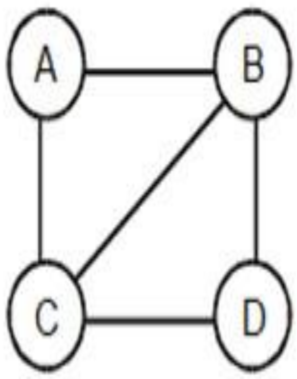
Step 1: Choose any node in the graph, designate it as the search node and mark it as visited.

Step 2: Using the adjacency matrix of the graph, find all the unvisited adjacent nodes to the search node and enqueue them into the queue Q.

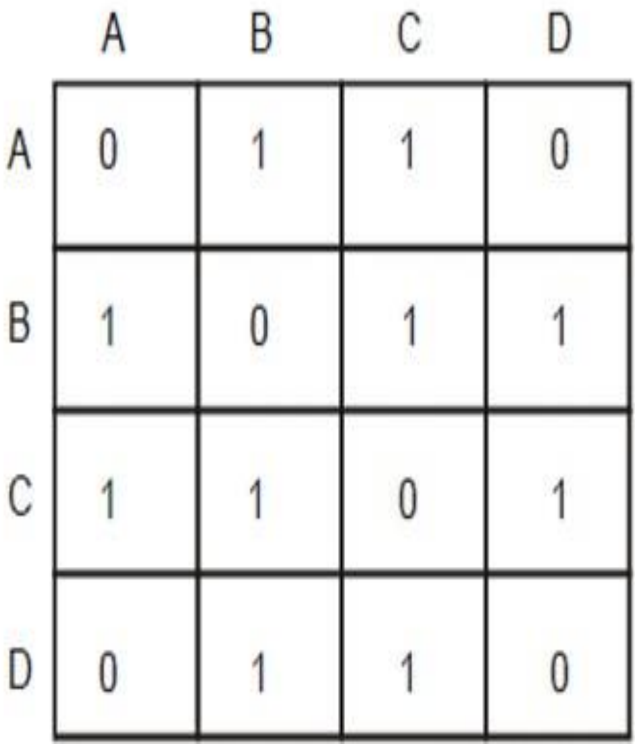
Step 3: Then the node is dequeued from the queue. Mark that node as visited and designate it as the new search node.

Step 4: Repeat step 2 and 3 using the new search node. Step 5: This process continues until the queue Q which keeps track of the adjacent nodes is empty.

Example:



Adjacency matrix

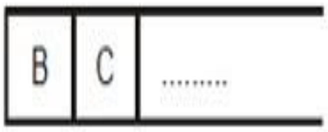


	A	B	C	D
A	0	1	1	0
B	1	0	1	1
C	1	1	0	1
D	0	1	1	0

Implementation

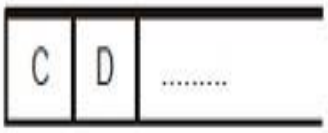
1. Let 'A' be the source vertex. Mark it to as visited.

2. Find the adjacent unvisited vertices of 'A' and enqueue then into the queue. Here B and C are adjacent nodes of A



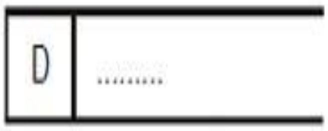
and B and C are enqueued.

3. Then vertex 'B' is dequeued and its adjacent vertices C and D are taken from the adjacency matrix for enqueuing. Since vertex C is already in the queue, vertex D alone is enqueued.



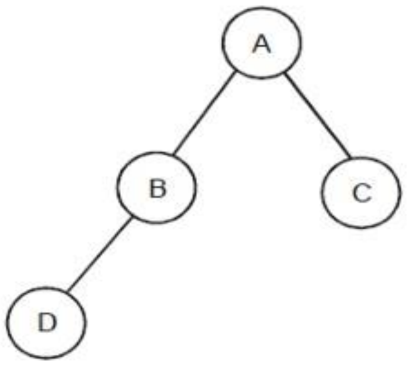
Here B is dequeued, D is enqueued.

4. Then vertex 'C' is dequeued and its adjacent vertices A, B and D are found out. Since vertices A and B are already visited and vertex D is also in the queue, no enqueue operation takes place.



Here C is dequeued

5. Then vertex 'D' is dequeued. This process terminates as all the vertices are visited and the queue is also empty.



Breadth first spanning tree

Applications of breadth first search

1. To check whether the graph is connected or not.

4.2.2 DEPTH-FIRST SEARCH ALGORITHM

Depth first Search or Depth first traversal is a recursive algorithm for searching all the vertices of a graph or tree data structure.

- Depth-first search begins at a starting node A which becomes the current node.
- Process a neighbour of A, then a neighbour of neighbour of A, and so on.
- During the execution of the algorithm, if we reach a path that has a node N that has already been processed, then we backtrack to the current node.
- Otherwise, the unvisited (unprocessed) node becomes the current node.

Algorithm

To implement the Depth first Search perform the following Steps :

Step: 1 Choose any node in the graph. Designate it as the search node and mark it as visited.

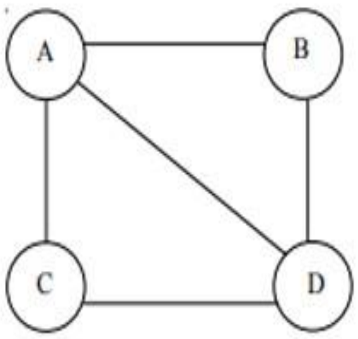
Step: 2 Using the adjacency matrix of the graph, find a node adjacent to the search node that has not been visited yet. Designate this as the new search node and mark it as visited.

Step: 3 Repeat step 2 using the new search node. If no nodes satisfying (2) can be found, return to the previous search node and continue from there.

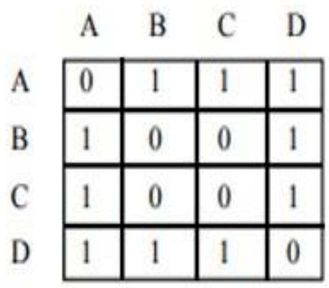
Step: 4 When a return to the previous search node in (3) is impossible, the search from the originally chosen search node is complete.

Step: 5 If the graph still contains unvisited nodes, choose any node that has not been visited and repeat step (1) through (4).

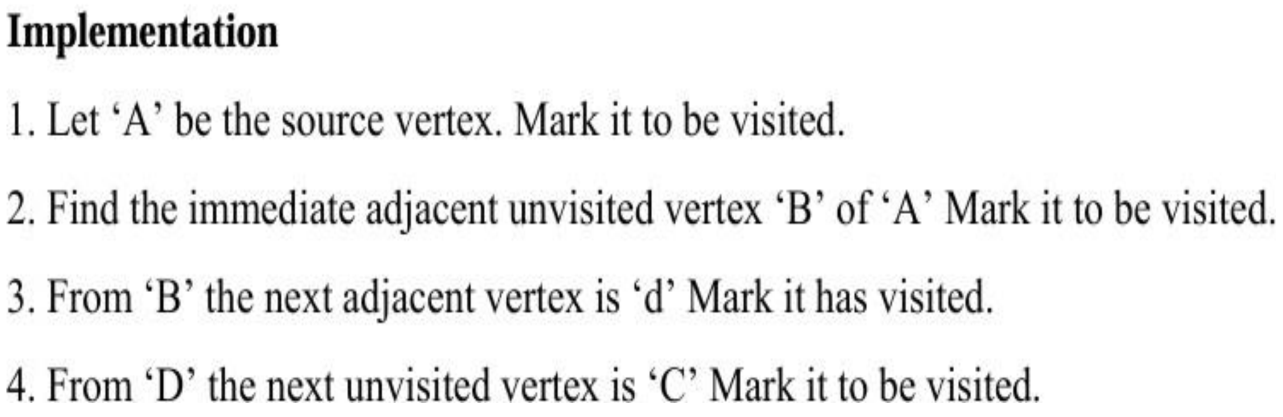
Example : -



Adjacency Matrix



	A	B	C	D
A	0	1	1	1
B	1	0	0	1
C	1	0	0	1
D	1	1	1	0



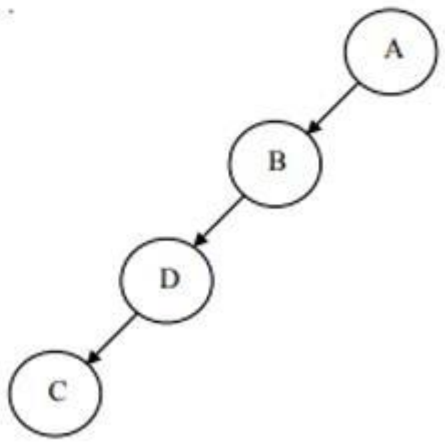
Implementation

1. Let ‘A’ be the source vertex. Mark it to be visited.

2. Find the immediate adjacent unvisited vertex ‘B’ of ‘A’ Mark it to be visited.

3. From ‘B’ the next adjacent vertex is ‘d’ Mark it has visited.

4. From ‘D’ the next unvisited vertex is ‘C’ Mark it to be visited.



Depth First Spanning Tree

Applications of Depth First Search

1. To check whether the undirected graph is connected or not.
2. To check whether the connected undirected graph is Bioconnected or not.
3. To check the a Acyclicity of the directed graph