Course title	: **CSE2001**
Course title	: **Data Structures and Algorithms**
Module	: **6**
Topic	: **1**

Graph Data Structure

Objectives

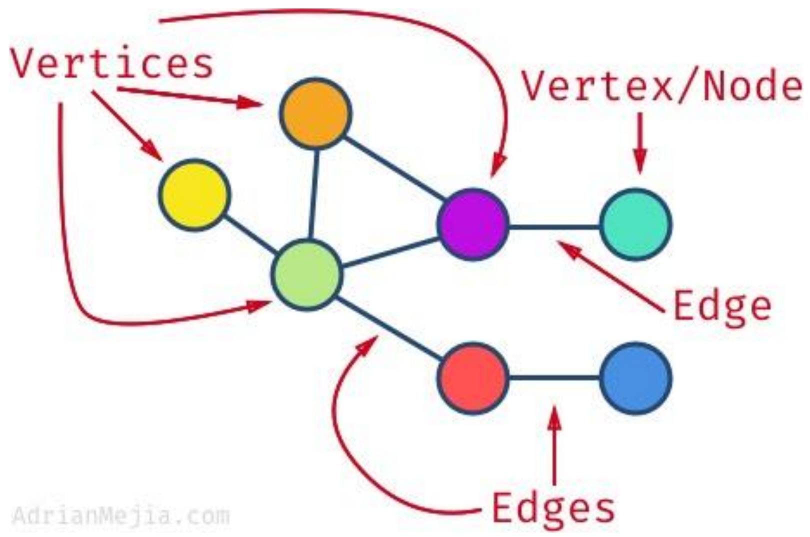
This session will give the knowledge about

- Introduction to Graphs
- Breadth First Search - BFS
- Depth First Search - DFS

Introduction to Graph

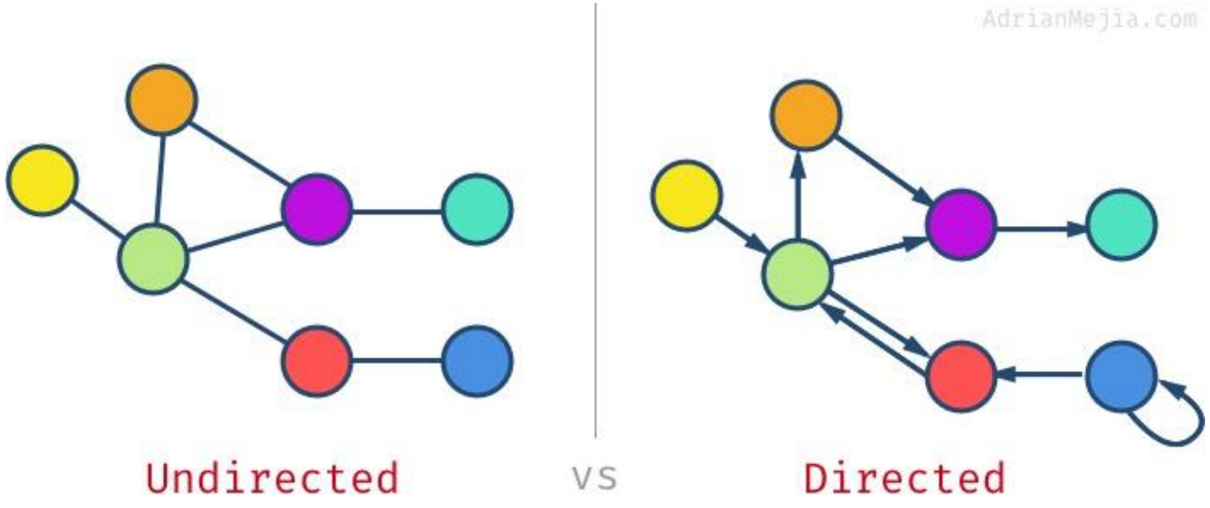
A graph is a collection of nodes (or vertices) and edges (or arcs)

- Each node contains an element
- Each edge connects two nodes together (or possibly the same node to itself) and may contain an edge attribute



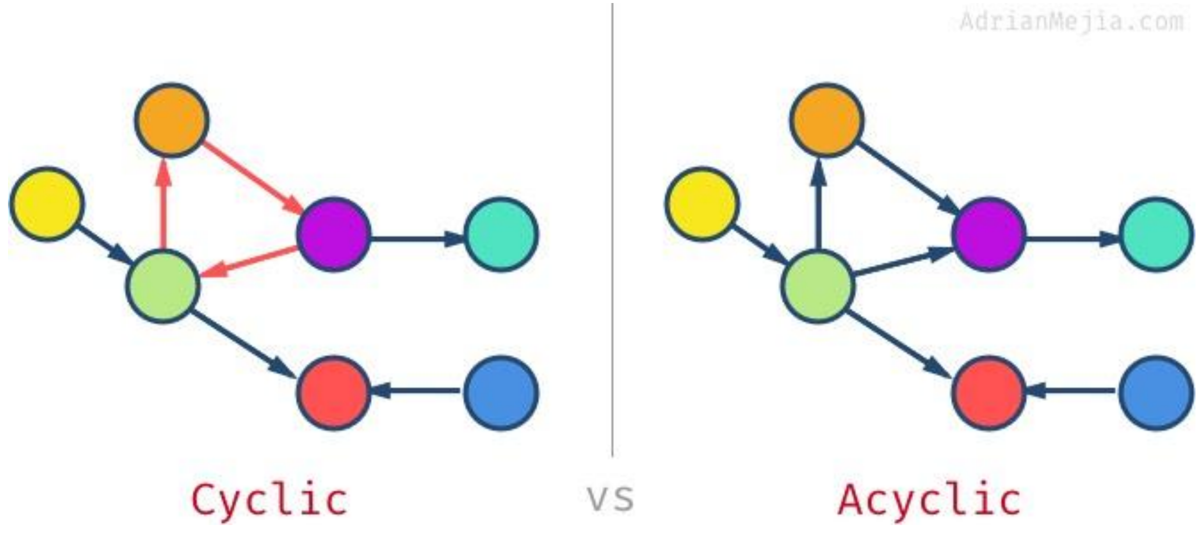
Introduction to Graph

- The degree is the number of edges connected to a vertex. E.g., the purple vertex has a degree of 3 while the blue one has a degree of 1.
- If the edges are bi-directional, then we have an undirected graph.
- If the edges have a direction, then we have a directed graph or di-graph for short.
- Vertex can have edges that go to itself (e.g., blue node), this is called self-loop.



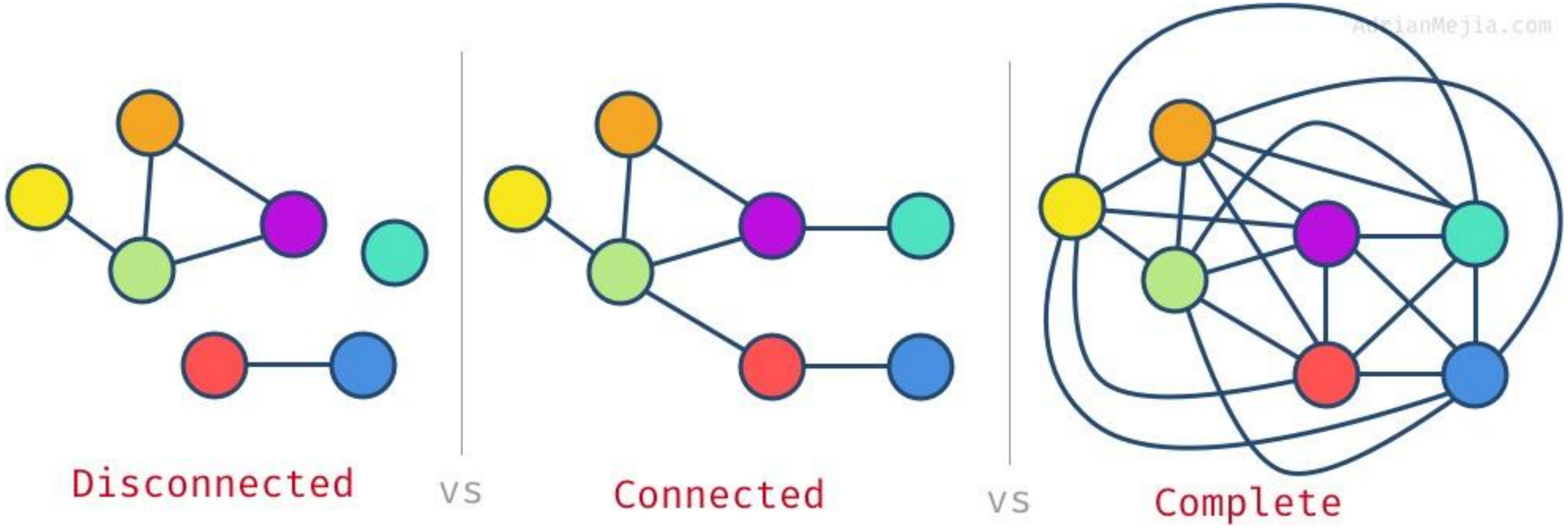
Introduction to Graph

- A graph can have cycles which means that if you traverse through the node, you could get the same node more than once. The graph without cycles is called acyclic graph.
- Also, acyclic undirected graphs are called tree. We are going to cover trees in depth in the next post.



Introduction to Graph

- If all nodes have at least one edge, then we have a connected graph.
- When all nodes are connected to all other nodes, then we have a complete graph.



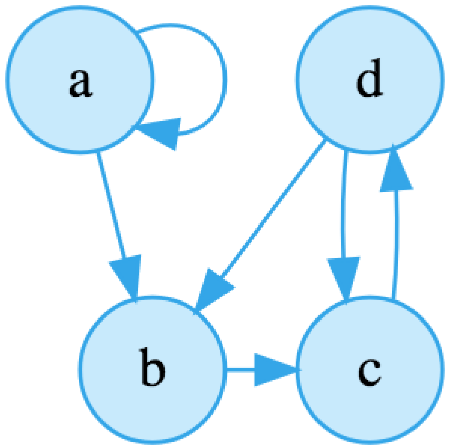
Graph Terminologies

- The size of a graph is the number of *nodes* in it
- The empty graph has size zero (no nodes)
- If two nodes are connected by an edge, they are neighbors (and the nodes are adjacent to each other)
- The degree of a node is the number of edges it has
- For directed graphs,
  - If a directed edge goes from node S to node D, we call S the source and D the destination of the edge
    - The edge is an out-edge of S and an in-edge of D
    - S is a predecessor of D, and D is a successor of S
  - The in-degree of a node is the number of in-edges it has
  - The out-degree of a node is the number of out-edges it has

Representing graphs

There are two primary ways of representing graph:

- Adjacency list
- Adjacency Matrix



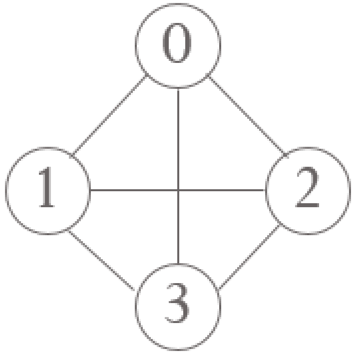
Adjacency Matrix

```
   a b c d e
a  1 1 - - -
b  - - 1 - -
c  - - - 1 -
d  - 1 1 - -
```

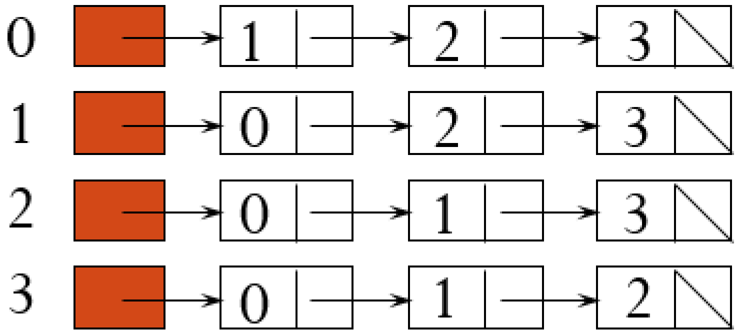
Adjacency List

```
a -> { a b }
b -> { c }
c -> { d }
d -> { b c }
```

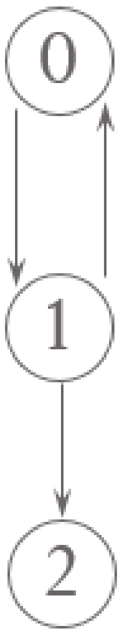
Representing graphs



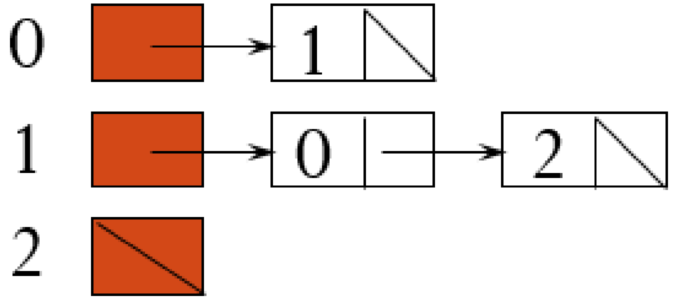
$$\begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix}$$



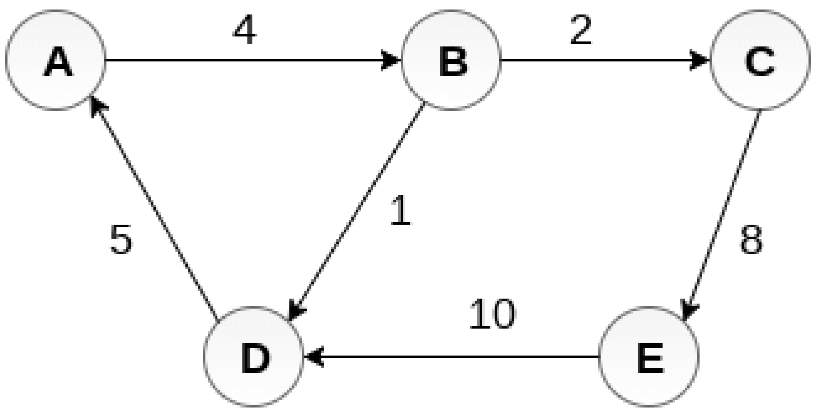
Representing graphs



$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}$$



Representing graphs

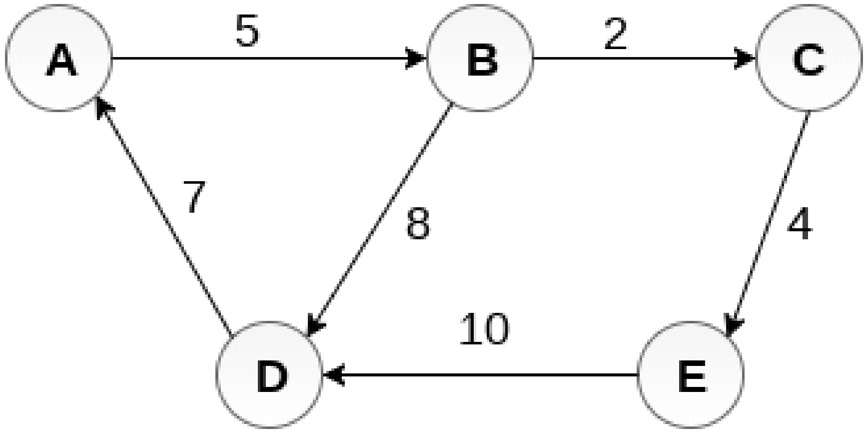


Weighted Directed Graph

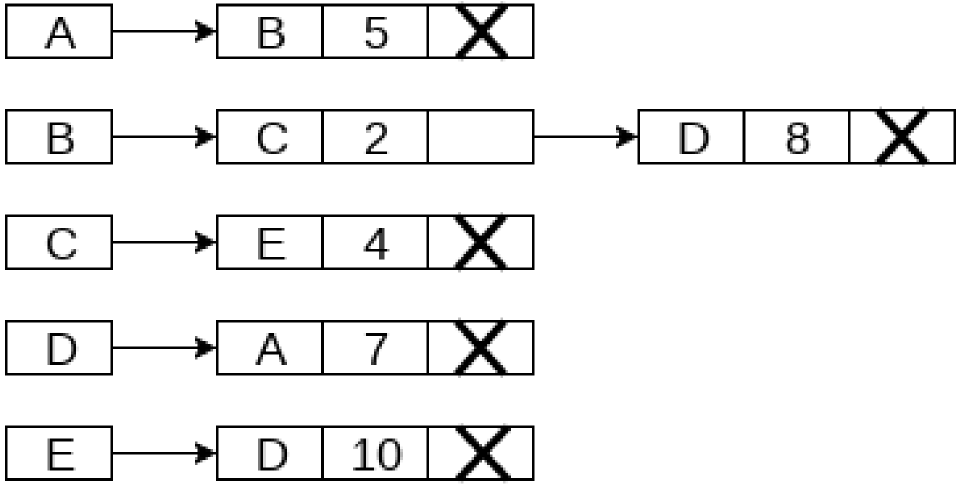
$$
\begin{array}{c|ccccc}
 & A & B & C & D & E \\
\hline
A & 0 & 4 & 0 & 0 & 0 \\
B & 0 & 0 & 2 & 1 & 0 \\
C & 0 & 0 & 0 & 0 & 8 \\
D & 5 & 0 & 0 & 0 & 0 \\
E & 0 & 0 & 0 & 10 & 0
\end{array}
$$

Adjacency Matrix

Representing graphs



Weighted Directed Graph



Adjacency List

Graph Traversal

- BFS (Breadth First Search)
  - Start from a vertex, visit all the reachable vertices in a breadth first manner
  - Uses Queue for non-recursive implementation
- DFS (Depth First Search)
  - Start from a vertex, visit all the reachable vertices in a depth first manner
  - Uses Stack for non-recursive implementation

Depth First Search (DFS) Algorithm

Algorithm

Step 1: SET STATUS = 1 (ready state) for each node in G

Step 2: Push the starting node A on the stack and set its STATUS = 2 (waiting state)

Step 3: Repeat Steps 4 and 5 until STACK is empty

Step 4: Pop the top node N. Process it and set its STATUS = 3 (processed state)

Step 5: Push on the stack all the neighbours of N that are in the ready state (whose

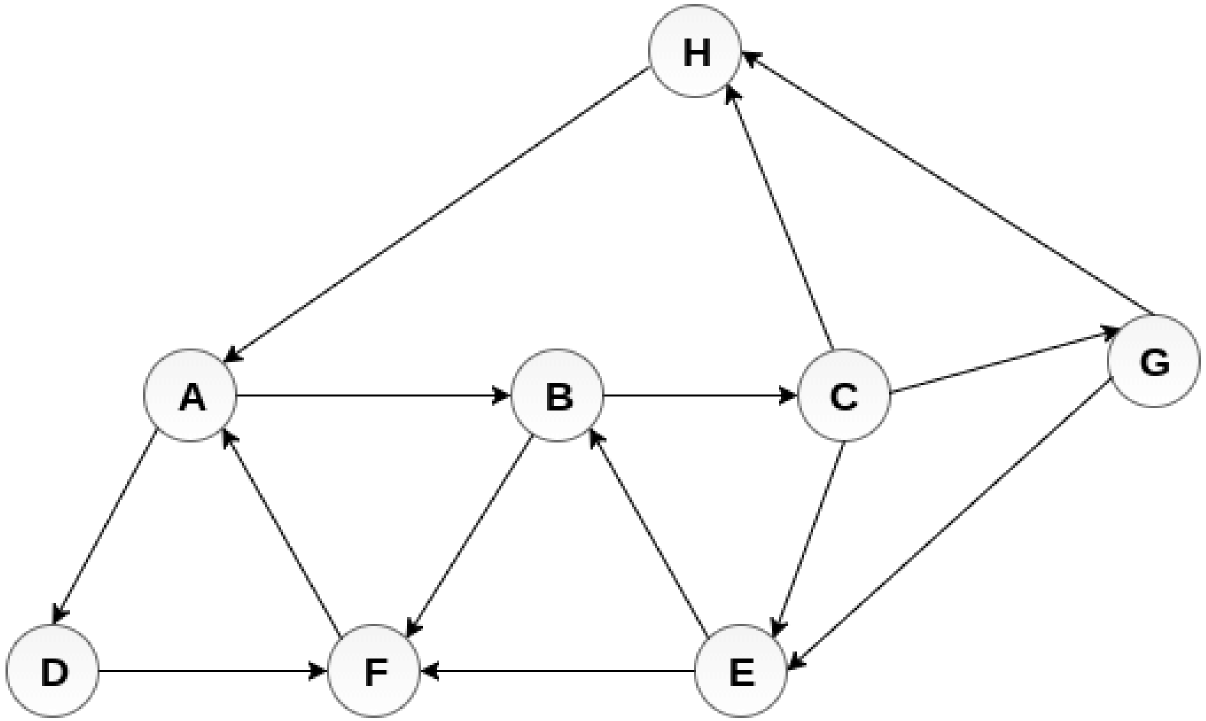
STATUS = 1) and set their

STATUS = 2 (waiting state)

[END OF LOOP]

Step 6: EXIT

DFS - Example

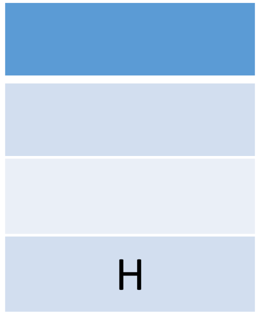


Adjacency Lists

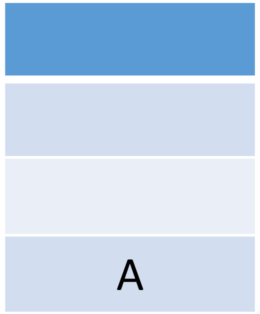
A : B, D
B : C, F
C : E, G, H
G : E, H
E : B, F
F : A
D : F
H : A

DFS - Example

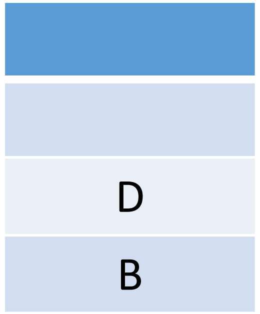
Step1: Push H onto the stack



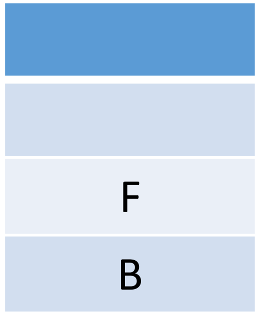
Step2: Pop H and Push all its adjacent



Step3: Pop A and Push all its adjacent

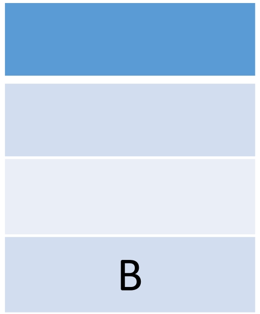


Step4: Pop D and Push all its adjacent

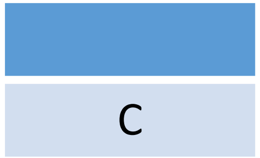


DFS - Example

Step5: Pop F and Push all its adjacent. But A is already visited. So no Push

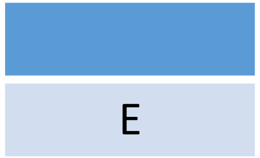


Step6: Pop B and Push all its adjacent. F is already visited



Step7: Pop C and Push all its adjacent. H is already visited

Step8: Pop G and Push all its adjacent. But B, F already visited. So no push



DFS - Example

Step9: Pop E and Push all its adjacent. But B, E is already visited. So no Push

Now the stack is empty. Stop the process. List all the popped elements

DFS = H → A → D → F → B → C → G → E

Breadth First Search (BFS) Algorithm

Algorithm

Step 1: SET STATUS = 1 (ready state) for each node in G

Step 2: Enqueue the starting node A and set its STATUS = 2 (waiting state)

Step 3: Repeat Steps 4 and 5 until QUEUE is empty

Step 4: Dequeue a node N. Process it and set its STATUS = 3 (processed state).

Step 5: Enqueue all the neighbours of N that are in the ready state

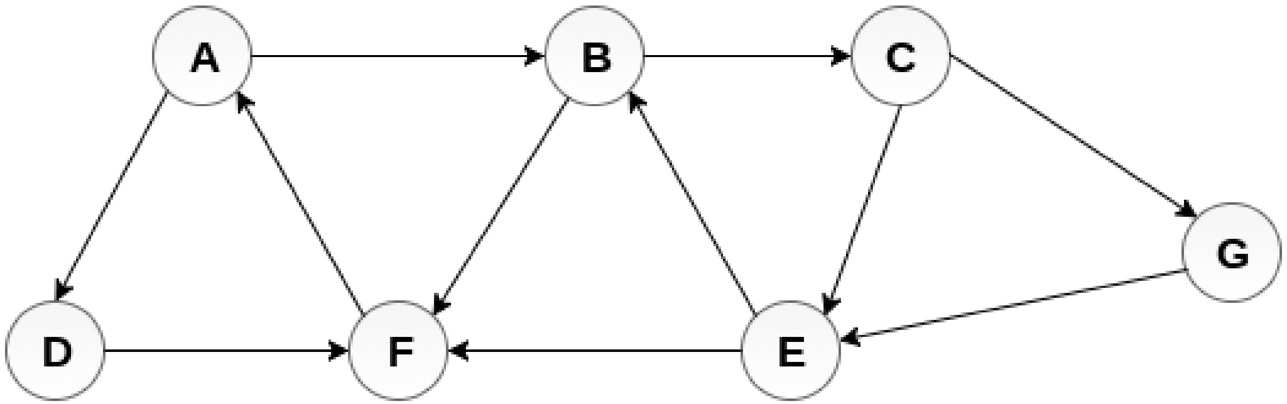
(whose STATUS = 1) and set

their STATUS = 2 (waiting state)

[END OF LOOP]

Step 6: EXIT

BFS - Example



Adjacency Lists

A : B, D

B : C, F

C : E, G

G : E

E : B, F

F : A

D : F

BFS - Example

Step1: Enque A onto the Queue Q1

Q1:

A				

Step2: Deque A and Enque all its adjacent

Q1:

B	D			

Step3: Deque B and Enque all its adjacent

Q1:

D	C	F		

BFS - Example

Step4: Deque D and Enque all its adjacent. But F is already inserted. So no Enque

Q1:

C	F			

Step5: Deque C and Enque all its adjacent

Q1:

F	E	G		

Step6: Deque F and Enque all its adjacent. But A already visited so no Enque.

Q1:

E	G			

BFS - Example

Step7: Deque E and Enque all its adjacent. But B, F is already inserted. So no Enque

Q1:

G				

Step8: Deque G and Enque all its adjacent. But E already visited

Q1:

Now queue is empty, Display all deque element is order

BFS: A, B, D, C, F, E, G

Summary

At the end of this session we learned about

- Graphs
- Breadth First Search - BFS
- Depth First Search - DFS