GRAPHS

Unit - 5

INTRODUCTION

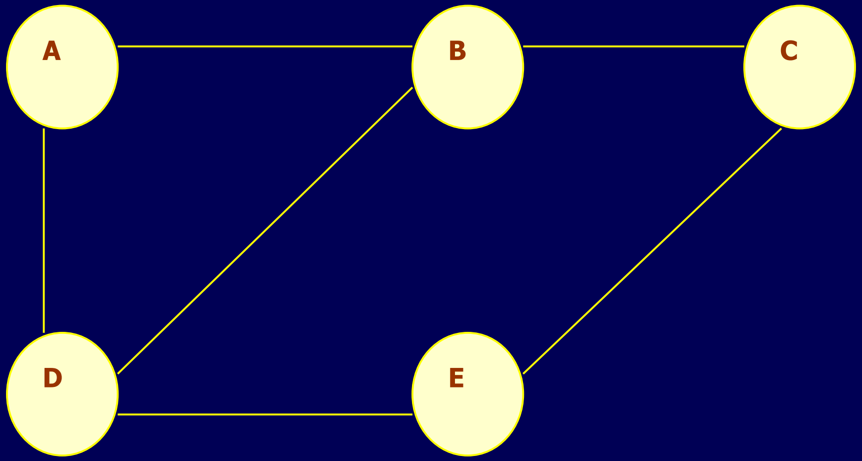
- A graph is an abstract data structure that is used to implement the graph concept from mathematics.
- A graph is basically, a collection of vertices (also called nodes) and edges that connect these vertices.
- A graph is often viewed as a generalization of the tree structure, where instead of a having a purely parent-to-child relationship between tree nodes, any kind of complex relationships between the nodes can be represented.

Why graphs are useful?

- Graphs are widely used to model any situation where entities or things are related to each other in pairs; for example, the following information can be represented by graphs:
- *City trees* in which the member nodes have an edge from one city to other cities
- *Transportation networks* in which nodes are airports, intersections, ports, etc. The edges can be airline flights, one-way roads, shipping routes, etc.

Definition

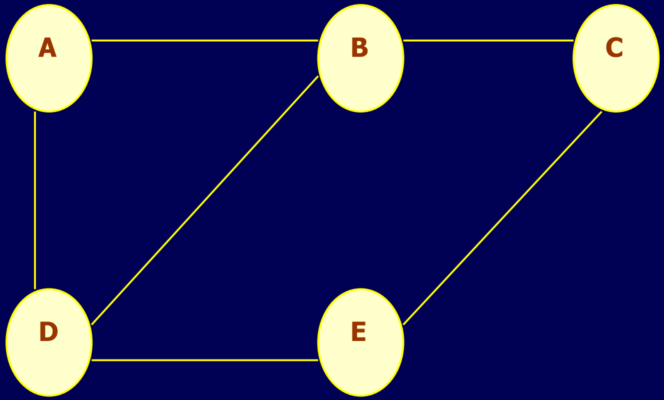
- A graph G is defined as an ordered set (V, E), where V(G) represent the set of vertices and E(G) represents the edges that connect the vertices.
- The figure given shows a graph with V(G) = { A, B, C, D and E} and E(G) = { (A, B), (B, C), (A, D), (B, D), (D, E), (C, E) }. Note that there are 5 vertices or nodes and 6 edges in the graph.



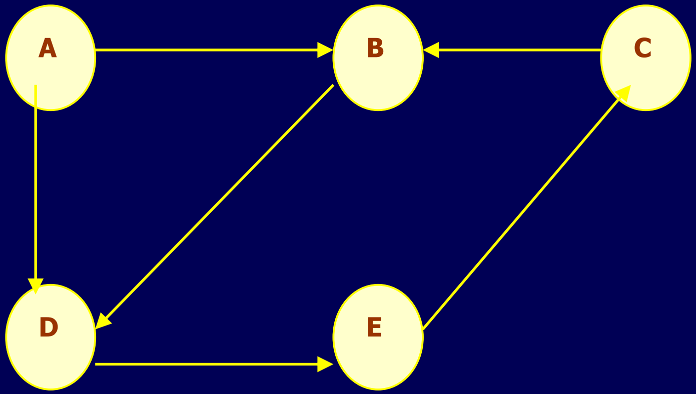
Definition

A graph can be directed or undirected.

In an undirected graph, the edges do not have any direction associated with them. That is, if an edge is drawn between nodes A and B, then the nodes can be traversed from A to B as well as from B to A.



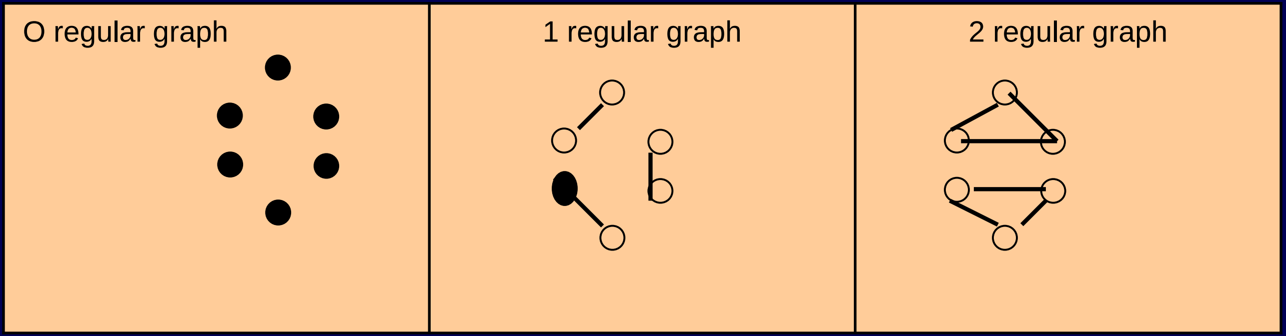
The above figure shows an undirected graph because it does not gives any information about the direction of the edges.



The given figure shows a directed graph. In a directed graph, edges form an ordered pair. If there is an edge from A to B, then there is a path from A to B but not from B to A. The edge (A, B) is said to initiate from node A (also known as initial node) and terminate at node B (terminal node).

Graph Terminology

- ***Adjacent Nodes or Neighbors:*** For every edge, e = (u, v) that connects nodes u and v; the nodes u and v are the end-points and are said to be the adjacent nodes or neighbors.
- ***Degree of a node:*** Degree of a node u, deg(u), is the total number of edges containing the node u. If deg(u) = 0, it means that u does not belong to any edge and such a node is known as an isolated node.
- ***Regular graph:*** Regular graph is a graph where each vertex has the same number of neighbors. That is every node has the same degree. A regular graph with vertices of degree *k* is called a *k*-regular graph or regular graph of degree *k*.



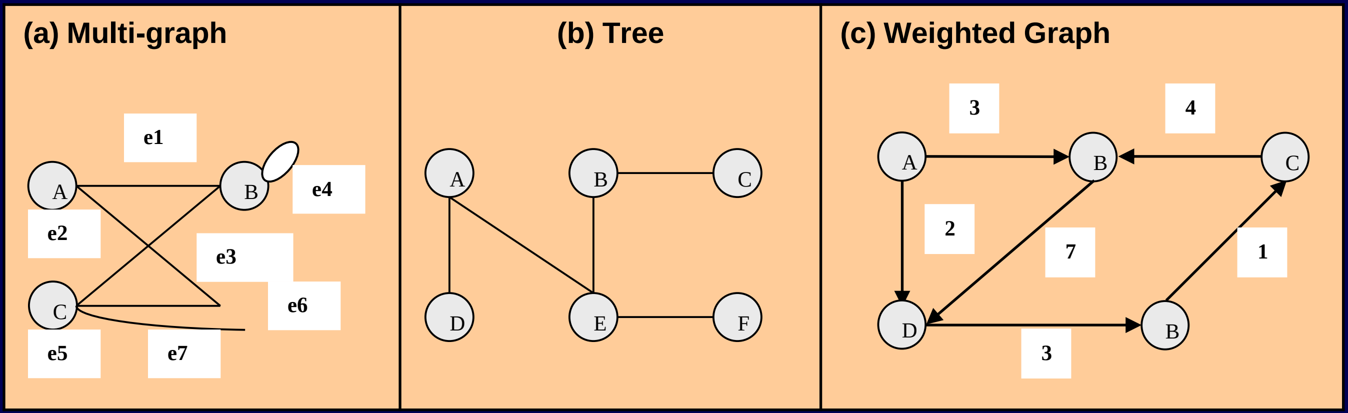
Graph Terminology

Path: A path P, written as P = {v0, v1, v2,….., vn), of length n from a node u to v is defined as a sequence of (n+1) nodes. Here, u = v0, v = vn and vi-1 is adjacent to vi for i = 1, 2, 3, …, n.

Closed path: A path P is known as a closed path if the edge has the same end-points. That is, if v0 = vn.

Simple path: A path P is known as a simple path if all the nodes in the path are distinct with an exception that v0 may be equal to vn. If v0 = vn, then the path is called a closed simple path.

- *Cycle:* A closed simple path with length 3 or more is known as a cycle. A cycle of length *k* is called a *k* – cycle.
- *Connected graph:* A graph in which there exists a path between any two of its nodes is called a connected graph. That is to say that there are no isolated nodes in a connected graph. A connected graph that does not have any cycle is called a tree.
- *Complete graph:* A graph G is said to be a complete, if all its nodes are fully connected, that is, there is a path from one node to every other node in the graph. A complete graph has n(n-1)/2 edges, where n is the number of nodes in G.
- *Labeled graph or weighted graph:* A graph is said to be labeled if every edge in the graph is assigned some data. In a weighted graph, the edges of the graph are assigned some weight or length. Weight of the edge, denoted by w(e) is a positive value which indicates the cost of traversing the edge.



Directed Graph

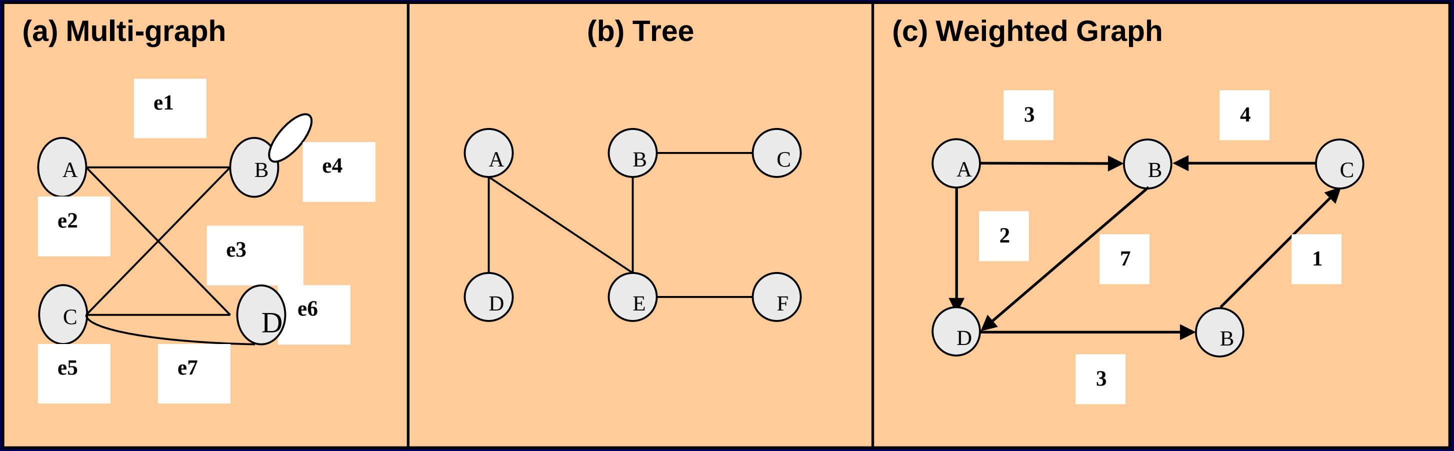
A directed graph G, also known as a digraph, is a graph in which every edge has a direction assigned to it. An edge of a directed graph is given as an ordered pair (u, v) of nodes in G. For an edge (u, v)-

- the edge begins at u and terminates at v
- u is known as the origin or initial point of e. Correspondingly, v is known as the destination or terminal point of e
- u is the predecessor of v. Correspondingly, v is the successor of u
- nodes u and v are adjacent to each other.

Terminology of a directed graph

- *Out-degree of a node:* The out degree of a node u, written as outdeg(u), is the number of edges that originate at u.
- *In-degree of a node*: The in degree of a node u, written as indeg(u), is the number of edges that terminate at u.
- *Degree of a node:* Degree of a node written as deg(u) is equal to the sum of in-degree and out-degree of that node. Therefore, deg(u) = indeg(u) + outdeg(u)
- *Source:* A node u is known as a source if it has a positive out-degree but an in-degree = 0.
- *Sink:* A node u is known as a sink if it has a positive in degree but a zero out-degree.

- *Multiple edges:* Distinct edges which connect the same end points are called multiple edges. That is, e = {u, v) and e' = (u, v) are known as multiple edges of G.
- *Loop:* An edge that has identical end-points is called a loop. That is, e = (u, u).
- *Multi- graph:* A graph with multiple edges and/or a loop is called a multi-graph.
- *Size of the graph:* The size of a graph is the total number of edges in it.

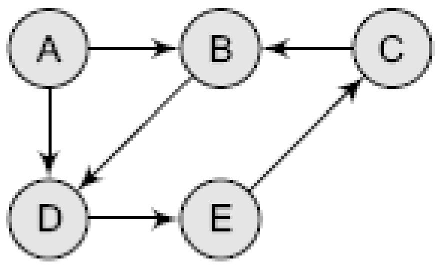


Adjacency Matrix Representation

- **An adjacency matrix is used to represent which nodes are adjacent to one another. By definition, we have learnt that, two nodes are said to be adjacent if there is an edge connecting them.**
- **In a directed graph G, if node v is adjacent to node u, then surely there is an edge from u to v. That is, if v is adjacent to u, we can get from u to v by traversing one edge. For any graph G having n nodes, the adjacency matrix will have dimensions of n X n.**
- **In an adjacency matrix, the rows and columns are labeled by graph vertices. An entry aij in the adjacency matrix will contain 1, if vertices vi and vj are adjacent to each other. However, if the nodes are not adjacent, aij will be set to zero.**

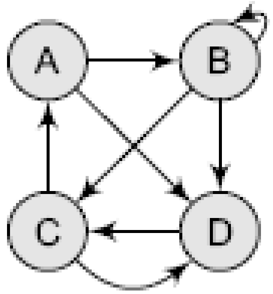
$$\begin{tikzcd} & \boxed{\textbf{1} \qquad \textbf{if } v_i \textbf{ is adjacent to } v_j\textbf{, that is there is an edge } (v_i, v_j)} \\ \boxed{aij} \arrow[ur] \arrow[dr] & \\ & \boxed{\textbf{0} \qquad \textbf{otherwise}} \end{tikzcd}$$

Since an adjacency matrix contains only 0s and 1s, it is called a bit matrix or a Boolean matrix. The entries in the matrix depend on the ordering of the nodes in G. therefore, a change in the order of nodes will result in a different adjacency matrix.



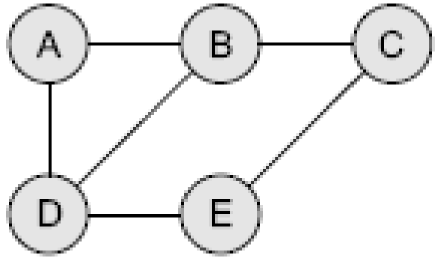
$$\begin{array}{c|ccccc} & A & B & C & D & E \\ \hline A & 0 & 0 & 0 & 1 & 0 \\ B & 0 & 0 & 0 & 1 & 0 \\ C & 0 & 1 & 0 & 0 & 0 \\ D & 0 & 0 & 0 & 0 & 1 \\ E & 0 & 0 & 1 & 0 & 0 \end{array}$$

(a) Directed graph



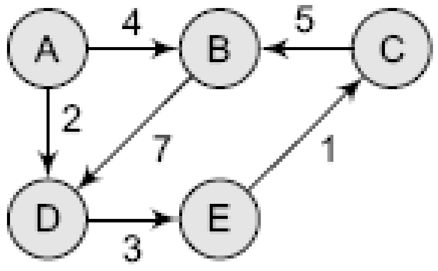
$$\begin{array}{c|cccc} & A & B & C & D \\ \hline A & 0 & 1 & 0 & 1 \\ B & 0 & 1 & 1 & 1 \\ C & 1 & 0 & 0 & 1 \\ D & 0 & 0 & 1 & 0 \end{array}$$

(b) Directed graph with loop



$$\begin{array}{c|ccccc} & A & B & C & D & E \\ \hline A & 0 & 1 & 0 & 1 & 0 \\ B & 1 & 0 & 1 & 1 & 0 \\ C & 0 & 1 & 0 & 0 & 1 \\ D & 1 & 1 & 0 & 0 & 1 \\ E & 0 & 0 & 1 & 1 & 0 \end{array}$$

(c) Undirected graph



$$\begin{array}{c|ccccc} & A & B & C & D & E \\ \hline A & 0 & 4 & 0 & 2 & 0 \\ B & 0 & 0 & 0 & 7 & 0 \\ C & 0 & 5 & 0 & 0 & 0 \\ D & 0 & 0 & 0 & 0 & 3 \\ E & 0 & 0 & 1 & 0 & 0 \end{array}$$

(d) Weighted graph

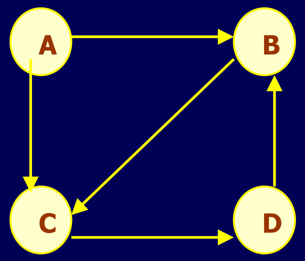
From adjacency matrix A1, we have learnt that an entry 1 in the ith row and jth column means that there exists a path of length 1 from vi to vj. Now consider, A2, A3 and A4

$$a_{ij}{}^{2} = \sum a_{ik}\, a_{kj}$$

Any entry $a_{ij} = 1$ if $a_{ik} = a_{kj} = 1$. That is, if there is an edge (v_i, v_k) and (v_k, v_j). This implies that there is a path from vi to vj of length 2.

Similarly, every entry in the ith row and jth column of A3 gives the number of paths of length 3 from node vi to vj.

In general terms, we can conclude that every entry in the ith row and jth column of An (where n is the number of nodes in the graph) gives the number of paths of length n from node vi to vj.



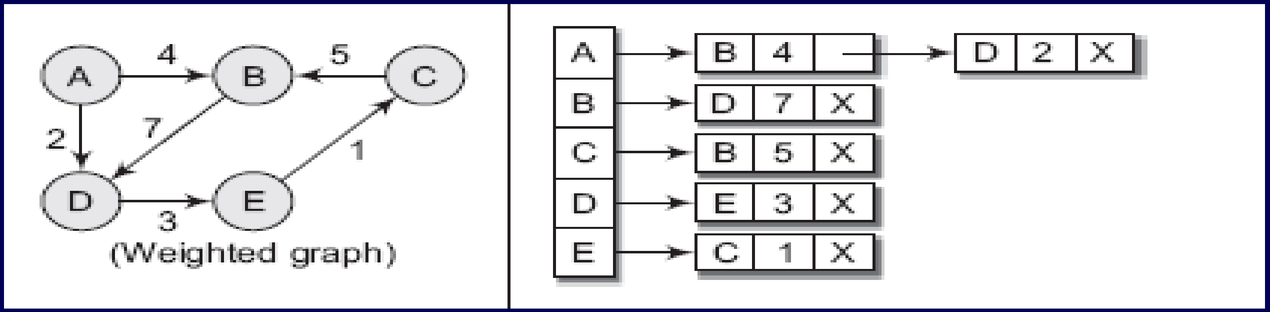
Adjacency List

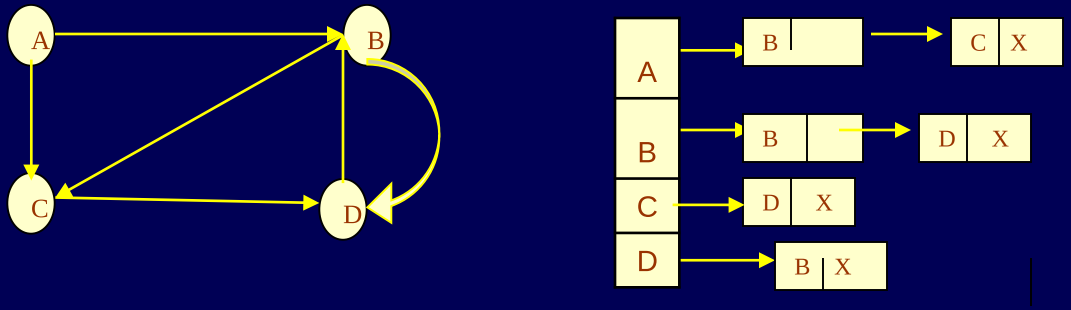
The adjacency list is another way in which graphs can be represented in computer's memory. This structure consists of a list of all nodes in G. Furthermore, every node is in turn linked to its own list that contains the names of all other nodes that are adjacent to itself.

The key advantage of using an adjacency list includes:

- It is easy to follow, and clearly shows the adjacent nodes of a particular node
- It is often used for storing graphs that have a small to moderate number of edges. That is an Adjacency list is preferred for representing sparse graphs in computer's memory; otherwise, an adjacency matrix is a good choice.

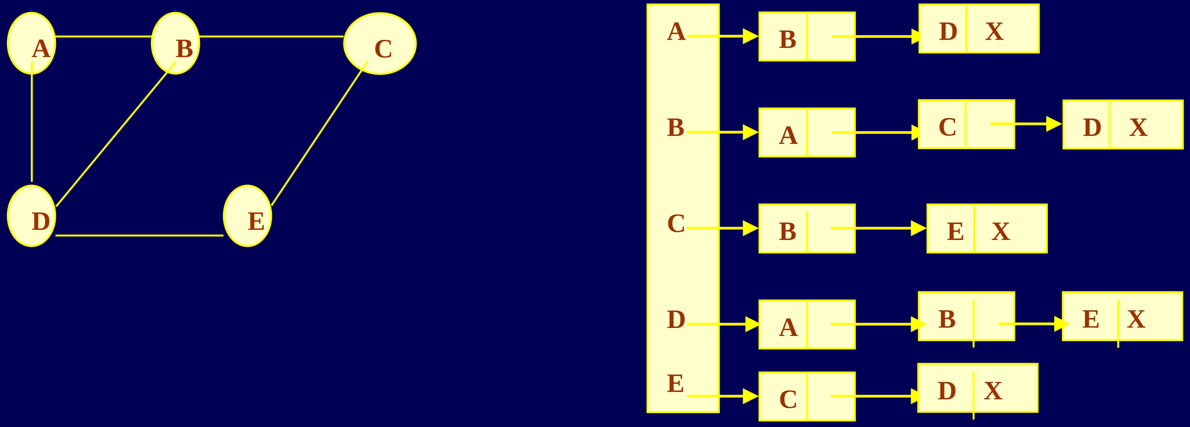
Adding new nodes in G is easy and straightforward when G is represented using an Adjacency list. Adding new nodes in an Adjacency matrix is a difficult task as size of the matrix needs to be changed and existing nodes may have to be reordered.





Graph G and its adjacency list

For a directed graph, the sum of lengths of all adjacency lists is equal to the number of edges in G. However, for an undirected graph, the sum of lengths of all adjacency lists is equal to twice the number of edges in G because an edge (u, v) means an edge from node u to v as well as an edge v to u. The adjacency list can also be modified to store weighted graphs.



GRAPH TRAVERSAL ALGORITHMS

By traversing a graph, we mean the method of examining the nodes and edges of the graph.

There are two standard methods of graph traversal which we will discuss in this section. These two methods are-

- Breadth first search : it uses a queue as an auxiliary data structure to store nodes for further processing,
- Depth first search : it uses a stack.

But both these algorithms will make use of a variable STATUS. During the execution of the algorithm, every node in the graph will have the variable STATUS set to 1, 2 or depending on its current state. Table I shows the value of status and its significance.

STATUS	STATE OF THE NODE	DESCRIPTION
1	Ready	The initial state of the node N
2	Waiting	Node N is placed on the queue or stack and waiting to be processed
3	Processed	Node N has been completely processed