Data Structures and Algorithms (CS-2001)

KALINGA INSTITUTE OF INDUSTRIAL TECHNOLOGY

School of Computer Engineering

4 Credit

Lecture Note

Chapter Contents

Sr #	Major and Detailed Coverage Area	Hrs
6	**Graphs**	5
	Graph ADT, Graph Operation – DFS, BFS	

Introduction

A graph is a representation of a set of objects where some pairs of objects are connected by links. The interconnected objects are represented by points termed as **vertices**, and the links that connect the vertices are called **edges**. Formally, a graph is a pair of sets **(V, E)**, where **V** is the set of vertices and **E** is the set of edges, connecting the pairs of vertices. Take a look at the following graph –
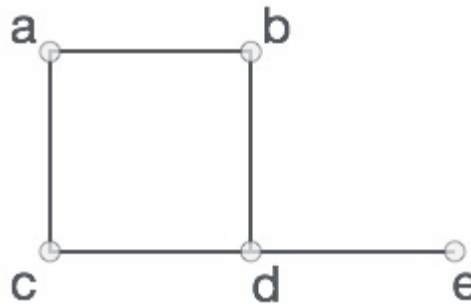
In the above graph,

V = {a, b, c, d, e}

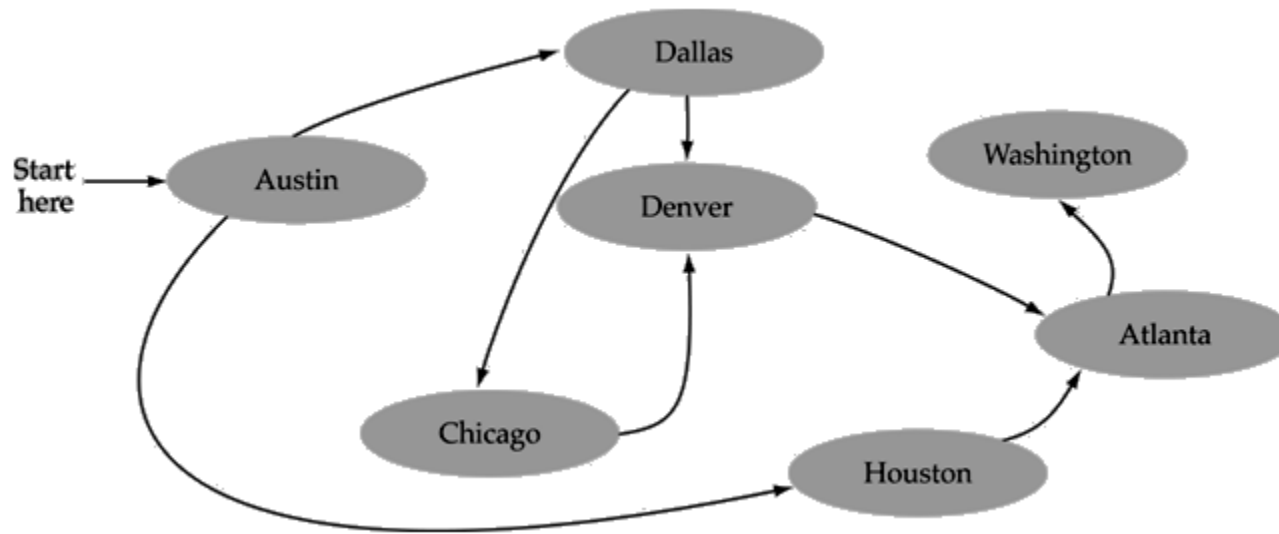E = {(a,b), (a,c), (b,d), (c,d), (d,e)}

Formal Definition

A graph G is defined as follows:

$$G = (V, E)$$

V(G): a finite, nonempty set of vertices

E(G): a set of edges (pairs of vertices)

Class Work

Define V(G) and E(G) of the above graph

School of Computer Engineering
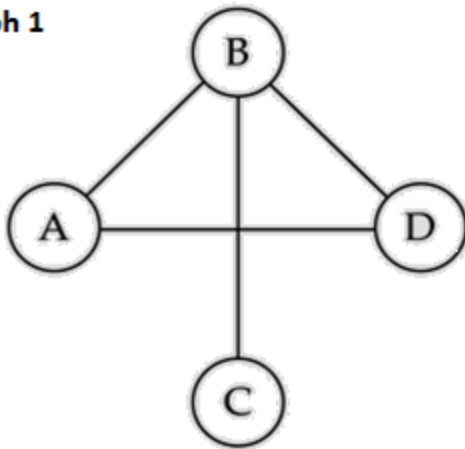
Directed and Undirected Graph

Undirected Graph

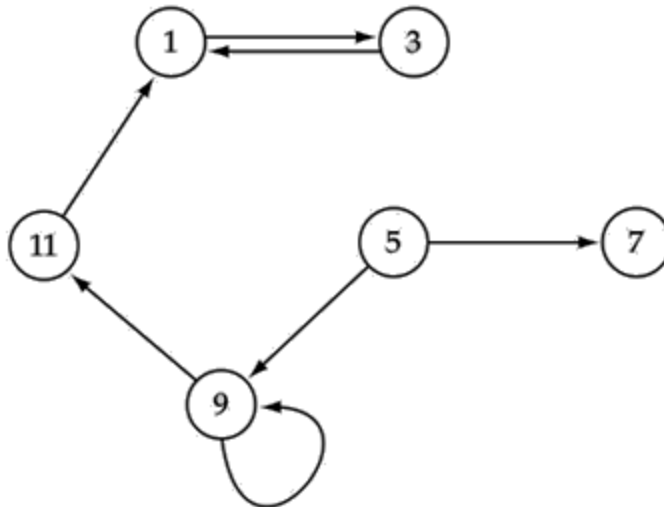When the edges in a graph have no direction, the graph is called undirected

Graph 1

V(Graph1) = { A, B, C, D }
E(Graph1) = { (A, B), (A, D), (B, C), (B, D) }

Directed Graph

When the edges in a graph have a direction, the graph is called directed (or digraph)

Graph2

V(Graph2) = { 1, 3, 5, 7, 9, 11 }
E(Graph2) = {(1,3),(3,1),(5,9),(9,11),(5,7), (9, 9), (11, 1) }

Trees vs. Graphs

|	Trees	Graphs
Loops	Tree is a special case of graph having no loops and no self-loops.	Graph can have loops as well as can have self-loops.
Root Node	There is exactly one root node and every child have only one parent.	There is no such concept of root node.
No. of edges	Always has n-1 edges.	No. of edges depend on the graph.
Model	Hierarchical model.	Network model.
Complexity	Less complex then graphs as having no cycles, no self-loops and still connected.	More complex in compare to trees as it can have cycles, loops etc.
Path	It is special form of graph i.e. **minimally connected graph** and having only one path between any two vertices.	There can be more than one path i.e. graph can have uni-directional or bi-directional paths (edges) between nodes.

Graph Terminology

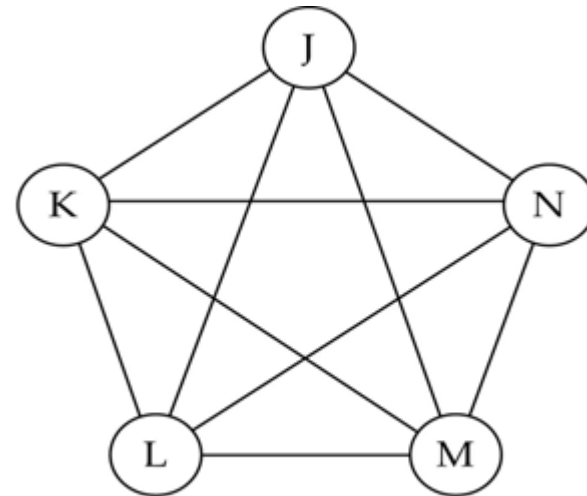❏ **Adjacent nodes:** two nodes are adjacent if they are connected by an edge

5 is adjacent to 7
7 is adjacent from 5

❏ **Path**: a sequence of vertices that connect two nodes in a graph

❏ **Complete graph**: a graph in which every vertex is directly connected to every other vertex

Complete directed graph.
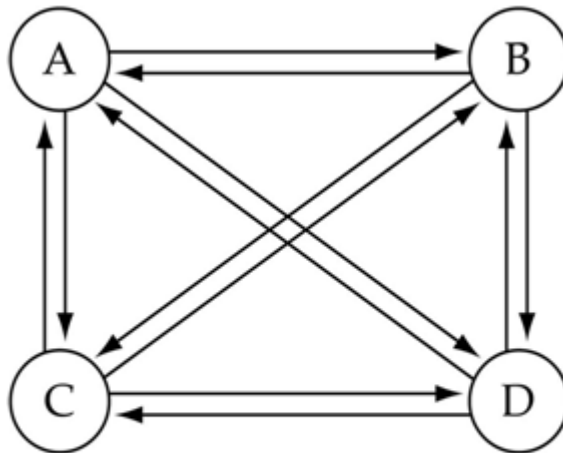
Complete undirected graph.

What is the number of edges in a complete directed graph with N vertices?

What is the number of edges in a complete undirected graph with N vertices?

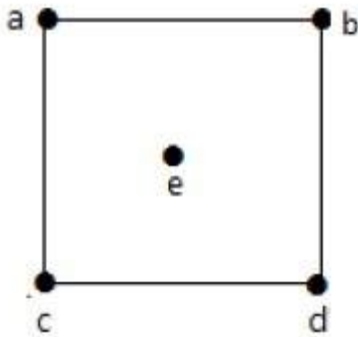**School of Computer Engineering**

Graph Terminology cont...

❑ **Loop:** In a graph, if an edge is drawn from vertex to itself, it is called a loop.

❑ **Degree of Vertex:** The degree of a graph vertex of a graph is the number of graph edges .

Degree of vertex in undirected graph

deg(a) = 2, deg(b) = 2, deg(c) = 2, deg(d) = 2, and deg(e) = 0

Degree of vertex in directed graph

Each vertex has an indegree and an outdegree. Indegree of vertex V is the **number of edges which are coming into the vertex** V. Outdegree of vertex V is the **number of edges which are going out from the vertex** V.
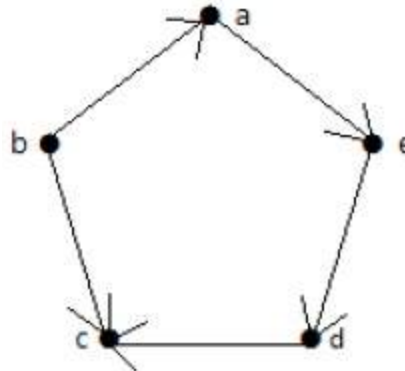
Vertex	Indegree	Outdegree
a	1	1
b	0	2
c	2	0
d	1	1
e	1	1

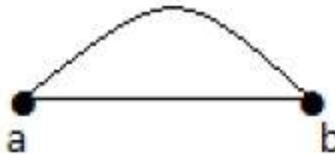**School of Computer Engineering**

Graph Terminology cont...

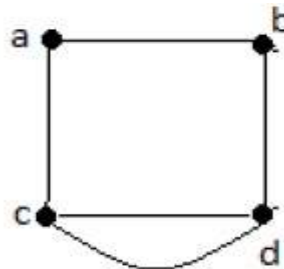❑ **Pendent Vertex:** A vertex with degree one is called a pendent vertex.

❑ **Isolated Vertex:** A vertex with degree zero is called an isolated vertex.

❑ **Parallel Edges:** In a graph, if a pair of vertices is connected by more than one edge, then those edges are called parallel edges.

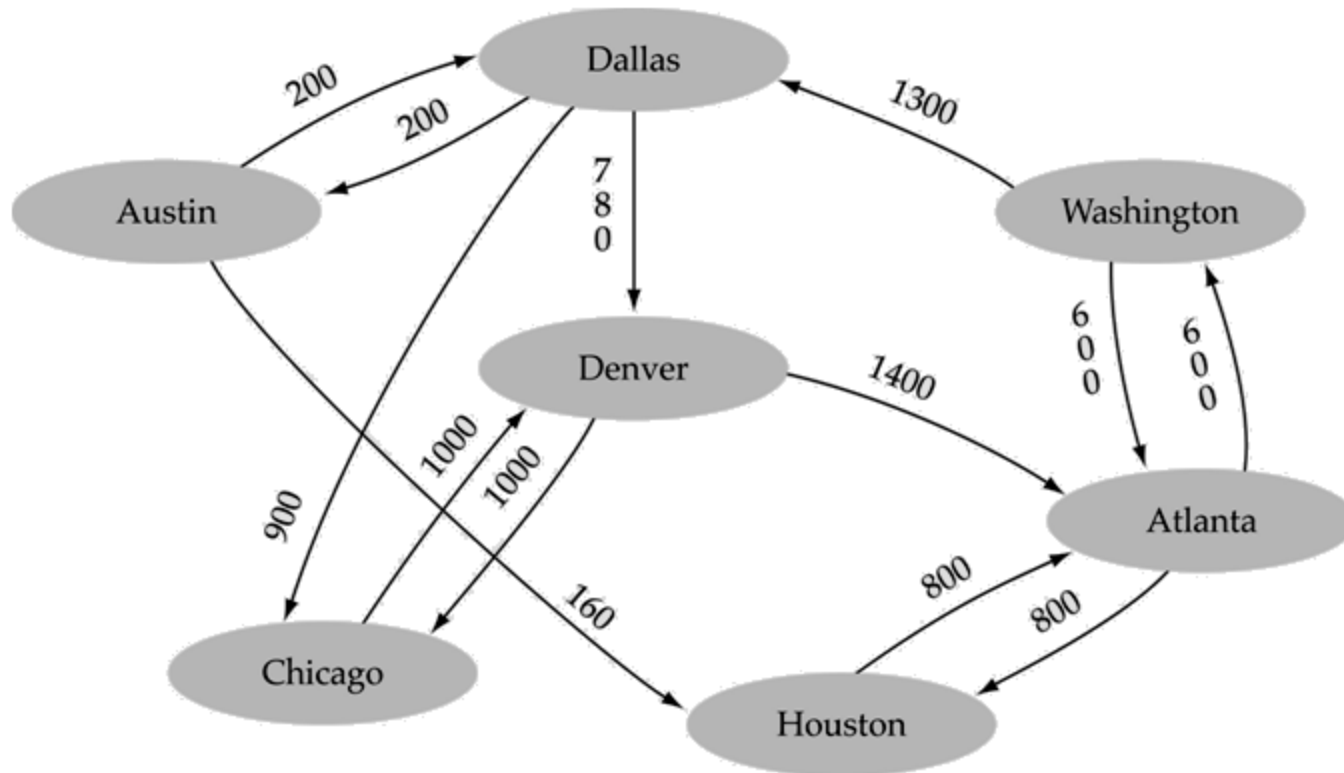❑ **Multi Graph:** A graph having parallel edges is known as a Multigraph.

Graph Terminology cont...

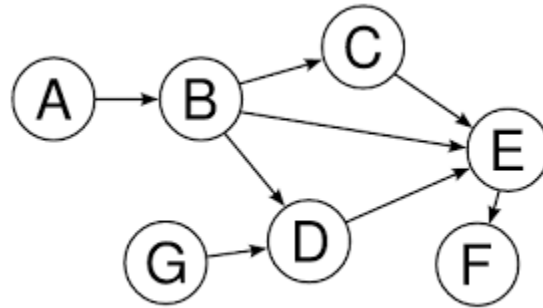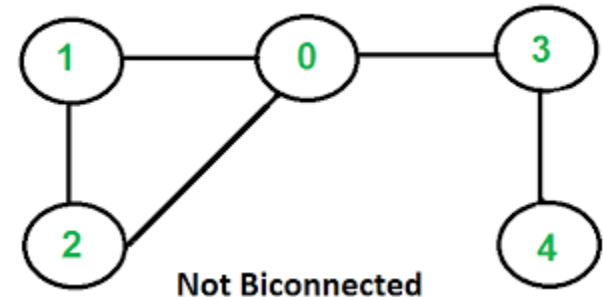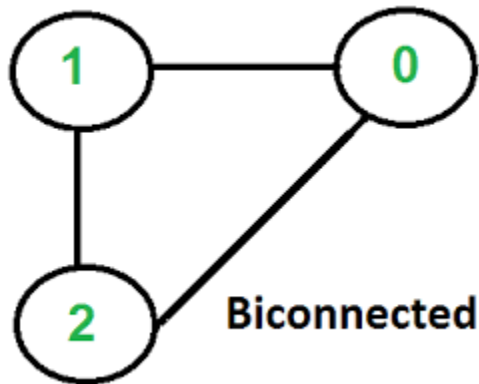**Weighted graph**: a graph in which each edge carries a value

Graph Terminology cont...

❏ **Directed Acyclic graph:** a direct graph with no cycles

❏ **Bi-connected graph:** It is a connected graph which cannot be broken down into any further pieces by deletion of any single vertex and incident edges.

School of Computer Engineering
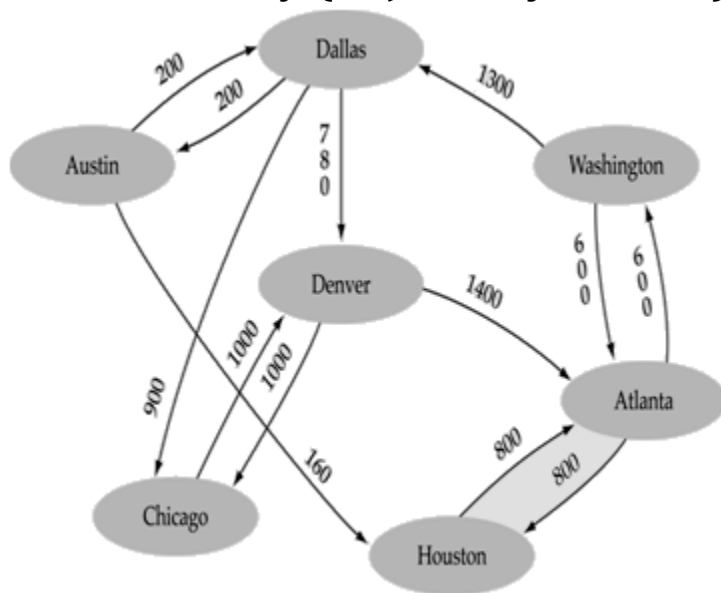
Representing Graphs in Memory

Let **G** be the graph. There are 2 ways of representing it in memory. The first way is called the **link representation of G** and the second way which uses arrays is called the **sequential representation of G**.

Sequential Representation (Array based)

❑ A 1D array is used to represent the vertices

❑ A 2D array (**adjacency matrix**) is used to represent the edges

(Array positions marked '•' are undefined)

Class Work

Let **G** be the **unweighted undirected** graph with 5 vertices. Represent it using Adjacency Matrix
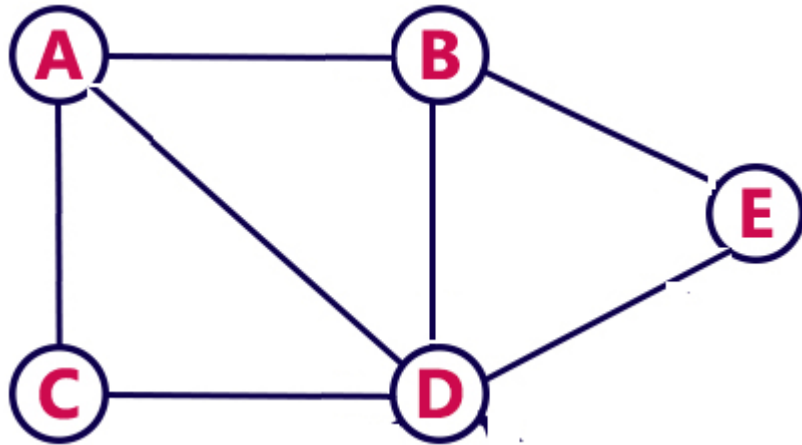
Let **G'** be the **unweighted directed graph** with 5 vertices. Represent it using Adjacency Matrix

Linked Representation of Graph

❑ A 1D array is used to represent the vertices

❑ A list is used for each vertex v which contains the vertices which are adjacent from v (**adjacency list**)
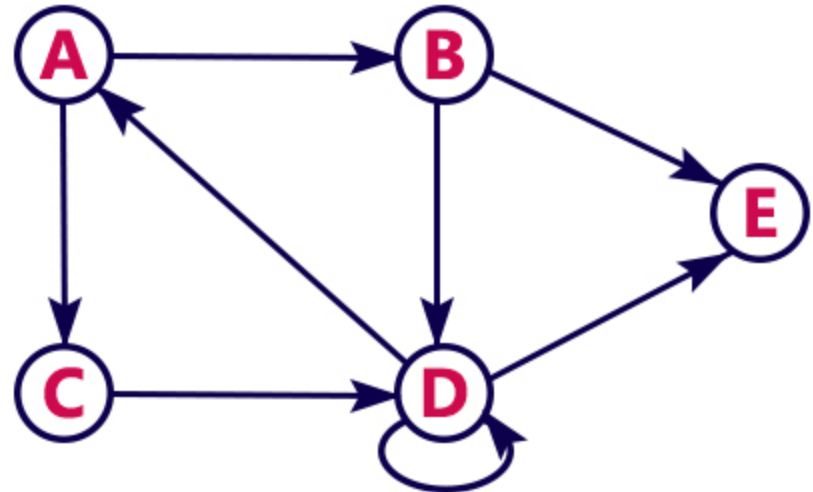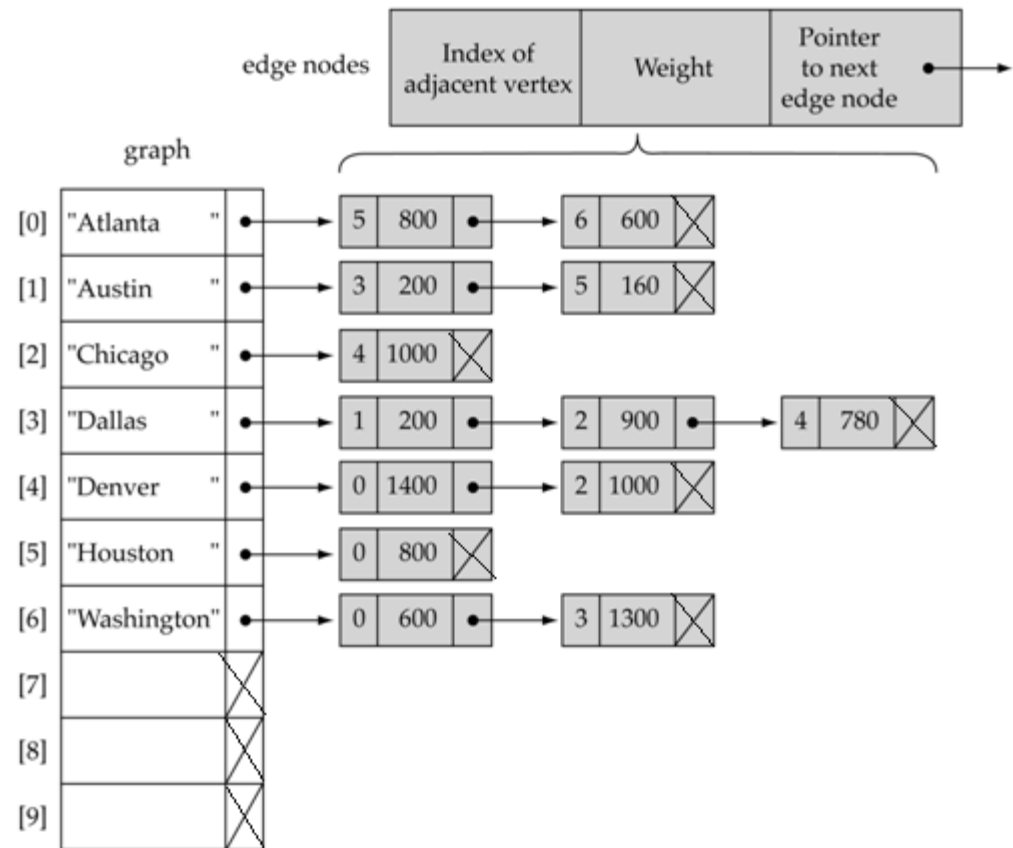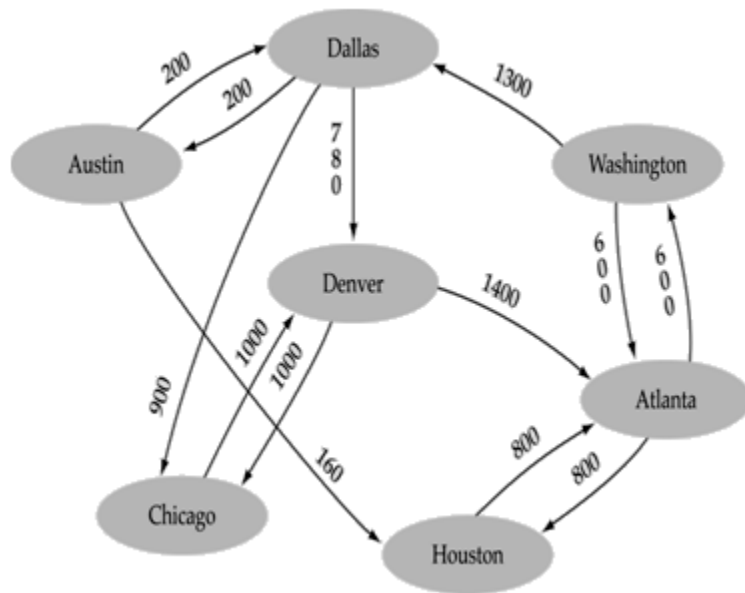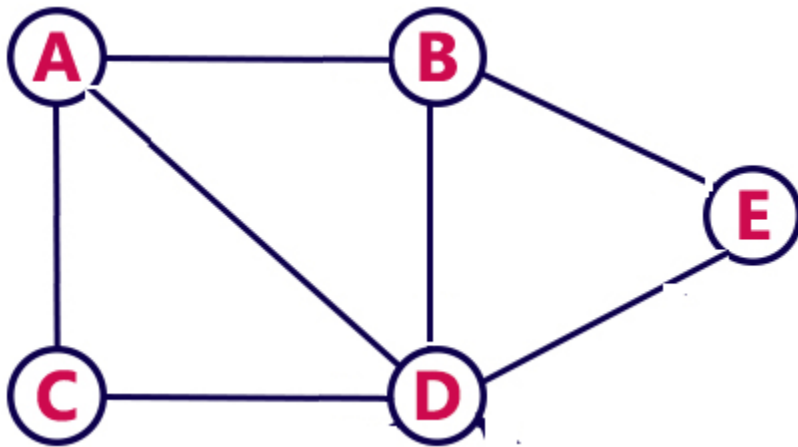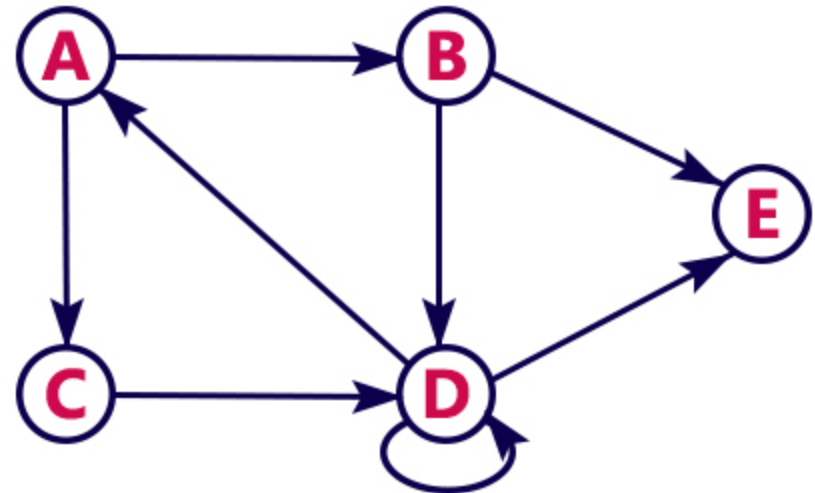
Class Work

Let **G** be the **unweighted undirected graph** with 5 vertices. Represent it using Adjacency List

Let **G′** be the **unweighted directed graph** with 5 vertices. Represent it using Adjacency List

Adjacency List vs. Adjacency Matrix

Adjacency Matrix

❑ Representation of graphs is very simple to implement.

❑ Wastes lot of memory space. Such matrices are found to be very sparse. This representation requires space for n^2 elements for a graph with n vertices.

❑ Degree of a vertex can easily be calculated by counting all **non-zero** entries in the corresponding row of the adjacency matrix.

❑ Presence of an edge between two vertices V_i and V_j can be checked in constant time. if(adj[i][j] == 1) then edge is present between vertices i and j, else edge is absent between vertices i and j

Adjacency List

❑ Very memory efficient when the graph has a large number of vertices but very few edges.

❑ For an undirected graph with n vertices and e edges, total number of nodes will be n + 2e. If e is large then due to overhead of maintaining pointers, adjacency list representation does not remain cost effective over adjacency matrix representation of a graph.

❑ Degree of a node in an undirected graph is given by the length of the corresponding linked list.

❑ Finding indegree of a directed graph represented using adjacency list will require O (e) comparisons. Lists pointed by all vertices must be examined to find the indegree of a node in a directed graph.

❑ Checking the existence of an edge between two vertices i and j is also time consuming. Linked list of vertex i must be searched for the vertex j.

School of Computer Engineering

Graph Theory Application

❑ **Electrical Engineering –** The concepts of graph theory is used extensively in designing circuit connections. The types or organization of connections are named as topologies. Some examples for topologies are star, bridge, series, and parallel topologies.

❑ **Computer Network** – The relationships among interconnected computers in the network follows the principles of graph theory.

❑ **Science** – The molecular structure and chemical structure of a substance, the DNA structure of an organism, etc., are represented by graphs.

❑ **Linguistics** – The parsing tree of a language and grammar of a language uses graphs.

❑ **General** – Routes between the cities can be represented using graphs. Depicting hierarchical ordered information such as family tree can be used as a special type of graph called tree.

❑ **Computer Science** – Graph theory is used for the study of algorithms. For example, Kruskal's Algorithm, Prim's Algorithm and Dijkstra's Algorithm

Graph Operation

❑ Search a vertex and the edge between two vertices

❑ Inserting a vertex

❑ Inserting edges between two vertices

❑ Deleting a vertex

❑ Deleting edges between two vertices

❑ Traversal

    ❑ Breadth-First Search

    ❑ Depth-First Search

Breadth-First Search

Breadth-First Search algorithm(BFS) traverses a graph in a breadth wards motion and uses a queue to remember to get the next vertex to start a search when a dead end occurs in any iteration.
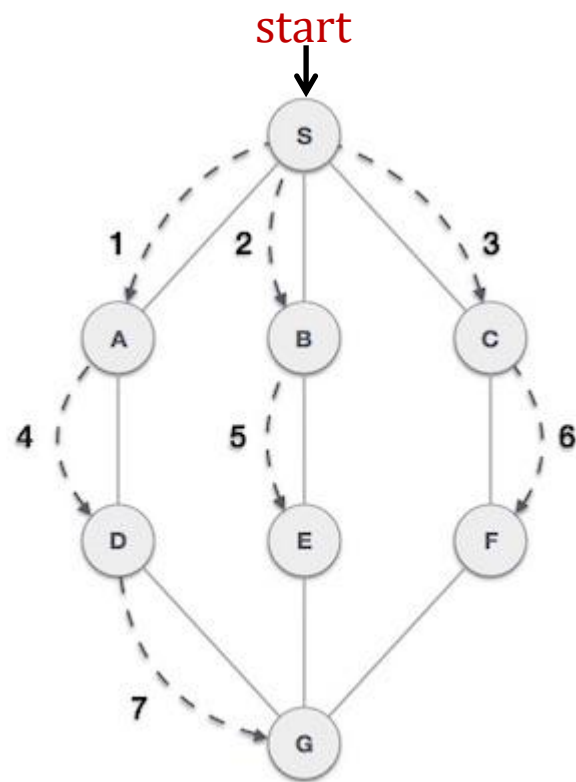
start

Pseudocode

Procedure **BFS**(graph G , node start)

begin

  Queue Q;

  char s, x;

  while (G has an unvisited node) do

    s = an unvisited node;

    visit(s);

    Enqueue(s, Q);

    while (Q is not empty) do

      x = Dequeue(Q);

      for (unvisited neighbor y of x) do

        visit(y);
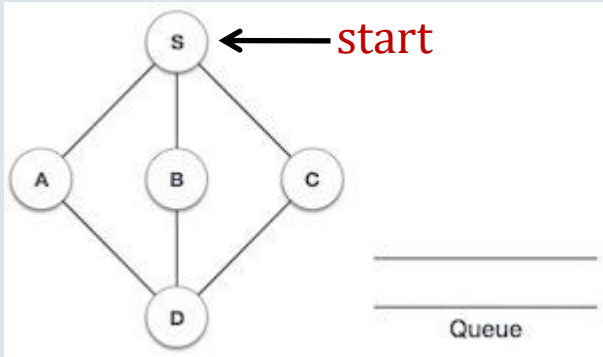
        Enqueue(y, Q);

      endfor

    endwhile
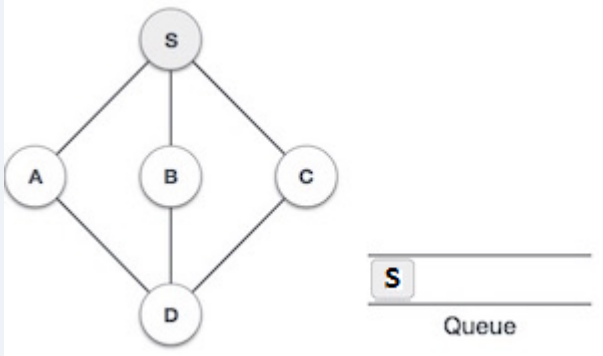
  endwhile

end

Breadth-First Search Example
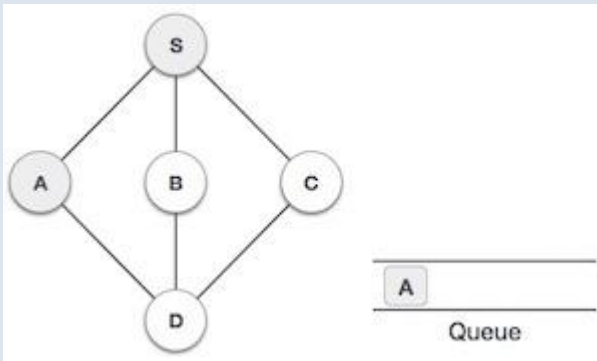
Step #	Traversal	Description	O/P Seq
1		Initialize the Queue	--
2		We start from visiting S (starting node), and mark it visited and put it onto the queue	S

School of Computer Engineering

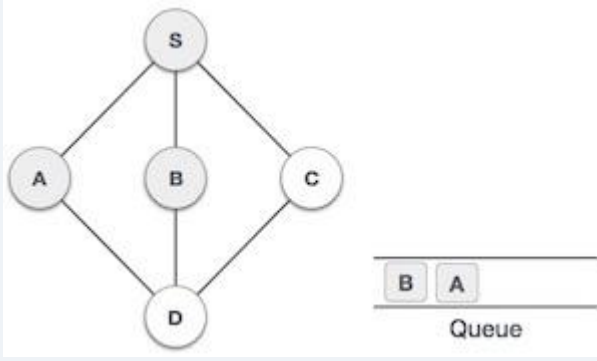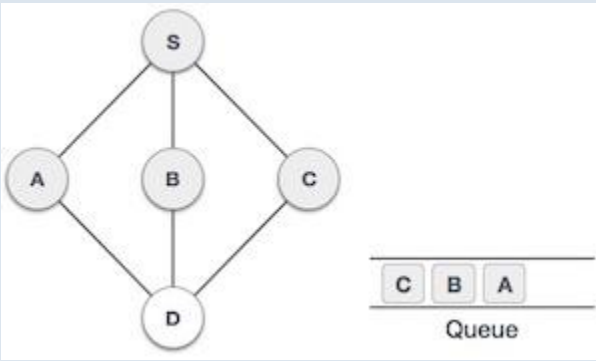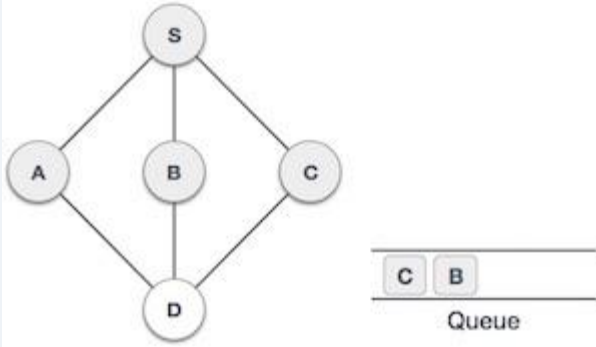| Step #	Traversal	Description	O/P Seq
3		Queue is not empty and S is dequed. We then see unvisited adjacent node from S. In this example, we have three nodes but alphabetically we choose A mark it visited and enqueue it.	S, A
4		Next unvisited adjacent node from S is B. We mark it visited and enqueue it.	S, A, B

Note: *comma used in Output Sequence (i.e. O/P Seq) is for illustration*

Breadth-First Search Example cont…
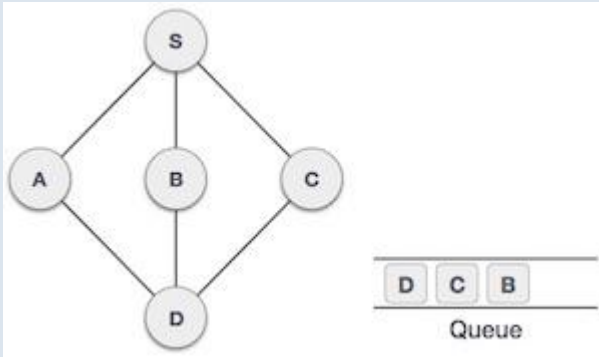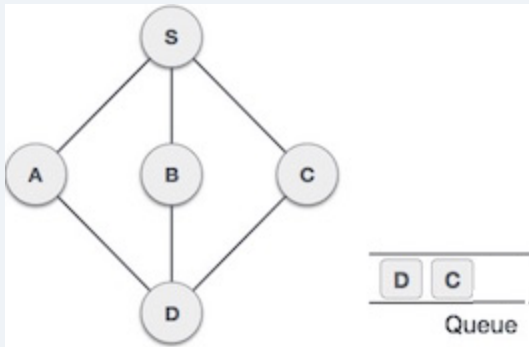
Step #	Traversal	Description	O/P Seq
5		Next unvisited adjacent node from S is C. We mark it visited and enqueue it.	S, A, B, C
6		Now **S** is left with no unvisited adjacent nodes. So we dequeue and find **A**.	S, A, B, C

Note: *comma used in Output Sequence (i.e. O/P Seq) is for illustration*

School of Computer Engineering

Breadth-First Search Example cont...

| Step #	Traversal	Description	O/P Seq
7		From A we have D as unvisited adjacent node. We mark it visited and enqueue it.	S, A, B, C, D
8		At this stage we are left with no unmarked (unvisited) nodes. But as per algorithm we keep on dequeuing in order to get all unvisited nodes. When the queue gets emptied the algorithm is over.	S, A, B, C, D

Note: *comma used in Output Sequence (i.e. O/P Seq) is for illustration*

School of Computer Engineering

Class Work

Display O/P sequence using BFS method for each Graph
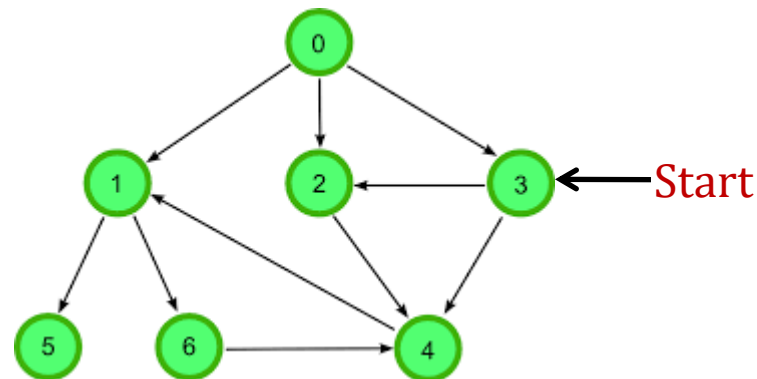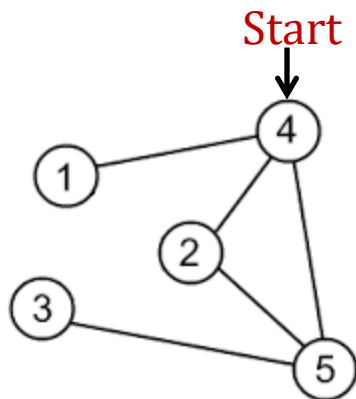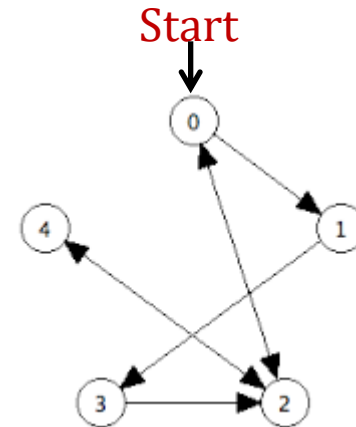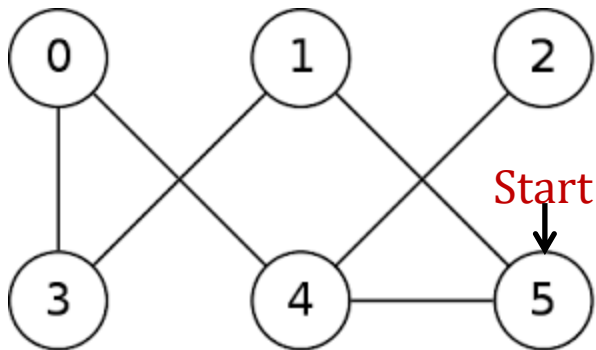
Depth-First Search

Depth-First Search algorithm(DFS) traverses a graph in a depth-ward motion and uses a stack to remember to get the next vertex to start a search when a dead end occurs in any iteration
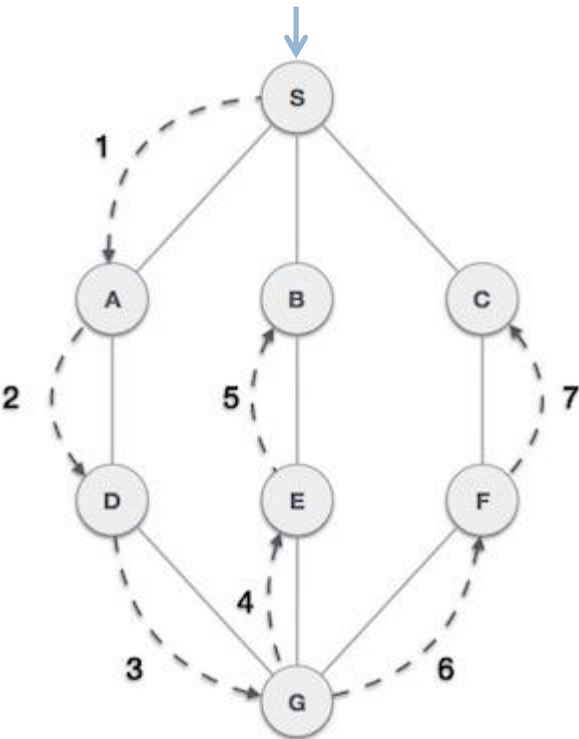
start

Pseudocode

Procedure **DFS**(graph G, node start)
begin
  Stack S;
  char s, x;
  while (G has an unvisited node) do
    s = an unvisited node;
    visit(s);
    push(s, S);
    while (S is not empty) do
      x = top(S);
      if (x has an unvisited neighbor y) then
        visit(y);
        push(y, S);
      else
        pop(S);
      endif
    endwhile
  endwhile
end
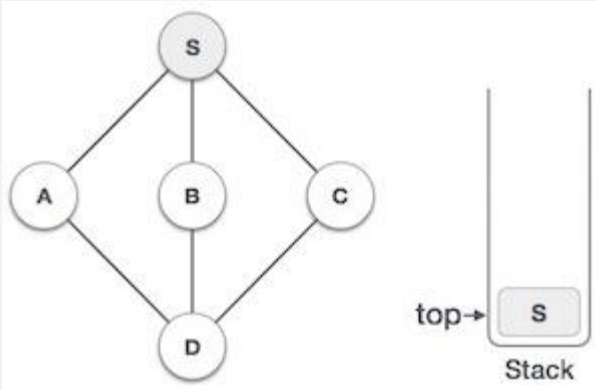
Depth-First Search Example
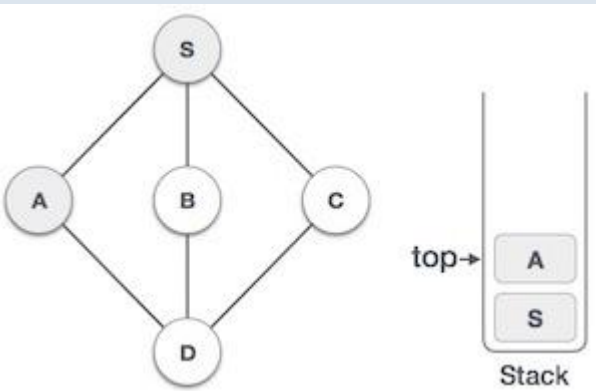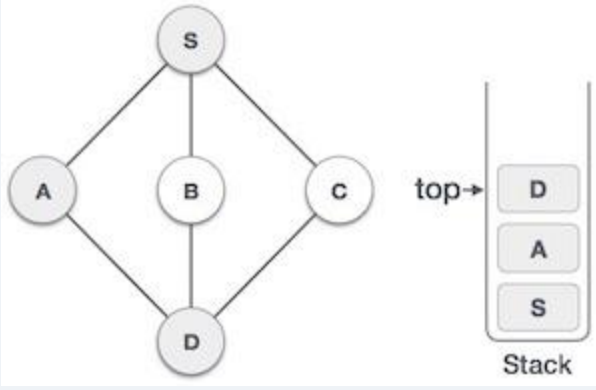
Step #	Traversal	Description	O/P Seq
1		Initialize the stack	--
2		Mark S as visited and put it onto the stack. Explore any unvisited adjacent node from S. We have three nodes and we can pick any of them. For this example, we shall take the node in alphabetical order.	S

Depth-First Search Example
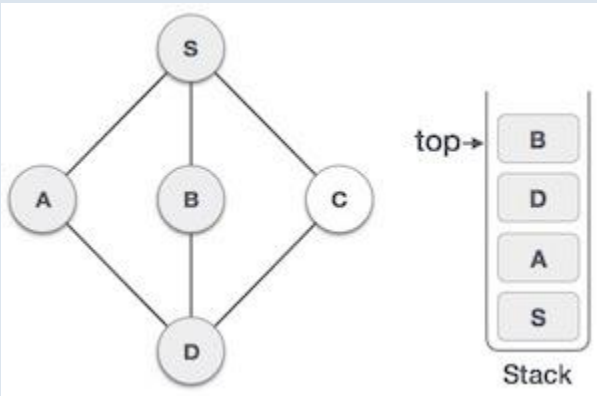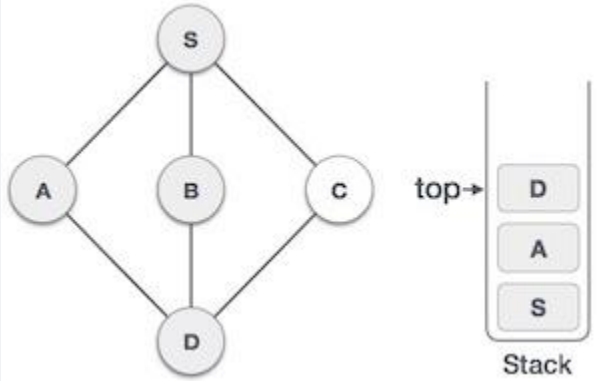
Step #	Traversal	Description	O/P Seq
3		Mark A as visited and put it onto the stack. Explore any unvisited adjacent node from A. Both S and D are adjacent to A but we are concerned for unvisited nodes only.	S, A
4		Visit D and mark it visited and put onto the stack. Here we have B and C nodes which are adjacent to D and both are unvisited. But we shall again choose in alphabetical order.	S, A, D

Note: *comma used in Output Sequence (i.e. O/P Seq) is for illustration*
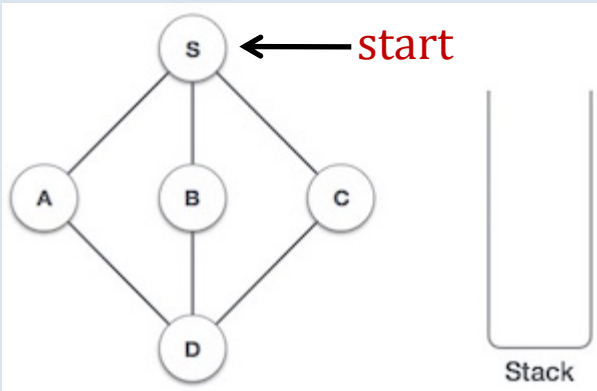
School of Computer Engineering

Depth-First Search Example

Step #	Traversal	Description	O/P Seq
5		We choose B, mark it visited and put onto stack. Here B does not have any unvisited adjacent node. So we pop B from the stack.	S, A, D, B
6		We check stack top for return to previous node and check if it has any unvisited nodes. Here, we find D to be on the top of stack.	S, A, D, B

Note: *comma used in Output Sequence (i.e. O/P Seq) is for illustration*

School of Computer Engineering

Depth-First Search Example

| Step #	Traversal	Description	O/P Seq
7		Only unvisited adjacent node is from D is C now. So we visit C, mark it visited and put it onto the stack.	S, A, D, B, C
8		As C does not have any unvisited adjacent node so we keep popping the stack until we find a node which has unvisited adjacent node. In this case, there's none and we keep popping until stack is empty.	S, A, D, B, C

Note: *comma used in Output Sequence (i.e. O/P Seq) is for illustration*

School of Computer Engineering

Class Work

Display O/P sequence using DFS method for each Graph
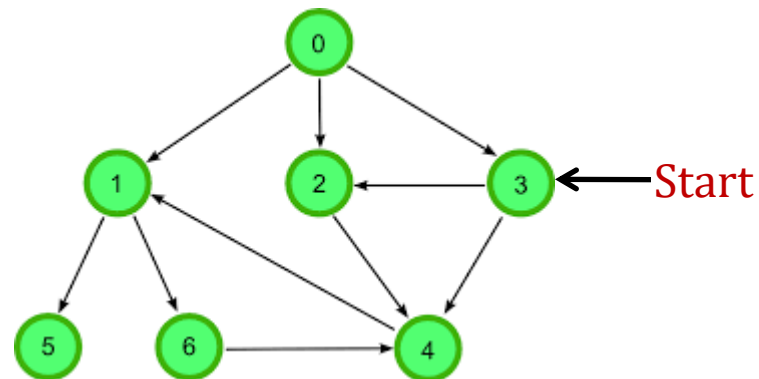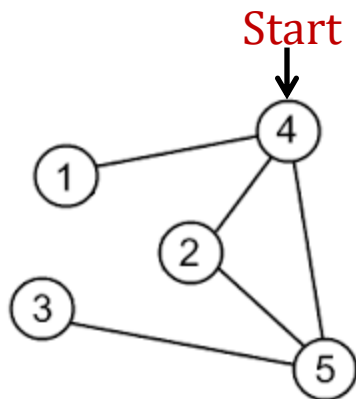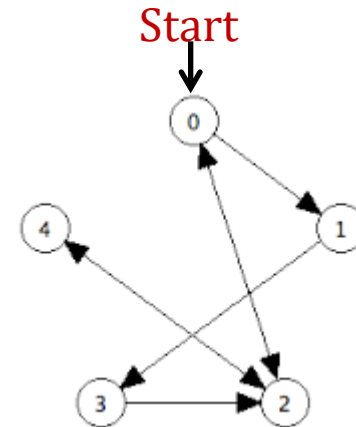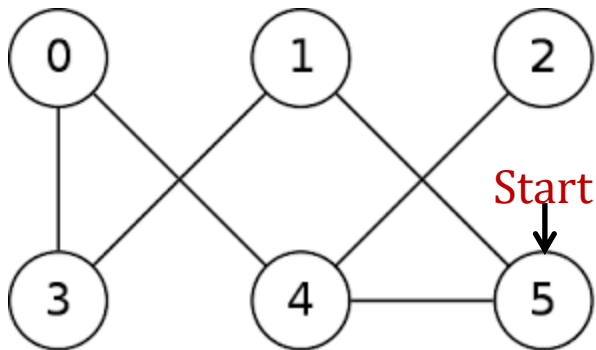
Assignments

Write C code snippet (function) to
1. Represent the graph using Adjacency Matrix
2. Represent the graph using Adjacency List
3. Find if there is a path between pair of vertices in an undirected and directed graph
4. Find the number of isolated vertices in an undirected graph.
5. Check if a given undirected graph is a tree (an undirected graph is a tree is there is cycle and is connected)
6. Find a mother vertex in a graph. Mother is a vertex v such that all other vertices in G can be reached by a path from v. There can be more than one mother vertices in a graph. Need to output anyone of them.
7. Detect a cycle in directed graph
8. Find the bridges in the undirected graph. An edge in an undirected connected graph is a bridge if removing it disconnects the graph

THANK YOU!

Home Work (HW)

❑ WAP using DFS method for directed graph traversal
❑ WAP using BFS method for directed graph traversal
❑ Write an algorithm that traverse the graph. If the graph is connected, it should return true and should return false if the graph is disjoint
❑ Write Pseudo code for the following undirected graph. Assume graph is represented in memory in adjacency matrix format
  ❑ Initialize the graph with new node named 'A' and 'B'
  ❑ Add edge between 'A' and 'B'
  ❑ Insert a new nodes i.e. 'C', 'D' and 'E' to the Graph
  ❑ Insert edge between 'C' & 'D' and 'A' & 'D' and 'B' & 'C'
  ❑ DFS
  ❑ BFS
  ❑ Search for the node 'B' and display the position of the node and its connecting nodes
  ❑ Delete the edge between 'C' & 'D'
  ❑ Delete the node 'B'
  ❑ Search for the node 'D' and display the position of the node and its connecting nodes

School of Computer Engineering

Home Work (HW) cont…

- ❑ Write a procedure which determines whether or not G is an undirected graph
- ❑ Write a procedure to
    - ❑ Print the list of successors of the given node N in the undirected graph G
    - ❑ Print the list of successors of the given node N in the directed graph G
    - ❑ Print the list of predecessors of the given node N in the undirected graph G'
    - ❑ Print the list of predecessors of the given node N in the directed graph G'
- ❑ Show that the sum of the degrees of the vertices of an undirected graph is twice the number of edges

Supplementary Reading

- ❑ http://www.geeksforgeeks.org/graph-data-structure-and-algorithms/
- ❑ http://www.algolist.net/Data_structures/Graph
- ❑ http://nptel.ac.in/courses/106102064/24
- ❑ https://www.youtube.com/watch?v=9zpSs845wf8
- ❑ https://www.tutorialspoint.com/data_structures_algorithms/graph_data_structure.htm

School of Computer Engineering

FAQ

Describe Graphs with example.

Graphs are widely-used structure in computer science and different computer applications. Graphs mean to store and analyze metadata, the connections, which present in data. For instance, consider cities in a country. Road network, which connects them, can be represented as a graph and then analyzed. We can examine, if one city can be reached from another one or find the shortest route between two cities. There are two important sets of objects, which specify graph and its structure. First set is V, which is called vertex-set. In the example with road network cities are vertices. Each vertex can be drawn as a circle with vertex's number inside. Next important set is E, which is called edge-set. E is a subset of V x V. Simply speaking, each edge connects two vertices, including a case, when a vertex is connected to itself (such an edge is called a loop). All graphs are divided into two big groups: directed and undirected graphs. The difference is that edges in directed graphs, called arcs, have a direction. These kinds of graphs have much in common with each other, but significant differences are also present. We will accentuate which kind of graphs is considered in the particular algorithm description. Edge can be drawn as a line. If a graph is directed, each line has an arrow.
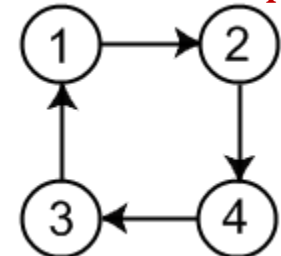
Vertices
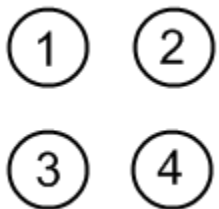
Undirected Graph

Directed Graph

School of Computer Engineering

FAQ
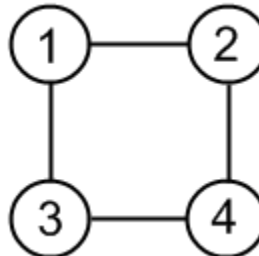
BFS:

BFS is a traversing algorithm where you should start traversing from a selected node (source or starting node) and traverse the graph layer-wise thus exploring the neighbor nodes (nodes which are directly connected to source node). You must then move towards the next-level neighbor nodes.

As the name BFS suggests, you are required to traverse the graph breadth-wise.

It uses a queue to remember to get the next vertex to start a search when a dead end occurs in any iteration

DFS:

The DFS is a recursive algorithm uses the idea of backtracking. It involves exhaustive searches of all the nodes by going ahead, if possible, else by backtracking.

Here, the word backtrack means that when you are moving forward and there are no more nodes along the current path, you move backwards on the same path to find nodes to traverse. All the nodes will be visited on the current path till all the unvisited nodes have been traversed after which the next path will be selected.

This recursive nature of DFS can be implemented using stacks.

School of Computer Engineering