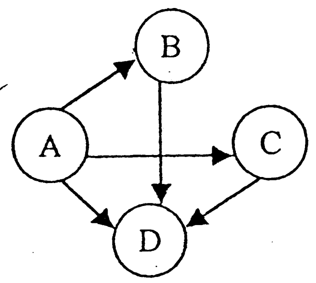
GRAPHS

A graph G consists of a finite non-empty set V of vertices & a set E of edges.
G = (V, E)



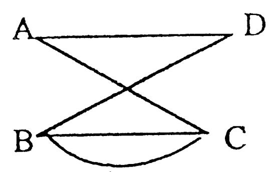
V={A, B, C, D}
E={(A, B)(A, C)(A, D)(B, D)(C, D)}

Directed graph (digraph)

A graph is said to be directed graph in which every edge is directed.
E={<A, B><A, C><A, D><B, D><C, D>}

Multi graph

A graph is said to be multi graph if there exist a loop or multiple edges in a given graph.



Multiple edges: -Distinct edges e &e' are called **multiple** edges if they connect the same end points ,that is if e=[u ,v] & e'=[u, v]

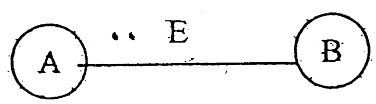
Loops: - An edge e is called a loop if it has **Identical** endpoints, that is if e=[u, v]

Simple graph

A graph without loops & parallel edges.

Incidence

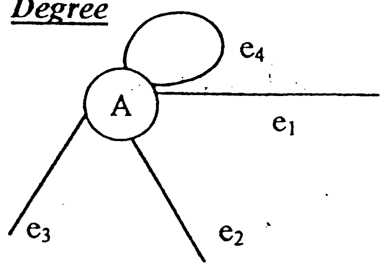
Incidence means when a vertex V is an end vertex of some edge E ,V & E are said to be incident with each other.



B is incident on E.

②

Degree



The no of edges incident on a vertex V is called the degree d(V) of vertex V. (For loops the edge is to be counted twice) So, the degree of A is 5.

In degree

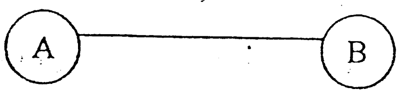
In degree of a vertex V_i is the no of edges with V_i as their end vertex.

Out degree

Out degree of a vertex V_i is the no of edges whose start vertex is V_i.

Adjacent

Two vertices are said to be adjacent if they are end vertices of the same edge.



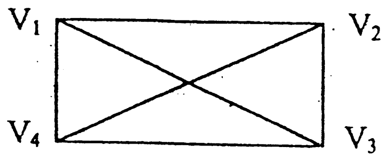
A is adjacent to B.
B is adjacent to A.

Theorem

*The maximum number of edges in any simple undirected graph with **n** vertices is*

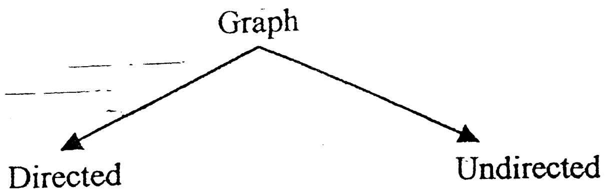
$$^{N}C_2 = n\,(n-1)/2$$

Complete graph



A graph will be complete if there is an edge between any two nodes. The no of edges in a complete graph having n nodes are

$= (n-1)+(n-2)+\ldots\ldots\ldots\ldots+2+1$
$= n(n-1)/2$

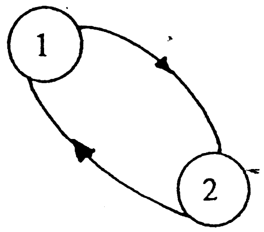


Directed

A directed graph is a graph in which all the edges have specific direction.

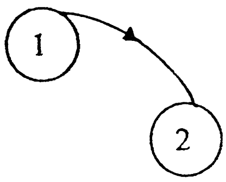
Connected Graph

Strongly connected A directed graph is strongly connected if for every pair of vertices Vi & Vj, there is a directed path form Vi to Vj & from Vj to Vi.



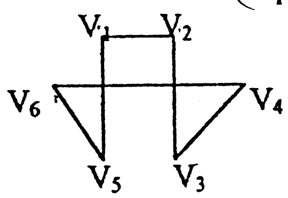
2 is adjacent to 1
1 is adjacent to 2

Weakly connected A directed graph is weakly connected if there is a directed path form Vi to Vj or from Vj to Vi.



Path

A path is a collection of vertex $V_1, V_2, \ldots\ldots V_n$. Where $(V_1, V_2), (V_2, V_3) \ldots\ldots (V_{n-1}, V_n)$ are the edges of the graph.



Length of a path

The length of the path is the number of edges in the path.

Simple path

A simple path is a path in which all nonterminal nodes cannot be repeated.

(4)

Cycle

A cycle is a simple path so that the terminal node coincides with the starting node

Subgraph & connected components

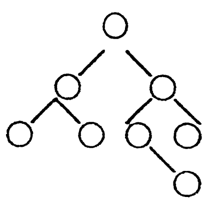
A *subgraph* of G is G' such that V (G')<=V (G) & E (G')<=E (G)

A *connected component* of a graph is a maximal connected subgraph.

Weighted graph

A weighted graph is a graph in which edges are assigned some weight or value.(such graph often known as a network.)

Tree



A tree is a connected graph, which is acycl i.e.-having no cycle.

Theorem

There is one and only one path between any two vertices in a tree T.

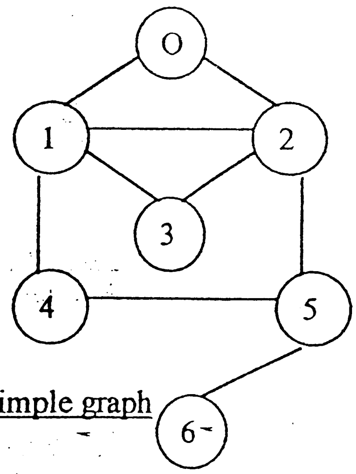
Proof (By contradiction)

Let a & b any two vertices, by combining this two we have cycle. But a tree cannot have a cycle. So, assuming two path is contradictory .so any two verti of the tree there is only one path.

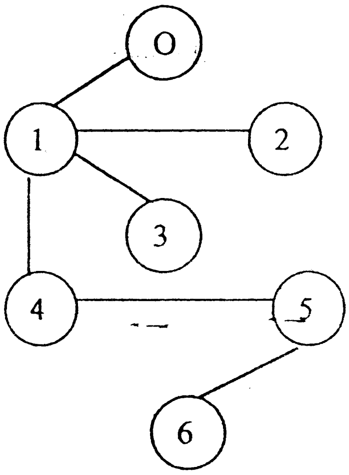
Spanning Tree

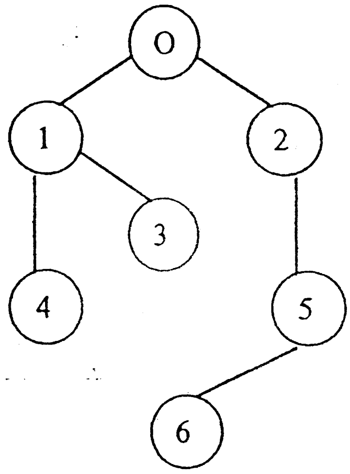
A subgraph of a graph G=(V,E) which is a tree containing all vertices of "V" is called a spanning tree of "G".

Note:-There will not be a unique spanning tree of a graph "G"



A simple graph





Minimal Spanning Tree

A spanning tree T of G where the sum of weights of all edges in T is the minimum is called the minimal cost spanning tree or the minimal spanning tree of G.(It is not necessarily unique)

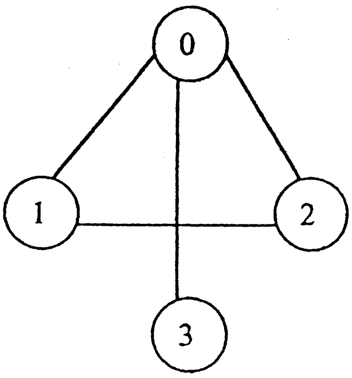
Graph representations

The most commonly used representation for graphs are

I. Adjacency matrices
II. Adjacency lists
III. Adjacency multilist

Adjacency matrix

The adjacency matrix of G is a two-dimensional array of size n*n (where n is the number of vertices in the graph) with the property that a [i][j]=1 if the edge (Vi, Vj) is in the set of edges, & a [i][j]=0 if there is no such edge.



$$
\begin{array}{c|cccc}
 & 0 & 1 & 2 & 3 \\
\hline
0 & 0 & 1 & 1 & 1 \\
1 & 1 & 0 & 1 & 0 \\
2 & 1 & 1 & 0 & 0 \\
3 & 1 & 0 & 0 & 0
\end{array}
$$

C representation of a graph

```
#define Maxnode 100
  int adj[Maxnode][Maxnode];
```

Write a routine to insert an edge in the graph.

```
Void insert (int adj [][100], int v1, int v2)
 {
   adj[v1][v2]=1;
 }
```

Write a routine to delete an edge from the graph.

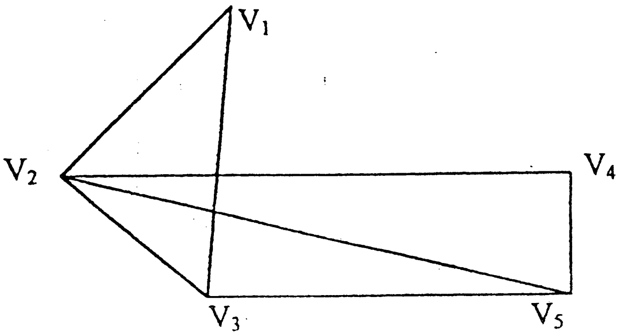
```
Void remove (int adj [][100], int v1, int v2)
 {
   adj [v1][v2]=0;
 }
```

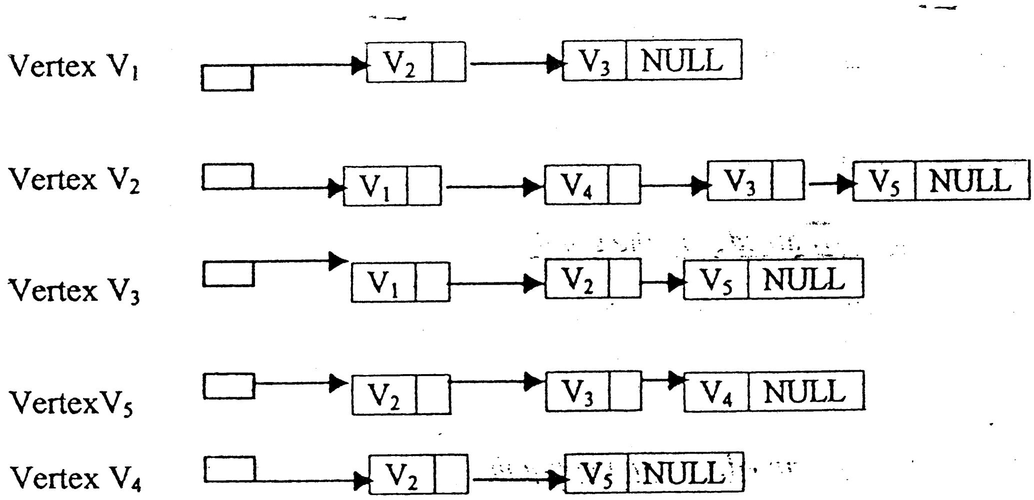
Adjacency list

In this representation the n rows of the adjacency matrix are represented as n link lists. There is one list for each vertex in the graph .The nodes in list i represent the vertices that are adjacent from vertex i. Each list has a head node.

C implementation

```
#define MAX 20
 struct node
 {
    int vno;
    struct node *next;
 };
 struct node *nodeptr;
 nodeptr adj-list[MAX];
```





8

Traversal of a Graph

A graph can be traversed in two ways: -

I. BFS (Breadth First Search)
II. DFS (Depth First Search)

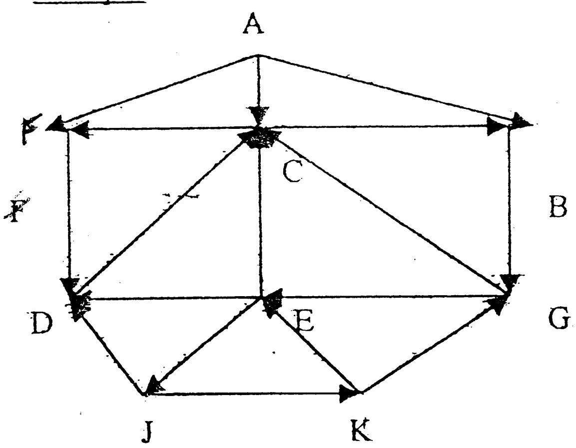
Depth First Search (DFS)

Imp

Algorithm

1. Initialize all the node of the graph to status 1.
2. Push the starting node in the stack & make it status 2
3. Repeat steps 4 & 5 until the stack becomes empty.
4. Pop the top node &n print it make it status 3
5. Push all the neighbor node of the deleted node whose status is 1& make the status 2
6. Exit.

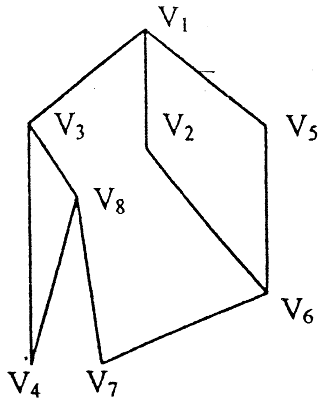
Example



We want to find & print all the nodes Reachable from the node J

I. Push J onto the stack	Stack: J
II. Print J	Stack: D, K
III. Print K	Stack: D, E, G
IV. Print G	Stack: D, E, C
V. Print C	Stack: D, E, F
VI. Print F	Stack: D, E
VII. Print E	Stack: D
VIII. Print D	Stack:

The stack is now empty, so the DFS of G starting at j is now complete. So the result is: -J K G C F E D



V_4
$\cancel{V_8}$
$\cancel{V_7}$
$\cancel{V_6}$
$\cancel{V_5}$
V_2
V_3
$\cancel{V_1}$
NULL

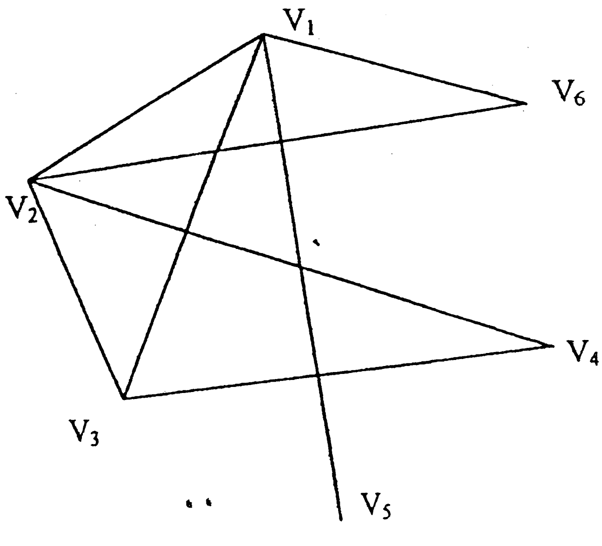
So the result is, V_1, V_5, V_6, V_7, V_8, V_4, V_2, V_3

Breadth First Search (BFS)

Algorithm

1. Initialize all the nodes of the graph to status 1.
2. Insert the starting node in the queue & make it status 2
3. Repeat steps 4 & 5 until the queue becomes empty.
4. Delete the front node & make it status 3
5. Insert all the neighbors of the deleted node in the graph (whose status is 1)& make the status 2
6. Exit.

Example

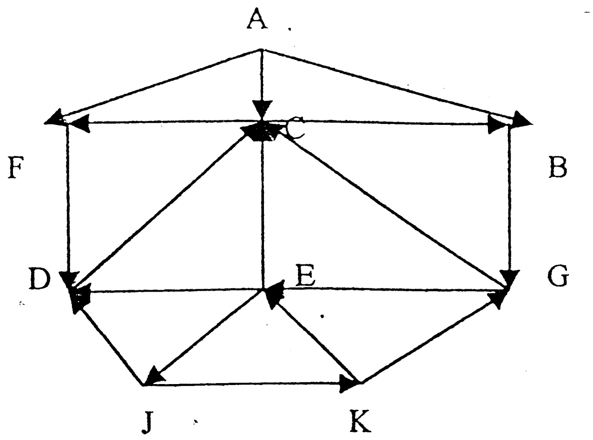


Rear front

$V_5\ V_6\ V_3\ V_4\ \cancel{V_1}\ \cancel{V_2}$

So, the result is,
$V_2\ V_1\ V_4\ V_3\ V_6\ V_5$

Example



We want minimum path p from A to J

1) FRONT=1
REAR=1

Queue: A
Origin:Φ

2) FRONT=2
REAR=4
Remove A

x
Queue: A F C B
Origin:Φ A A A

3) FRONT=3
REAR=5
Remove (AF)

x x
Queue: A F C B D
Origin:Φ A A A F

4) FRONT=4
REAR=5
Remove (AFC)

x x x
Queue: A F C B D
Origin:Φ A A A F

5) FRONT=5
REAR=6
Remove (AFCB)

x x x x
Queue: A F C B D G
Origin:Φ A A A F B

Contd.......

xx x xx

6) FRONT=6 Queue: A F C B D G

REAR=6 Origin: Φ A A A F B

Remove (AFCBD)

xx x x x x

7) FRONT=7 Queue: A F C B D G E

REAR=7 Origin: Φ A A A F B G

Remove (AFCBDG)

xx x x x xx

8) FRONT=8 Queue: A F C B D G E J

REAR=8 Origin: Φ A A A F B G E

Remove (AFCBDG E)

We stop as soon as J is added to the queue, since J is our final destination. We now backtrack from J, using the array *origin* to find the path P. Thus

J ⟵ E ⟵ G ⟵ B ⟵ A is required path.

12

Graphs

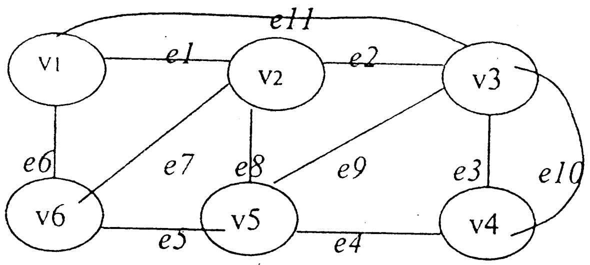
1. Define the following terms. Associate appropriate figures to illustrate the terms

- Walk
- Open & closed walk
- Path
- Length of the path
- Circuit
- Connected graph
- Degree of a vertex
- Clique/Complete Graph

- Indegree
- Sink
- Cycle
- Network.

Ans:- **Walk:-***A walk in a graph G=(V,E) is a finite alternating sequence of vertices & edges(v1,e1,v2,e2,v3……………….)beginning & ending with vertices, in such a way that each edge is incident with vertices preceding & following it.*

Open & closed walk:- *a walk is open if its end vertices or terminal vertices are distinct, otherwise it is considered as a closed walk.*



Open walk:-v1,e1,v2,e2,v3,e3,v4,e4,v5,e9,v3,e10,v4
Closed walk:-v1,e1,v2,e2,v3,e3,v4,e4,v5,e9,v3,e11,v1
Circuit: *- in a closed walk if no vertex appears more than once, except the terminal vertices ,then the closed walk is called the circuit.*
v1,e1,v2,e2,v3,e3,v4,e4,v5,e5,v6,e6,v1.

2. What are the various ways of representation of graph in memory?Explain each of them.

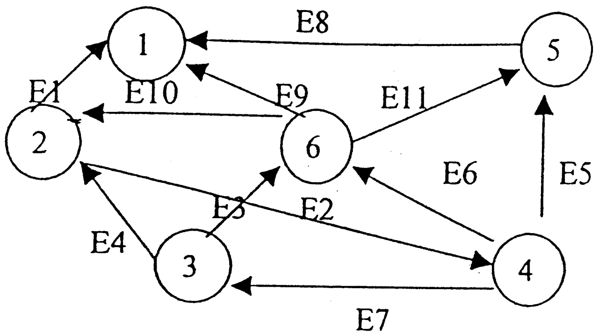
3)

3. For the following graph obtain:

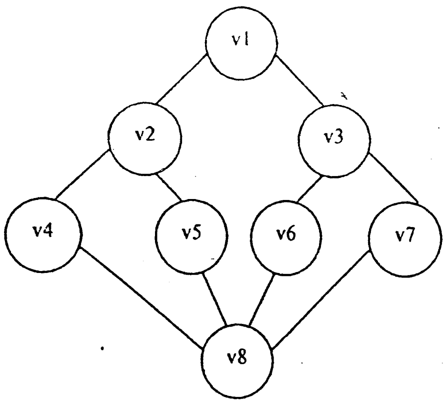
a) The indegree & out degree of each vertex.

b) Its adjacency matrix.

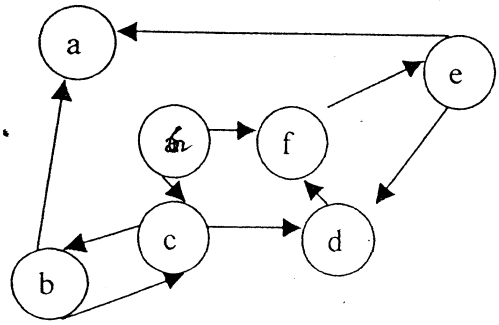
c) Its adjacency list representation.



4. Explain the BFS & DFS traversals of the following undirected graph.
Write down the required algorithm of the above two traversals.



5. For the following directed graph (digraph) find the DFS & BFS



6. The adjacency matrix of an undirected graph is

a) Unit matrix   b) an asymmetric matrix

c) symmetric matrix   d) None of the above.