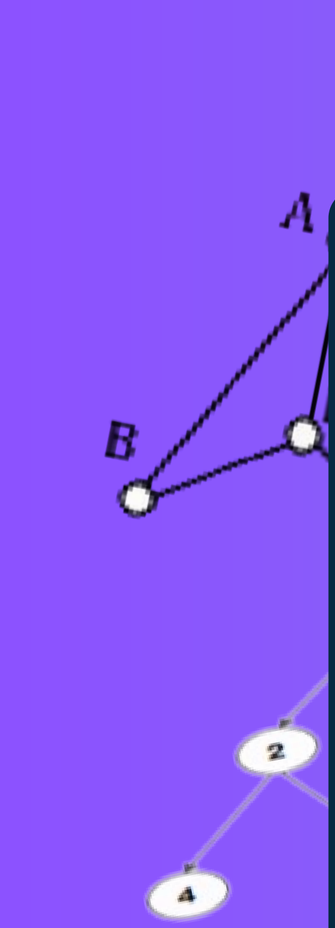
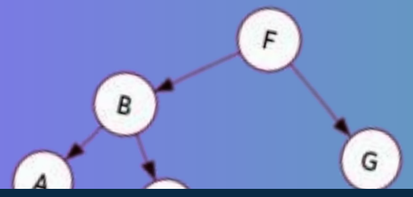
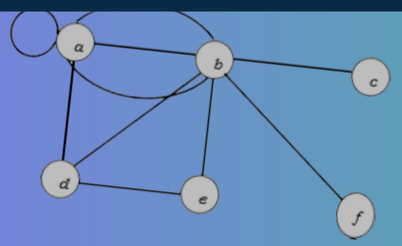
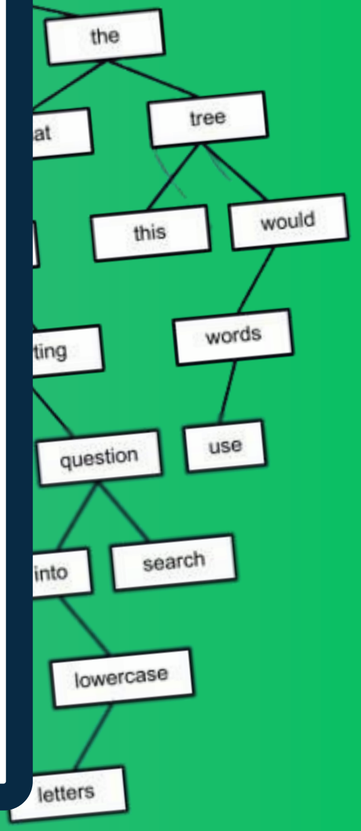




DBM20083-DISCRETE MATHEMATICS

Chapter 3





CHAPTER 3: GRAPHS AND TREES

3.1 Derive concept of Graphs

- 3.1.1 Define graph
- 3.1.2 Identify the graph terminology
  - 3.1.2a Simple graph
  - 3.1.2b Multigraphs
  - 3.1.2c Pseudograph
- 3.1.3 Explain the properties of graph.
- 3.1.4 Construct graph representations.
  - 3.1.4a Simple graph
  - 3.1.4b Directed graph
  - 3.1.4c Weighted graph
  - 3.1.4d Connected graph

CHAPTER 3: GRAPHS AND TREES

3.1.5 Compare the different types of graphs:

- 3.1.5a Discrete graph
- 3.1.5b Complete graph
- 3.1.5c Linear graph
- 3.1.5d Bipartitegraph

3.1.6 Define graph

3.1.7 Identify the graph terminology

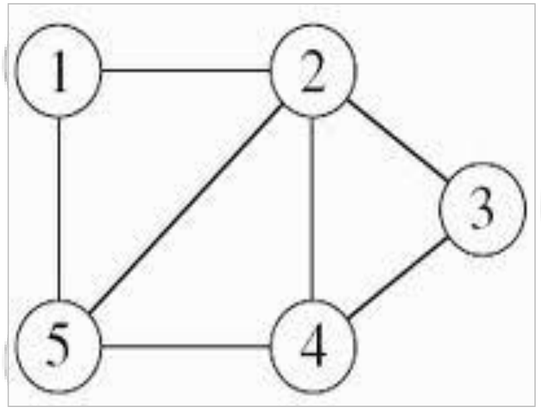
3.1.8 Explain the properties of graph.

3.1.1 DEFINE GRAPH

- Problem can be modeled as a graph. Since graphs are drawn with dots and lines, they look like road maps.
- **Computer network can be modeled using a graph** in which **the vertices of the graph represent the data centers and the edges represent communication links.**

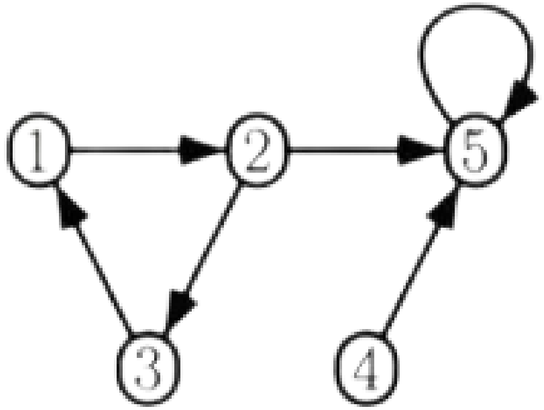
An undirected graph G consists of set V of vertices and set E of edges such that each edge is associated with an unordered pair of vertices.

Example:



A directed graph G (also known as digraph) is a graph in which edge of the graph has a direction. Such edge is known as directed edge.

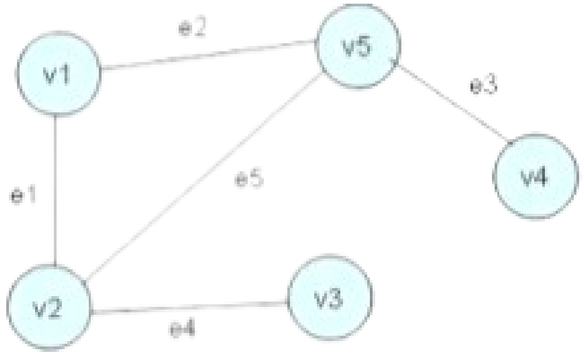
Example:



IMPORTANT: CONCEPT OF GRAPHS

VERTICES/NODES

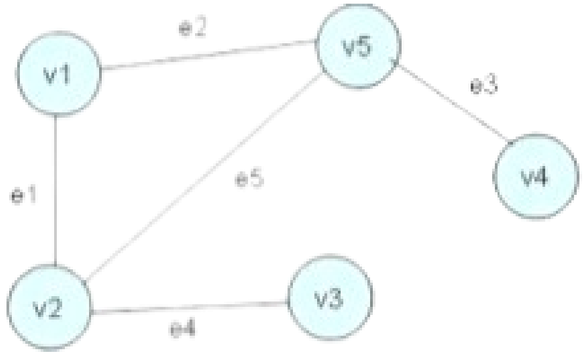
A set of $\boldsymbol{V} = \boldsymbol{V}\ (\boldsymbol{G})$ whose elements are called **vertices, points or nodes**.



Vertices, V ={v1, v2, v3, v4, v5}

EDGES

A set of $\boldsymbol{E} = \boldsymbol{E}\ (\boldsymbol{G})$ whose elements are called **edges**.

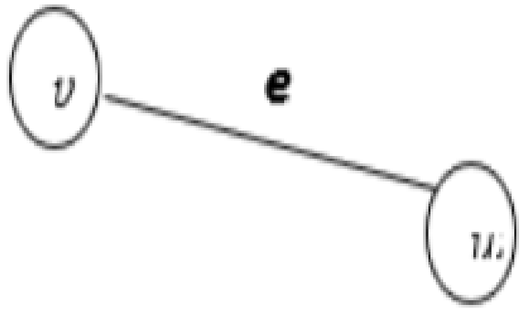


Edges, E = {e1, e2, e3, e4, e5}

IMPORTANT: CONCEPT OF GRAPHS

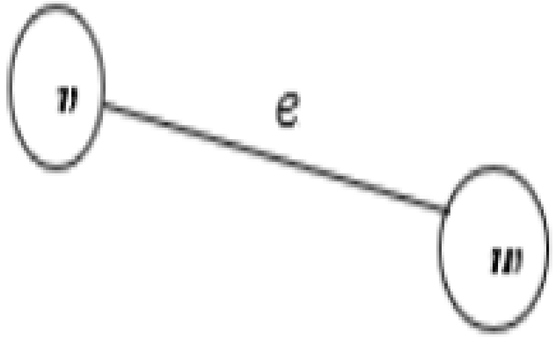
INCIDENT ON A VERTEX

If edge e is associated with vertices v and w, **e is said to be incident on v and w.**

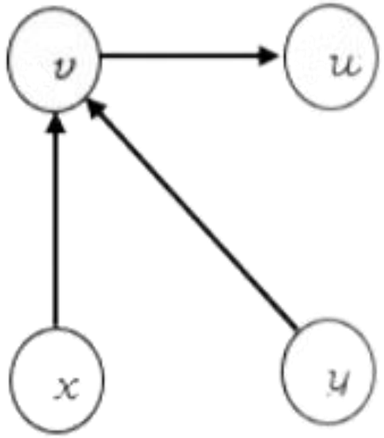


INCIDENT ON AN EDGE

If edge e is associated with vertices v and w, **v and w are said to be incident on e.**



DEGREE ON A VERTEX



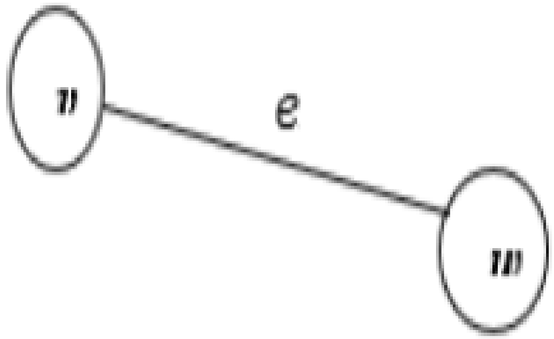
Vertices	v	w	x	y
In degree	2	1	0	0
Out degree	1	0	1	1

Degree of vertex v is 3, because there are **3 edges incident on v**

IMPORTANT: CONCEPT OF GRAPHS

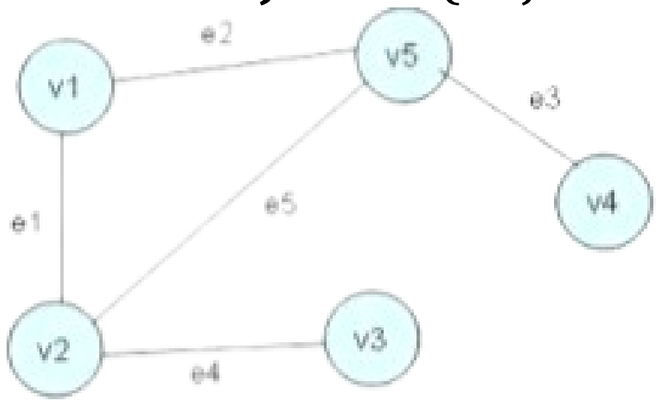
ADJACENT VERTICES

If edge e is associated with vertices v and w, v and w are said to be adjacent (adjoining).



ADJACENT EDGES

Adjacent edges are edges that originate from one same vertex. If edge e_1 and e_2 is associated with vertices v_1 , **e_1 and e_2 are said to be adjacent** (adjoining)



IMPORTANT: CONCEPT OF GRAPHS

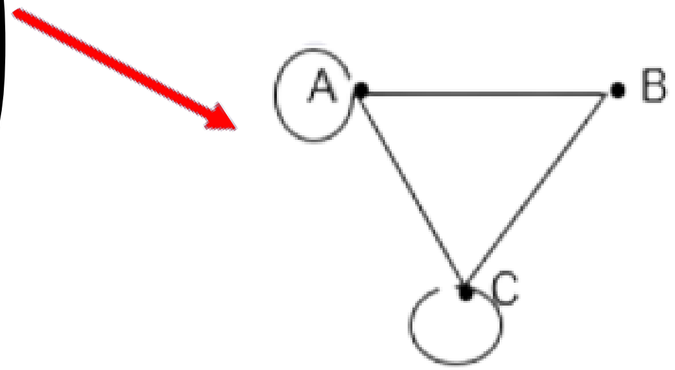
ADJACENCY MATRIX

A matrix that representing a graph with rows and columns labeled by vertices.

• Adjacent vertices is labeled 1, 0 if there no adjacent vertices.

Example: Given coordinates A, B and C

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

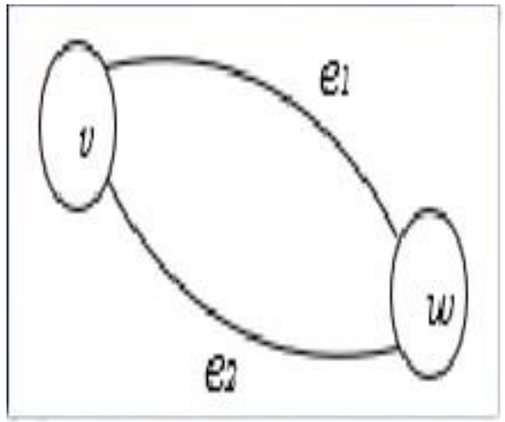


ADJACENT EDGES

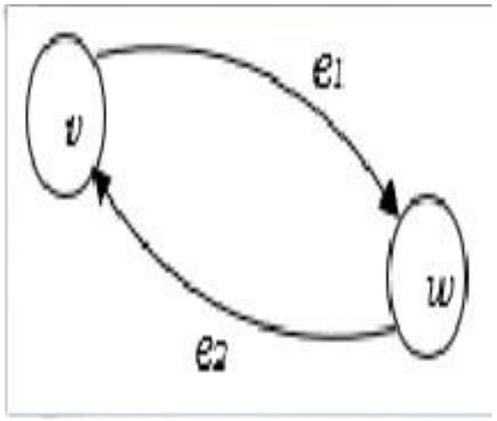
- Edges that are **incident on the same pair of vertices**.

Example:

(**e_1** and **e_2** are parallel edges)



EXAMPLE 1



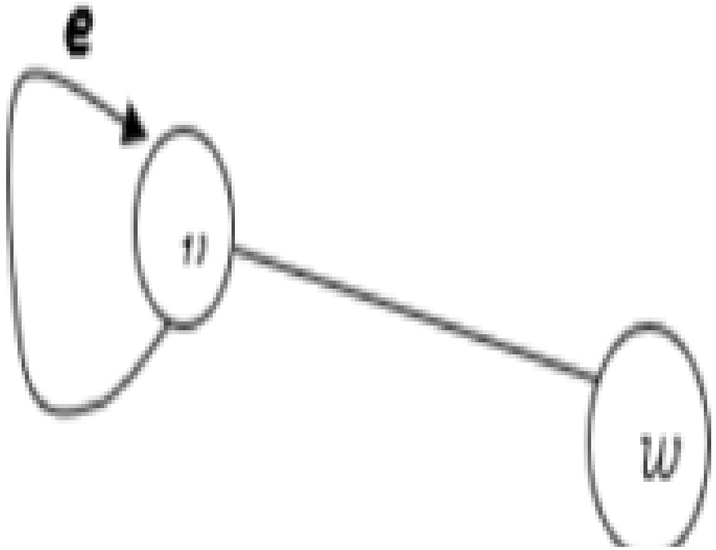
EXAMPLE 2

IMPORTANT: CONCEPT OF GRAPHS

LOOP

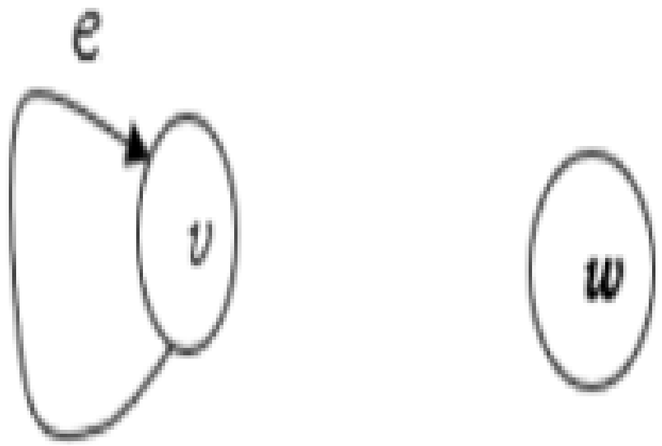
An edge incident on a single vertex

Example: (**e** is a loop)



ISOLATED VERTEX

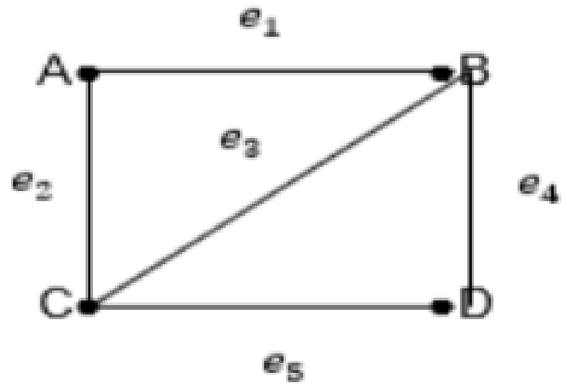
A VERTEX THAT IS NOT INCIDENT ON ANY EDGE EXAMPLE: (**W**: ISOLATED VERTEX)



3.1.2 IDENTIFY THE GRAPH TERMINOLOGY

1. A simple graph is an
   - **Unweighted**
   - **Undirected graph**
   - **Containing no loops**
   - **No multiple edges/parallel edges**

EXAMPLE



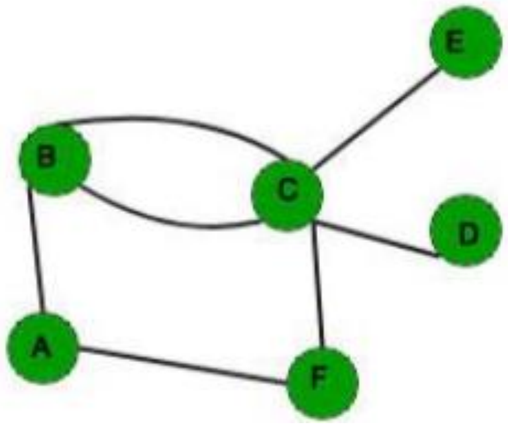
- Vertices, V = A, B, C, D
- Edges, E: $E = \{e_1, e_2, e_3, e_4, e_5\} = \{\{A, B\}, \{A, C\}, \{B, C\}, \{B, D\}, \{C, D\}\}$
- Vertices **A and B** are said to be **adjacent** if there is an edge $e = A, B$. We say that e_1 is an incident of A and B.
- **Adjacent edges** are edges that originate from one same vertex. Example e_1 and e_2.

3.1.2 IDENTIFY THE GRAPH TERMINOLOGY

2. **A multigraph** is a graph that may contain

- **multiple edges/parallel edges** (definition: two or more edges connecting the same two vertices)

EXAMPLE

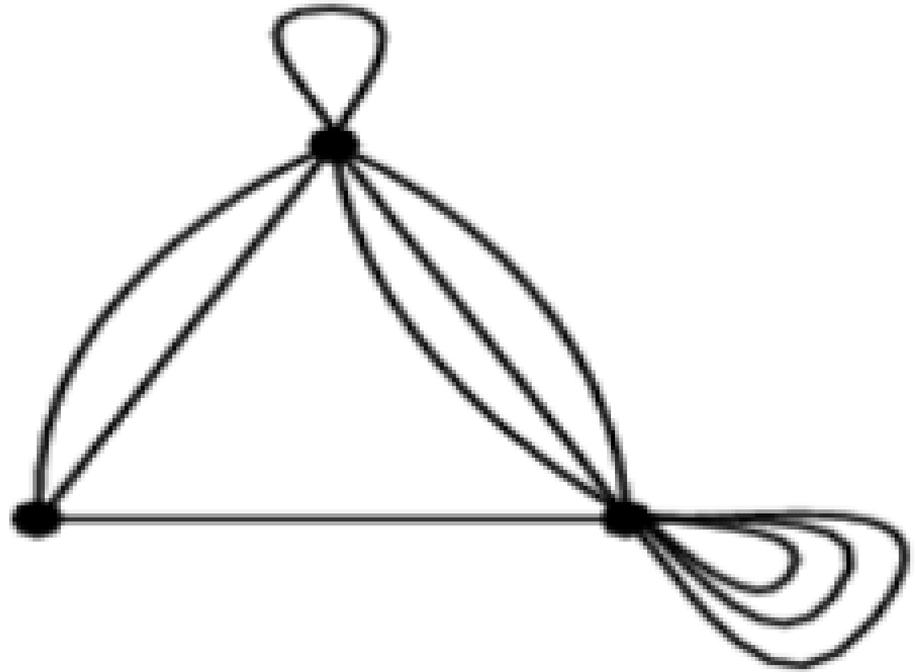


- IT CONTAINS **MULTIPLE EDGES** WHICH **CONNECT THE SAME TWO VERTICES B AND C.**

3.1.2 IDENTIFY THE GRAPH TERMINOLOGY

3. **A pseudograph** is a non-simple graph in which

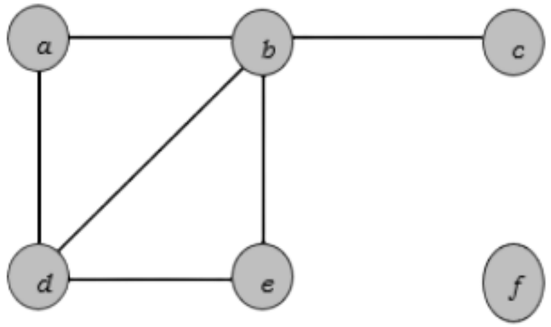
- **multiple edges/parallel edges** (definition: two or more edges connecting the same two vertices)
- **loops** are permitted.



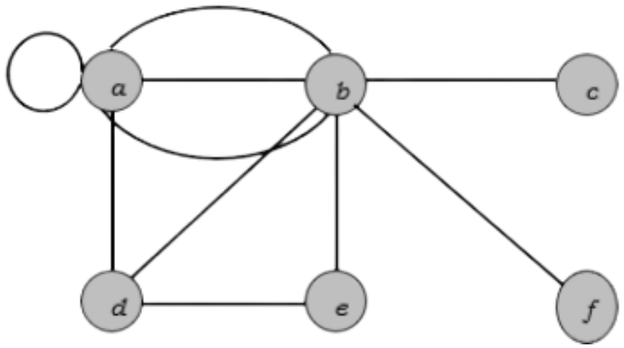
Exercise A

1. Find the **number of vertices**, the **number of edges**; identify all **parallel edges**, **loops** and **isolated vertices** for the following graph.

a)



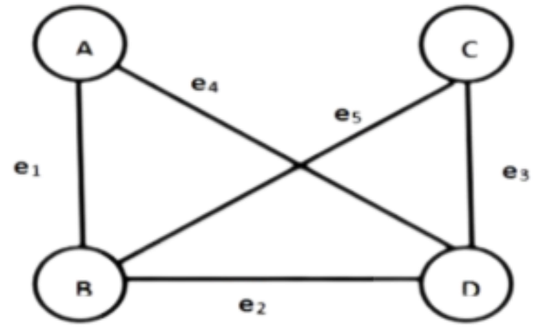
b)



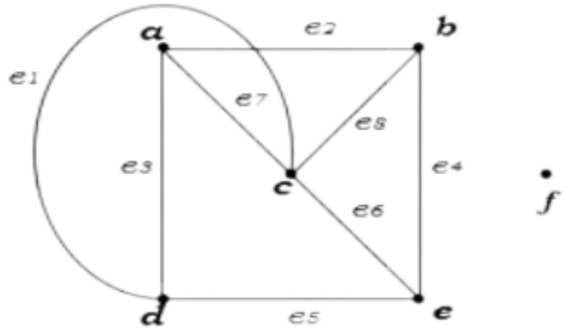
Exercise A

2. Find the **number of vertices**, the **number of edges**; identify all **parallel edges**, **loops** and **isolated vertices** for the following graph.

a)



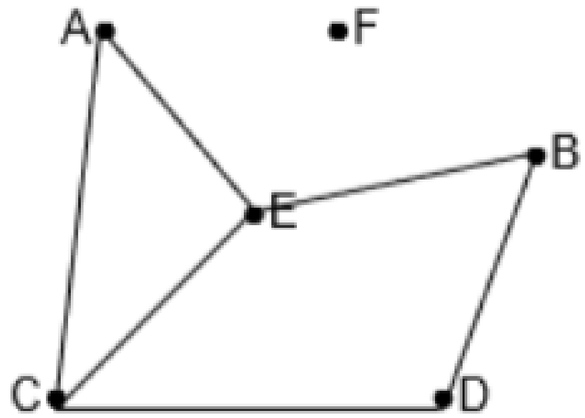
b)



3.1.3 PROPERTIES OF GRAPH

- The **ordered pair** (a, b) is a pair of objects, for example graph G = G (V, E), V = {a, b, c, d}, the order pair for the vertices are **E = {(a, b), (a, d), (b, d), (a, c), (c, d)}**
- The **size of a graph** is the **number of its edges**.
- **Degree of a vertex** written $\boldsymbol{deg}\ (\boldsymbol{v})$ = the **number of edges** which are incident on v. The vertex v is said to be **even or odd (parity)** according as $deg\ (v)$ is even or odd.
- **A vertex $\boldsymbol{v}$ is isolated** if it is does **not belong to any edge**.
- **Sum of the degrees = twice the number of edges**.

EXAMPLE



$deg\ (A)$ = 2 → A is even vertex

$deg\ (B)$ = 2 → B is even vertex

$deg\ (C)$ = 3 → C is odd vertex

$deg\ (D)$ = 2 → D is even vertex **(loop must be counted twice)**

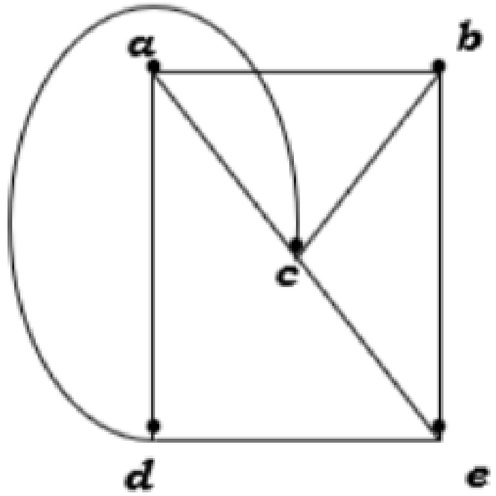
$deg\ (E)$ = 3 → D is even vertex

***Isolated vertex = F**

***Sum of degrees = 6 × 2 = 12 (6 EDGES)**

Exercise 3B

1. Find the degree of each vertex for the following graph.



2. Draw a graph having the given properties or explain why no such graph exists.

a) Six vertices each of degree 3

b) Five vertices each of degree 2

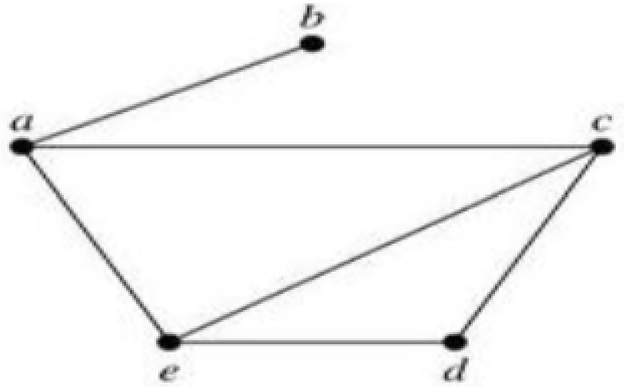
Exercise 3C

1. Draw a graph with the adjacency matrix

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

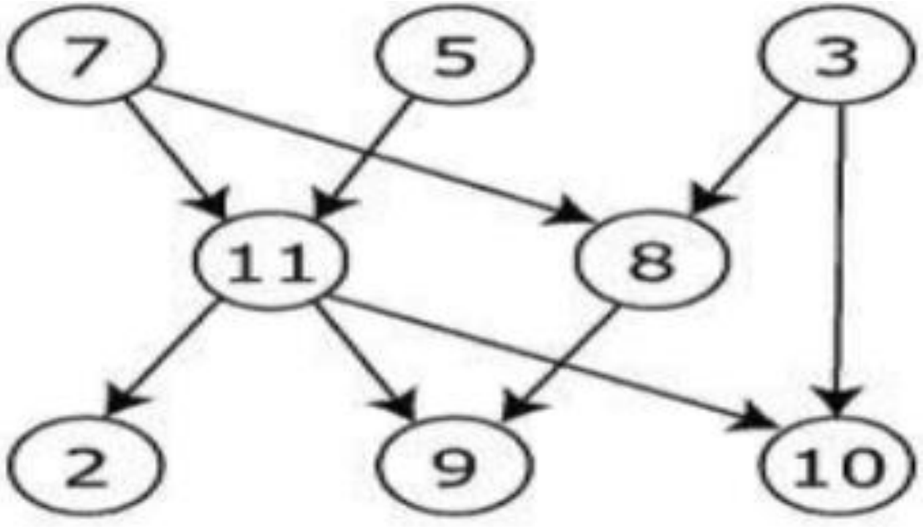
with respect to the ordering of vertices a,b,c and d.

2. Use an adjacency matrix to represent the graph shown below.



EXERCISE 3C

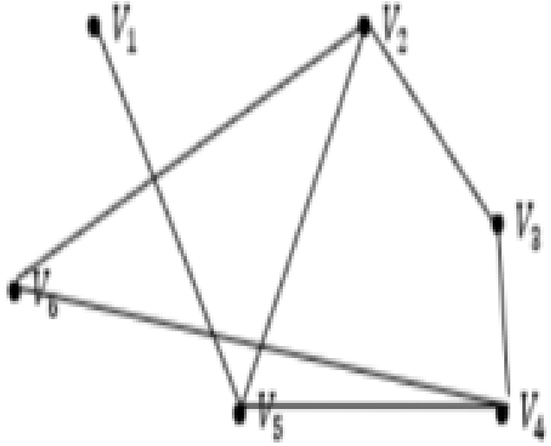
3. From the graph shown below,
   a) List down in a table form the **in degree** and the **out degree** of each vertices
   b) Determine the **parity** (even or odd) **for each vertex** and
   c) Determine the **sum of degrees**



3.1.3 PROPERTIES OF GRAPH

- A vertex with degree 1 is called a **leaf vertex** and the incident edge is referred to as a **pendant edge**.
- A **path** is a sequence of edges that begins at a vertex of a graph and travels from vertex to vertex along edges of the graph.

EXAMPLE



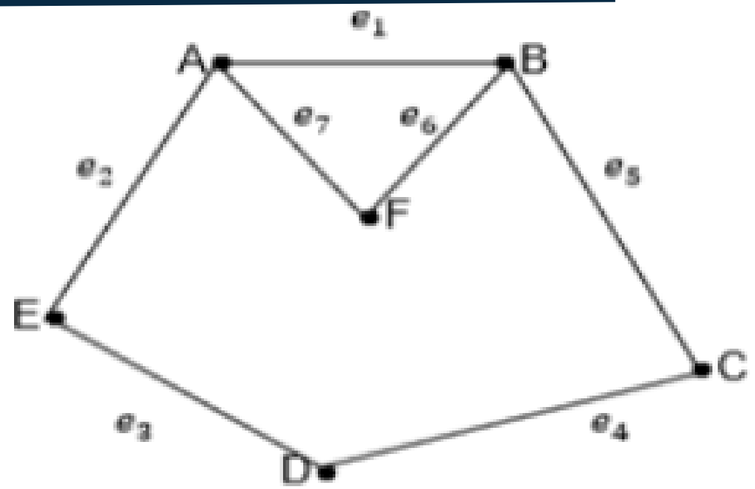
$A = (V_1, V_5, V_4, V_3)$
$B = (V_1, V_5, V_2, V_3)$
$C = (V_1, V_5, V_2, V_6, V_4, V_6, V_2, V_3)$
$D = (V_1, V_5, V_3)$

- $\boldsymbol{A}$, $\boldsymbol{B}$ and $\boldsymbol{C}$ are path because there exists an edge that connects every consecutives vertex.
- $\boldsymbol{D}$ **is not a path** because there is no edge that connects V_5 and V_3 directly.
- $\boldsymbol{A}$ and $\boldsymbol{B}$ also be called **trails** because each edge is only traversed once. (trails = simple path)
- Path $\boldsymbol{A}$ and $\boldsymbol{B}$ are also be called **simple path** where the vertex are only passed through once.

3.1.3 PROPERTIES OF GRAPH

- The **length** of the path is **the number n of edges that it contains**.
- The **distance** between two vertices is described by **the length of the shortest path that joins them**.
- A **closed path/circuit** is a path that starts and ends at the same vertex.
- A **cycle** is a closed path with **at least 3 edges** and there are **no other vertices or edges repeated except the starting vertex/node**.

EXAMPLE



- From vertex A to B there exists three different paths:

$$P = (e_1)\ ;\ Q = (e_2, e_3, e_4, e_5)\ ;\ R = (e_7, e_6)$$

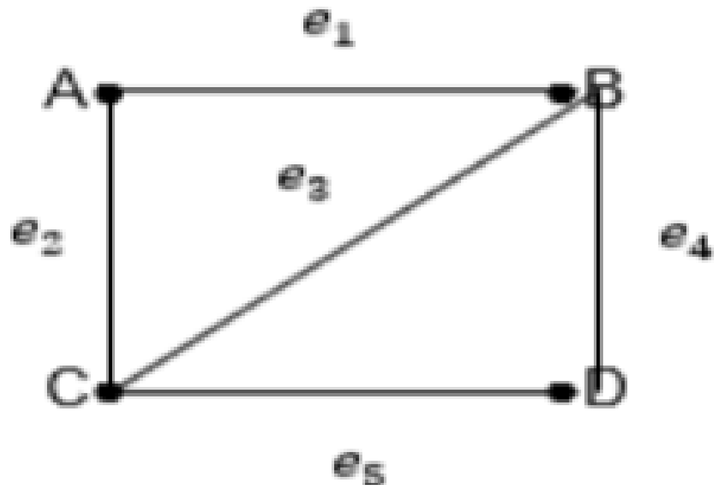
- The length of path $P = 1$, the length of path $Q = 4$ and the length of path $R = 2$
- The distance from A to B ($\boldsymbol{d\ A, B}$) is 1 (shortest path)
- Closed path: $S = (A, B, F, A)$; $T = (B, C, D, E, A, F, B)$
- Cycle: Path S and T

3.1.4 GRAPH REPRESENTATIONS

1) **A simple graph** is an

- **Unweighted**
- **Undirected graph**
- **Containing no loops**
- **No multiple edges/parallel edges.**

EXAMPLE



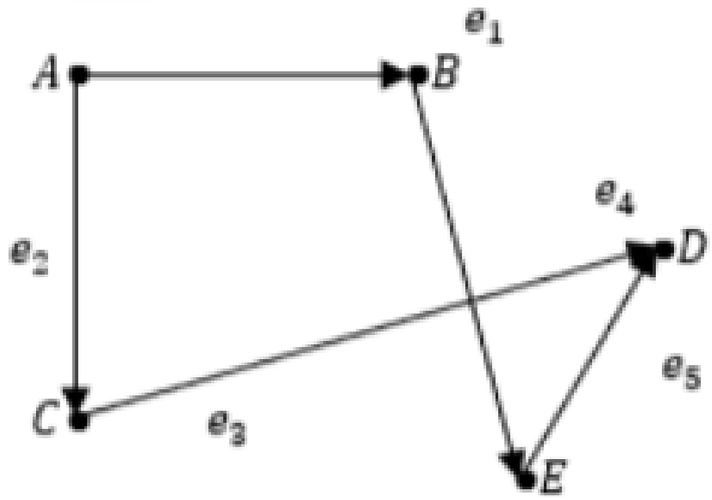
- Vertices, V = A, B, C, D
- Edges, E: $E = \{e1, e2, e3, e4, e5\} = \{(A, B)\ , (A, C)\ , (B, C)\ , (B, D)\ , (C, D)\}$
- Vertices **A and B** are said to be **adjacent** if there is an edge $e = A, B$. We say that $e1$ is an incident of A and B.
- **Adjacent edges** are edges that originate from one same vertex. Example $e1$ and $e2$.

3.1.4 GRAPH REPRESENTATIONS

2) **Digraphs (directed graph)**

- defined by $D = V, E$ where V is the set of vertex and E is the set of directed edges.

EXAMPLE



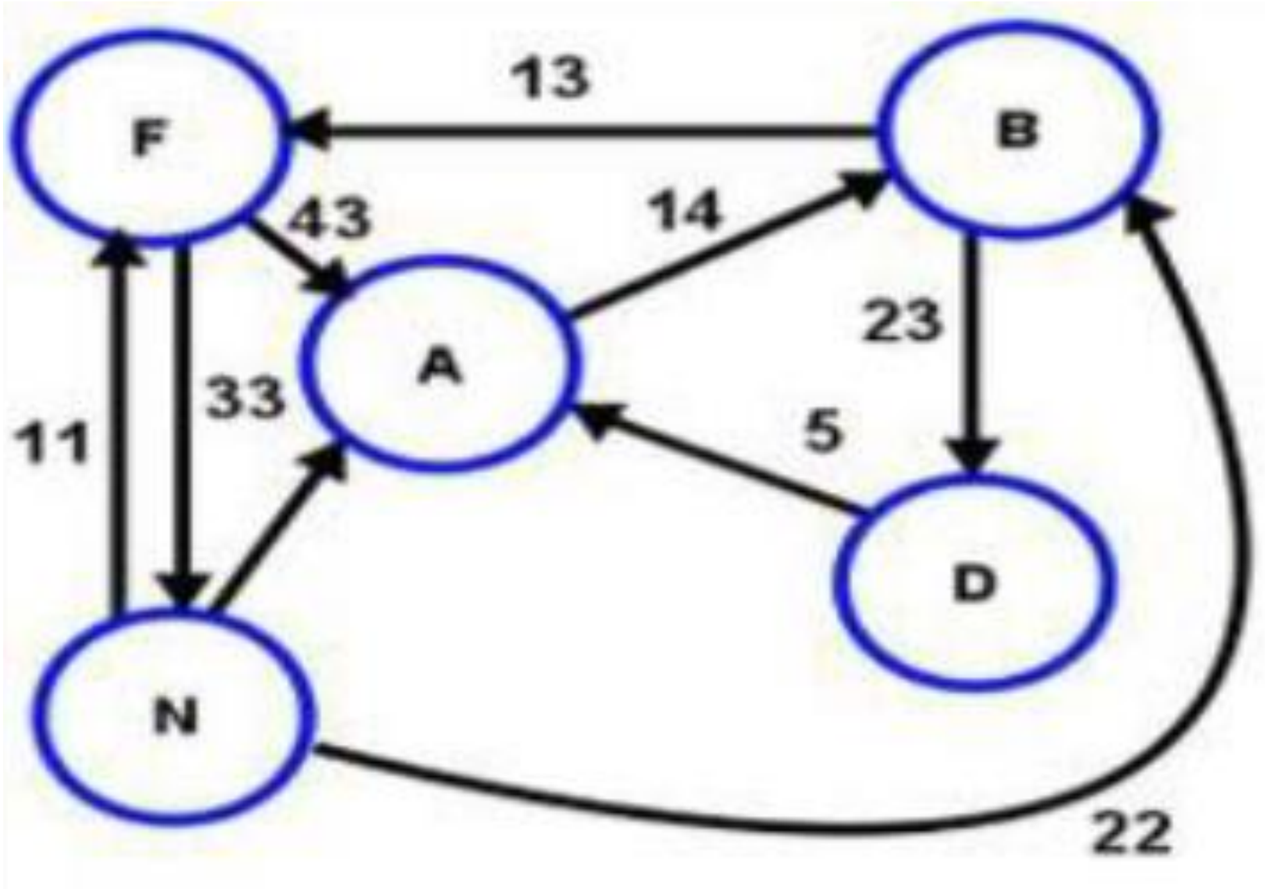
- Vertices, V = A, B, C, D, E
- Edges, E: $E = e1, e2, e3, e4, e5 = A, B\ , A, C\ , C, D\ , B, E\ , E, D$
- $\{A, B\} \neq \{B, A\}$
- For and edge A, C vertex ***A*** is called the head of edge and vertex ***C*** is the **tail** of edge.
- The head vertex is said to be adjacent to the tail but not vice versa. Example edge $\{A, B\}$, **A is adjacent to B BUT vertex B is NOT ADJACENT to A**.

3.1.4 GRAPH REPRESENTATIONS

3) **Weighted graph**

- a **number (weight)** is **assigned to each edge**.
- Weights are usually real numbers, such as costs, lengths or capacities, etc. depending on the problem at hand.

EXAMPLE

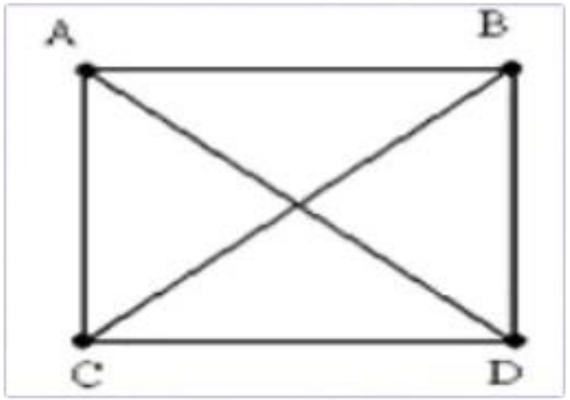


3.1.4 GRAPH REPRESENTATIONS

4) **Connected Graph**

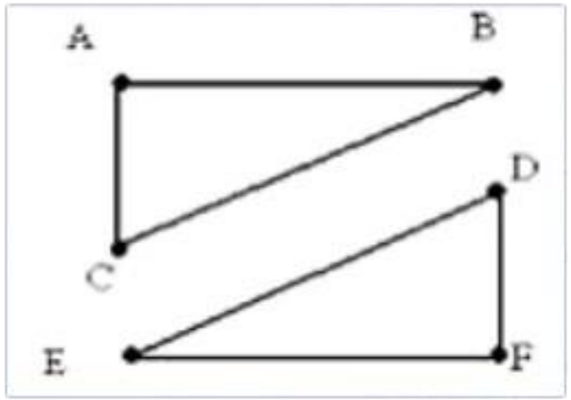
- An undirected graph is called **connected** **if there is a path between every pair of distinct vertices of the graph**.
  - * **Path**: path is a sequence of edges that begins at a vertex of a graph and travels from vertex to vertex along edges ofthe graph.

EXAMPLE

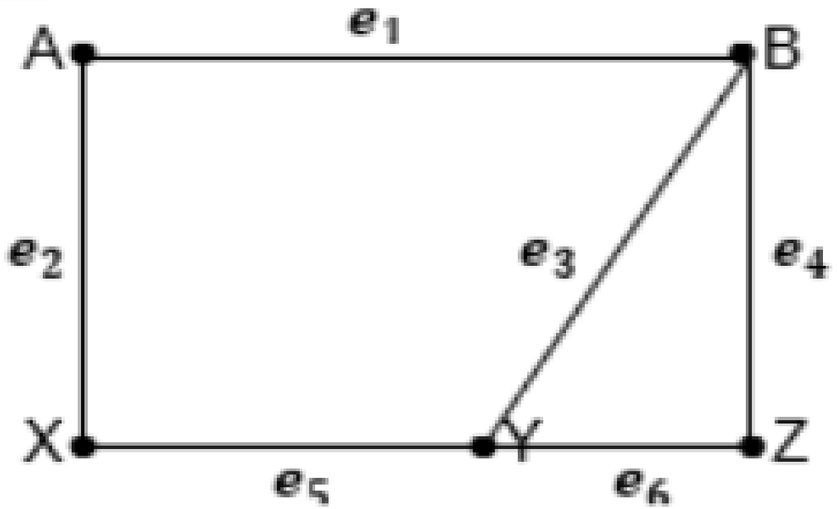


CONNECTED GRAPH

DISCONNECTED GRAPH



EXAMPLE



Let G be a graph in the figure above. Find :

a) All simple paths from vertex A to vertex Z.

b) d(A, Z)

c) All closed path (circuit) starting from vertex A.

d) k-cycle for k = 3, k = 4, k = 5 and k = 6

Solution

a) (A, B, Z), (A, B, Y, Z), (A, X, Y, Z), (A, X, Y, B, Z)

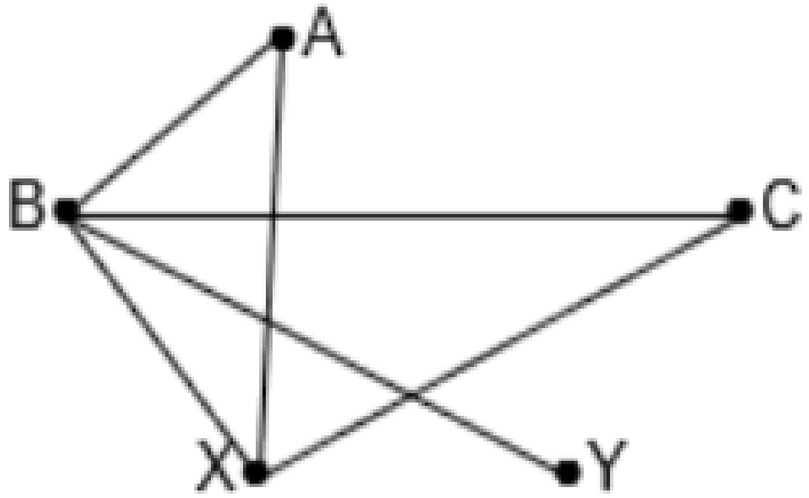
b) d(A, Z) = 2

c) (A, B, Y, X, A), (A, X, Y, B, A), (A, B, Z, Y, X, A), (A, X, Y, Z, B, A)

d)

- 3-cycle = (B, Y, Z, B), (B, Z, Y, B)
- 4-cycle = (A, B, Y, X, A), (A, X, Y, B, A)
- 5-cycle = (A, B, Z, Y, X, A), (A, X, Y, Z, B, A)
- 6-cycle = no 6-cycle

EXAMPLE



Let G be a graph in the figure below. Determine whether each of the following is a closed path(cycle), trail or simple path?

a) (B, A, X, C, B)
b) (X, A, B, Y)
c) (B, X, Y, B)
d) (B, A, X, C, B, Y)
e)(X, C, A, B, Y)
f) (X, B, A, X, C)
g) (X, B, A, X, B)
h) (X, C, B, A)

Solution

a) Closed path (cycle) & trail.

b) Simple path & trail.

c) Not a path because there is no edge between X and Y.

d) Trail.

e) Not a path because there is no edge between C and A.

f) Trail.

g) This path is not a simple path, trail or cycle.

h) Simple path & trail.

SUMMARY OF GRAPH TERMINOLOGY IN GRAPH THEORY

The **size** of a graph is the number of its edges.
The **degree of a vertex** written $\boldsymbol{deg(v)}$ is equal to the ***number of edges*** which are incident on v
The **sum of the degrees** of the vertices of a graph is equal to ***twice the number of edges***
The vertex v is said to be **even or odd (parity)** according as $deg(v)$ is even or odd
A vertex v is **isolated** if it is does not belong to any edge
A vertex with degree 1 is called a **leaf vertex**
The incident edge of vertex with degree 1 is referred as a **pendant edge**
A **path** is the sequence of connected vertices.
A **simple path** is a path where the vertices are only passed through once
A **trail** is a path where each edge is traveled once, meaning that there are no repeated edges (all edges are distinct)
The **length** of the path is the number n of edges that it contains

SUMMARY OF GRAPH TERMINOLOGY IN GRAPH THEORY

The **distance** between two vertices is described by the length of the shortest path that joins them

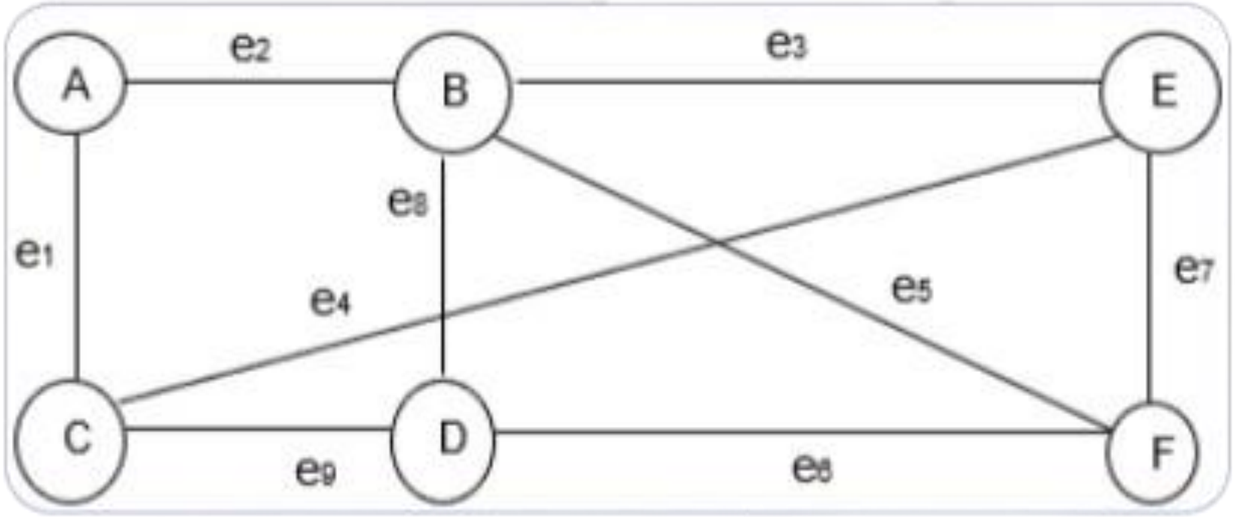
A **cycle / simple cycle** is a closed path with at least 3 edges, and no repeated vertices and

An **acyclic** is a graph that has no cycles in it

A **closed path or circuit** is a path that starts and ends at the same vertex

A graph is called **planar** if it can be drawn in the plane without any edges crossing each other

EXAMPLE



Distance from A to F : d(A, F) = 2 The path: (A, B, F)

Size of the graph: 9

Path from B to C: (B, A, C) @ (B, E, C) @ (B, F, D, C)

Trail from A to D: (A, C, D) @ (A, B, D) @ (A, B, F, D)

Simple cycle start from A: (A, B, D, C, A) @ (A, B, E, F, D, C, A)

Exercise 3d

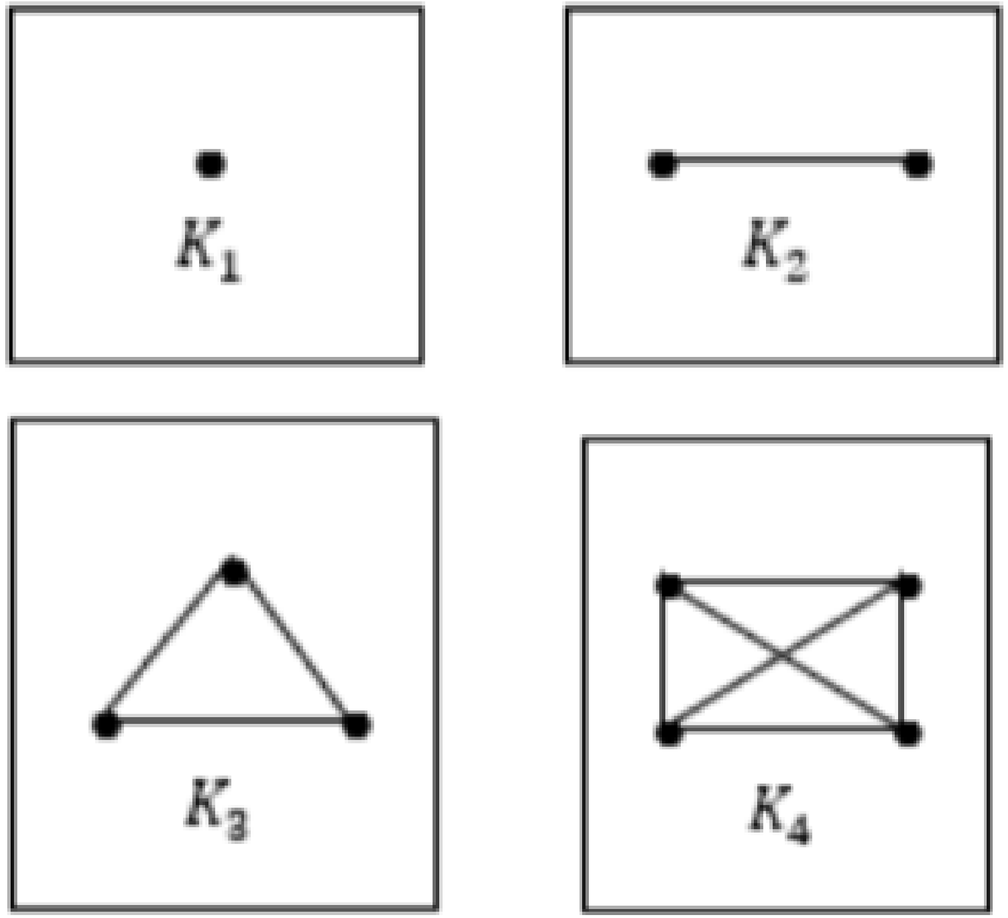
4. From the graph,
   a) Determine the **parity** (even or odd) **for each vertex**
   b) Identify the **leaf vertex**
   c) Identify the **pendant edge**
   d) Identify the **distance**, D from a to g
   e) Identify the **size** of the graph
   f) Find the 2 **simple path**, P_1, P_2 from a to c
   g) Find a **trail**, T from d to e
   h) Find 3 **simple cycle**, labelled as C_1, C_2 and C_3

3.1.5 DIFFERENT TYPES OF GRAPHS

1) **COMPLETE GRAPH**

- All nodes are connected to every other node in the graph
- The complete graph that has n nodes is written as K_n
- Example: A complete graph with 3 nodes is written as K_3
- Formula to calculate the edge in complete graph:
- The number of edges = $\frac{n(n-1)}{2}$

EXAMPLE



3.1.5 DIFFERENT TYPES OF GRAPHS

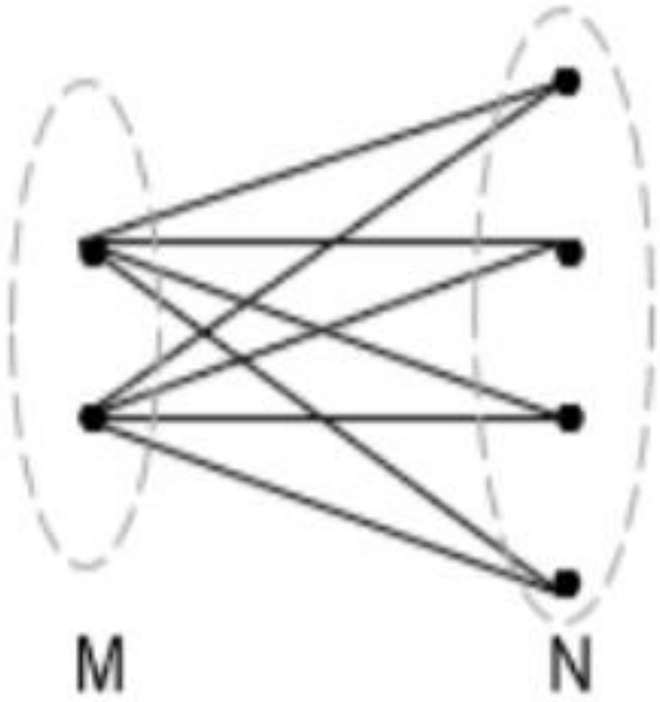
2) **BIPARTITE GRAPH**

- A bipartite graph has vertex that can be divided into two subsets M and N. Each edge connects a vertex in M to a vertex in N.
- Represented by $K_{m,n}$ where m is a number of vertices in M and n is the number of vertices in N.

(M ≤ N)

EXAMPLE

Example complete bipartite graph $K_{2,4}$ has $2 \times 4 = 8$ edges

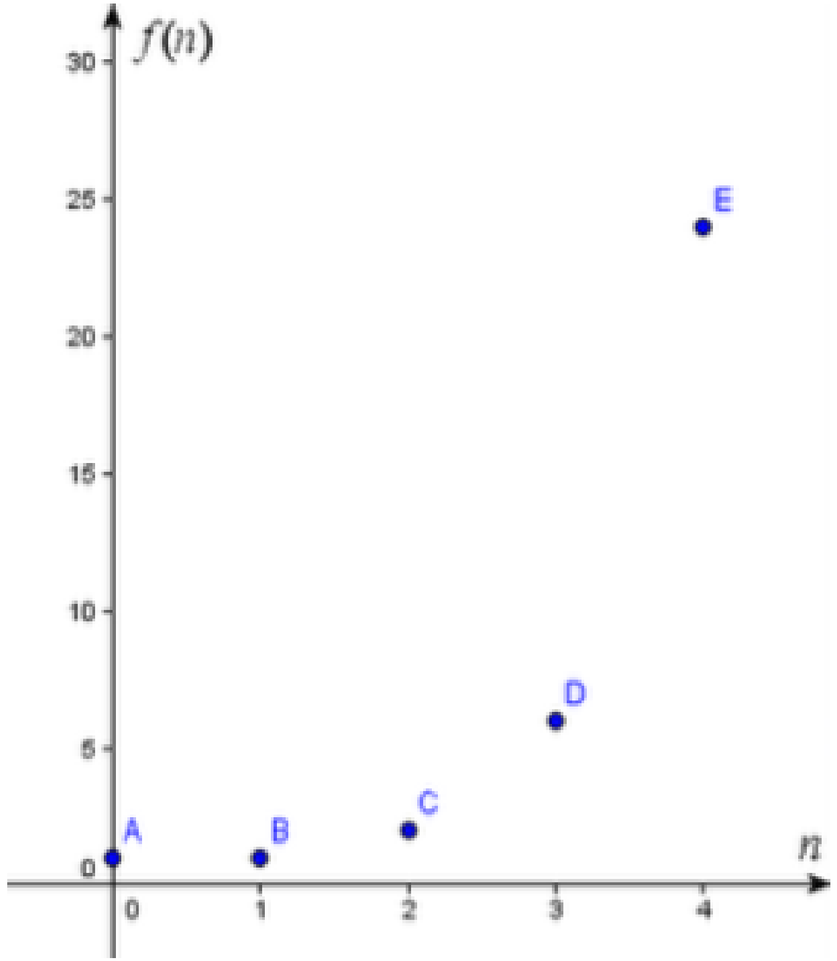


3.1.5 DIFFERENT TYPES OF GRAPHS

3) **DISCRETE GRAPH**

- A discrete graph consists of only certain **discrete points**. Or in other word, the **value of y is independent to x**
- Example: If you graphed y = x2 for all integer values of x, then you would only have certain values: (0,0) (1,1) (2,4) (3,9) etc. (-1,1) (-2,4) (-3,9) etc. You do **NOT connect the dots** in this case. Therefore, it is called **discrete graph**

EXAMPLE

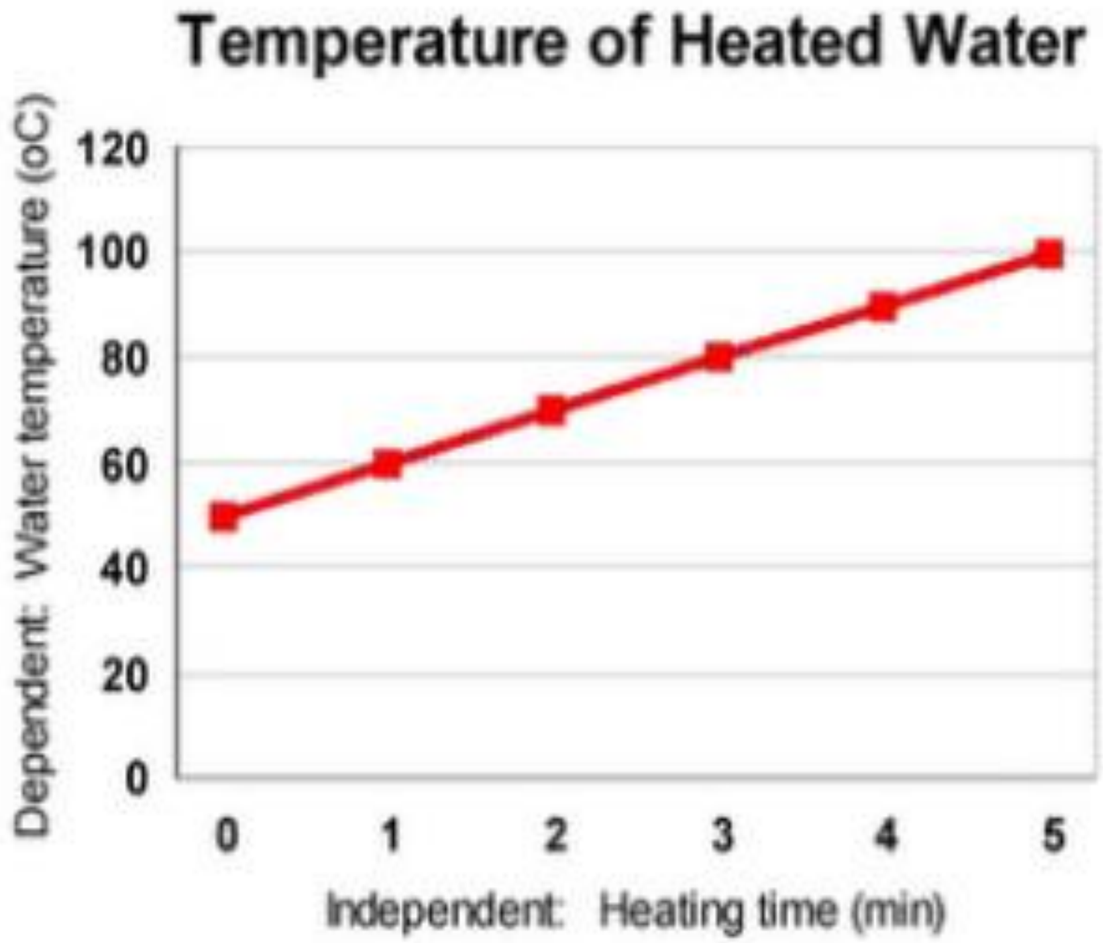


3.1.5 DIFFERENT TYPES OF GRAPHS

4) **LINEAR GRAPH**

- **A continuous graph** contains all points within a given interval or perhaps the entire number line. Or in other word, the **value of y is dependent to x**
- Example: If you graphed $y = x^2$ for all values of x, it would be a continuous graph from one side of the graph to the other

EXAMPLE


Temperature of Heated Water
Dependent: Water temperature (oC)
120
100
80
60
40
20
0
0
1
2
3
4
5
Independent: Heating time (min)

Exercise 3D

1) Calculate the number of edges in complete graph $\mathbf{K_8}$.

2) How many vertices and edges are in $\mathbf{K_{100}}$?

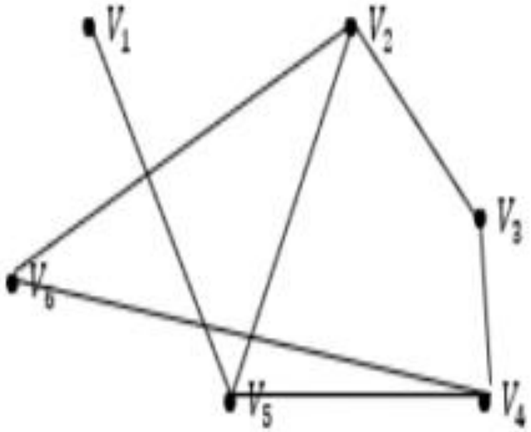
3) Draw Complete bipartite graph $\mathbf{K_{3,3}}$.

3.1.6 PATHS, CYCLES AND PLANARITY IN GRAPH

PATHS

- A **path** is a sequence of edges that begins at a vertex of a graph and travels from vertex to vertex along edges of the graph.

EXAMPLE



$A = (V_1, V_5, V_4, V_3)$
$B = (V_1, V_5, V_2, V_3)$
$C = (V_1, V_5, V_2, V_6, V_4, V_6, V_2, V_3)$
$D = (V_1, V_5, V_3)$

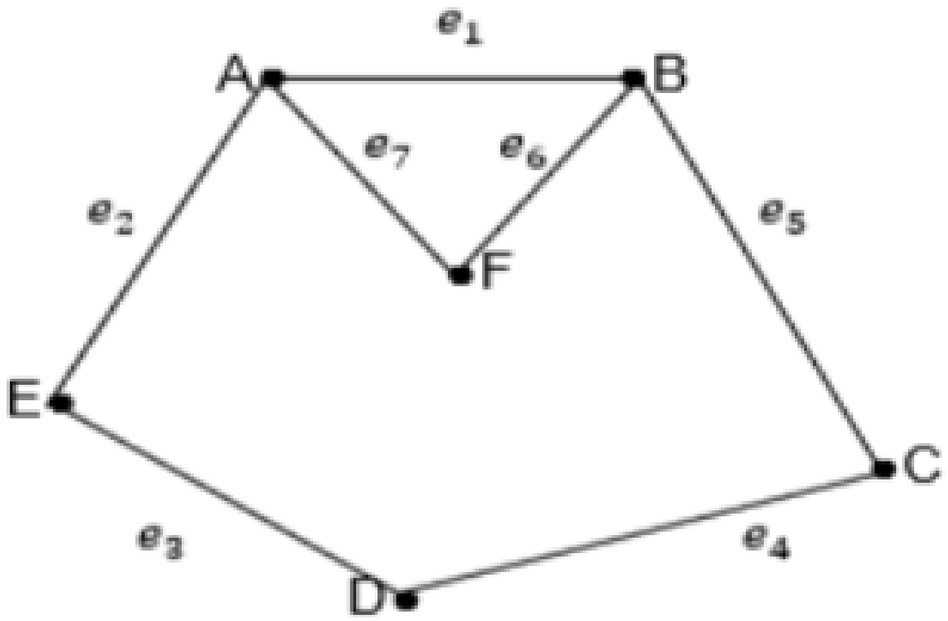
- ***A***, ***B*** and ***C*** **are path** because there exists an edge that connects every consecutives vertex.
- ***D*** **is not a path** because there is no edge that connects V_5 and V_3 directly.
- ***A*** and ***B*** also be called **trails** because each edge is only traversed once. (trails = simple path)
- Path ***A*** and ***B*** are also be called **simple path** where the **vertex are only passed through once**.

3.1.6 PATHS, CYCLES AND PLANARITY IN GRAPH

CIRCUIT & CYCLE

- A **closed path/circuit** is a path that starts and ends at the same vertex.
- A **cycle** is a closed path with **at least 3 edges** and there are **no other vertices or edges repeated except the starting vertex/node**.

EXAMPLE



- From vertex A to B there exists three different paths:

  $P = (e_1)$; $Q = (e_2, e_3, e_4, e_5)$; $R = (e_7, e_6)$
- **Closed path**: $S = (A, B, F, A)$; $T = (B, C, D, E, A, F, B)$
- **Cycle**: Path S and T

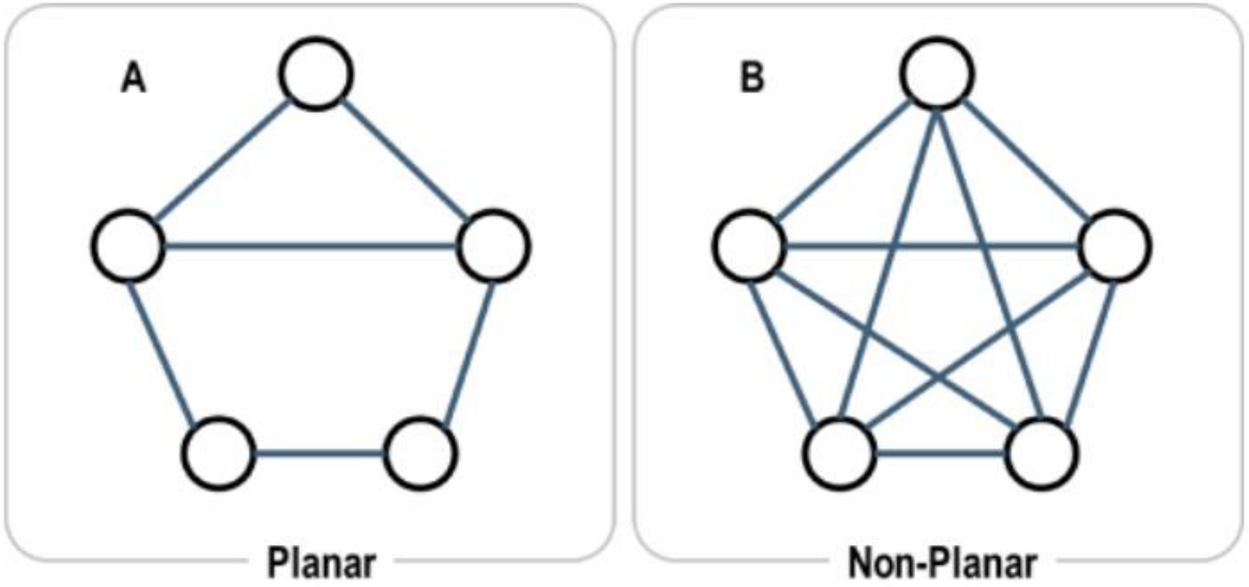
3.1.6 PATHS, CYCLES AND PLANARITY IN GRAPH

PLANAR GRAPH

- A **graph** is called **planar** if it can be drawn in the plane **without any edges crossing each other**.

EXAMPLE

Example:

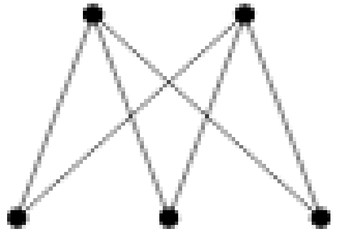


Graph A is planar since no link is overlapping with another. Graph B is non-planar since many links are overlapping. Also, the links of graph B cannot be reconfigured in a manner that would make it planar.

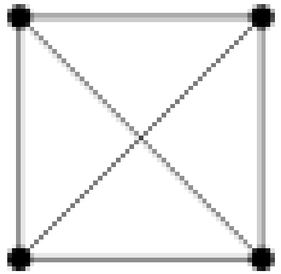
Exercise 3d

Is this a planar graph or not?

(a)



(b)



3.1.7 ISOMORPHIC GRAPHS

DEFINITION:

- Two or more graphs that have the same
  - **number of vertices**
  - **number of edges**
  - **degrees for each distinct vertices**
  - **the graphs have bijective function***

(* a function giving an exact pairing of the elements of two sets, which it is one-to-one & onto function)

EXAMPLE

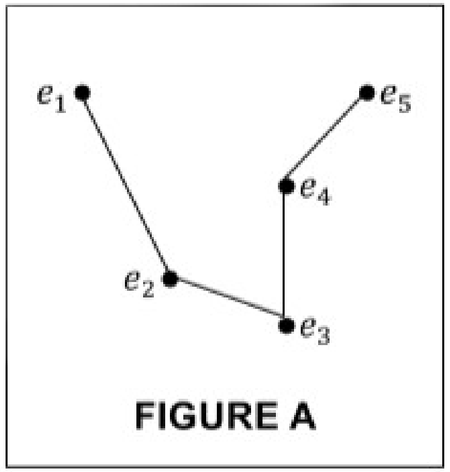


FIGURE A

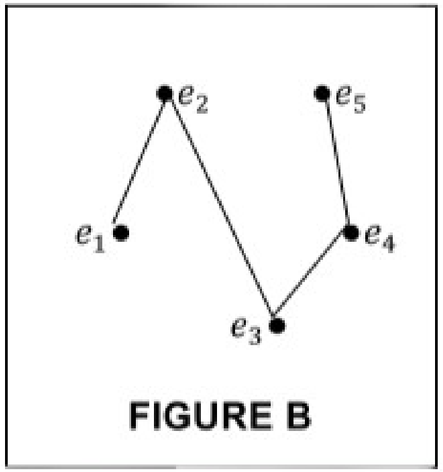


FIGURE B

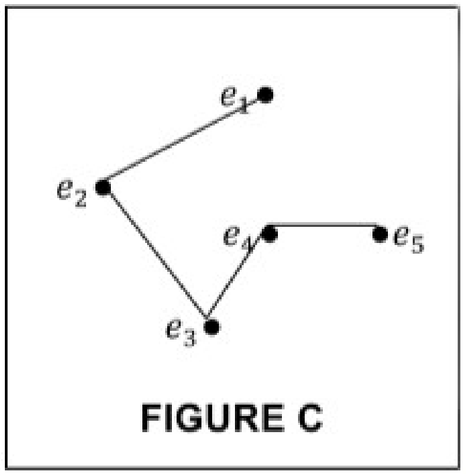


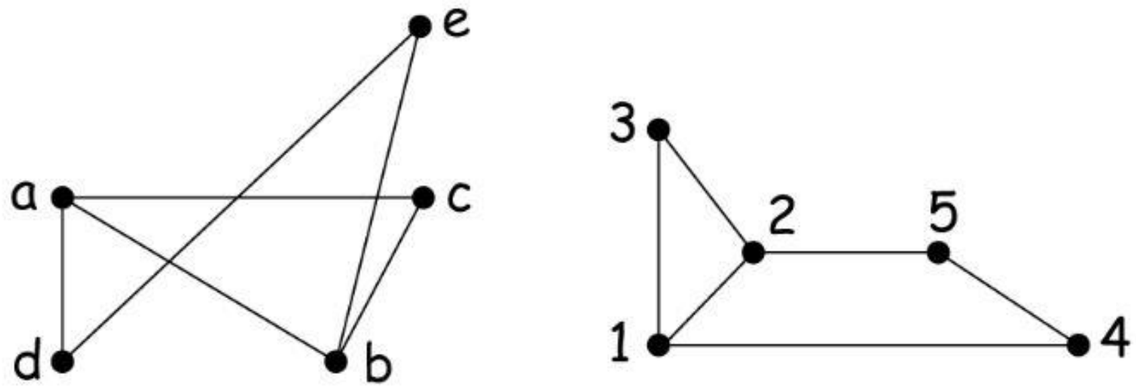
FIGURE C

For each of the graph,

G = {(e_1, e_2), (e_2, e_3), (e_3, e_4), (e_4, e_5)} the set of edges are E = { e_1, e_2, e_3, e_4, e_5}.

Therefore, Figure A, B and C are **isomorphic graphs**.

EXAMPLE



The algorithm to determine isomorphism of 2 or more graphs…

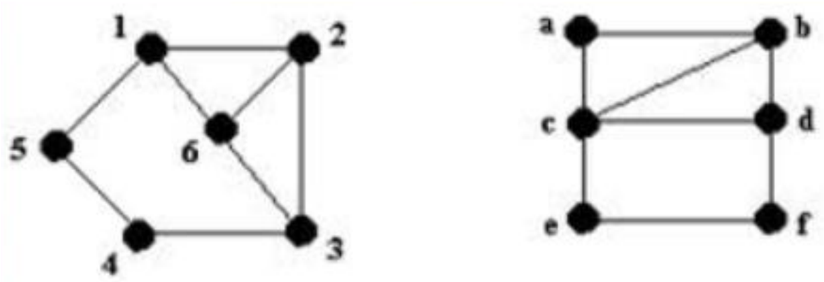
SOLUTION

Step		Graph G		Graph H	
1.	No. of vertices	5		5	
2.	No. of edges	6		6	
3.	Degree of vertex	Deg(a)	3	Deg(1)	3
		Deg(b)	3	Deg(2)	3
		Deg(c)	2	Deg(3)	2
		Deg(d)	2	Deg(4)	2
		Deg(e)	2	Deg(5)	2
4.	Illustrate the graph G & H to ensure it have bijective function! f(a)=1 f(b)=2 f(c)=3 f(d)=4 f(e)=5				

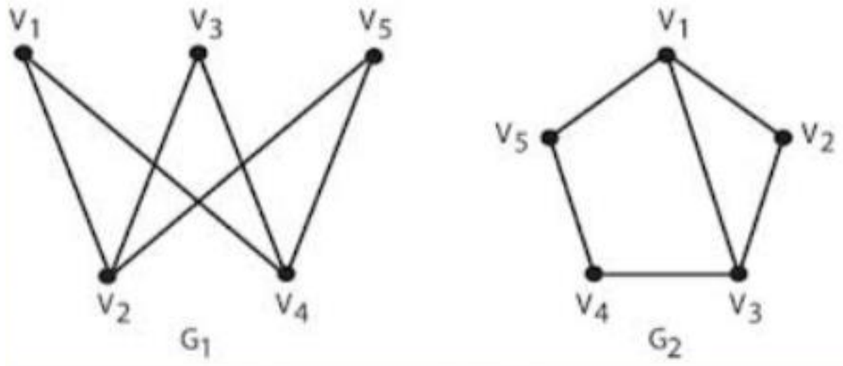
Exercise E

Determine whether the graphs below are isomorphic or not.

a)



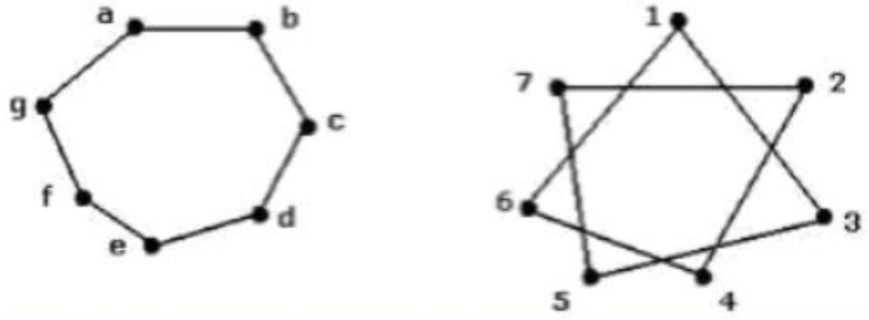
b)



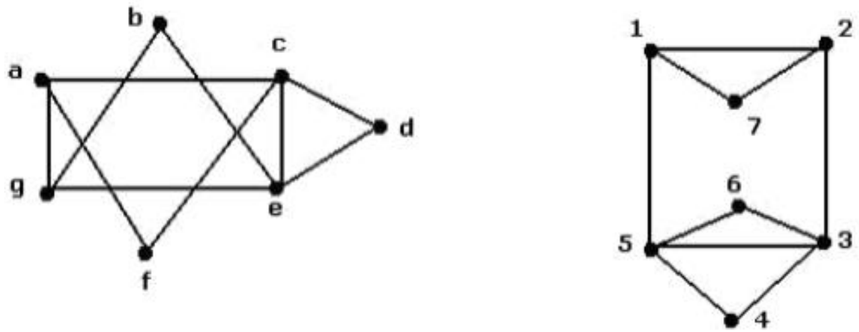
Exercise E

Determine whether the graphs below are isomorphic or not.

c)



d)



3.1.8 EULER PATHS & EULER CIRCUITS

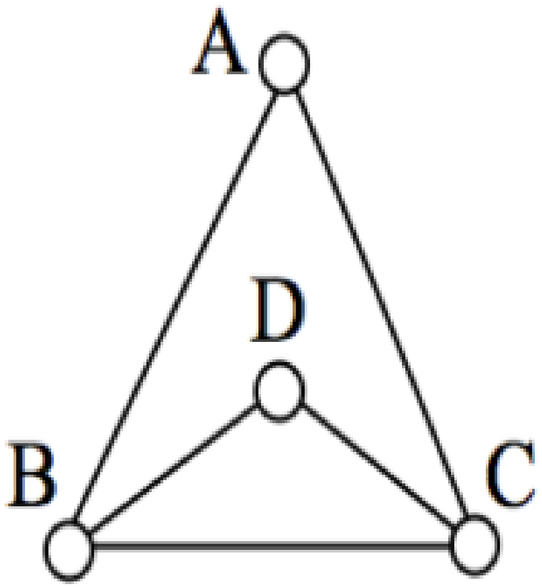
Euler circuit in a graph G

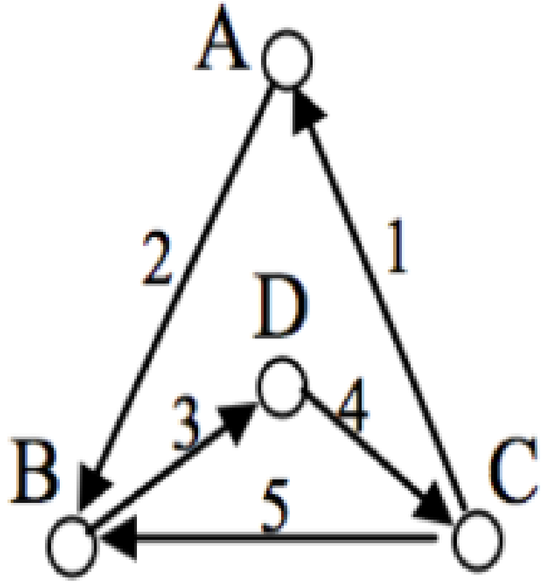
- An **Euler circuit** is a **circuit that uses every edge in a graph with no repeats**. Being a circuit, it must **start and end at the same vertex**.
- A connected multigraph has an **Euler circuit** if and only if **every vertex have even degree**.

Euler path in G

- An **Euler path** is a **path that uses every edge in a graph with no repeats**. Being a path, it **does not have to return to the starting vertex**.
- A connected multigraph has an **Euler path but not an Euler circuit** if and only if it has **exactly two vertices of odd degree**.

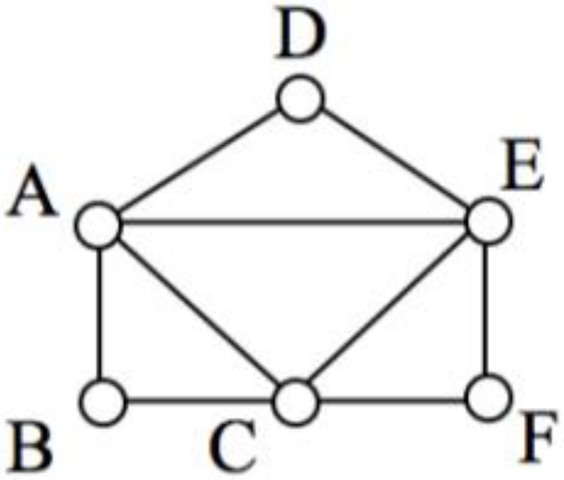
EXAMPLE

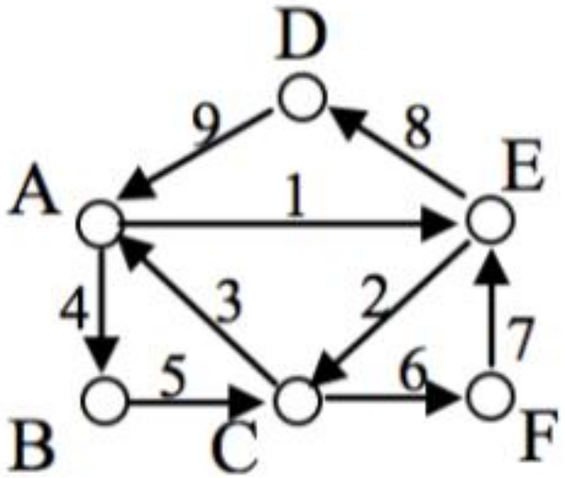




In the graph shown above, there are several **Euler paths**. One such path is **(C,A,B,D,C,B)**. The path is shown in arrows to the right, with the order of edges numbered.

EXAMPLE



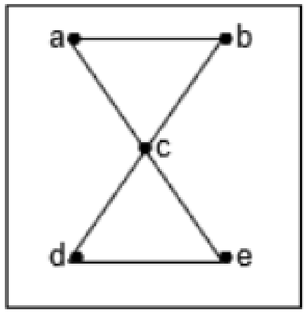


The graph above has several possible **Euler circuits**. Here's a couple, **starting and ending at vertex A:** **(A,D,E,A,C,E,F,C,B,A)** and **(A,E,C,A,B,C,F,E,D,A)**. The second is shown in arrows.

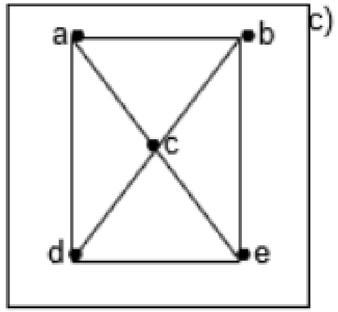
EXAMPLE

Determine whether each of the graph has an Euler circuit or Euler path.

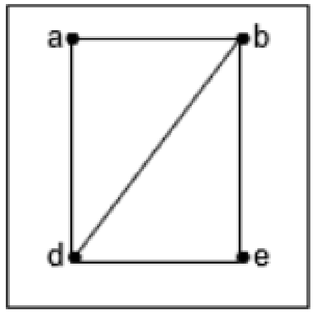
b)

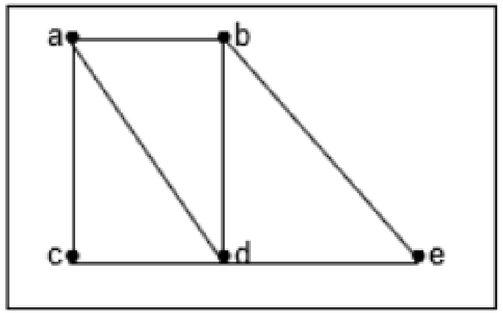


c)



d)





Solution:

a) Euler circuit – all the vertices have an even degree.

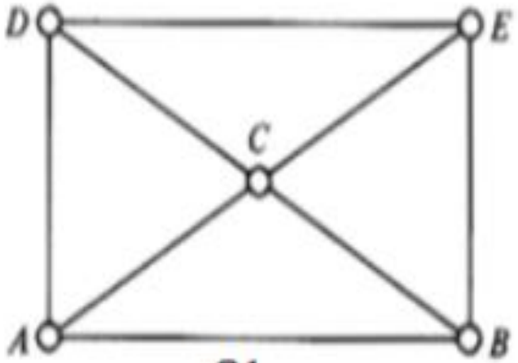
b) Neither Euler circuit nor Euler path.

c) Euler path – two of the vertices have an odd degree.

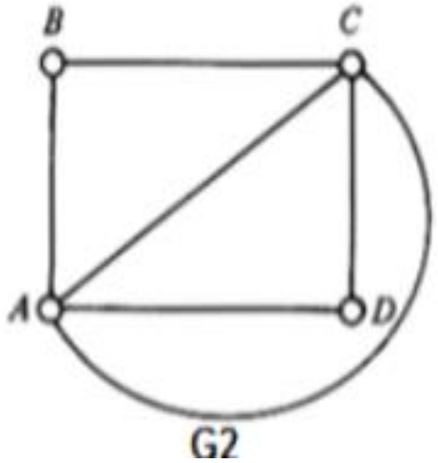
d) Euler path – two of the vertices have an odd degree.

EXERCISE F

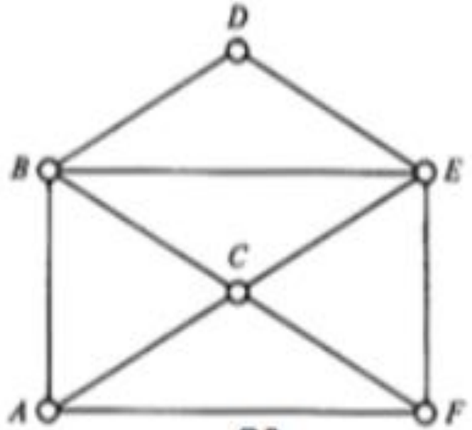
1. Give the characteristic of Euler path and Euler circuit.
2. Which of the following graphs G1, G2 and G3 has Euler path or Euler circuit? If any, list down the Euler path or Euler circuit.



G1



G2



G3

CHAPTER 3: GRAPHS AND TREES

3.1.9 Construct the Hamilton paths and Hamilton circuits in graphs

3.1.10 Apply graphs theories in Travelling Salesman Problem (TSP)

3.2 Follow concept of trees

3.2.1 Describes trees

3.2.2 Identify the properties of trees

3.2.3 Describe the terminology and characteristics of trees

3.2.4 Sketches the spanning trees

3.2.4 Construct the minimal spanning trees using

3.1.4a Prim's Algorithm

3.1.4b Kruskal's Algorithm

3.1.9 HAMILTON PATHS & HAMILTON CIRCUITS

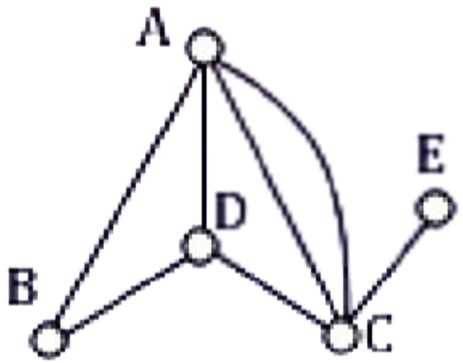
Hamilton circuit in G

- A **Hamilton circuit** is a simple circuit in a graph G that **passes through every vertex exactly once, and starts and finish at the same vertex**.

Hamilton Path in G

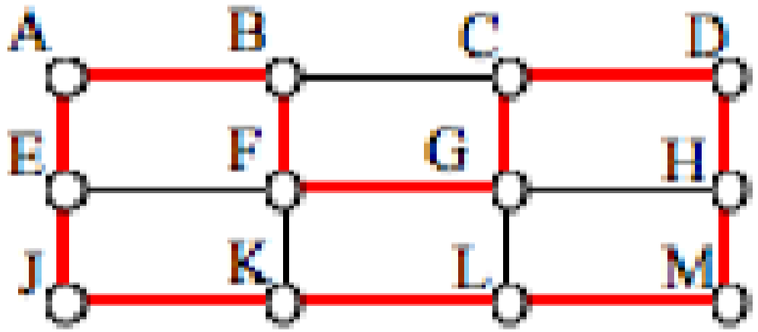
- A **Hamilton path** is a simple path in a graph G that **passes through every vertex exactly once.**

EXAMPLE



We can see that once we travel to vertex E there is **no way** to leave **without returning to C, so there is no possibility of a Hamiltonian circuit**. If we **start at vertex E** we can find several **Hamiltonian paths, such as (E,C,D,A,B) and (E,C,A,B,D)**

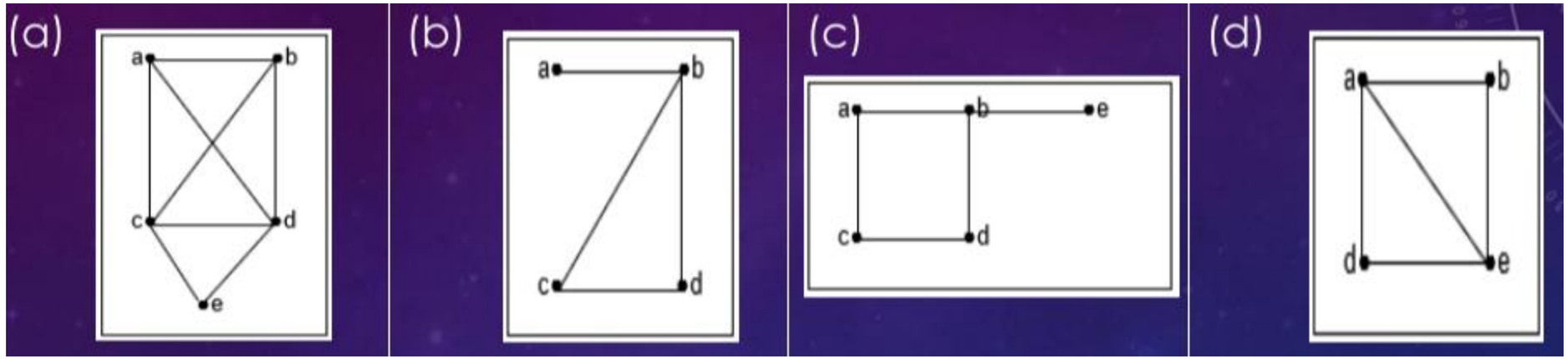
EXAMPLE



- One Hamiltonian circuit is shown on the graph above.
- There are several other Hamiltonian circuits possible on this graph. Notice that the circuit only has to visit every vertex once; it does not need to use every edge.
- This circuit could be notated by **the sequence of vertices visited, starting and ending at the same vertex**: (**A**,B,F,G,C,D,H,M,L,K,J,E,**A**). Notice that the **same circuit could be written in reverse order, or starting and ending at a different vertex**.

EXAMPLE

Determine whether each of the graph has a Hamilton path
or Hamilton circuit or both.



Solution:

a) **Hamilton circuit (a, b, d, e, c, a) & Hamilton path (a, b, c, d, e)**

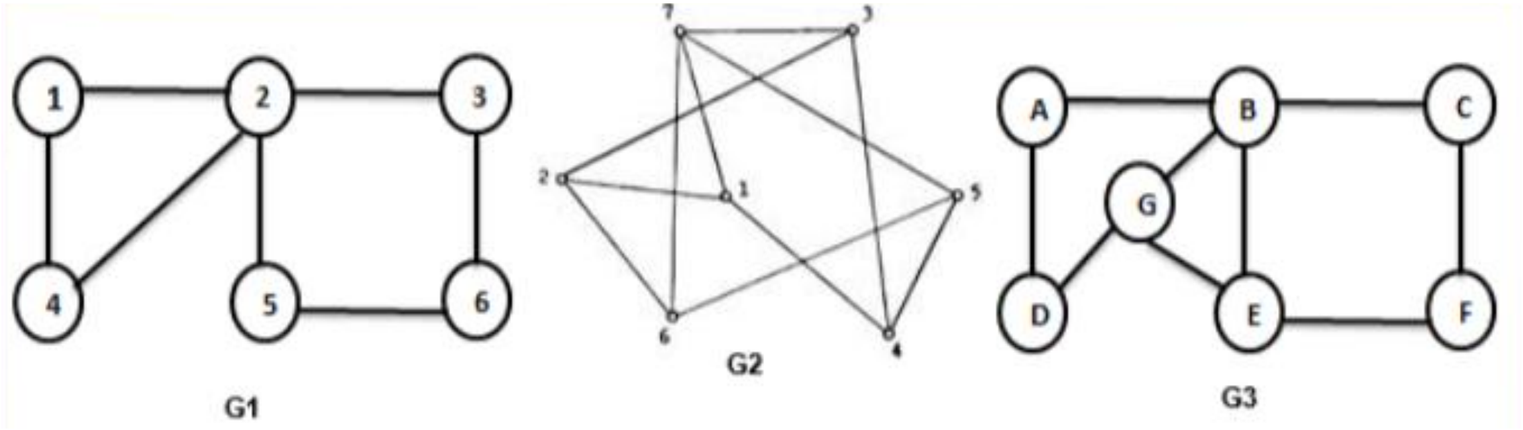
b) **Hamilton path (a, b, c, d)**

c) **Hamilton path (a, c, d, b, e)**

d) **Hamilton circuit (a, b, e, d, a) & Hamilton path (b, a, e, d)**

EXERCISE G

1. Which of the following graphs G1, G2 and G3 has **Hamilton path or Hamilton circuit**? If any, list down the Hamilton path or Hamilton circuit.



DIFFERENCES BETWEEN HAMILTON PATH AND EULER PATH

Hamilton Path

- Covers all the vertices of a graph exactly once. It can contain some edge multiple times

Euler Path

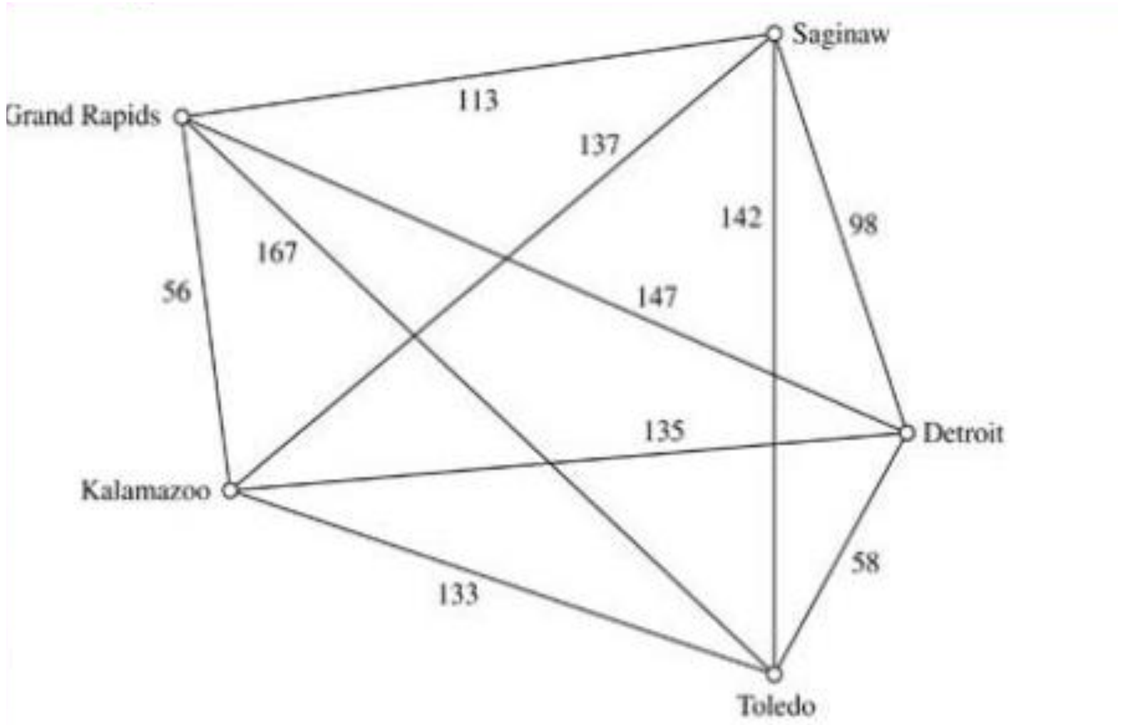
- Covers all the edges of a graph exactly once. It can contain some vertex multiple times.

3.1.10 APPLY GRAPHS THEORIES IN TRAVELLING SALESMAN PROBLEM (TSP)

Traveling Salesman Problem (TSP) is to find the circuit with minimum total weight in a weighted, complete, undirected graph that visits each vertex exactly once and returns to its starting point. It is equivalent to Hamilton circuit concept

STEPS IN SOLVING TSP

EXAMPLE



Find the shortest route that the salesperson could use if he travels to each city in the map below exactly once, starting and ending in Detroit.

SOLUTION

The possible route:

Route	Total Distance
Detroit – Toledo – Grand Rapids – Saginaw – Kalamazoo – Detroit	610
Detroit – Toledo – Grand Rapids — Kalamazoo – Saginaw – Detroit	516
Detroit – Toledo – Kalamazoo – Saginaw – Grand Rapids – Detroit	588
Detroit – Toledo – Kalamazoo – Grand Rapids – Saginaw – Detroit	458
Detroit – Toledo – Saginaw – Kalamazoo – Grand Rapids – Detroit	540
Detroit – Toledo – Saginaw – Grand Rapids – Kalamazoo – Detroit	504
Detroit – Saginaw – Toledo – Grand Rapids – Kalamazoo – Detroit	598
Detroit – Saginaw – Toledo – Kalamazoo – Grand Rapids – Detroit	576
Detroit – Saginaw – Kalamazoo – Toledo – Grand Rapids – Detroit	682
Detroit – Saginaw – Grand Rapids – Toledo – Kalamazoo – Detroit	646
Detroit – Grand Rapids – Saginaw – Toledo – Kalamazoo – Detroit	670
Detroit – Grand Rapids – Toledo – Saginaw – Kalamazoo – Detroit	728

By referring to the table in the slide before,

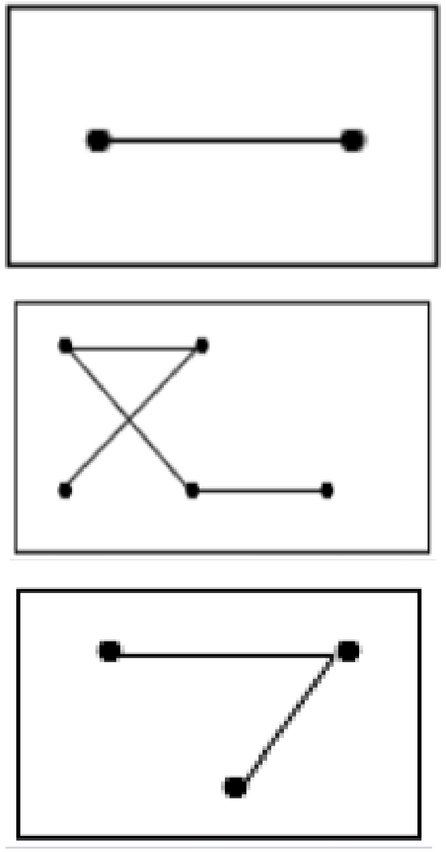
The shortest route is:

Detroit – Toledo – Kalamazoo – Grand Rapids – Saginaw – Detroit

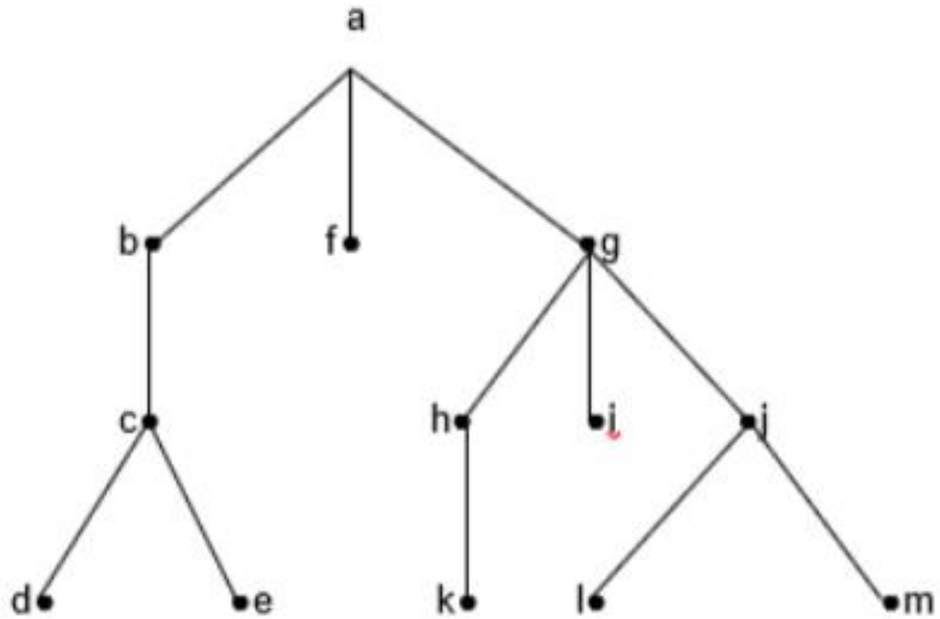
The total distance is 458

3.2.1 DESCRIBES TREES

- A tree is another data structure that you can use to store pieces of information, or rather, a bunch of elements.
- **A tree** is a
  - **connected**
  - **undirected graph** with no simple circuit
  - **cannot contain multiple edges or loops**
- Therefore a tree must be a **simple graph**.



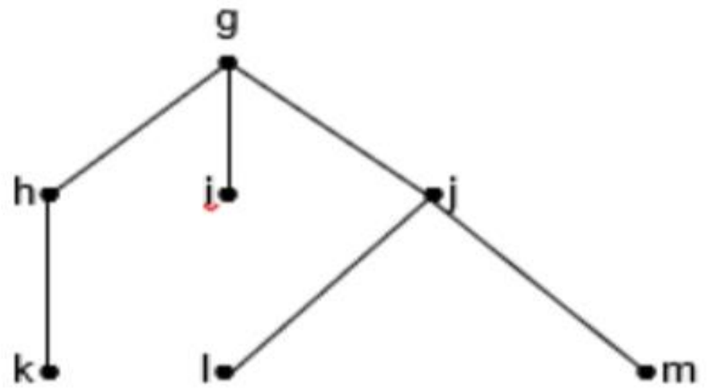
A **rooted tree** is a tree in which one vertex has been designated as the root and every edge is directed away from the root.



A Rooted Tree

- The **root** is **a**.
- The **parent** of **h**,**i** and **j** is **g**.
- The **children** of **b** is **c**. The **children of j** are **l** and **m**.
- **h**, **i** and **j** are a **siblings**.
- The **ancestor** of **e** are **c**, **b** and **a**.
- The **descendants** of **b** are **c**, **d** and **e**.
- The **internal vertices** are **a**, **b**, **c**, **g**, **h** and **j**. (vertices that have children)
- The **leaves** are **d, e, f, i, k, l** and **m**.(vertices that have no children)

If a is a vertex in a tree, the subtree with a as its root is the subgraph of the tree consisting of a and its descendants and
all edges incident to these descendants.



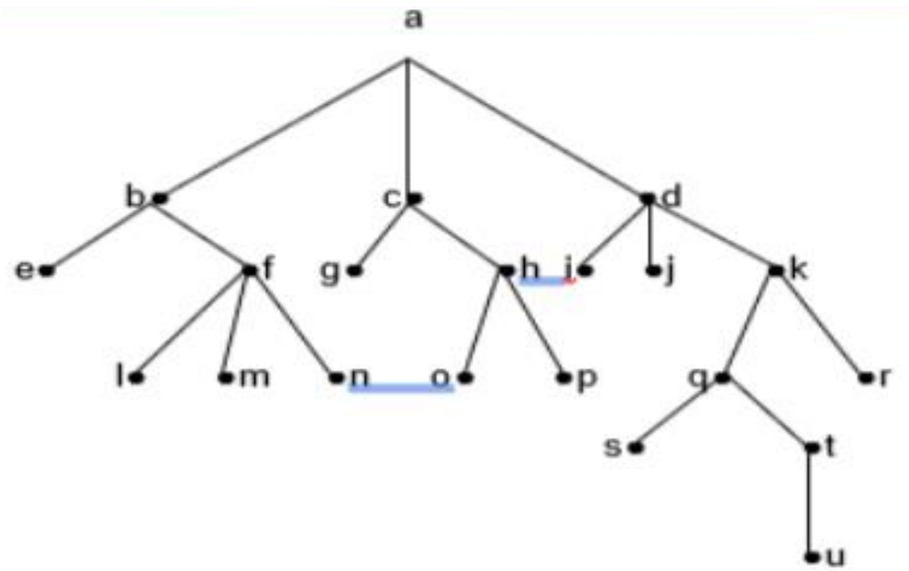
The Subtree Rooted At g

- The **level** of a vertex v in rooted tree is the length of the unique path from the root to this vertex.
- The level of the root is defined to be zero.
- The **height** of a rooted tree is the maximum of the levels of vertices.
- A rooted tree is called an **m-ary tree** if **every internal vertex has no more than m children**.
- The tree is called a **full m-ary tree** if **every internal vertex has exactly m children**.
- An **m-ary tree** with **m = 2** is called a **binary tree**.

3.2.2 THE PROPERTIES OF TREES

- A tree with n vertices has $\boldsymbol{n - 1}$ edges. (no. of edges = n – 1; n is no. of vertices)
- A full m-ary tree with i internal vertices contains $\boldsymbol{n = mi + 1}$ vertices
- A full m-ary tree with:
  - n vertices has $i = \frac{(n-1)}{m}$ internal vertices and $l = \frac{[(m-1)n+1]}{m}$ leaves.
  - i internal vertices has $n = mi + 1$ vertices and $l = (m - 1)i + 1$ leaves.
  - l leaves has $n = \frac{(ml-1)}{(m-1)}$ vertices and $i = \frac{(l-1)}{(m-1)}$ internal vertices.

Example



Question:

a) Which vertex is the root?

b) Which vertices are internal?

c) Which vertices are leaves?

d) Which vertices are children of k?

e) Which vertex is the parent of h?

f) Which vertices are siblings of o?

g) Which vertices are ancestors of m?

h) Which vertices are descendants of b?

i) Is the rooted tree above a full m-ary tree for some positive integer m?

j) What is the level of each vertex of the rooted tree above?

k) Draw a subtree that is rooted at d.

Solution:

a) A

b) a, b, c, d, f, h, k, q, t

c) e, g, i, j, l, m, n, o, p, r, s, u

d) q and r

e) C

f) P

g) f, b, a

h) e, f, l, m, n

i) No

j)

Level 0 = a

Level 1 = b, c, d

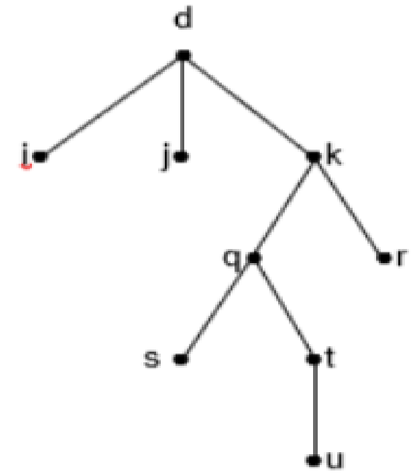
Level 2 = e, f, g, h, i, j, k

Level 3 = l, m, n, o, p, q, r

Level 4 = s, t

Level 5 = u

k)



Example:

a) How many edges does a tree with 10,000 vertices have?

b) How many vertices does a full 5-ary tree with 100 internal vertices have?

c) How many edges does a full binary tree with 1000 internal vertices have?

d) How many leaves does a full 3-ary tree with 100 vertices have?

Solution:

a) 10000 – 1 = 9999 edges

b) n = mi + 1 = 5 100 + 1 = 501 vertices

c) n = mi + 1 = 2 1000 + 1 = 2001 vertices

d) $l = \frac{[(m-1)n+1]}{m} = \frac{[(3-1)(100)+1]}{3} = 67$ leaves

Exercise H

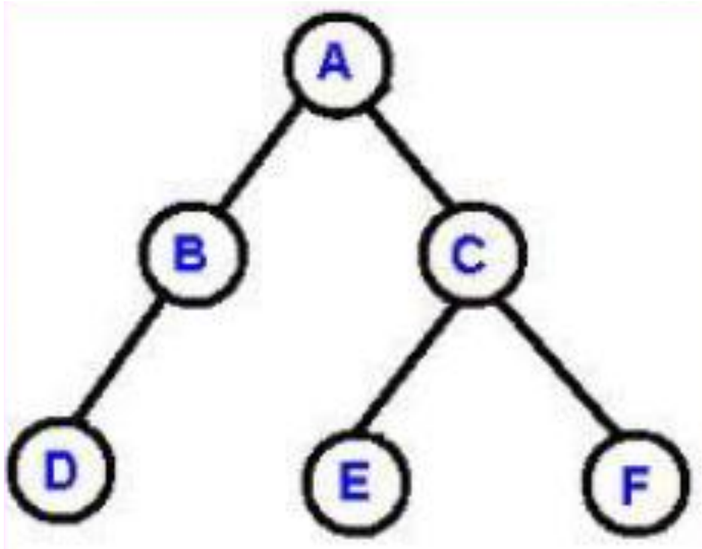
a) How many edges does a full 3-ary tree with 500 internal vertices have?

b) How many leaves does a full binary tree with 1001 vertices have?

3.2.3 DESCRIBE THE TERMINOLOGY AND CHARACTERISTICS OF TREES

- **Binary trees** are among the most important type of rooted trees.
- **Every vertex** in a binary tree has **at most two children**.
- Each child is either a **left child** or a **right child**.

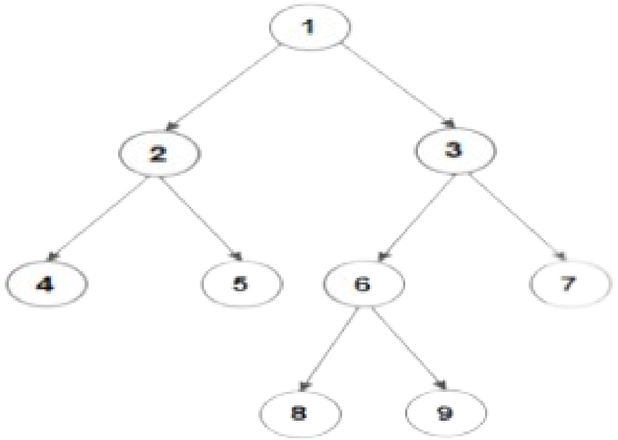
EXAMPLE



- **Parent** = A
- **Left child of vertex A** = vertex B
- **Right child of vertex A** = vertex C

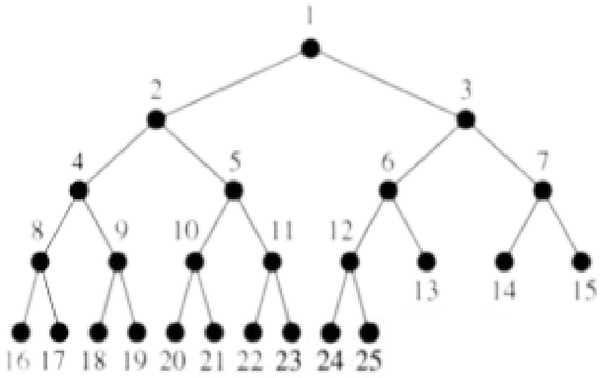
- **Full binary tree** is a binary tree in which each vertex has either two children or no children at all.
- A rooted binary tree is a rooted tree in which every vertex has at most two children.
- **A full binary tree** is a tree in which **every vertex other than the leaves has two children**.
- A **complete binary tree** is a **binary tree in which every level except possibly the last is completely filled**.

EXAMPLE



FULL BINARY TREE

COMPLETE BINARY TREE



Complete Binary Tree

Exercise 1

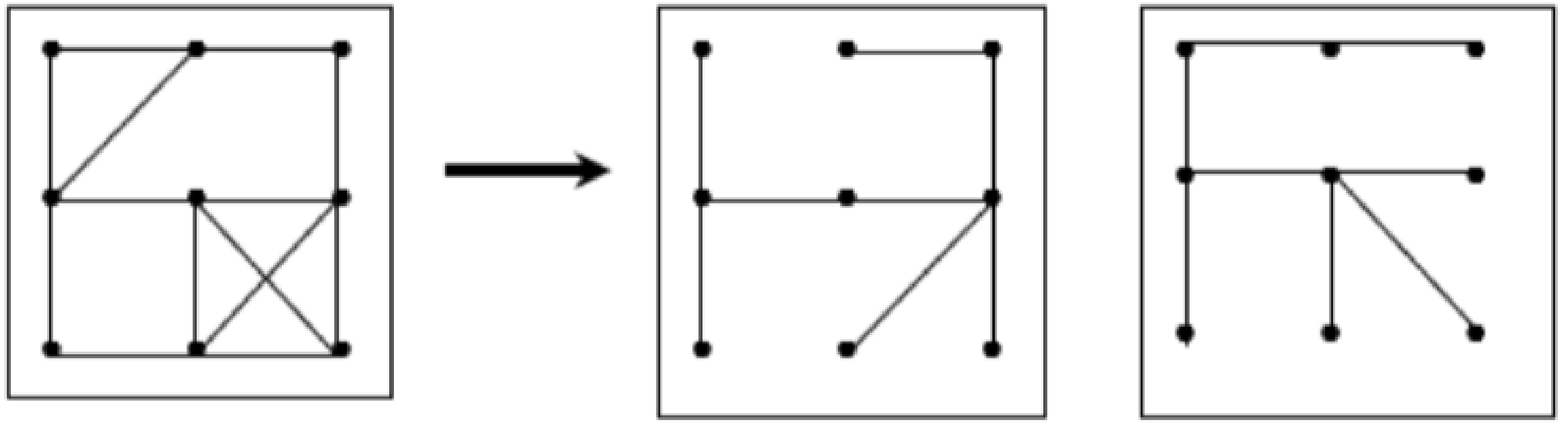
Draw a rooted tree having the following properties:

a) Full binary tree, four internal vertices and five terminal vertices (leaves)

b) Full binary tree, six internal vertices and seven terminal vertices

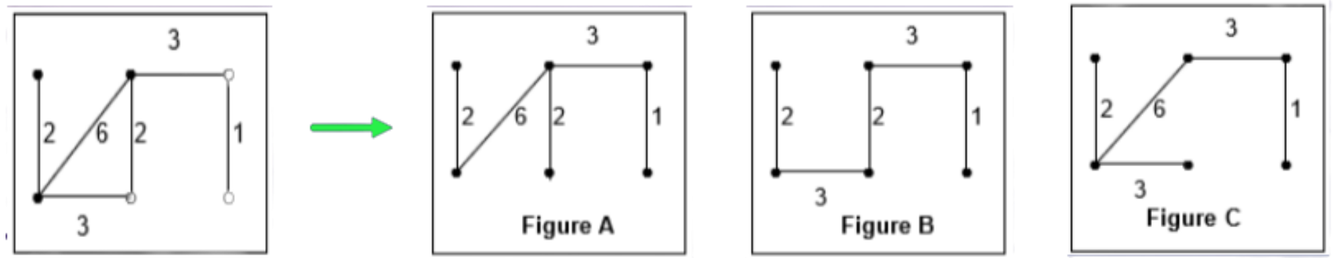
3.2.4 SPANNING TREES

- A spanning tree of G is a **subgraph** of G that is a tree **containing every vertex of G**.
- A simple graph G with a spanning tree must be **connected** because there is a path in the spanning tree between any two vertices.
- Consider the graph below:



3.2.4 SKETCHES MINIMAL SPANNING TREES

- A **minimum spanning tree** in a **connected weighted graph** is a **spanning tree that has the smallest possible sum of weights of its edges**.
- Consider the graph below: **(WITH 3 DIFFERENT SPANNING TREE)**



- Minimum Weight for each of the spanning tree : Figure A=14; Figure B=11 and Figure C=15.
- Therefore, **Figure B has the minimum spanning tree because it has the smallest sum of weight**.

3.2.5 CONSTRUCT MINIMAL SPANNING TREES

Prim's Algorithm

- It has a **starting point**.

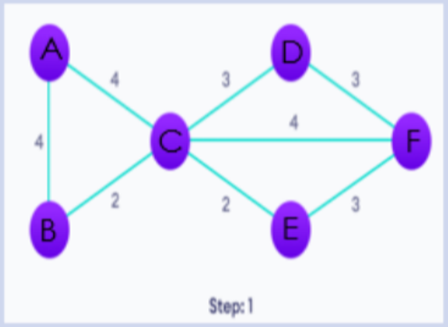
The steps for implementing Prim's algorithm are as follows:

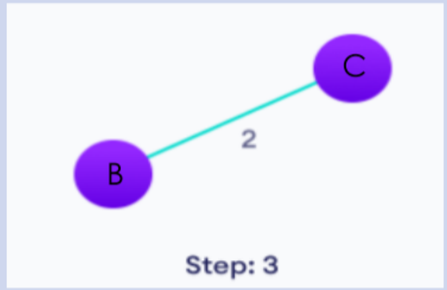
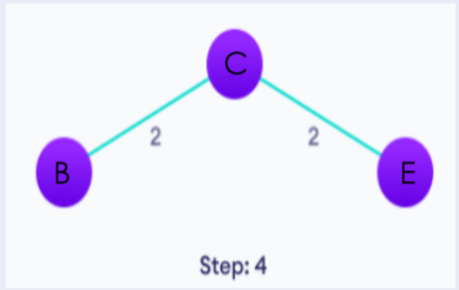
- Initialize the minimum spanning tree with a vertex chosen at random.
- **Find all the edges that connect the tree to new vertices, find the minimum and add it to the tree**
- Keep repeating step 2 until we get a minimum spanning tree

Kruskal's Algorithm

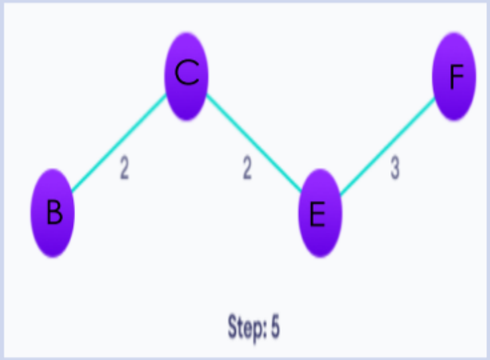
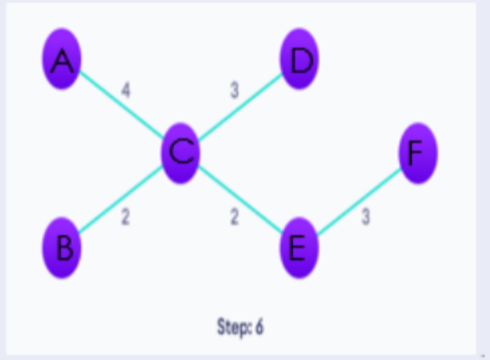
- start with the minimum weight of an edge

PRIM'S ALGORITHM

STEPS	EXPLANATION
1. 	Start with a weighted graph
2. 	Choose a vertex (For this case, we choose C as the starting point)

STEPS	EXPLANATION
3. 	Choose the shortest edge from this vertex and add it
4. 	Choose the nearest vertex not yet in the solution

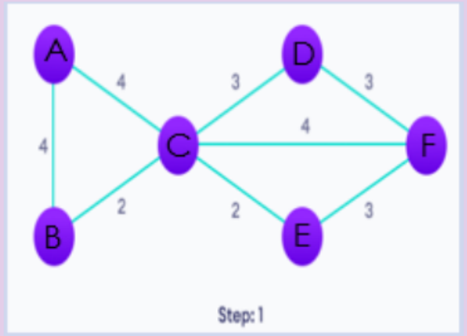
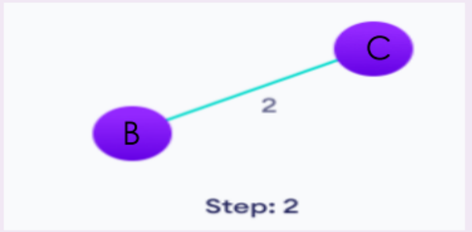
PRIM'S ALGORITHM

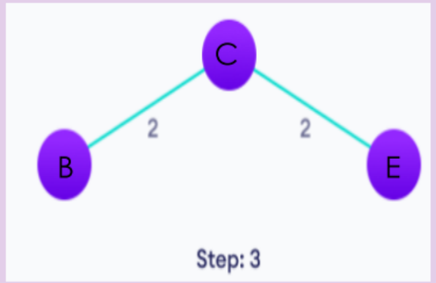
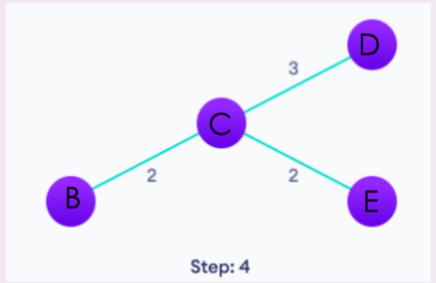
STEPS	EXPLANATION
5. 	Choose the nearest edge not yet in the solution, if there are multiple choices, choose one at random
6. 	Repeat until you have a spanning tree

SUMMARY OF PRIM'S ALGORITHM (VERTEX C AS THE STARTING POINT)

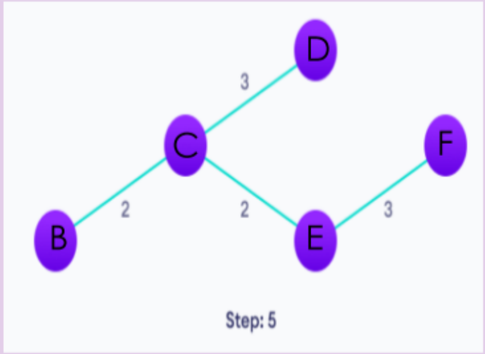
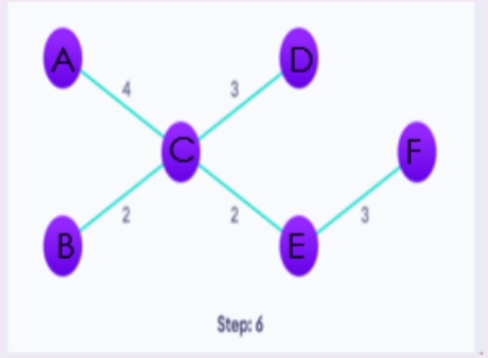
FROM	TO	WEIGHT
C	B	2
C	E	2
E	F	3
C	D	3
C	A	4
		TOTAL WEIGHT=14

KRUSKAL'S ALGORITHM

STEPS	EXPLANATION
1. 	Start with a weighted graph
2. 	Choose the edge with the least weight, if there are more than 1, choose anyone

STEP	EXPLANATION
3. 	Choose the next shortest edge and add it
4. 	Choose the next shortest edge that doesn't create a cycle and add it

KRUSKAL'S ALGORITHM

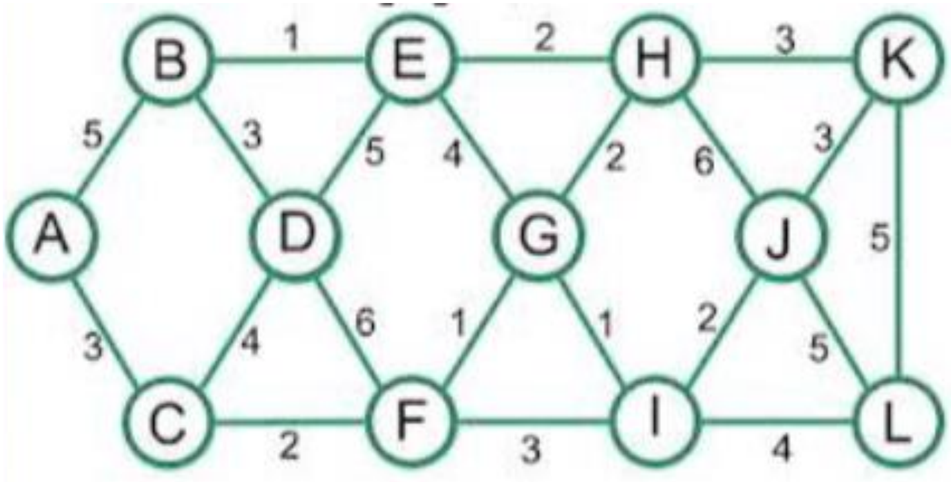
STEP	EXPLANATION
5. 	Choose the next shortest edge that doesn't create a cycle and add it
6. 	Repeat until you have a spanning tree

SUMMARY OF KRUSKAL'S ALGORITHM

FROM	TO	WEIGHT
B	C	2
C	E	2
C	D	3
E	F	3
C	A	4
		TOTAL WEIGHT=14

EXERCISE J

a) Based on the following figure,



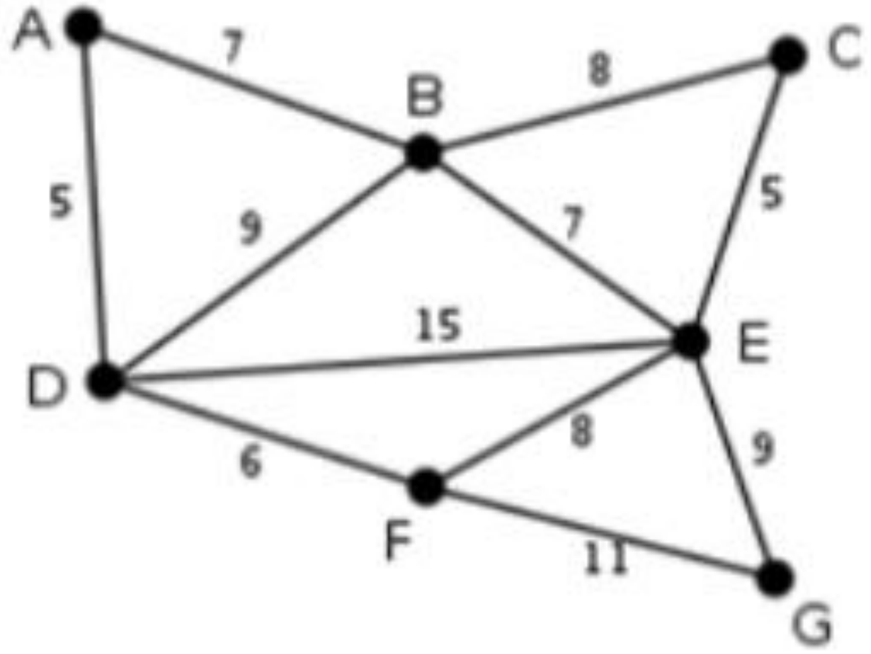
i. Draw 2 possible spanning trees

ii. Find a minimum spanning tree by using Kruskal's and Prim's algorithms (compare the findings)

iii. Draw the minimum spanning tree.

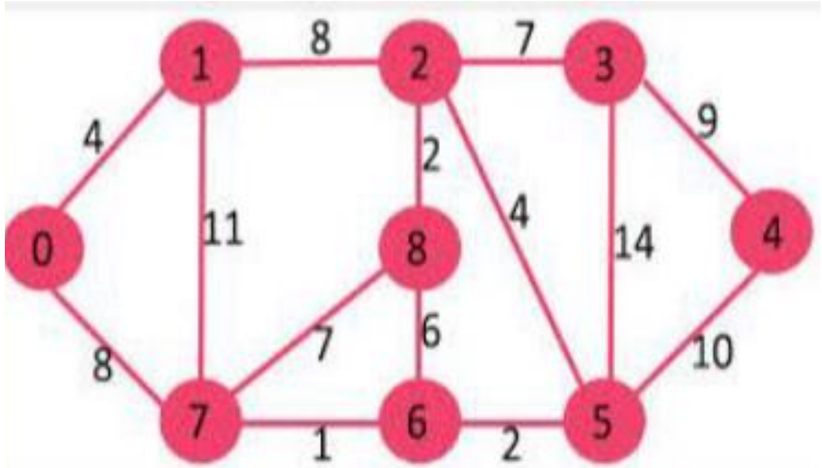
Exercise J

b) By referring to the following weighted graph, find



EXERCISE J

c) Based on the following graph:



i. Draw the minimum spanning tree by using Prim's algorithm.

ii. Calculate the shortest path.

CHAPTER 3: GRAPHS AND TREES

3.2.6 Build a binary search tree

3.2.7 Organize tree traversals:

- 3.1.5a Pre-order traversal
- 3.1.5b In-order traversal
- 3.1.5c Post-order traversal

3.2.8 Distinguish the full binary trees and complete binary trees

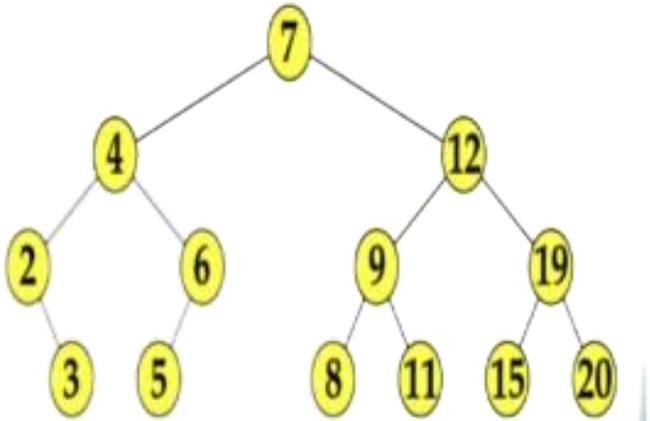
3.2.9 Associate tree theories in searching and Travelling Salesman Problem (TSP) solving.

3.2.6 BUILD A BINARY SEARCH TREE

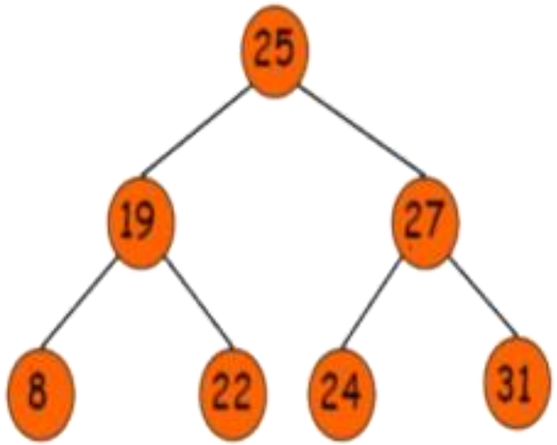
A **Binary Search Tree (BST)** is a tree in which all the nodes follow the below-mentioned properties:

- The **value** of the key of the **left sub-tree is less than the value of its parent** (root) node's key.
- The **value** of the key of the **right sub-tree is greater than or equal to the value of its parent** (root) node's key.

EXAMPLE



BST



NOT BST
(SINCE 24 IS
LESS THAN 25)

STEP 1

Insert the first data item in a vertex (label as the root)

STEP 2

Insert the next data items in the tree accordingly. Begin at the root, if the **data item to be added is less than the current vertex, move it to the left (as left child).** Put an edge incident on the vertex.

STEP 3

If the **data item is greater than the current vertex, move it to the right.**

STEP 4

Repeat until all data item was added into the binary search tree.

EXAMPLE

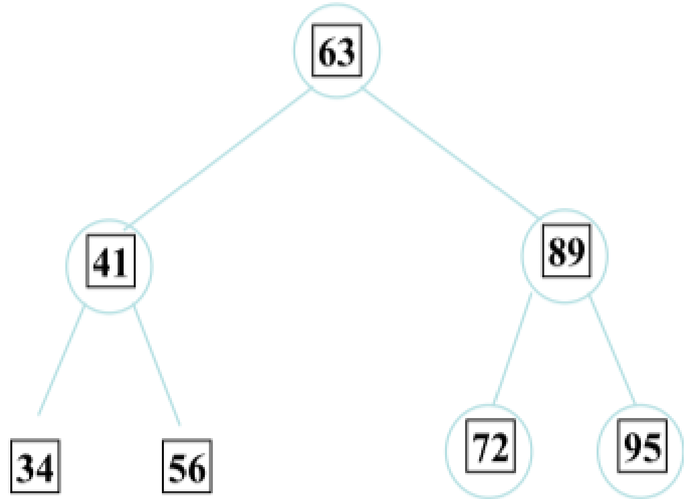
Construct a BST from set of data 63 89 72 41 56 95 34.

Solution:

Label each number given according to the sequences.

63	89	72	41	56	95	34
1	2	3	4	5	6	7

The **first number will become the root**. You need to follow the steps.



EXAMPLE

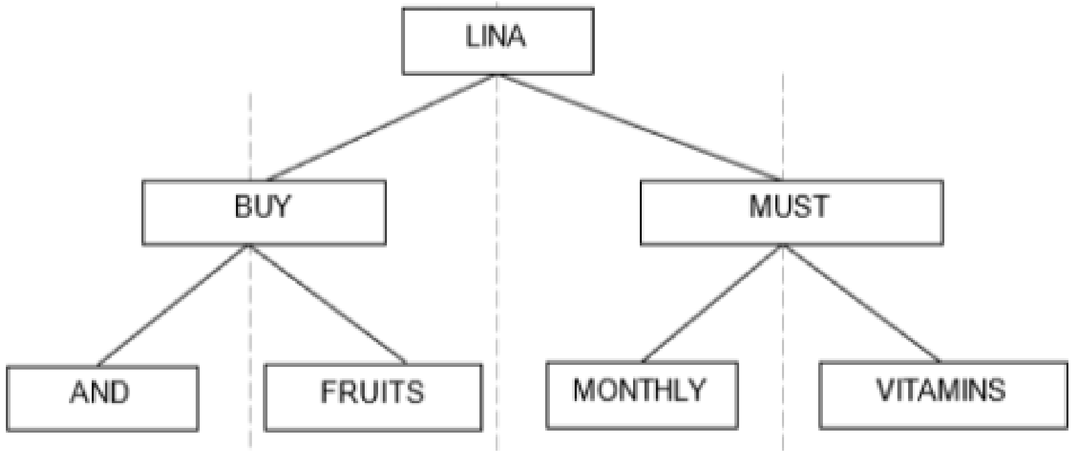
Construct a BST from the words “LINA MUST BUY FRUITS AND VITAMINS MONTHLY”.

Solution:

Label each word given according to the sequences.

LINA	MUST	BUY	FRUITS	AND	VITAMINS	MONTHLY
1	**2**	**3**	**4**	**5**	**6**	**7**

The **first number will become the root**. You need to follow the steps.



Tips:

Arrange the data items which in left and right of the root, apply the alphabetical order.

EXAMPLE

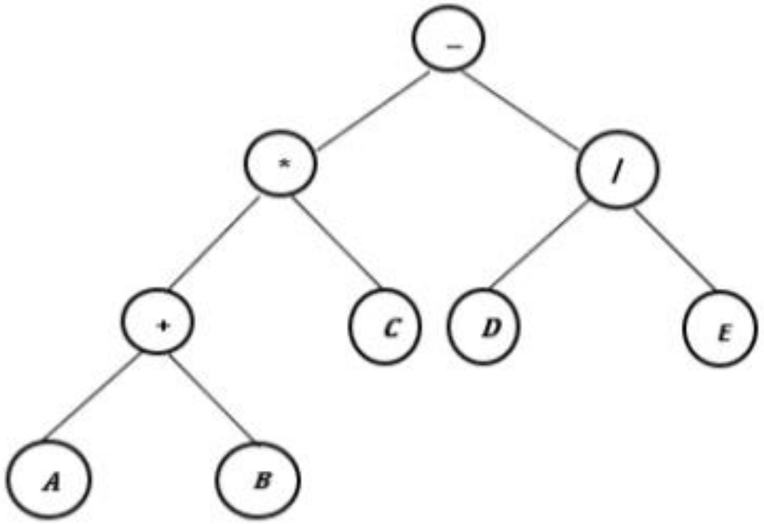
Construct a BST for the given arithmetic expression : $A + B * C - D/E$

Solution:

Label each word given according to the sequences.

$$(A \quad + \quad B \quad * \quad C \quad - \quad D \quad / \quad E)$$
$$1 \quad 2 \quad 3 \quad 4 \quad 5 \quad 6 \quad 7 \quad 8 \quad 9$$

The **first number will become the root**. You need to follow the steps.



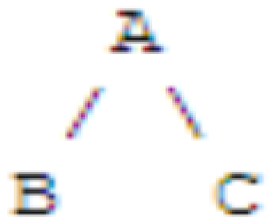
Tips:

Arrange the data items where the internal vertices correspond to the operators and the terminal vertices correspond to the operands.

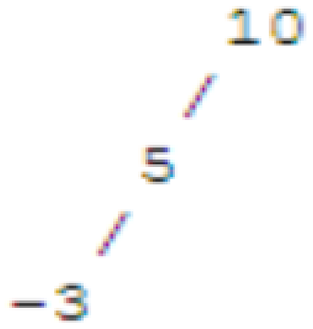
Exercise K

Which of the following binary trees are BSTs? If a tree is **not** a BST, why?

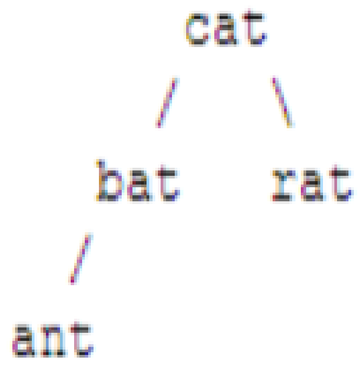
a)



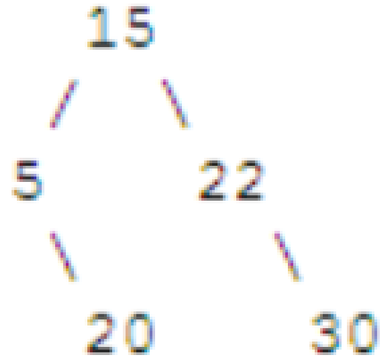
b)



c)



d)



Exercise L

1. Create a BST from the following set of data, with the first number is the root

   a) 14 4 17 19 15 7 9 3 16 10

   b) 9 23 17 4 30 7 14 16 15 19

   c) 19 27 16 15 14 30 39 17 9 3

2. Create a BST from the following sequence of words (sentence), with the first word is the root.

   a) MOST STUDENTS IN THIS CLASS HAS STUDIED BASIC LOGIC

   b) EVERYTHING HAPPENS FOR A REASON, EVEN WHEN WE ARE NOT WISE TO SEE IT

Exercise L

3. Represent the following arithmetic expression as binary tree:

a) (A + B) ∗ (C − D)

b) (((A + B ∗ C + D) ∗ E) − ((A + B) ∗ (C − D))

3.2.7 ORGANIZE TREE TRAVERSALS

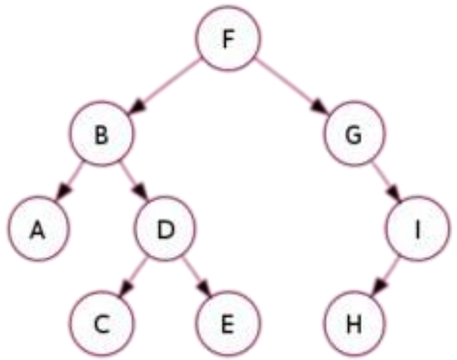
- Tree traversals is a process of visiting (examining and/or updating) each node in a trees, exactly once, in a systematic way.
- Method of searching Binary Tree:
  - **1) Pre-order traversal**
  - **2) In-order traversal**
  - **3) Post-order traversal**

PRE-ORDER TRAVERSAL

To traverse a non-empty binary tree in pre-order, perform the following operations recursively at each node, starting with the root node:

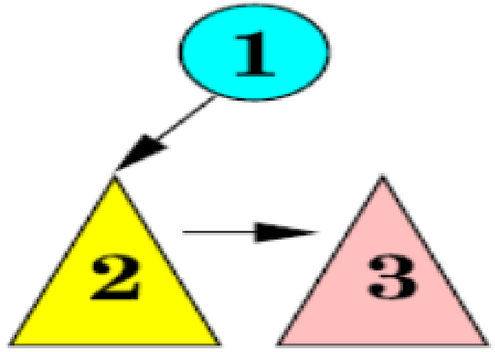
- Visit the root.
- Traverse the left subtree.
- Traverse the right subtree.

Example:



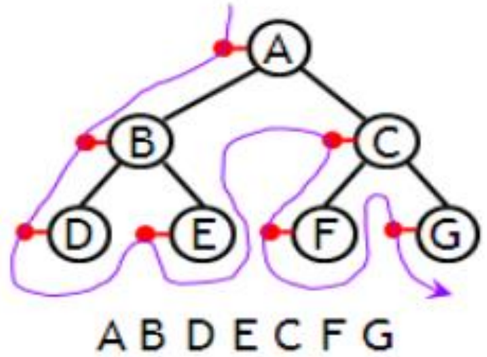
Pre-order: **F, B, A, D, C, E, G, I, H**

Example:



Pre-order: **1, 2, 3**

Example:



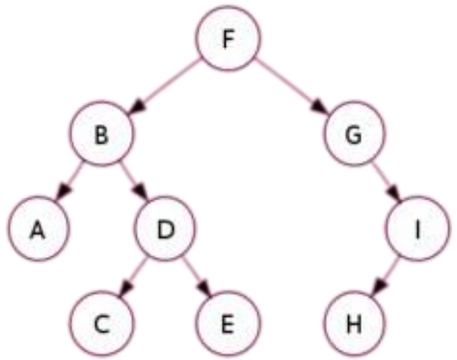
Pre-order: **A, B, D, E, C, F, G**

IN-ORDER TRAVERSAL

To traverse a non-empty binary tree in pre-order, perform the following operations recursively at each node, starting with the root node:

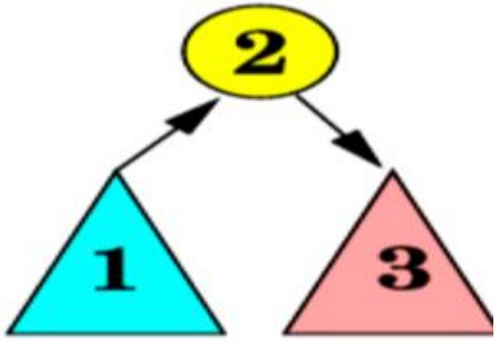
- Traverse the left subtree.
- Visit the root.
- Traverse the right subtree.

Example:



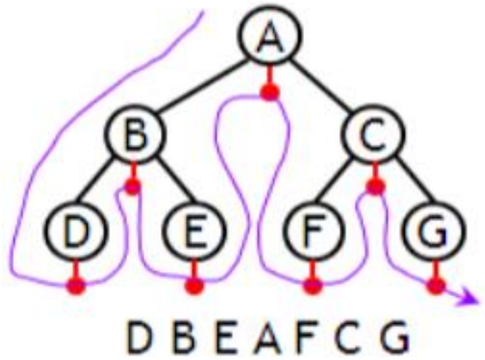
In-order: **A, B, C, D, E, F, G, H, I**

Example:



In-order: **1, 2, 3**

Example:



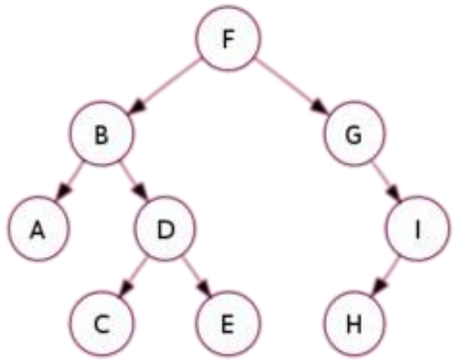
In-order: **D, B, E, A, F, C, G**

POST-ORDER TRAVERSAL

To traverse a non-empty binary tree in pre-order, perform the following operations recursively at each node, starting with the root node:

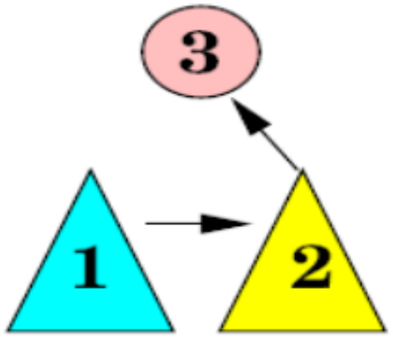
- Traverse the left subtree.
- Traverse the right subtree.
- Visit the root.

Example:



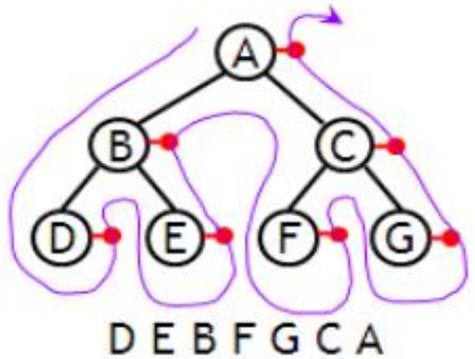
Post-order: **A,C,E,D,B,H,I,G,F**

Example:



Post-order: **1, 2, 3**

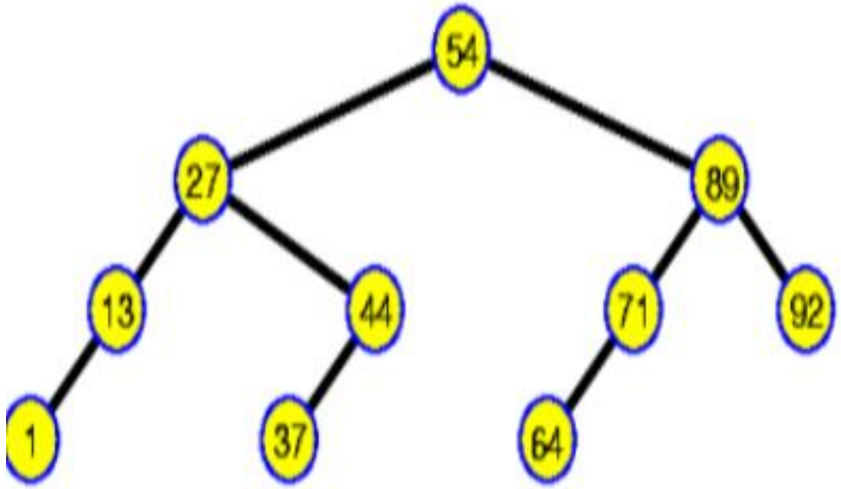
Example:



Post-order: **D, E, B, F, G, C, A**

EXAMPLE

Find the tree traversals (pre-order, in-order & post-order) of the following tree:



Solution:

Pre-order : 54,27,13,1,44,37,89,71,64,92

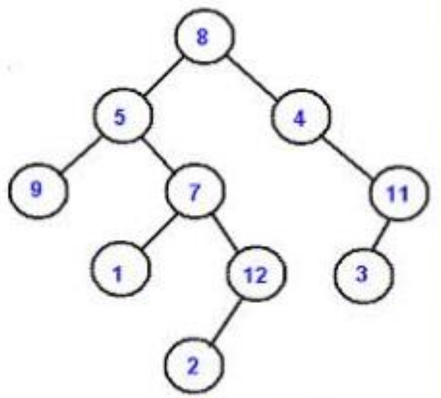
In-order : 1,13,27,37,44,54,64,71,89,92

Post-order :1,13,37,44,27,64,71,92,89,54

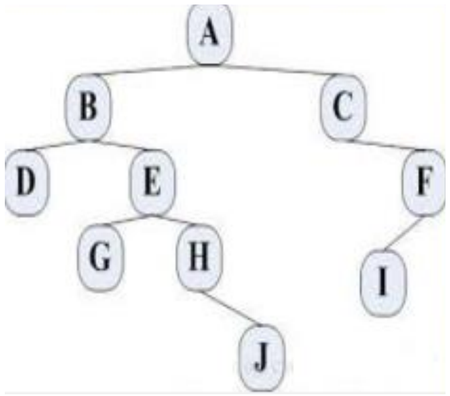
Exercise M

Find the tree traversals (pre-order, in-order & post-order) of the following tree:

a)

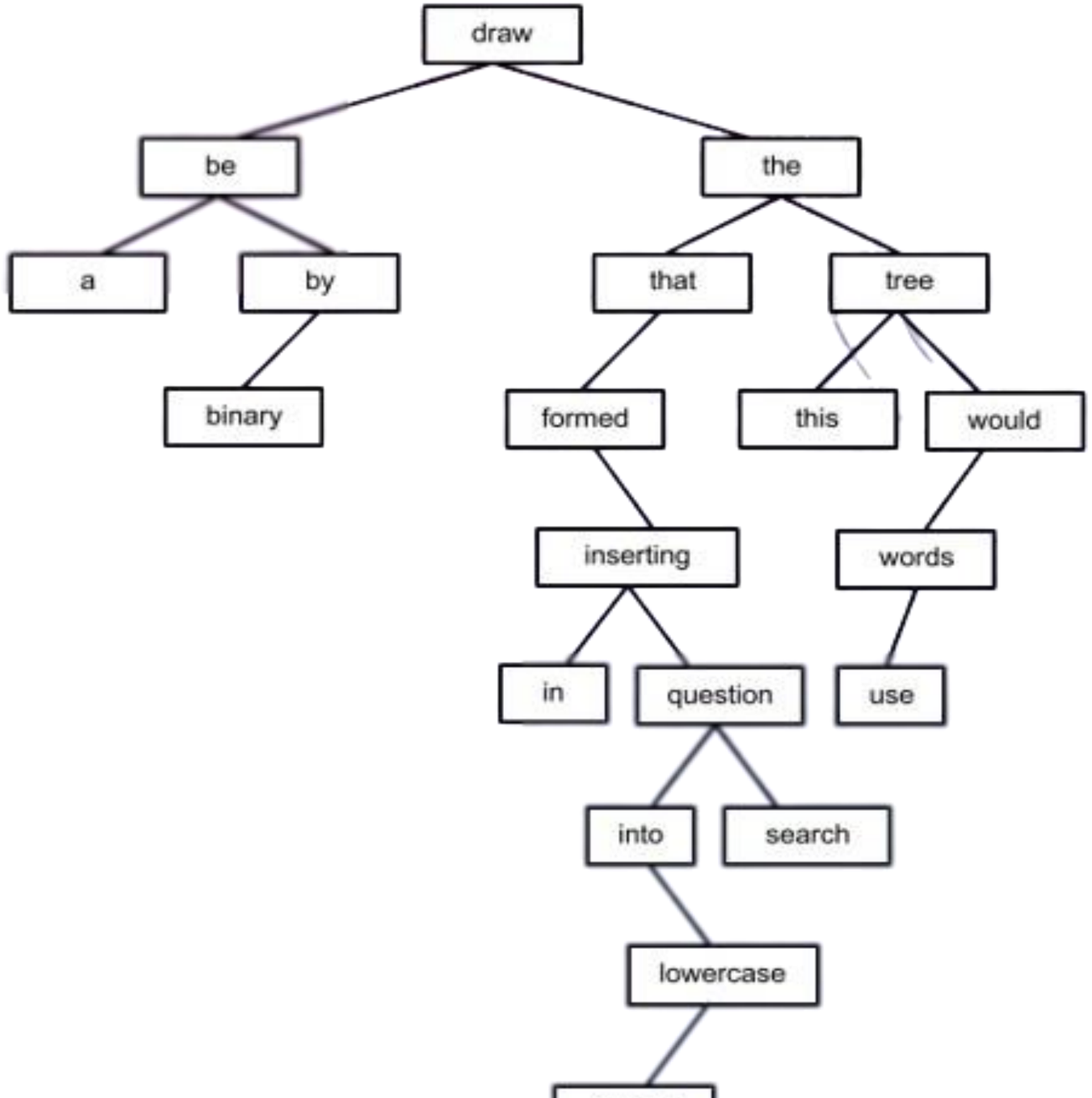


b)



EXERCISE M

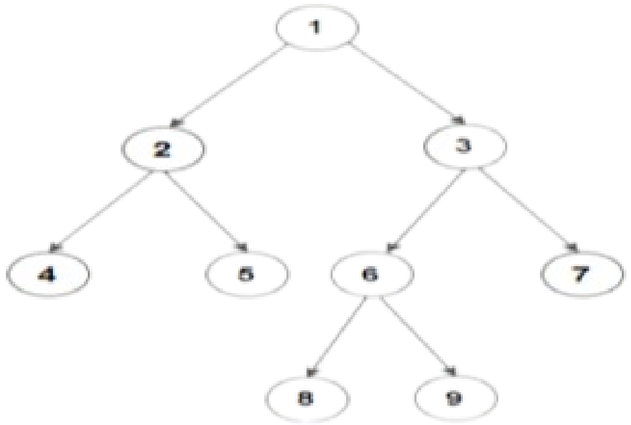
c)


draw
be
the
a
by
that
tree
binary
formed
this
would
inserting
words
in
question
use
into
search
lowercase

3.2.8 DISTINGUISH FULL BINARY TREES AND COMPLETE BINARY TREES

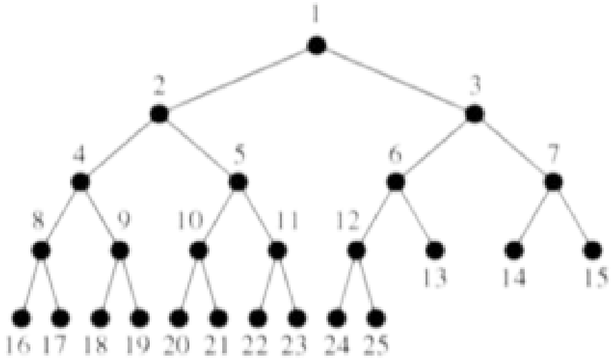
- **Full binary tree** is a binary tree in which each vertex has either two children or no children at all.
- A rooted binary tree is a rooted tree in which every vertex has at most two children.
- **A full binary tree** is a tree in which **every vertex other than the leaves has two children**.
- A **complete binary tree** is a **binary tree in which every level except possibly the last is completely filled**.

EXAMPLE



FULL BINARY TREE

COMPLETE BINARY TREE



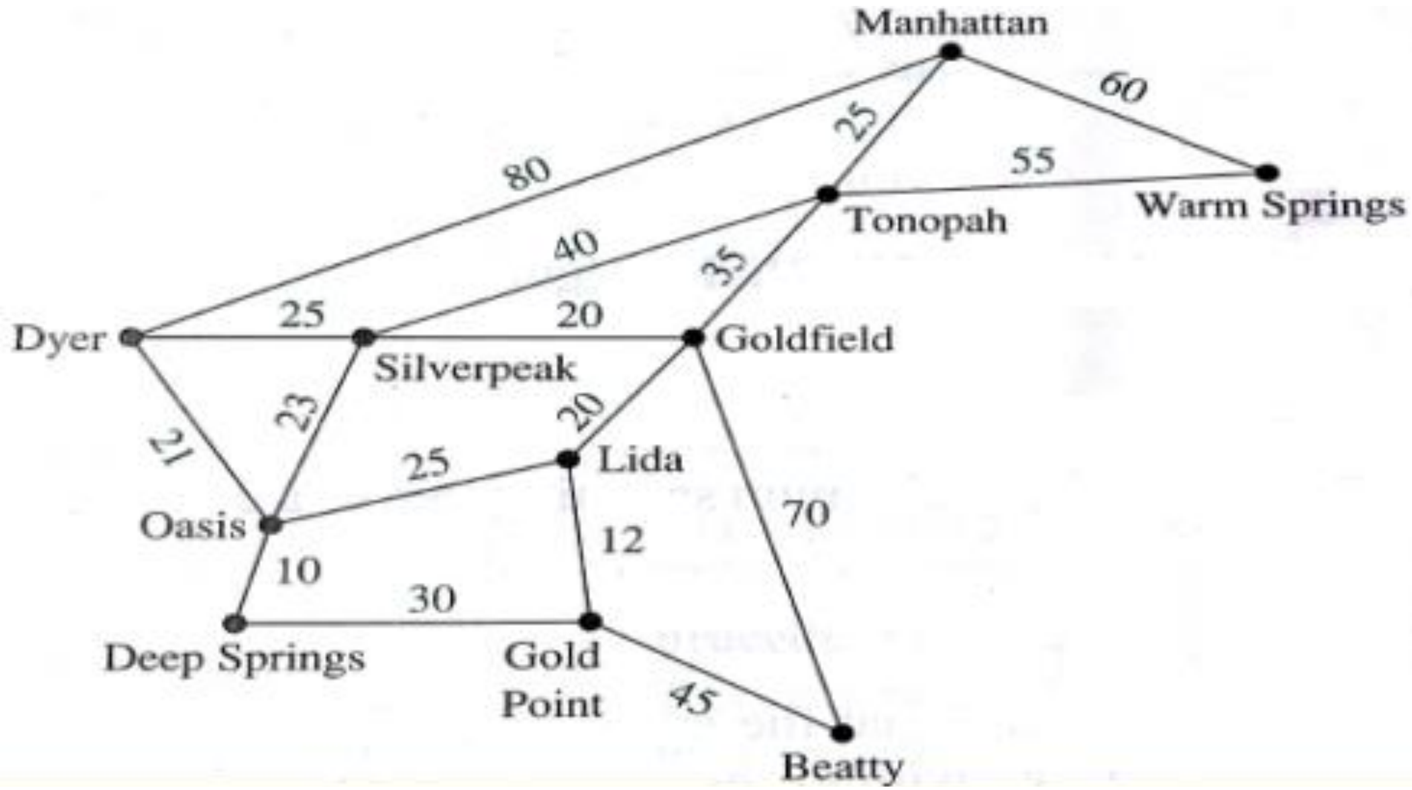
Complete Binary Tree

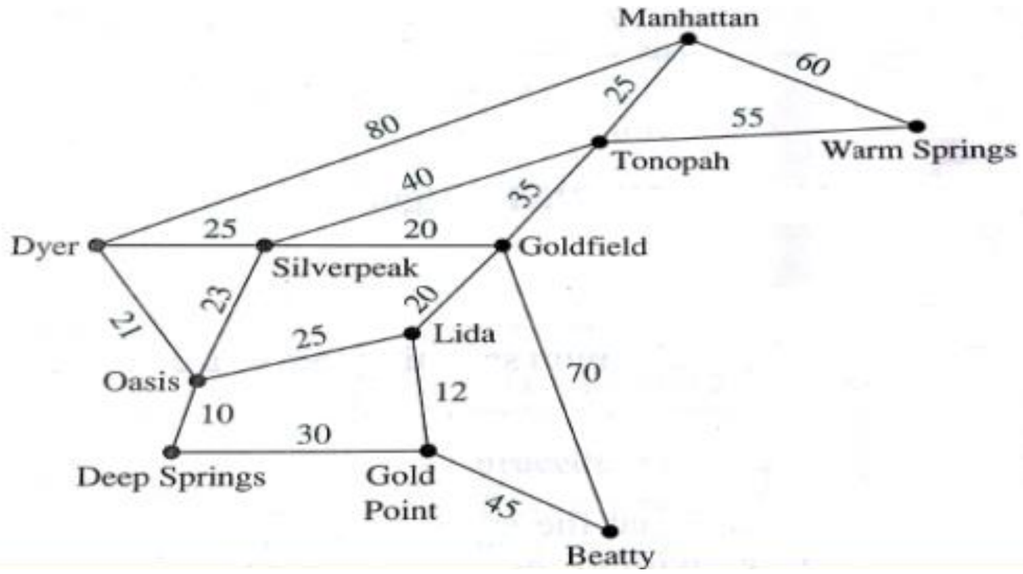
3.2.9 TRAVELLING SALESMAN PROBLEM (TSP) IN TREE THEORY

Apply tree theories in searching and Travelling Salesman Problem (TSP) solving. Normally it will use Prim’s Algorithm.

EXAMPLE

From the map below, find the shortest route that the salesperson could use to travel to each city exactly once starting at Manhattan.





Solution:

By using **Prim's Algorithm**, we will get: