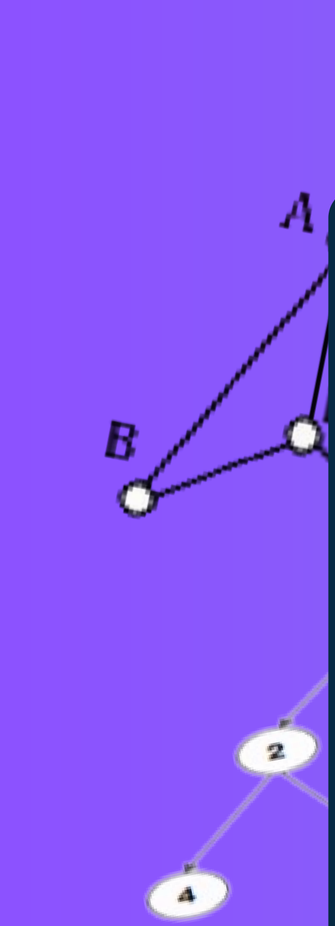
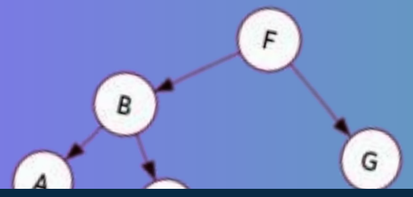
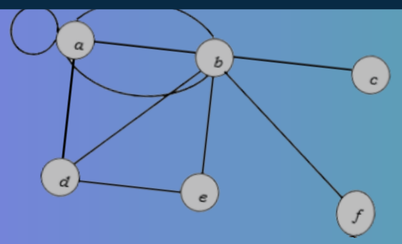
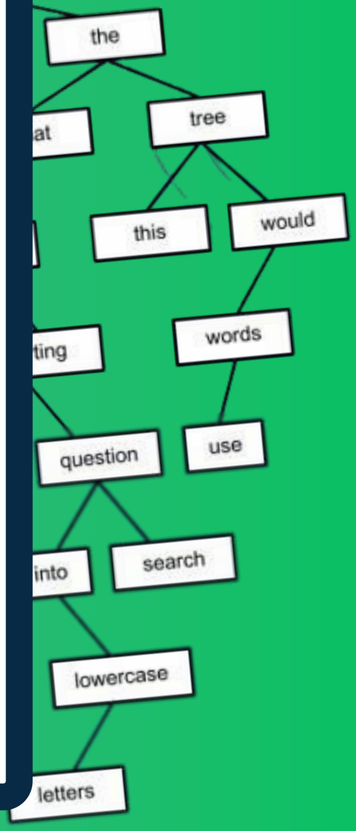




DBM20153-DISCRETE MATHEMATICS

Chapter 2





CHAPTER 2: DIRECTED GRAPHS

2.1 Discover concepts of Graphs

- 2.1.1 Explain the properties of directed graphs
- 2.1.2 Identify subgraph
- 2.1.3 Construct paths and cycles.
  - a) Hamiltonian
  - b) Eulerian

2.2 Discover concept of Trees.

- 2.2.1 Explain spanning trees
- 2.2.2 Construct the minimum spanning trees
  - a) Kruskal’s algorithm
  - b) Prim’s algorithm

2.1 DISCOVER CONCEPT OF GRAPHS

- 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.**

Directed graphs

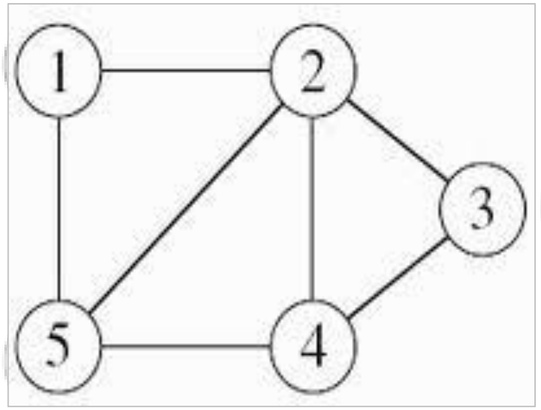
Digraphs (directed graph)

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

Differences between undirected & directed graphs

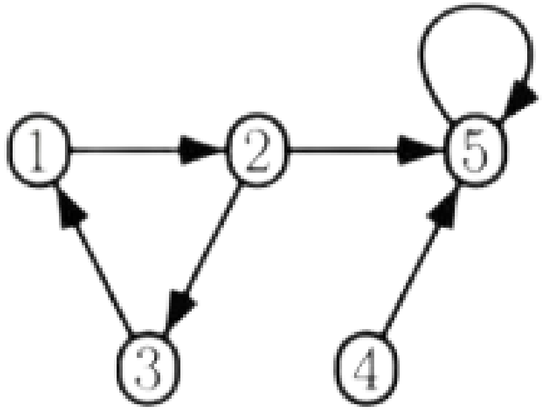
An undirected graph G consists of set V of vertices and set E of edges such that each edge is associated with an unordered pair of vertices.

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.

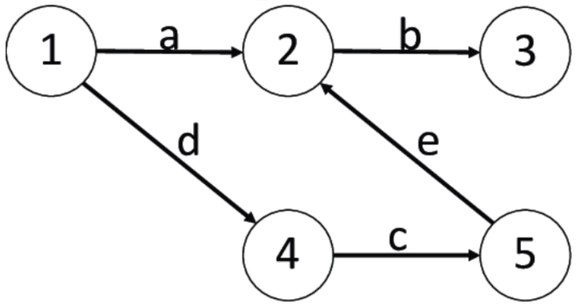
Example:



2.1.1 PROPERTIES OF DIRECTED GRAPHS

VERTICES/NODES

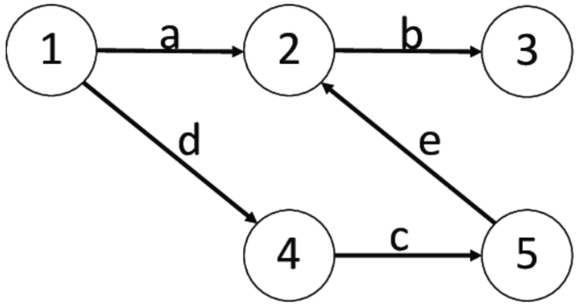
A set of $\boldsymbol{V} = \boldsymbol{V}\ (\boldsymbol{G})$ whose elements are called **vertices, points or nodes**.



Vertices, V ={1, 2, 3,4, 5}

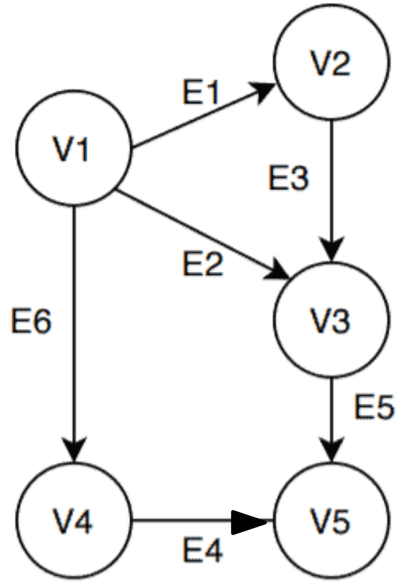
EDGES

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



Edges, E = {a, b, c,d, e}

EXAMPLE



- Vertices, V = V1, V2, V3, V4, V5
- Edges, E: E = {E1, E2, E3, E4, E5} = {{v1, v2} , {v1, v3} , {v2, v3} , {v3, v5} , {v4, v5}}

INDEGREE & OUTDEGREE ON A VERTEX

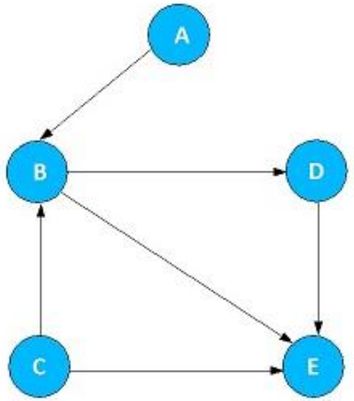
Each vertex in a directed graph has two different degree:

- ❑ **Indegree**
- ❑ **Outdegree.**

Indegree is the number of incoming edges to a vertex, while outdegree is the number of outgoing edges from a vertex.

INDEGREE & OUTDEGREE ON A VERTEX

Example 1:



Vertices	A	B	C	D	E
In degree	0	2	0	1	3
Out Degree	1	2	2	1	0
Total Degree	1	4	2	2	3

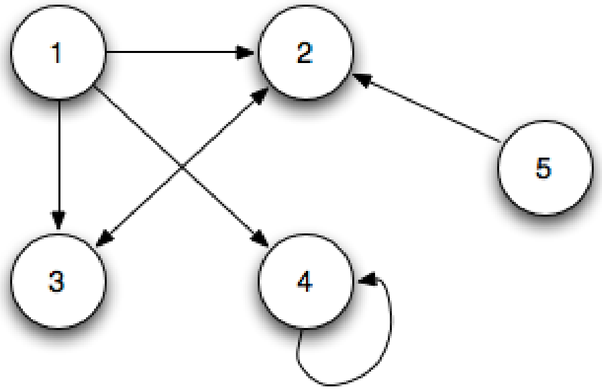
Degree of vertex B is 4, because there are **4 edges incident on B**

2.1.1 PROPERTIES OF DIRECTED GRAPHS

LOOP

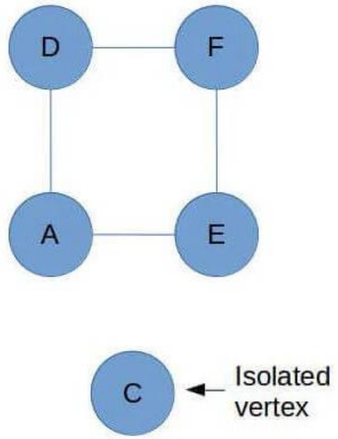
An edge incident on a single vertex

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



ISOLATED VERTEX

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

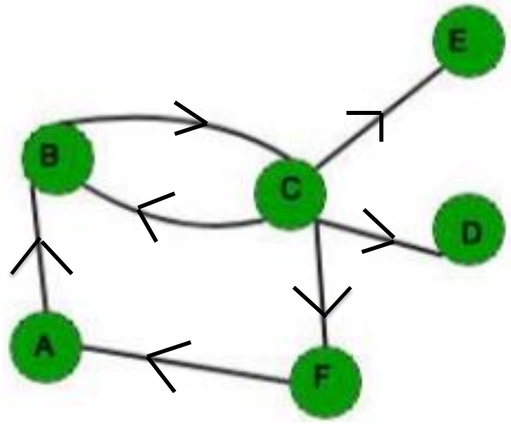


2.1.1 PROPERTIES OF DIRECTED GRAPHS

A multigraph is a graph that may contain

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

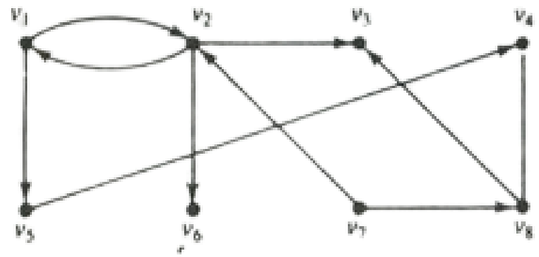
EXAMPLE



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

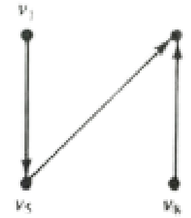
2.1.2 IDENTIFY SUBGRAPH

- Let V' be a subset of V and let E' be a subset of E whose endpoints belong to V' . Then $H(V', E')$ **is a directed graph and called subgraph of of** $G(\mathrm{V}, E)$
- Consider the digraph below.

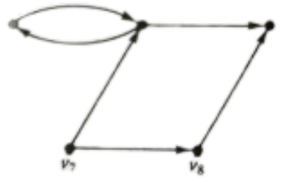


Graph G

- From the directed graph G, we can create subgraphs.
- Graph H and Graph I below are subgraph generated from graph G



Graph H

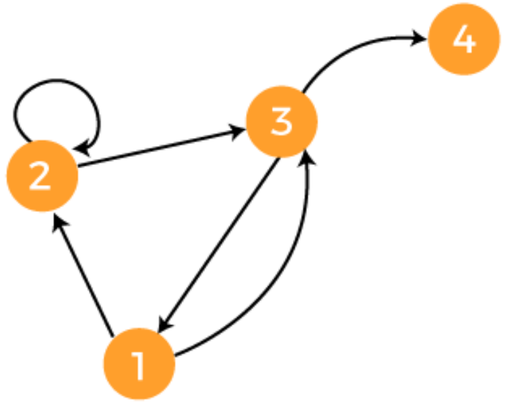


Graph I

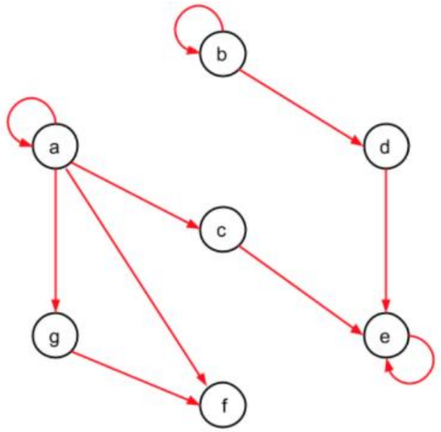
EXERCISE A

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

a)

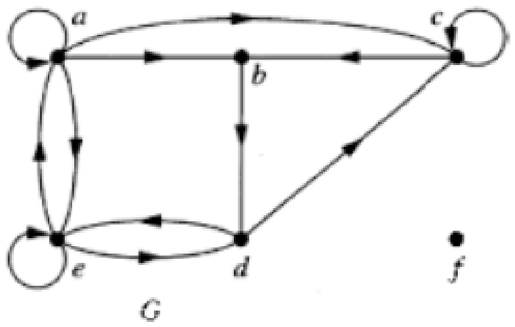


b)

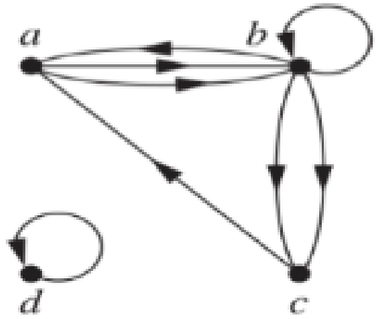


Exercise A cont...

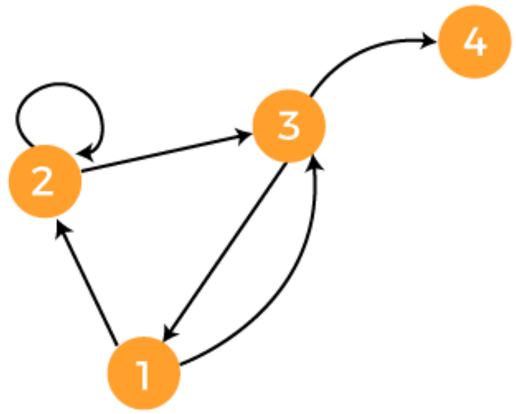
c)



d)



EXAMPLE



deg (1) = 3 → A is even vertex

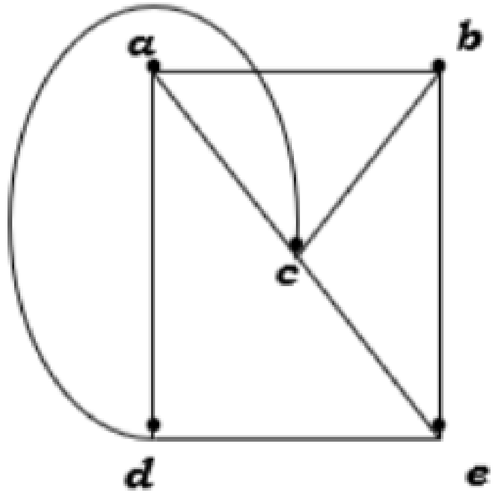
deg (2) = 4 → B is even vertex **(loop must be counted twice)**

deg (3) = 4 → C is odd vertex

deg (4) = 1 → D is even vertex deg (E) = 3 → D is even vertex

Exercise B

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



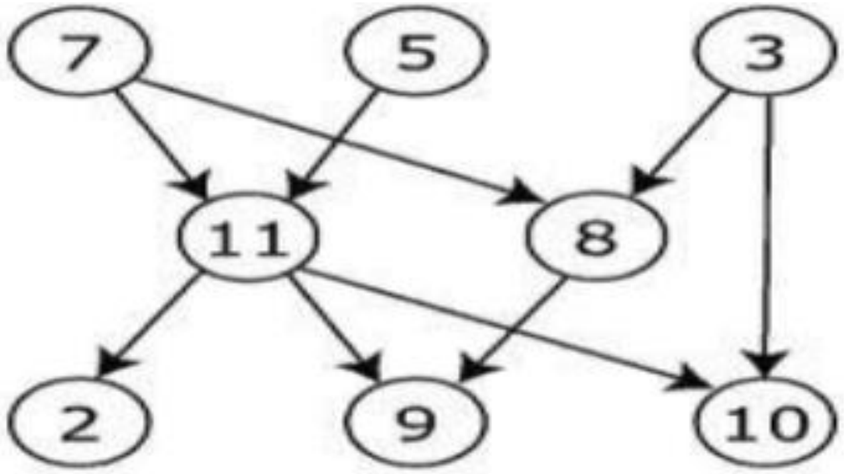
2. Draw a graph having the given properties or explain why no such graph exists.

a) Six vertices each of degree 3

b) Five vertices each of degree 2

EXERCISE B

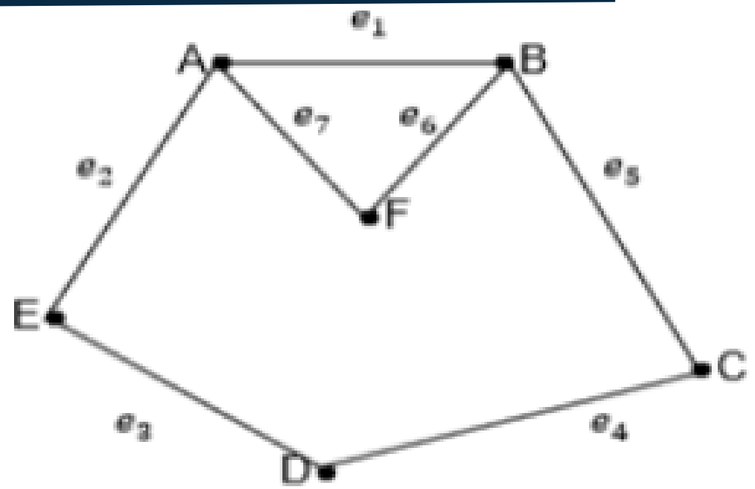
3. From the graph shown below,
   a) List down in a table form the **in degree** and the **out degree** of each vertices
   b) Determine the **parity** (even or odd) **for each vertex** and
   c) Determine the **sum of degrees**



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.
- 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**.

EXAMPLE



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

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

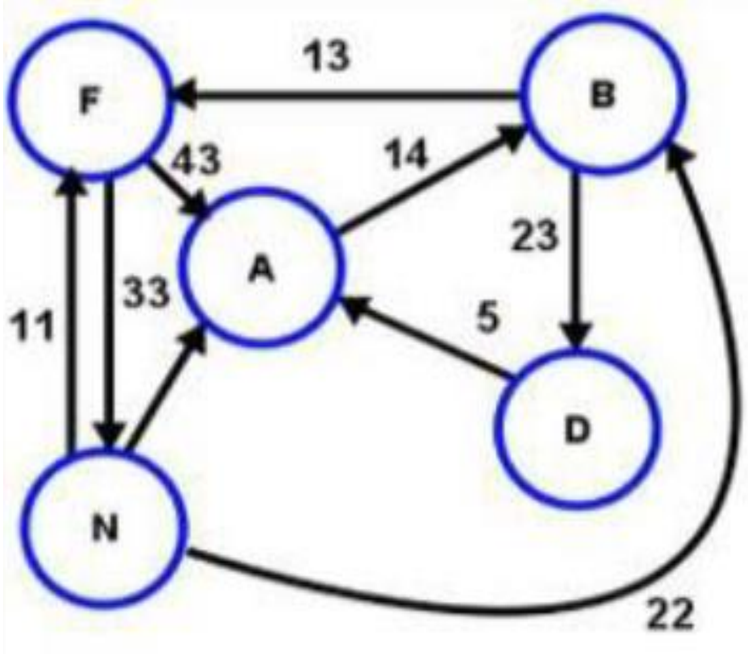
- The length of path $P = 1$, the length of path $Q = 4$ and the length of path $R = 2$
- The distance from A to B ($\boldsymbol{d\ A, B}$) is 1 (shortest path)

GRAPH REPRESENTATIONS

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

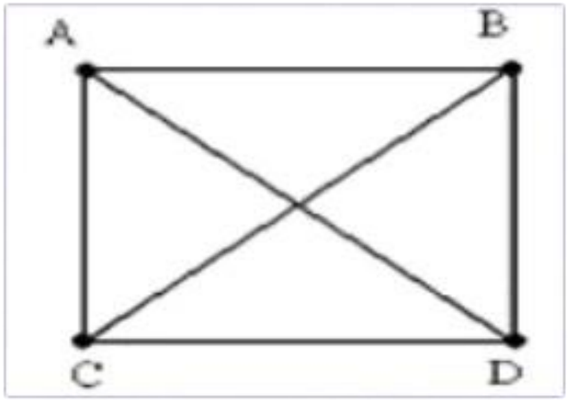


Connected and disconnected graphs

Connected Graph

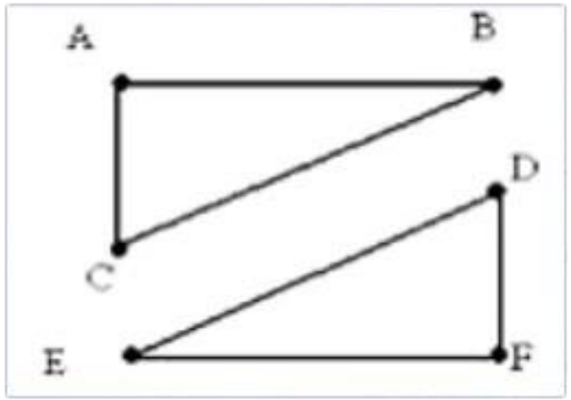
- An undirected graph is called **connected** **if there is a path between every pair of distinct vertices of the graph**.
  - * **Path**: path is a sequence of edges that begins at a vertex of a graph and travels from vertex to vertex along edges ofthe graph.

EXAMPLE



CONNECTED GRAPH

DISCONNECTED GRAPH

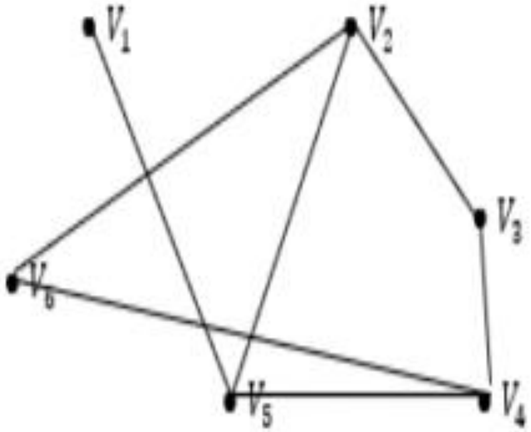


2.1.3 CONSTRUCT PATHS AND CYCLES

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



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

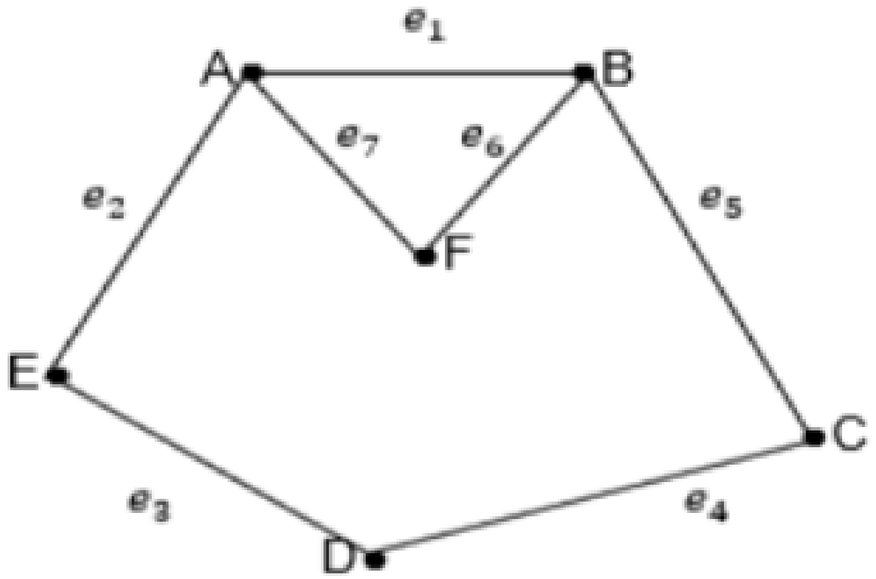
- ***A***, ***B*** and ***C*** **are path** because there exists an edge that connects every consecutives vertex.
- ***D*** **is not a path** because there is no edge that connects V_5 and V_3 directly.
- ***A*** and ***B*** also be called **trails** because each edge is only traversed once. (trails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

2.1.3 CONSTRUCT PATHS AND CYCLES

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**.

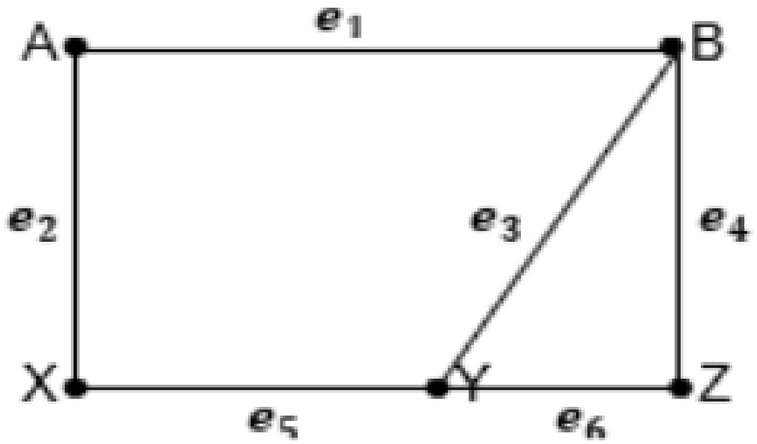
EXAMPLE



- 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

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)

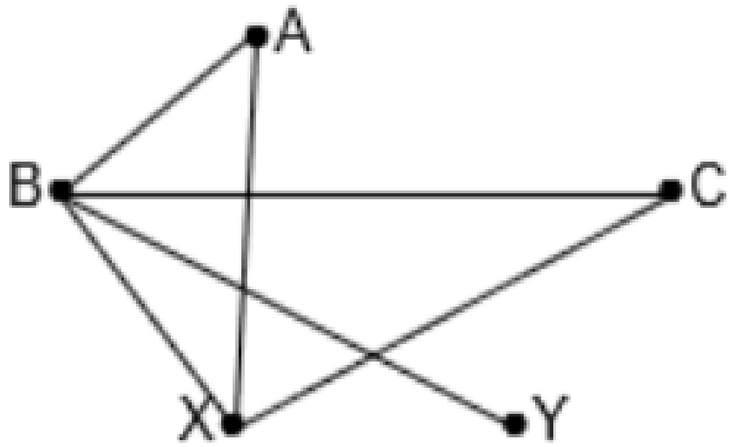
b) d(A, Z) = 2

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.

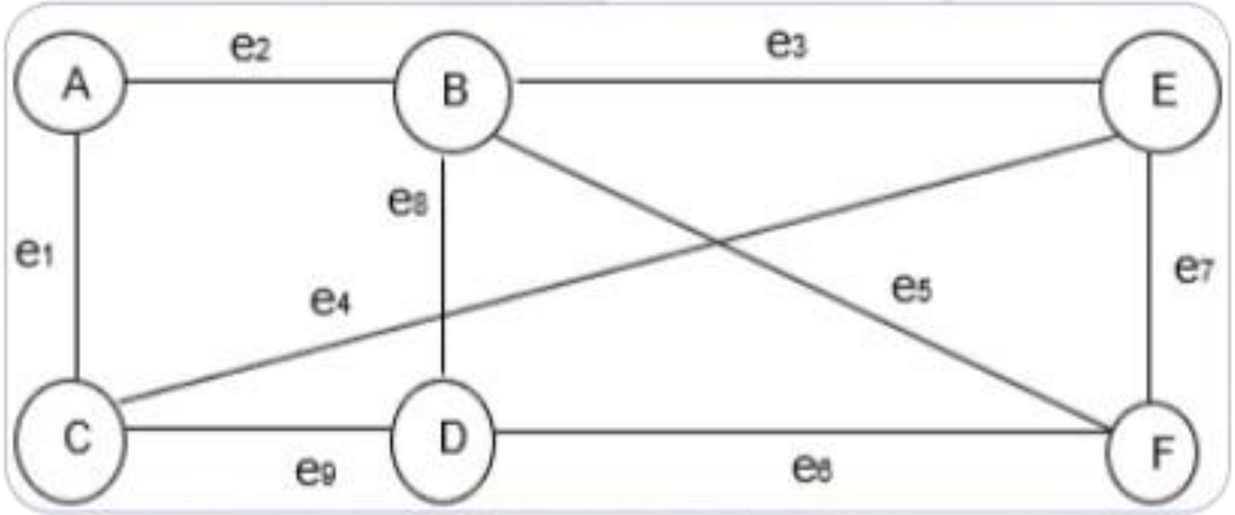
e) Not a path because there is no edge between C and A.

f) Trail.

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

h) Simple path & trail.

EXAMPLE



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

Size of the graph: 9

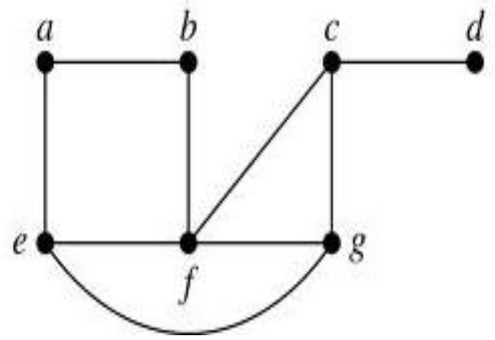
Path from B to C: (B, A, C) @ (B, E, C) @ (B, F, D, C)

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 3C

1. 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

2.1.3 CONSTRUCT PATHS AND CYCLES

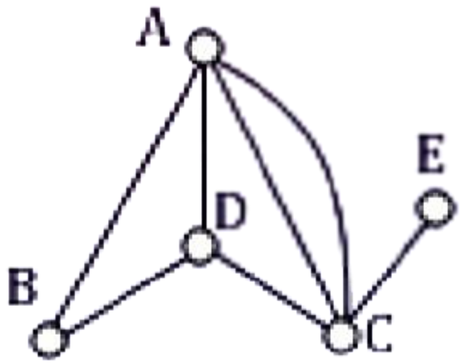
Hamilton circuit in G

- 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

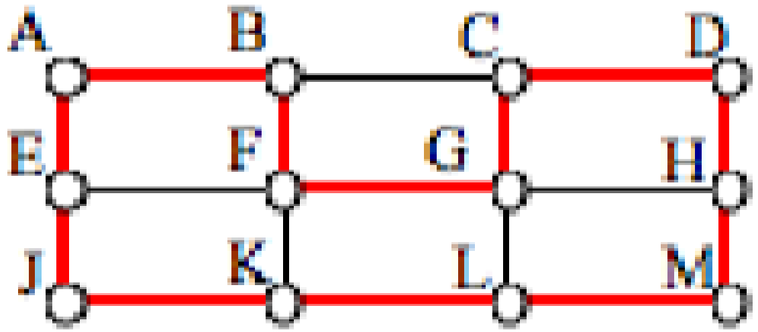
- A **Hamilton path** is a simple path in a graph G that **passes through every vertex exactly once.**

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)**

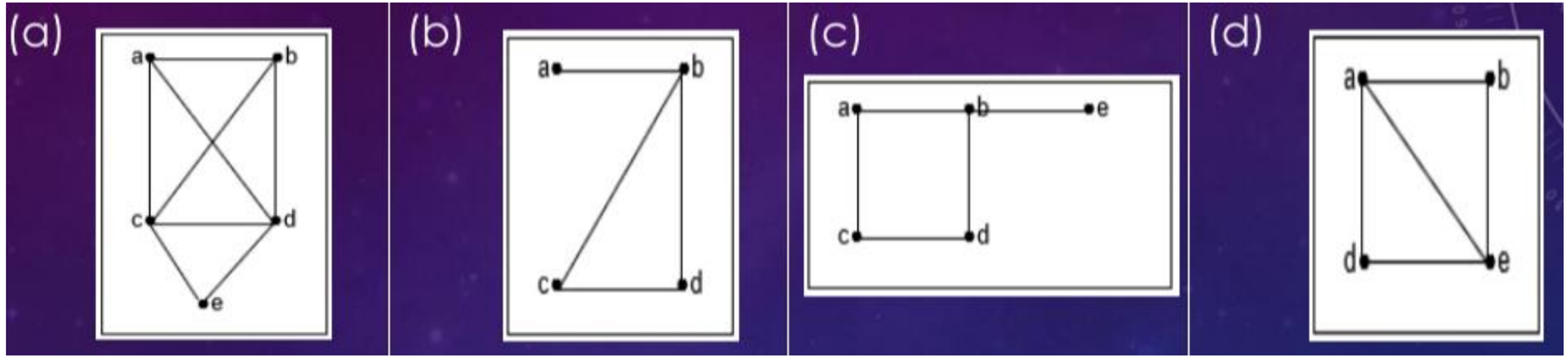
EXAMPLE



- 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)**

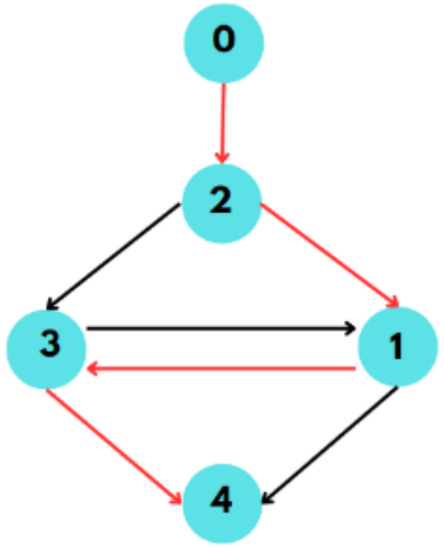
c) **Hamilton path (a, c, d, b, e)**

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

Exercise D

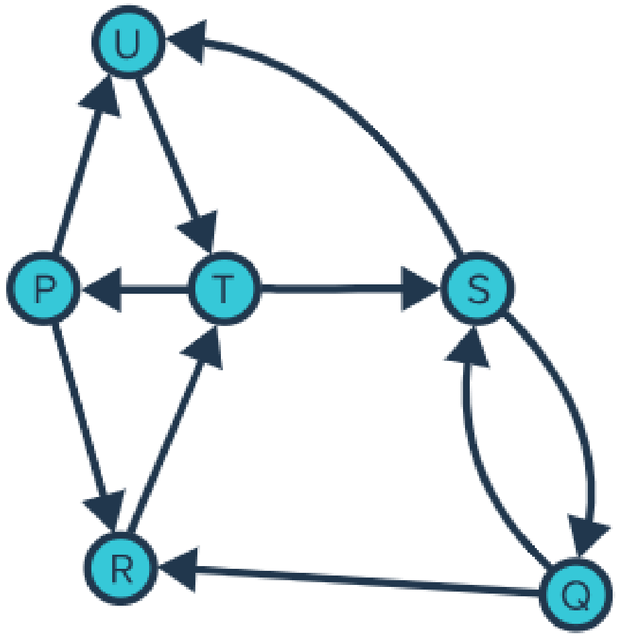
1. Base on diagram below, does **Hamilton path** or **Hamilton circuit** exist? If any, list down the Hamilton path or Hamilton circuit.

   (a)



Exercise D cont…

(b)



2.1.3 CONSTRUCT PATHS AND CYCLES

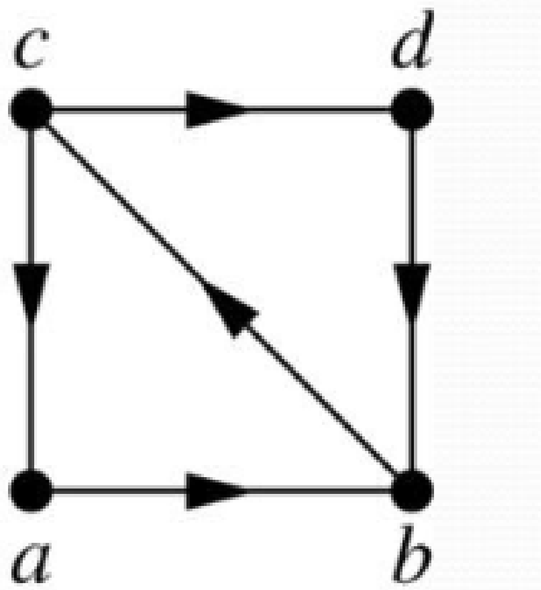
Euler circuit in a graph G

- 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

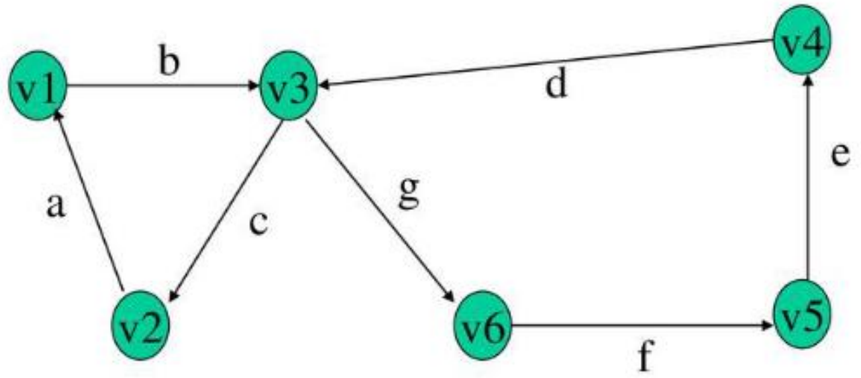
- An **Euler path** is a **path that uses every edge in a graph with no repeats**. Being a path, it **does not have to return to the starting vertex**.
- A connected multigraph has an **Euler path but not an Euler circuit** if and only if it has **exactly two vertices of odd degree**.

EXAMPLE



In the graph shown above, there are several **Euler paths**. One such path is **(c,a,b,c,d,b)**.

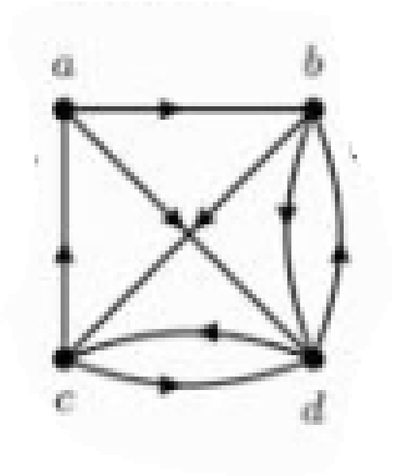
EXAMPLE

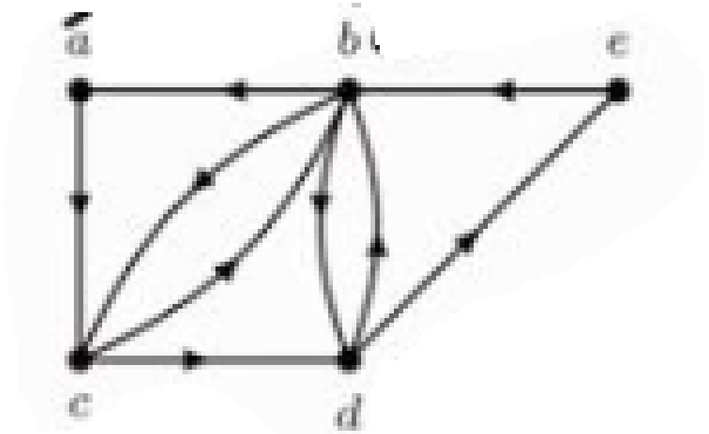


- The graph above has an **Euler circuits**.
- An Euler circuit in a directed graph is a directed circuit that visits every edge in G exacly once.
- Here's an example, **starting and ending at vertex V3: (v3,v2,v1,v3,v6,v5,v4,v3)**.

EXAMPLE

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





Solution:

a) Euler path – exactly two vertices have an even degree **(vertex a & vertex d).**

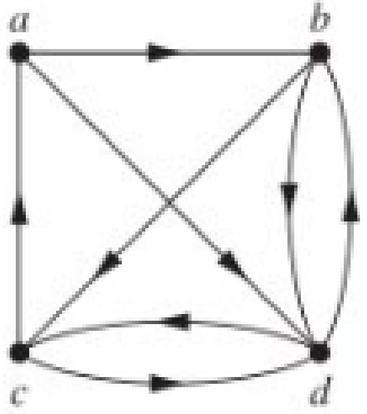
b) Euler circuit – all the vertices have an even degree.

EXERCISE E

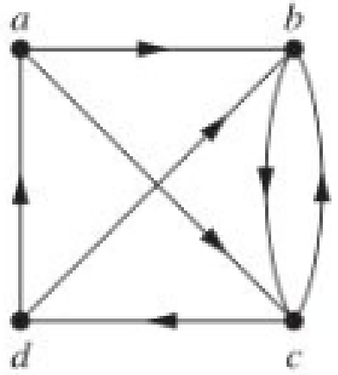
1. Give the characteristic of Euler path and Euler circuit.

2. Which of the following graphs has Euler path or Euler circuit? If any, list down the Euler path or Euler circuit.

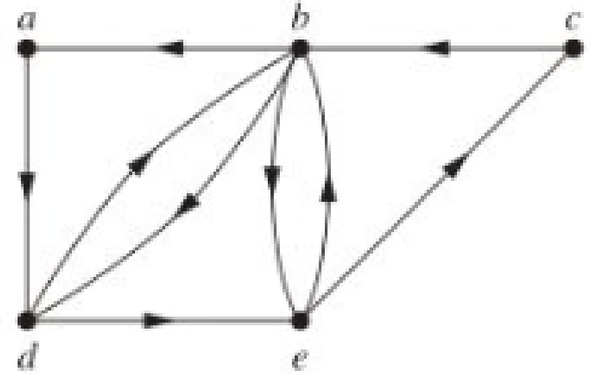
a)



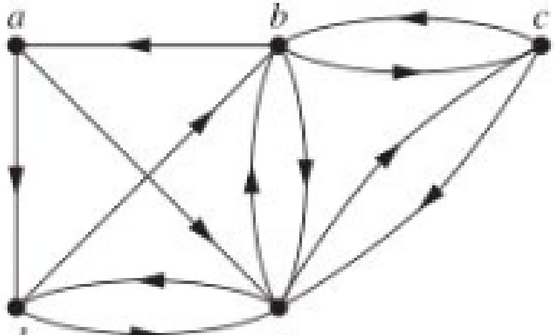
b)

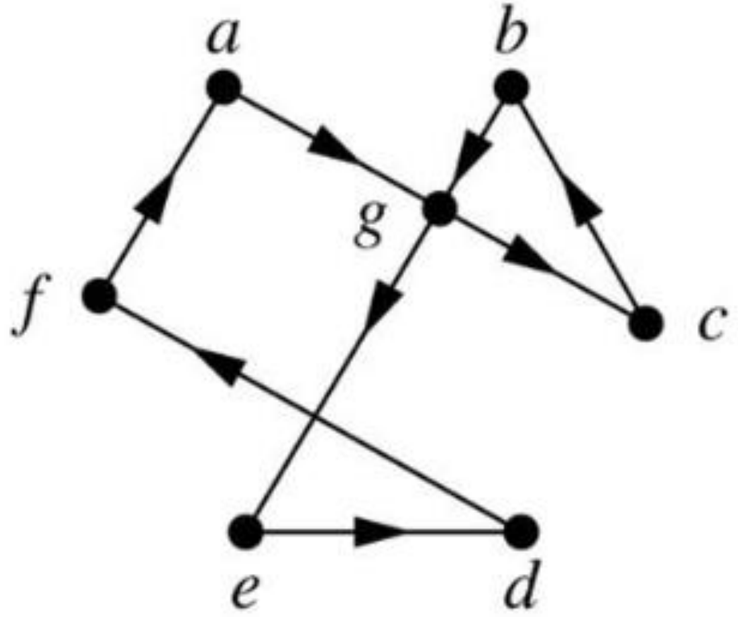


c)



d)





Based on the diagram, determine whether the graph has an Euler Path and Euler Circuit. Hence, construct such path and circuit if they exist or state the reason if they do not exist.

DIFFERENCES BETWEEN HAMILTON PATH AND EULER PATH

Hamilton Path

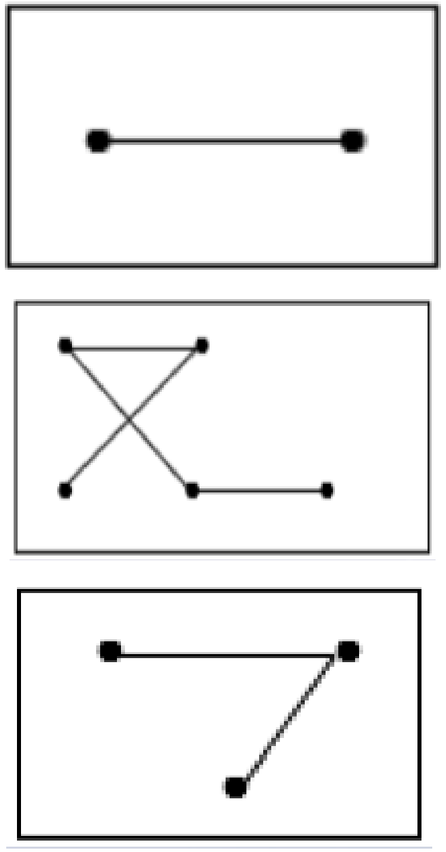
- Covers all the vertices of a graph exactly once. It can contain some edge multiple times

Euler Path

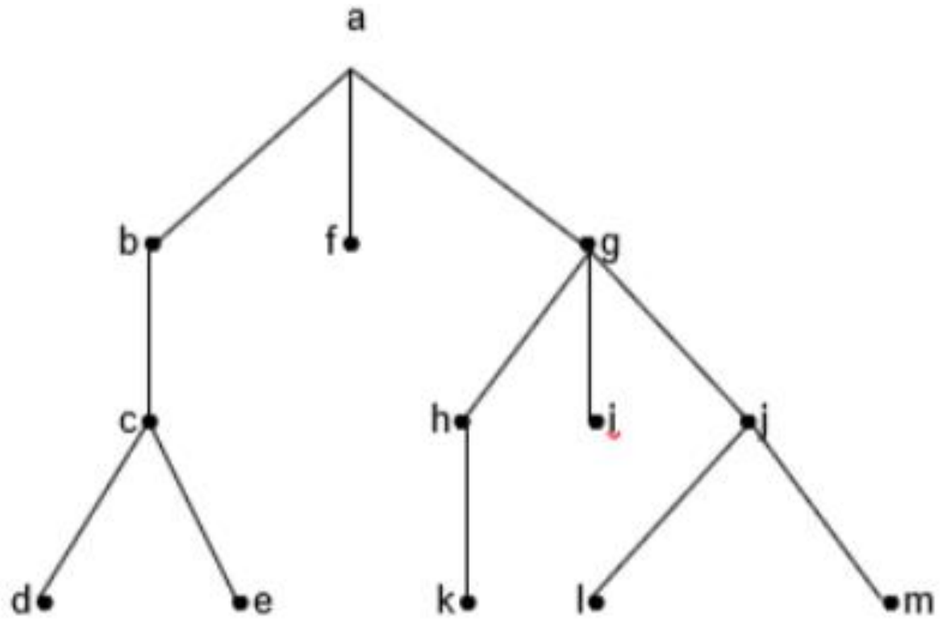
- Covers all the edges of a graph exactly once. It can contain some vertex multiple times.

2.2 DISCOVER CONCEPTS OF 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**.



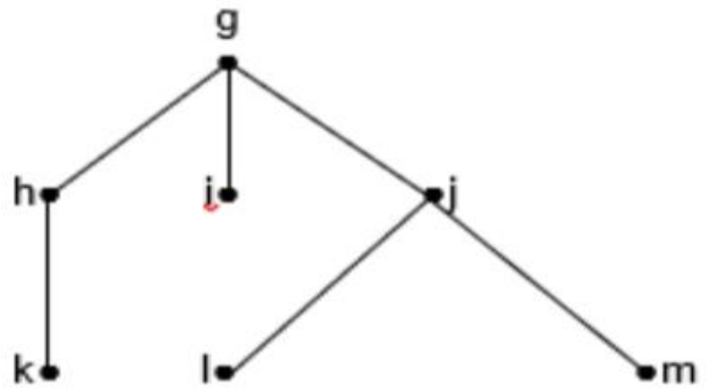
A **rooted tree** is a tree in which one vertex has been designated as the root and every edge is directed away from the root.



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.



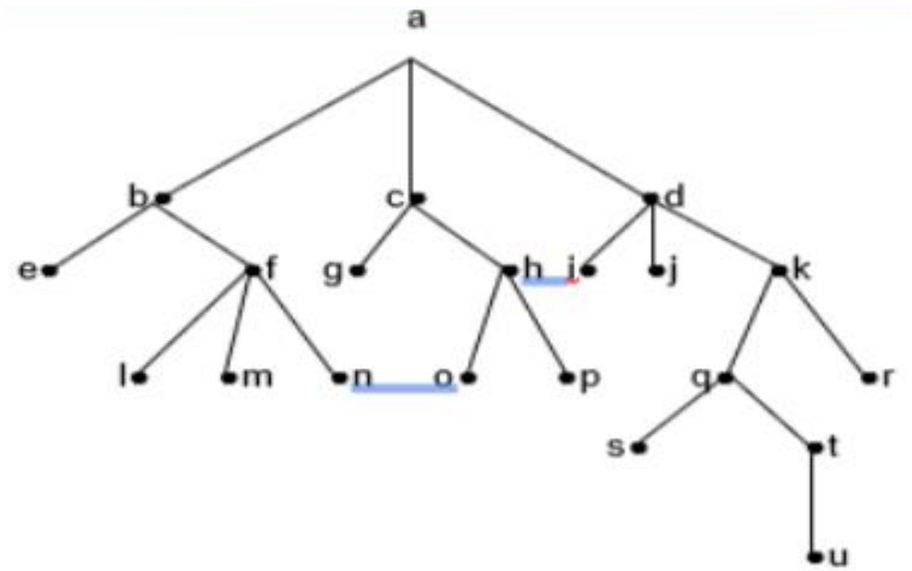
The Subtree Rooted At g

- 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

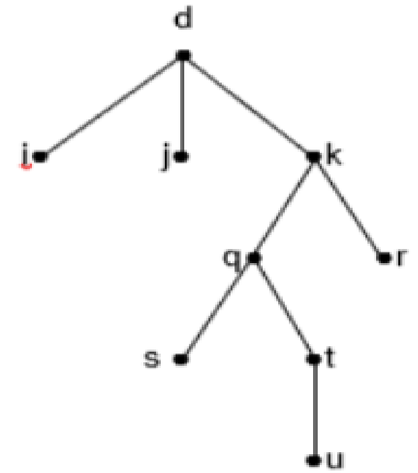
Level 2 = e, f, g, h, i, j, k

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

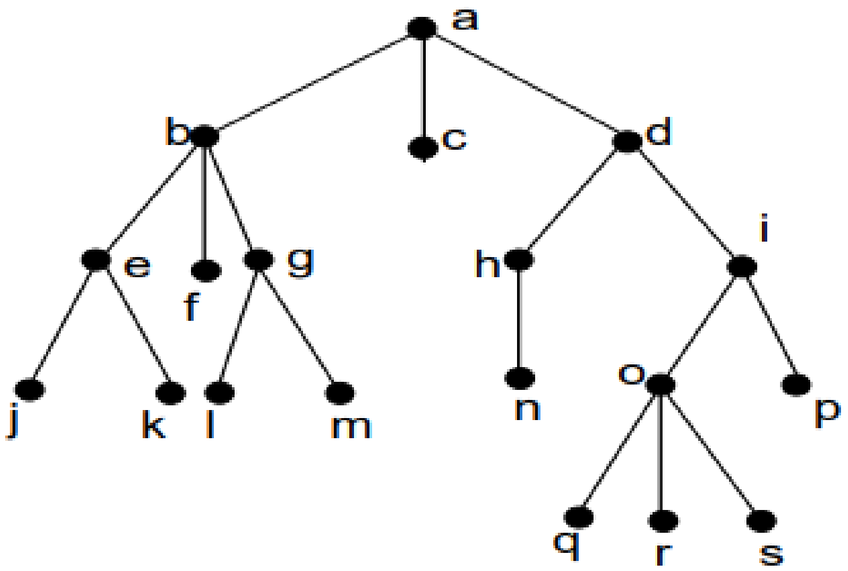
Level 4 = s, t

Level 5 = u

k)



EXERCISE G



Question:

a) Which vertex is the root?

b) Which vertices are internal?

c) Which vertices are leaves?

d) Which vertices are children of 0?

e) Which vertex is the parent of g?

f) Which vertices are siblings of k?

g) Which vertices are ancestors of m?

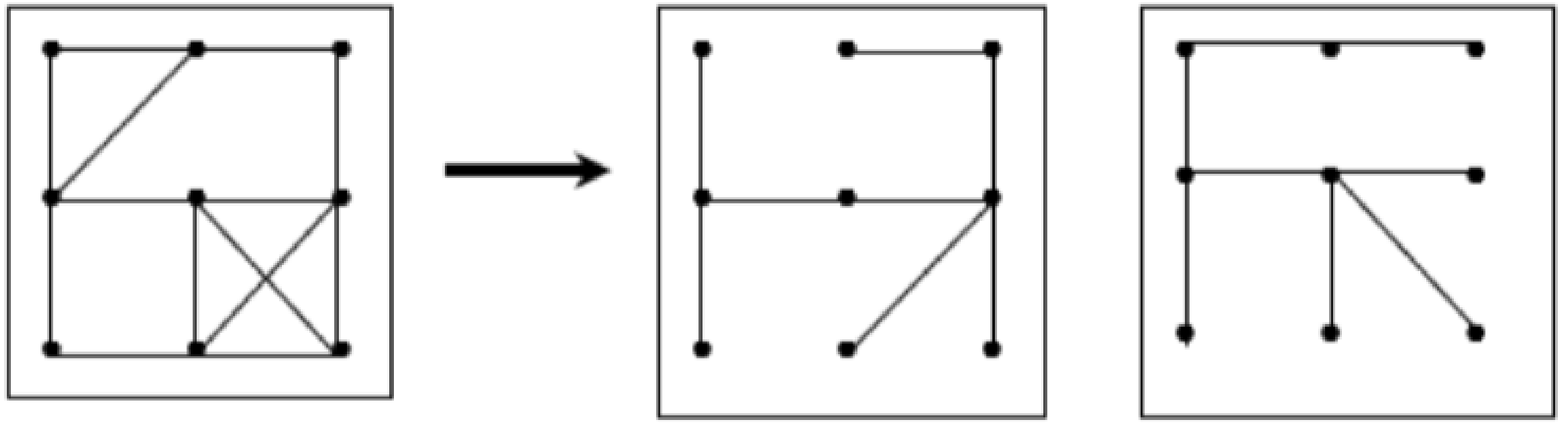
h) Which vertices are descendants of d?

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

k) Draw a subtree that is rooted at b.

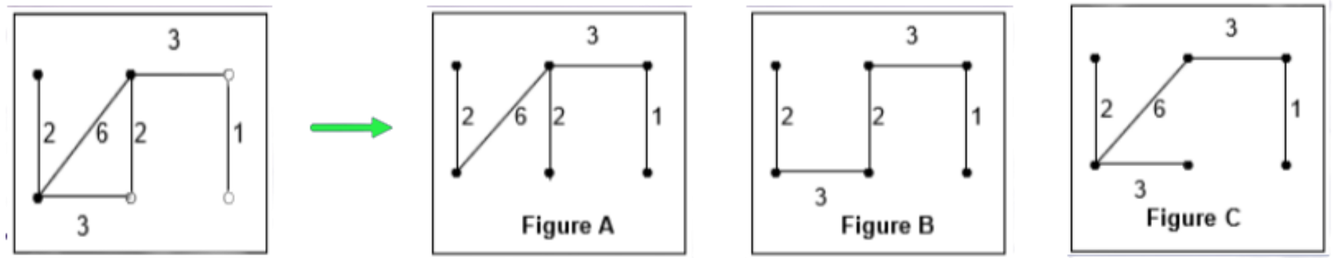
2.2.1 SPANNING TREES

- 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:



2.2.2 CONSTRUCT THE MINIMUM 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**.

2.2.2 CONSTRUCT THE MINIMUM SPANNING TREES

Prim’s Algorithm

- It has a **starting point**.

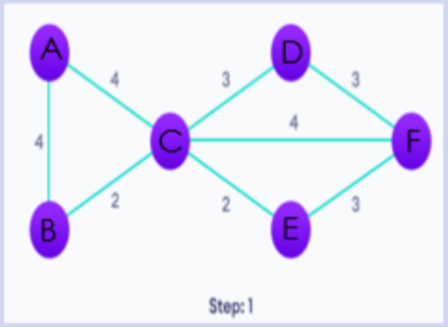
The steps for implementing Prim's algorithm are as follows:

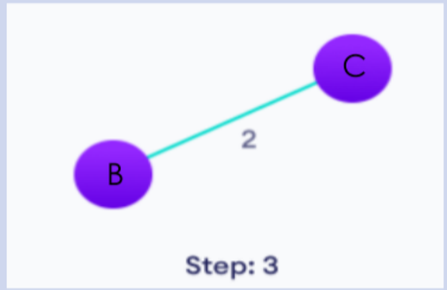
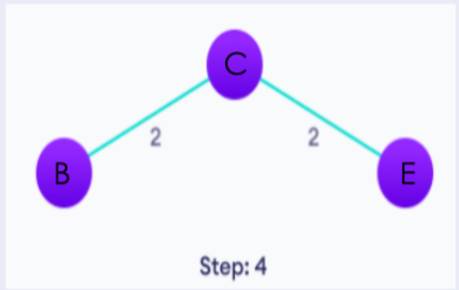
- Initialize the minimum spanning tree with a vertex chosen at random.
- **Find all the edges that connect the tree to new vertices, find the minimum and add it to the tree**
- Keep repeating step 2 until we get a minimum spanning tree

Kruskal’s Algorithm

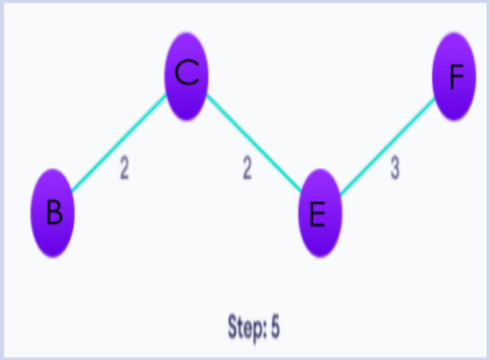
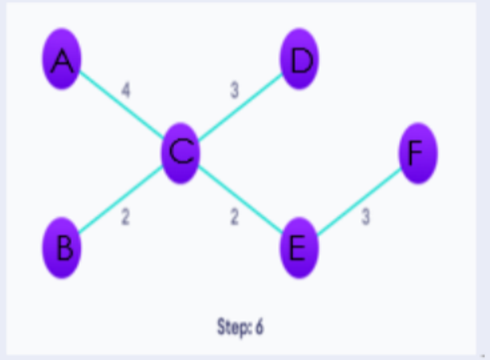
- start with the minimum weight of an edge

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

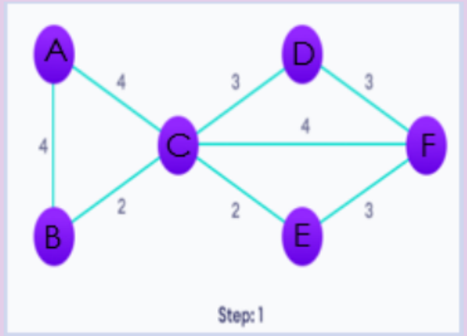
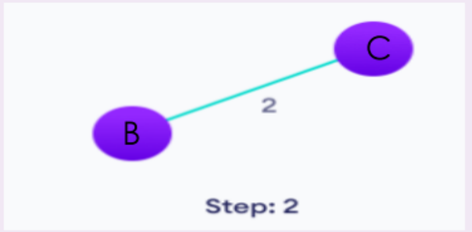
PRIM'S ALGORITHM

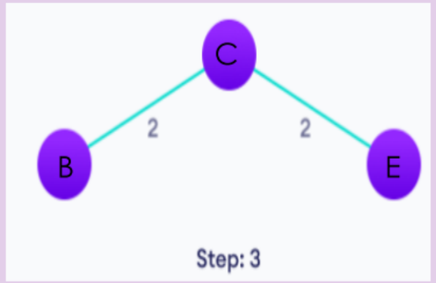
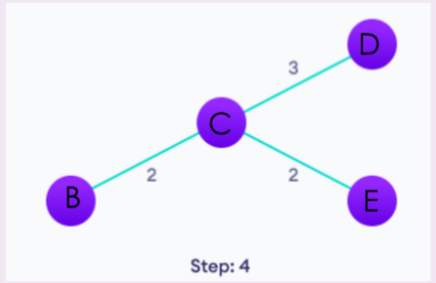
STEPS	EXPLANATION
5. 	Choose the nearest edge not yet in the solution, if there are multiple choices, choose one at random
6. 	Repeat until you have a spanning tree

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

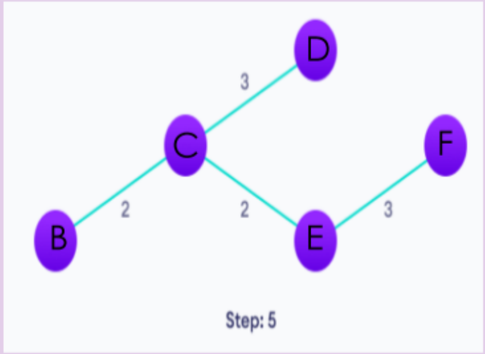
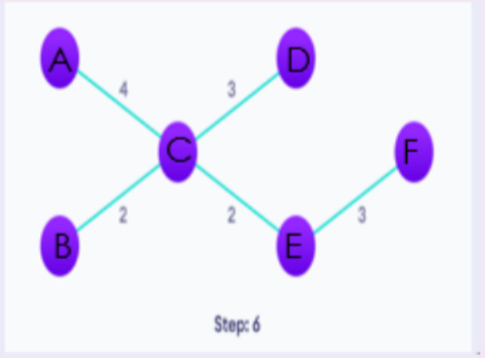
FROM	TO	WEIGHT
C	B	2
C	E	2
E	F	3
C	D	3
C	A	4
		TOTAL WEIGHT=14

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

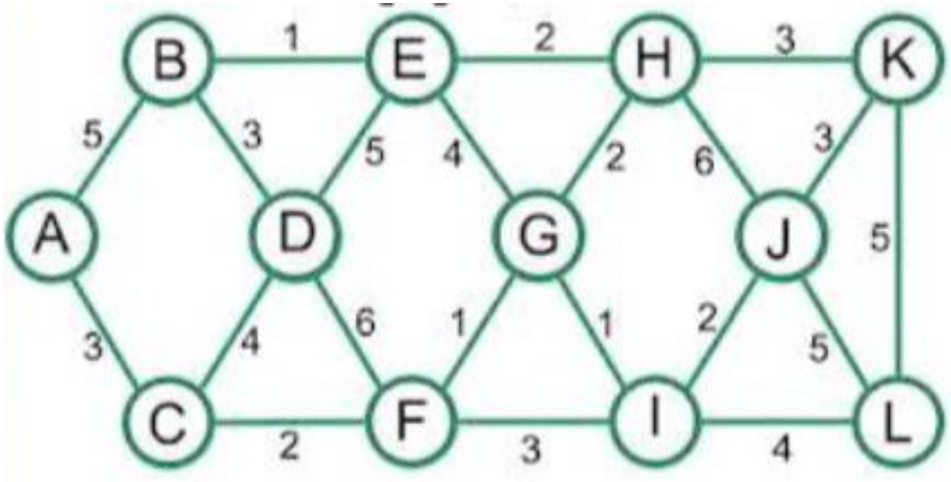
STEP	EXPLANATION
5. 	Choose the next shortest edge that doesn't create a cycle and add it
6. 	Repeat until you have a spanning tree

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 H

a) Based on the following figure,



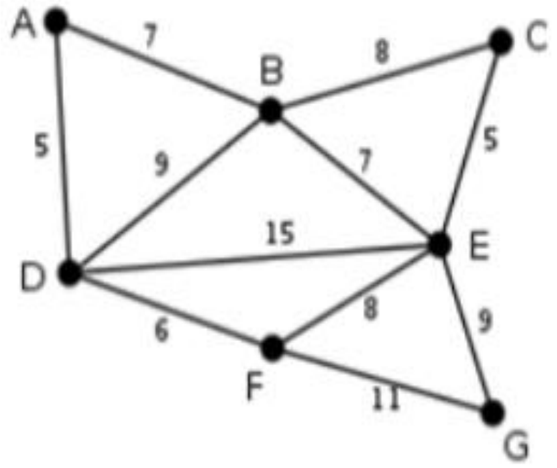
i. Draw 2 possible spanning trees

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

iii. Draw the minimum spanning tree.

EXERCISE H CONT...

b) By referring to the following weighted graph,

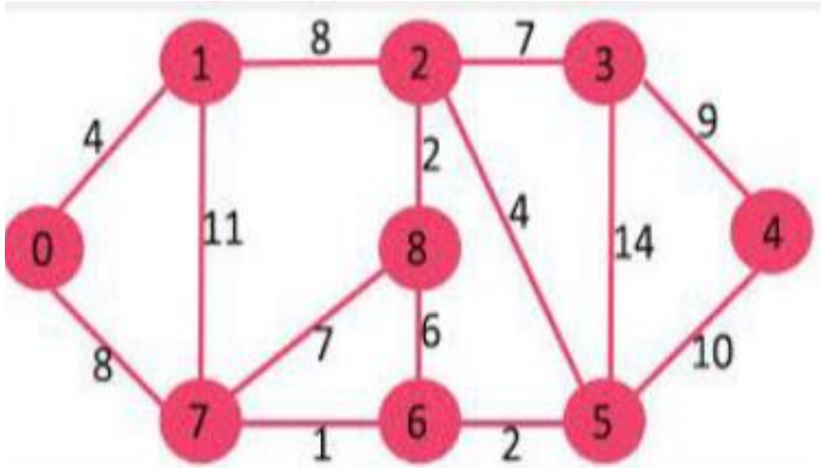


i. Draw the minimum spanning tree by using Kruskal algorithm.

ii. Calculate the shortest path.

EXERCISE H CONT...

c) Based on the following graph:



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

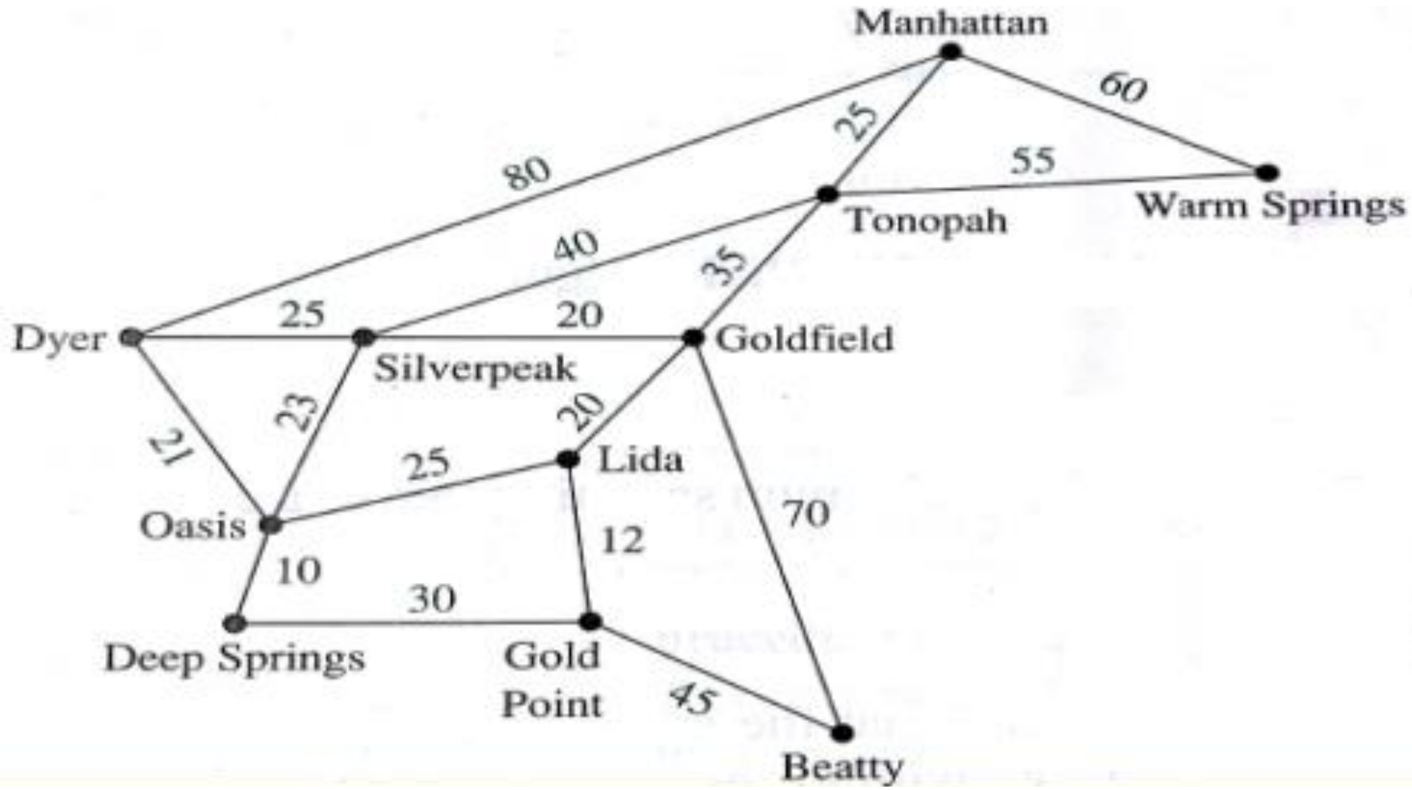
ii. Calculate the shortest path.

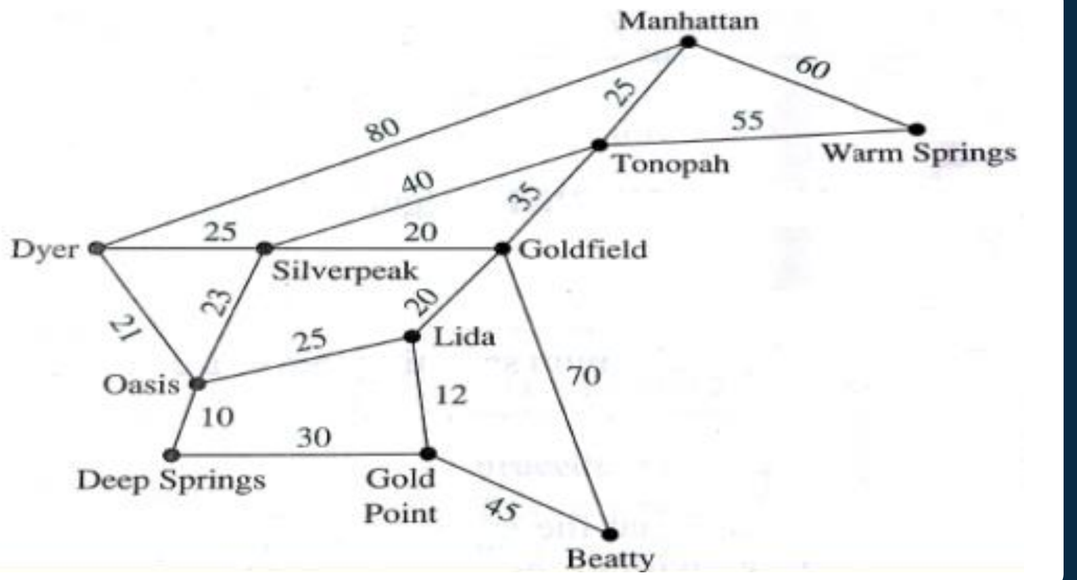
TRAVELLING SALESMAN PROBLEM (TSP) IN TREE THEORY

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: