Mathematics Concepts For Computing

AQ010-3-1-MCFC

A · P · U
ASIA PACIFIC UNIVERSITY
OF TECHNOLOGY & INNOVATION

Chapter 7

Graph and Tree

Topic & Structure of the lesson

- Introduction
- Definition
- Degree of a Vertex
- Graphs & Representations
- Paths & Circuits
- Trees

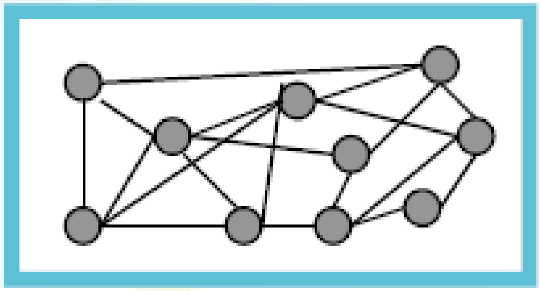
Graph

General meaning in everyday math:

- A plot or chart of numerical data using a coordinate system.

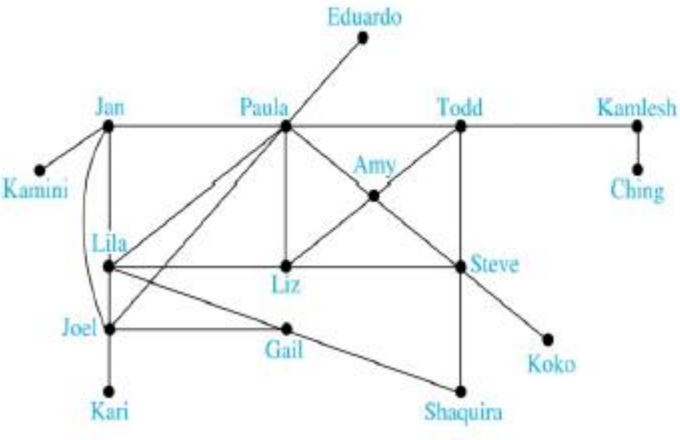
Technical meaning in discrete mathematics:

- A particular class of discrete structures (to be defined) that is useful for representing relations and has a convenient webby-looking graphical representation.



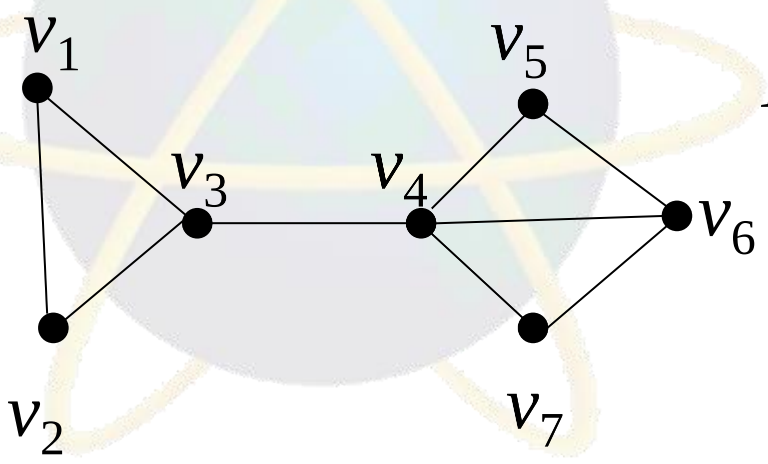
Application of Graphs

- Social networks
  - A friendship graph: two people are connected if they are Facebook friends.
- Communications networks
- Information networks
  - In a *web graph, web pages are represented* by vertices and links are represented by directed edges.
- Transportation networks



Def 1. A graph $G = (V, E)$ consists of V, a nonempty set of vertices (or nodes), and E, a set of edges. Each edge has either one or two vertices associated with it, called its endpoints. An edge is said to connect its endpoints.

eg.



$G=(V, E)$, where

$V=\{v_1, v_2, \ldots, v_7\}$

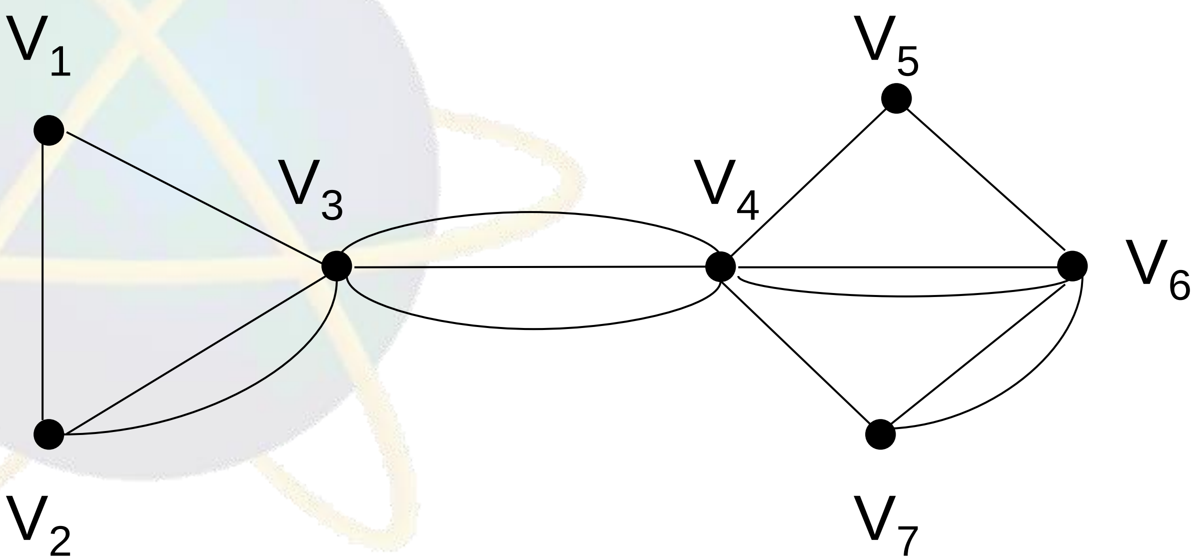
$E=\{\{v_1,v_2\}, \{v_1,v_3\}, \{v_2,v_3\}$
$\{v_3,v_4\}, \{v_4,v_5\}, \{v_4,v_6\}$
$\{v_4,v_7\}, \{v_5,v_6\}, \{v_6,v_7\}\}$

Def A graph in which each edge connects two different vertices and where no two edges connect the same pair of vertices is called a simple graph.

Def Multigraph:

simple graph + multiple edges (multiedges)

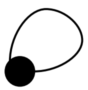
eg.



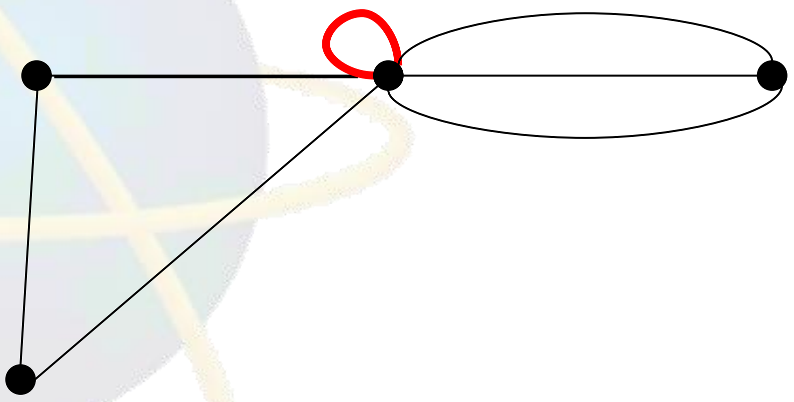
Def. Pseudograph:

simple graph + multiedge

+ loop

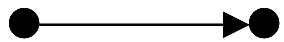
(a loop: )

eg.



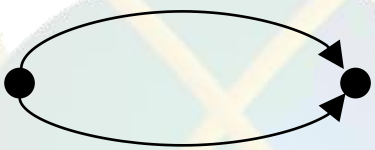
Def 2. Directed graph (digraph):

simple graph with each edge directed

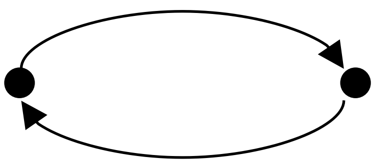


Note:  is allowed in a directed graph

Note:



The two edges (u,v),(u,v) are multiedges.



The two edges (u,v), (v,u) are not multiedges.

Def. Directed multigraph: digraph+multiedges

Graph Terminology

Def 1. Two vertices u and v in a undirected graph G are called adjacent (or neighbors) in G if $\{u, v\}$ is an edge of G.

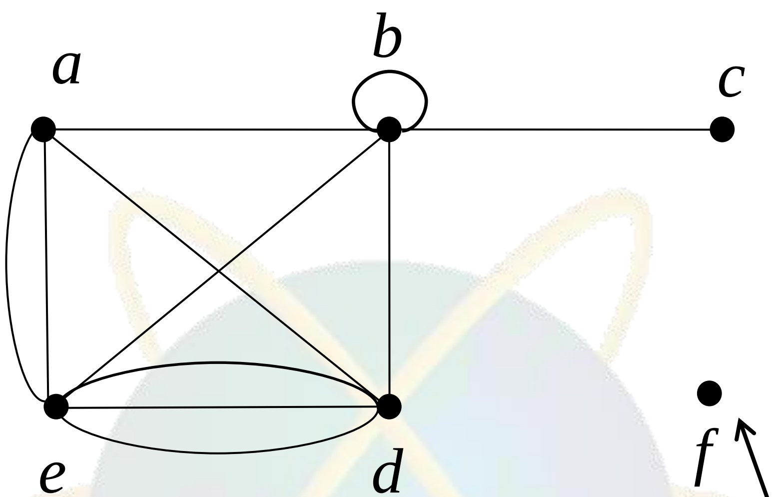
Note : adjacent: a vertex connected to a vertex
incident: a vertex connected to an edge

Def 2. The degree of a vertex v, denoted by deg(v), in an undirected graph is the number of edges incident with it.

(Note : A loop adds 2 to the degree.)

Example

What are the degrees of the vertices in the graph $\boldsymbol{H}$?



$\boldsymbol{H}$

Sol :

$\deg(a)=4$

$\deg(b)=6$

$\deg(c)=1$

$\deg(d)=5$

$\deg(e)=6$

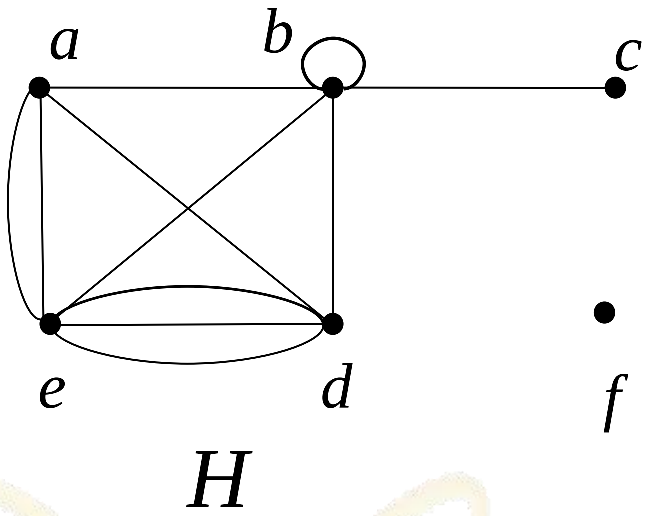
$\deg(f)=0$

Def. A vertex of degree 0 is called isolated.

Thm 1. (The Handshaking Theorem)
Let $\boldsymbol{G} = (\boldsymbol{V}, \boldsymbol{E})$ be an undirected graph with e edges (i.e., $|\boldsymbol{E}| = \boldsymbol{e}$). Then

$$\sum_{v \in V} \deg(v) = 2e$$

eg.



The graph H has 11 edges, and

$$\sum_{v \in V} \deg(v) = 22$$

Example

How many edges are there in a graph with 10 vertices each of degree six?

Sol :

$$10 \cdot 6 = 2e \quad \Rightarrow \quad e = 30$$

Example

Draw a simple graph whose degree sequence is :

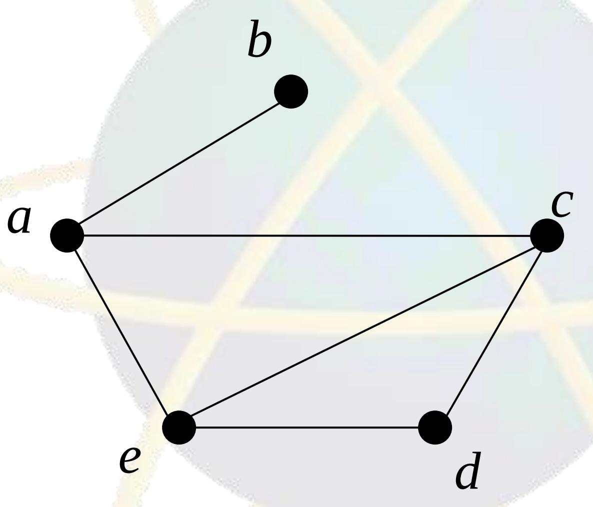
(a) (1, 2, 2, 2, 3, 4)

(b) (2, 2, 2, 2, 3, 3, 4, 4)

Representing Graphs – Adjacency List

※Adjacency list

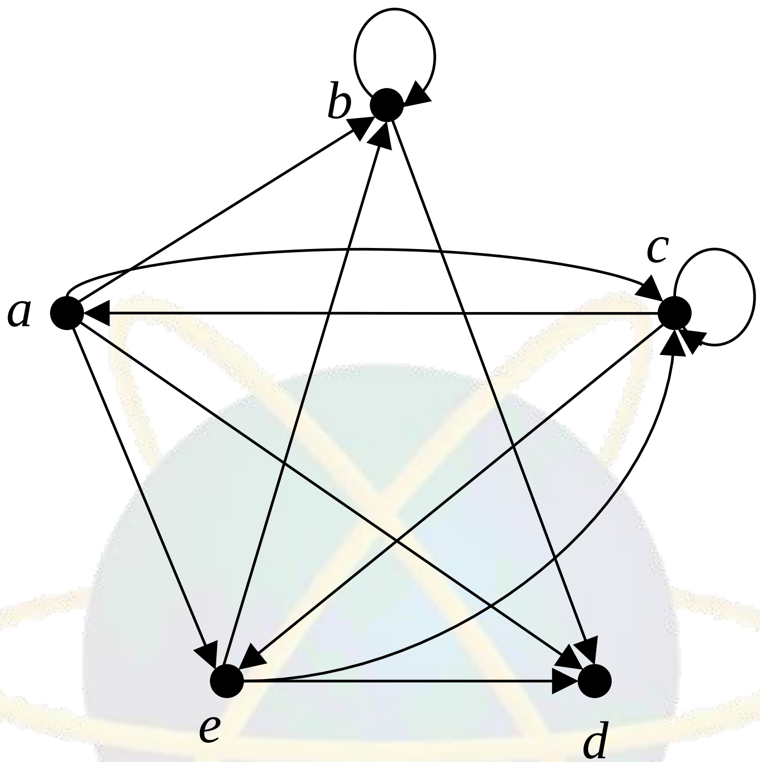
Example 1. Use adjacency lists to describe the simple graph given below.



Sol :

Vertex	Adjacent Vertices
a	b,c,e
b	a
c	a,d,e
d	c,e
e	a,c,d

Example 2.



Initial vertex	Terminal vertices
a	b,c,d,e
b	b,d
c	a,c,e
d	
e	b,c,d

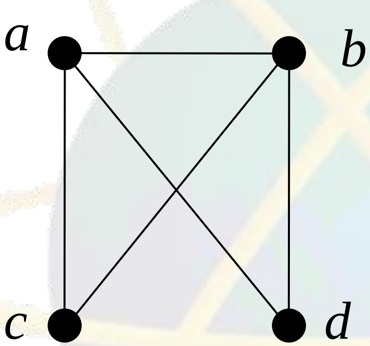
Representing Graphs – Adjacency Matrix

Def. $G=(V, E)$: simple graph, $V=\{v_1,v_2,\ldots,v_n\}$.

A matrix A is called the adjacency matrix of G

if $A=[a_{ij}]_{n\times n}$, where $a_{ij} = \begin{cases} 1, \text{ if } \{v_i,v_j\}\in E, \\ 0, \text{ otherwise.} \end{cases}$

Example 3.

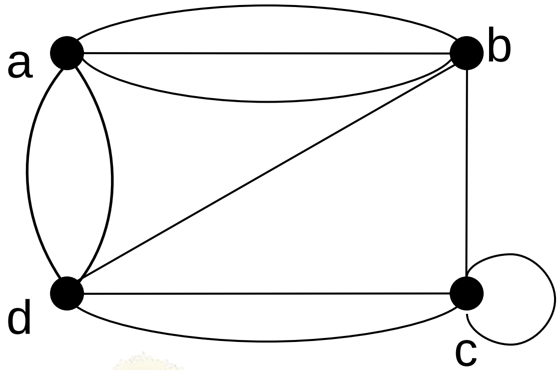


$$A_1 = \begin{array}{c} \\ a \\ b \\ c \\ d \end{array} \begin{array}{c} \begin{array}{cccc} a & b & c & d \end{array} \\ \begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \end{bmatrix} \end{array}$$

Note:

1. There are n! different adjacency matrices for a graph with n vertices.
2. The adjacency matrix of an undirected graph is symmetric.

Example 5. (Pseudograph)



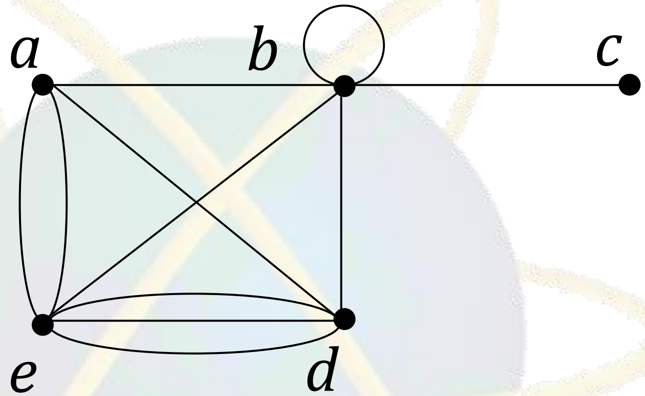
$$\Longrightarrow \quad A = \begin{array}{c} \\ a \\ b \\ c \\ d \end{array} \begin{array}{c} \begin{array}{cccc} a & b & c & d \end{array} \\ \begin{bmatrix} 0 & 3 & 0 & 2 \\ 3 & 0 & 1 & 1 \\ 0 & 1 & 1 & 2 \\ 2 & 1 & 2 & 0 \end{bmatrix} \end{array}$$

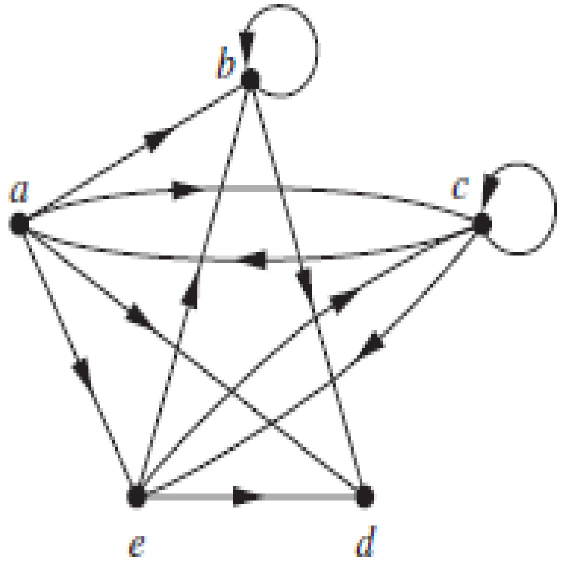
Def. If $A=[a_{ij}]$ is the adjacency matrix for the directed graph, then

$$a_{ij} = \begin{cases} 1 & \text{, if } v_i \longrightarrow v_j \\ 0 & \text{, otherwise} \end{cases}$$

Example

- Use adjacency list and adjacency matrix to represent the graph:



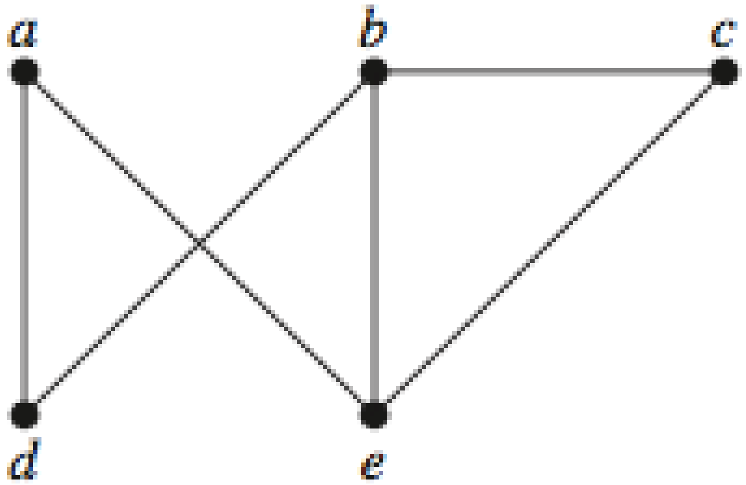


Paths and Circuits

- A path is sequence of adjacent vertices and edges.
- Simple path is a path that does not contain a repeated edge.
- A simple path is a circuit if it begins and ends at the same vertex.

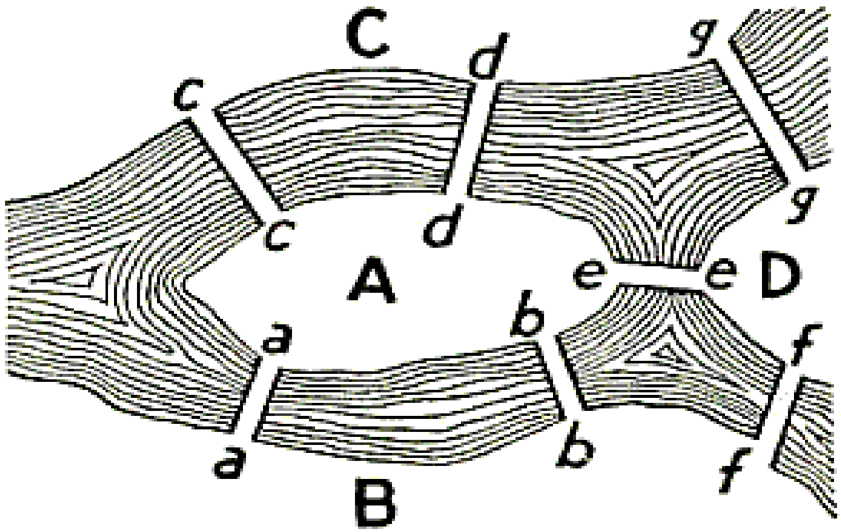
Example

Does each of these lists of vertices from a path in the following graph? Which paths are simple? Which are circuits? What are the lengths of those that are paths?

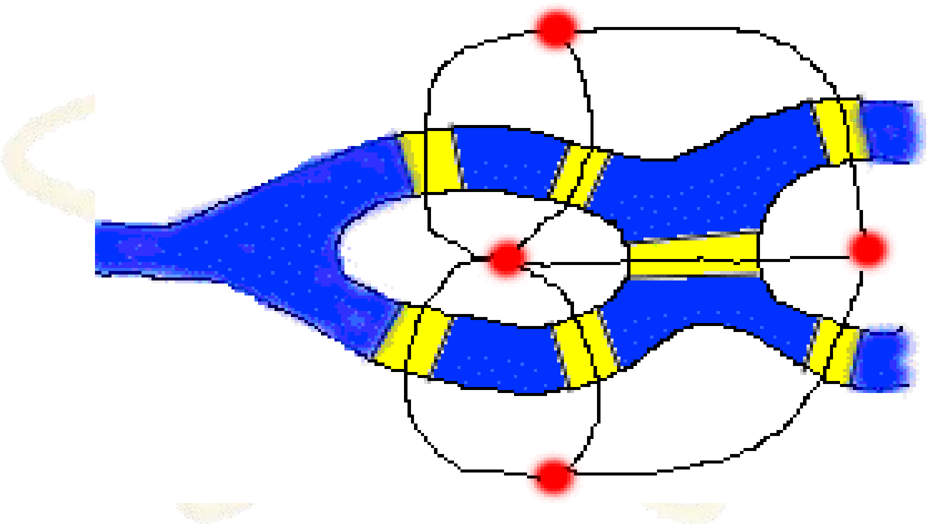


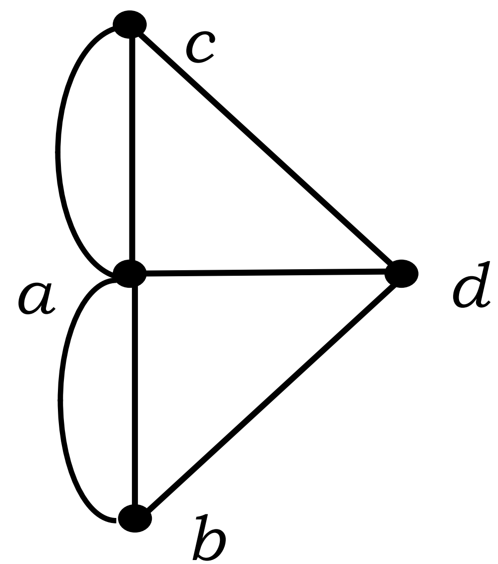
a) a, e, b, c, b
b) a, e, a, d, b, c, a
c) e, b, a, d, b, e
d) c, b, d, a, e, c

Konigsberg- in days past.



Is it possible to start at some location in the town, travel across all the bridges once without crossing any bridge twice, and return to the starting point ?





Euler Paths and Circuits

Def 1:

An *Euler circuit* in a graph *G* is a simple circuit containing every edge of *G*.

An *Euler path* in *G* is a simple path containing every edge of *G*.

Thm. 1:

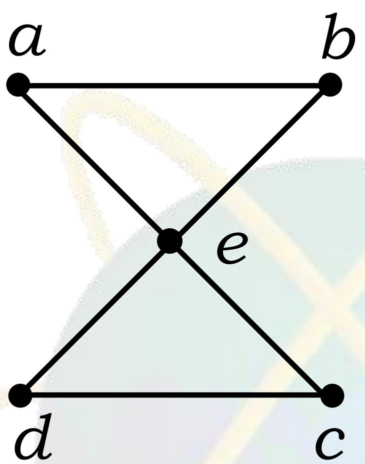
A connected multigraph with at least two vertices has an Euler circuit if and only if each of its vertices has even degree.

Thm. 2:

A connected multigraph has an Euler path (but not an Euler circuit) if and only if it has exactly 2 vertices of odd degree.

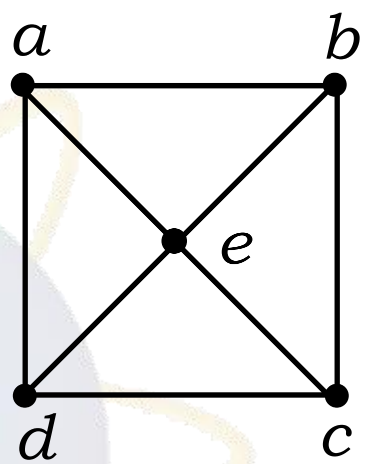
Example

Which of the following graphs has an Euler *circuit*?

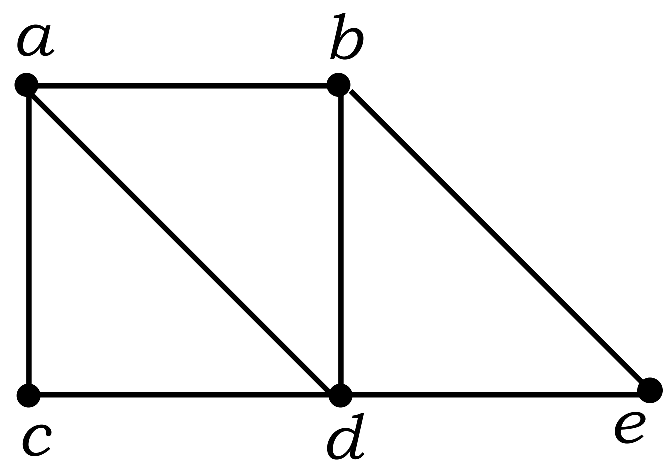


yes

(a, e, c, d, e, b, a)



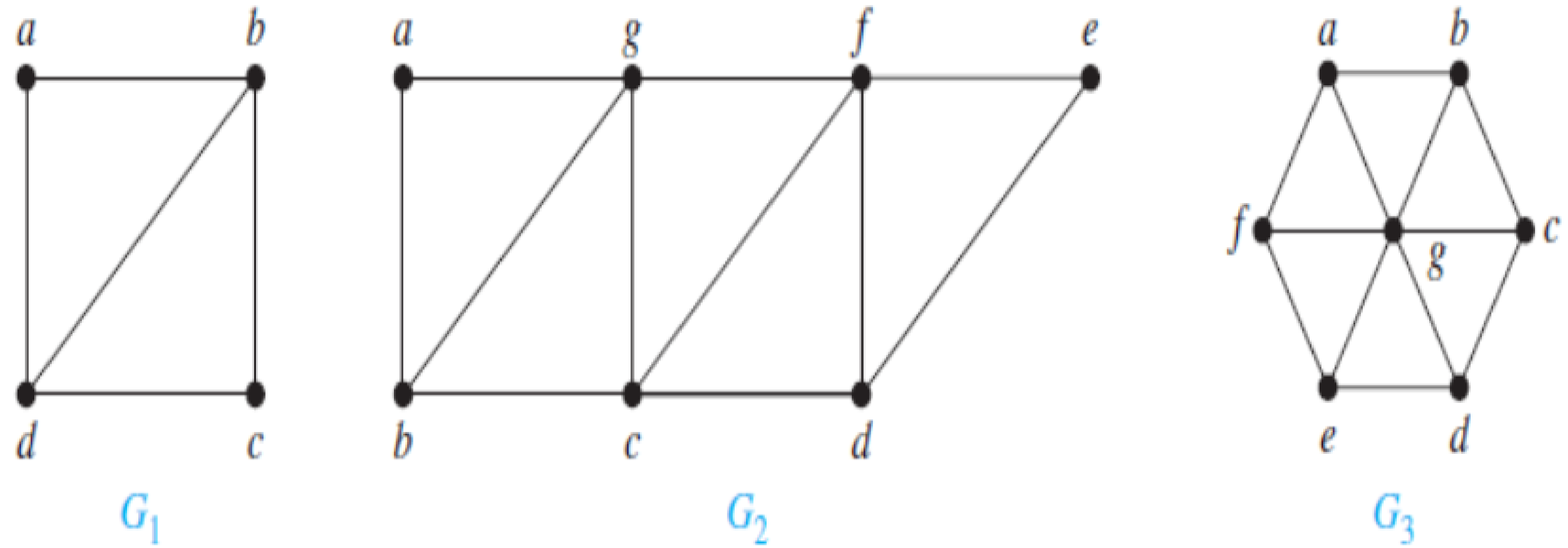
no



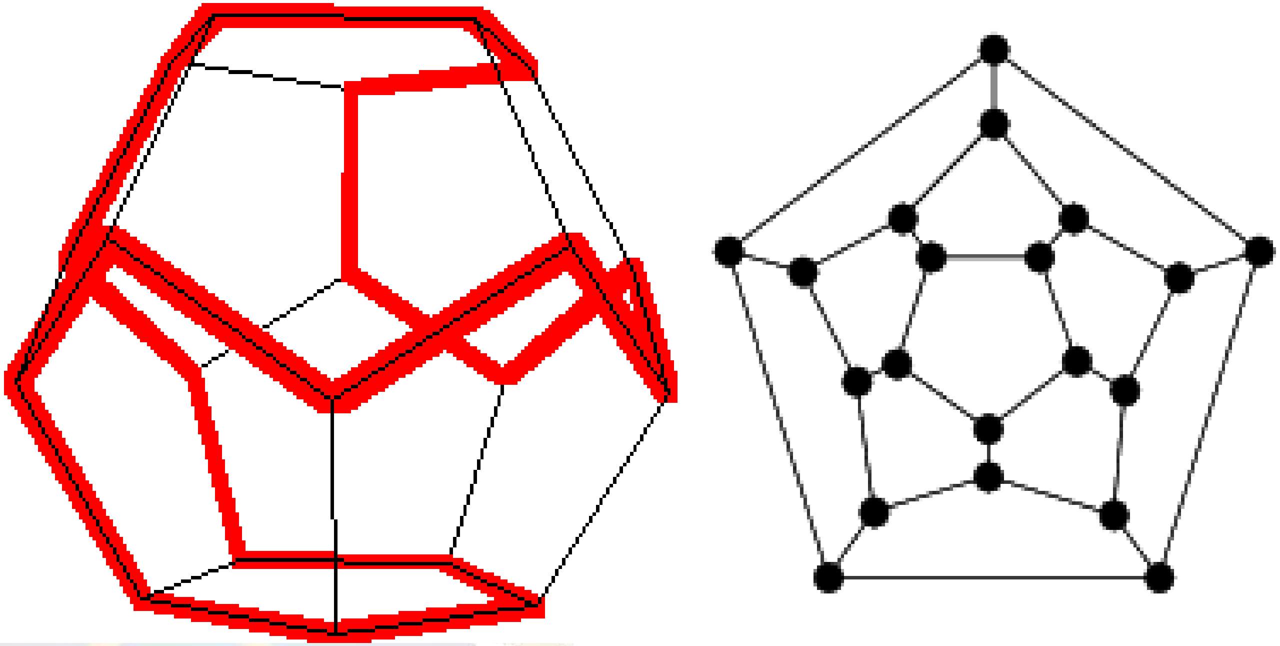
no, but has Euler path

(a, c, d, a, b, d,, e, b)

- **Example:** G1 contains exactly two vertices of odd degree, namely, b and d. Hence, **it has an Euler path** that must have b and d as its endpoints. One such Euler path is d, a, b, c, d, b. Similarly, G2 has exactly two vertices of odd degree, namely, b and d. So, it has an **Euler path that** must have b and d as endpoints. One such Euler path is b, a, g, f, e, d, c, g, b, c, f, d**. G3 has no Euler path** because it has six vertices of odd degree.



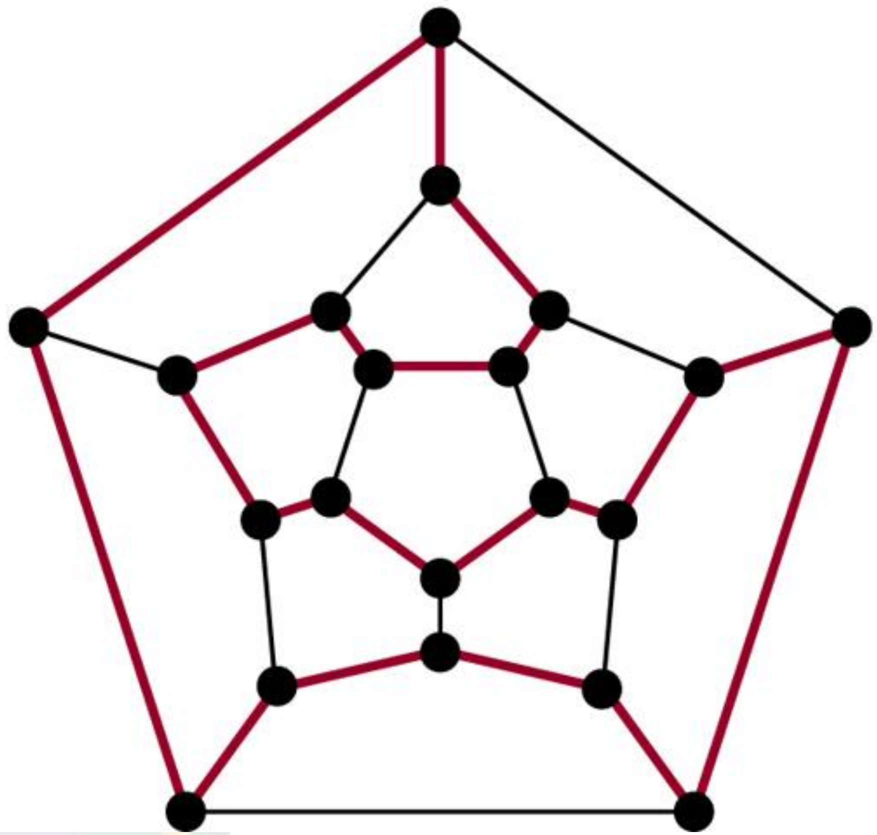
Hamilton Circuits



Dodecahedron puzzle and it equivalent graph

Is there a circuit in this graph that passes through each vertex exactly once?

Hamilton Circuits

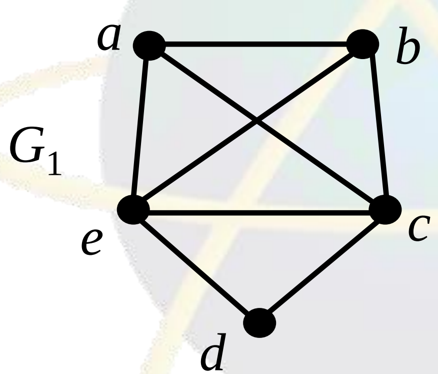


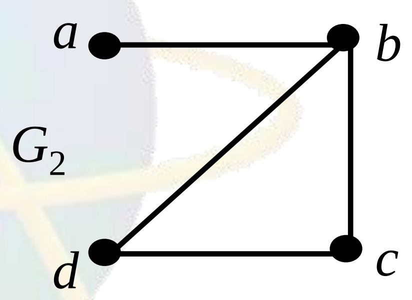
Yes; this is a circuit that passes through each vertex exactly once.

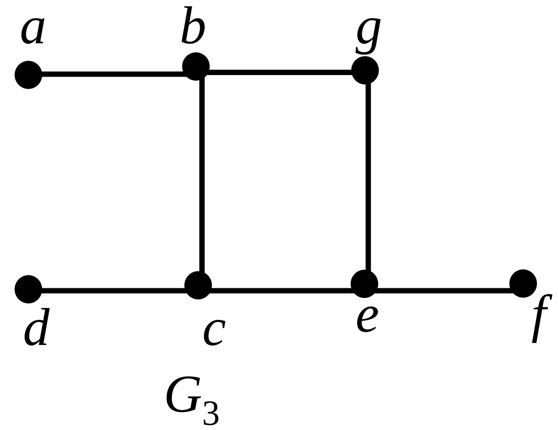
Hamilton Paths and Circuits

Def. 2: A *Hamilton path* is a path that traverses each vertex in a graph G exactly once.
A *Hamilton circuit* is a circuit that traverses each vertex in G exactly once.

Example 1. Which of the following graphs have a Hamilton circuit or a Hamilton path?







Hamilton circuit: G_1 Hamilton path: G_2

Theorem

If a graph G has a Hamilton circuit, then G has a subgraph H with the following properties:

1. H contains every vertex of G.
2. H is connected.
3. H has the same number of edges as vertices.
4. Every vertex of H has degree 2.

Example of Hamilton Circuit: Travelling Salesman Problem

A Hamilton circuit or path may be used to solve practical problems that require visiting “vertices”, such as:

- road intersections
- pipeline crossings
- communication network nodes

A classic example is the Travelling Salesman Problem – finding a Hamilton circuit in a complete graph such that the total weight of its edges is minimal.

Summary

Property	Euler	Hamilton
Repeated visits to a given vertices allowed?	Yes	No
Repeated traversals of a given edge allowed?	No	No
Skipped vertices allowed?	No	No
Skipped edges allowed?	No	Yes

Special Simple Graphs

- **Complete Graphs:** A complete graph on n vertices, denoted by K_n, is a simple graph that contains exactly one edge between each pair of distinct vertices.
- **Non-complete**: A simple graph for which there is at least one pair of distinct vertex not connected by an edge is called non-complete.

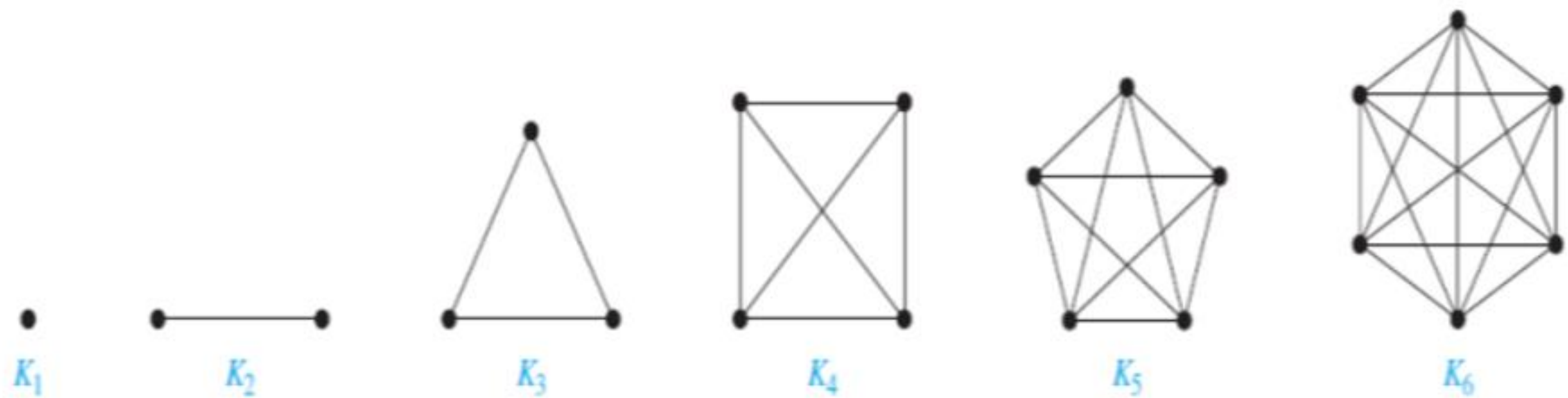


Figure: The Graphs K_n for $1 \leq n \leq 6$

Special Simple Graphs

- **Cycles:** A cycle C_n, $n \geq 3$, consists of n vertices $v_1, v_2, \ldots, v_n$ and edges $\{v_1, v_2\}, \{v_2, v_3\}, \ldots, \{v_{n-1}, v_n\}$, and $\{v_n, v_1\}$.

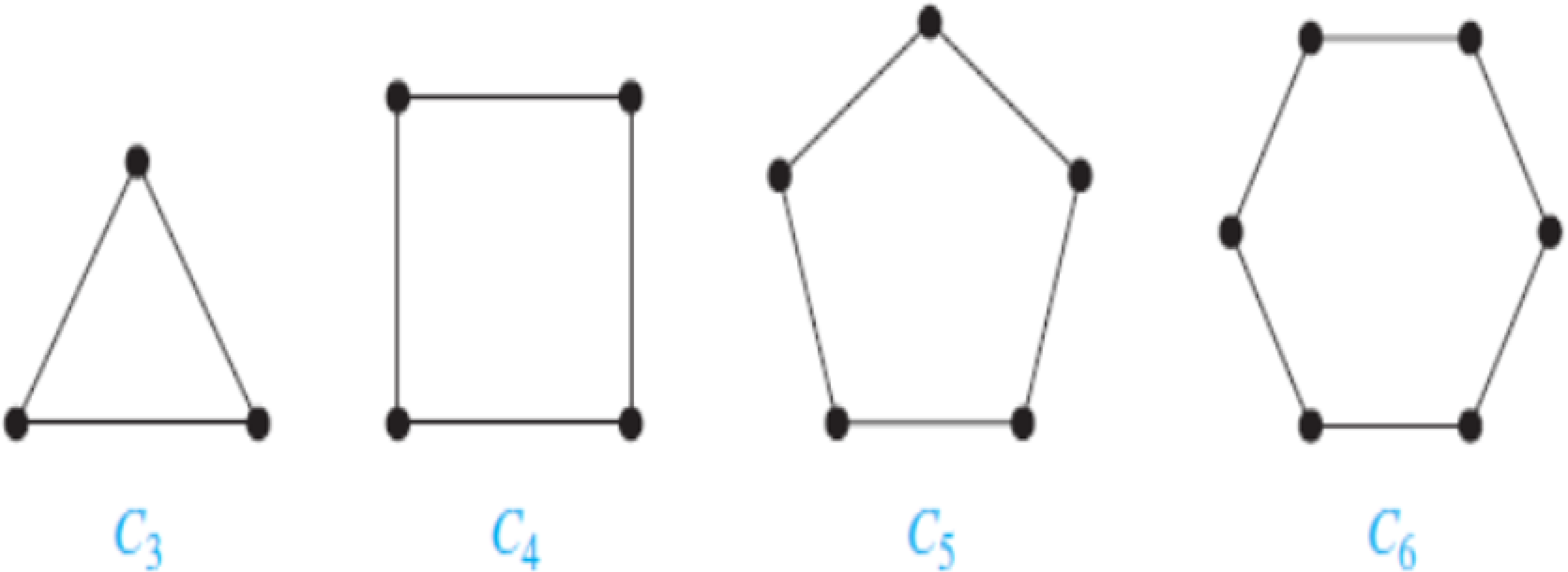


Figure 4.5: The Cycles C_3, C_4, C_5, and C_6

Special Simple Graphs

- **Wheel:** We obtain the wheel W_n when we add an additional vertex to the cycle C_n, for $n \geq 3$, and connect this new vertex to each of the n vertices in C_n by adding new edges.

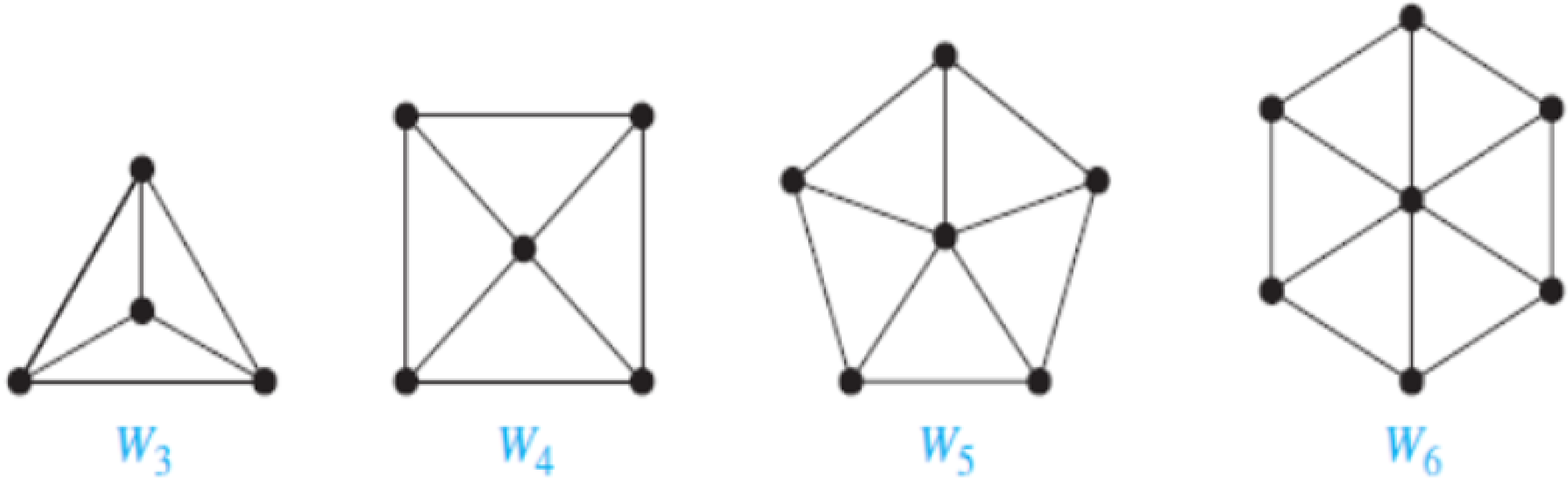


Figure 4.6: The Wheels W_3, W_4, W_5, and W_6

- **Regular Graph:** A graph is regular if every vertex has the same degree.
  - Example: The complete graph K_n is regular of degree n-1.
  - Example: A cycle graph is regular of degree 2.

Special Simple Graphs

- **Bipartite Graphs:** A simple graph G is called bipartite if its vertex set V can be partitioned into two disjoint sets V1 and V2 such that every edge in the graph connects a vertex in V1 and a vertex in V2 ***[so that no edges in G connect either two vertices in V1 or two vertices in V2].*** When this condition holds, we call the pair (V1, V2) a bipartition of the vertex set V of G.

Special Simple Graphs

- **Example I:** Is C_3 bipartite?
  - No, because there is no way to partition the vertices into two sets so that there are no edges with both endpoints in the same set.
- **Example2:** Is C_6 bipartite?
  - C_6 is bipartite because its vertex set can be partitioned into the two sets $V_1 = \{v_1, v_3, v_5\}$ and $V_2 = \{v_2, v_4, v_6\}$, and every edge of C_6 connects a vertex in V_1 and a vertex in V_2.

Figure: C_6 change into Bipartite

Special Simple Graphs

- **Complete Bipartite**: Graphs A complete bipartite graph $K_{m,n}$ is a graph that has its vertex set partitioned into two subsets of m and n vertices, respectively with an edge between two vertices if and only if one vertex is in the first subset and the other vertex is in the second subset.

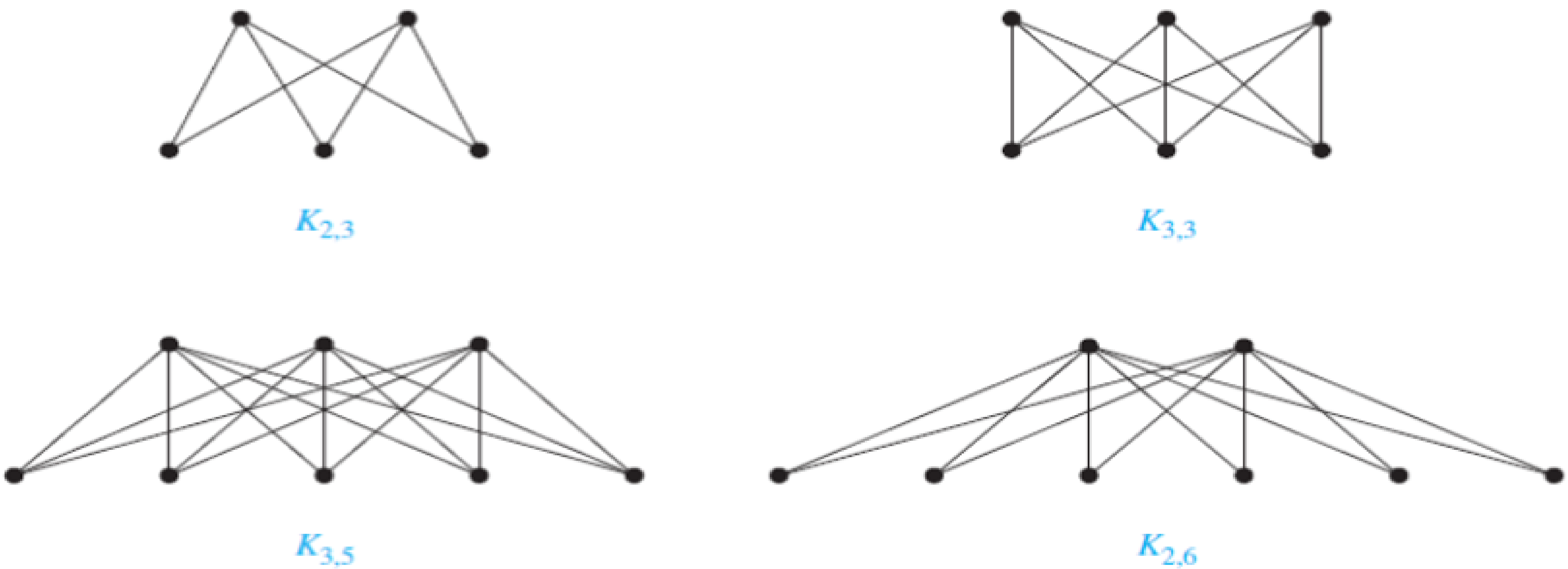
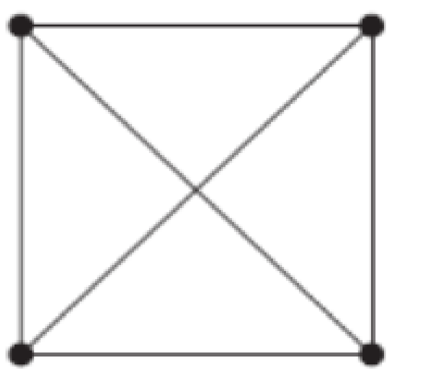


Figure: Complete Bipartite

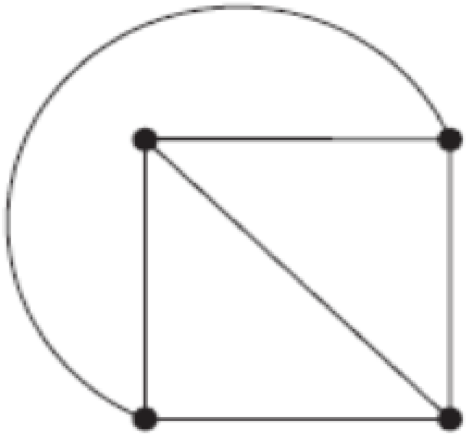
Planner Graph

- **Planar Graph**: A graph is called planar if it can be drawn in the plane without any edges crossing, i.e., it can be drawn on the plane in such a way that its edges intersect only at their endpoints. In other words, it can be drawn in such a way that no edges cross each other.

- ***Example 1: Is K_4 planar?***
  - ✓ Solution: K_4 is planar because it can be drawn without crossings, as shown in Figure.



Graph K_4



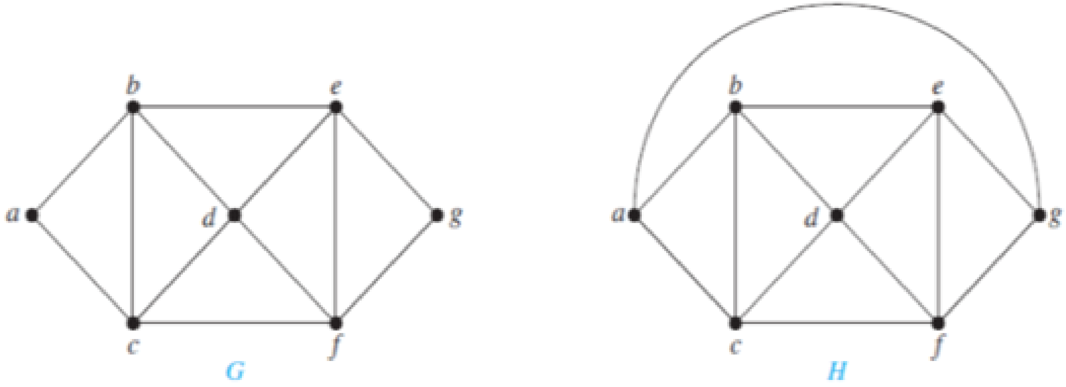
K_4 drawn with no crossing

Graph Coloring

- **Graph Coloring:** A coloring of a simple graph is the assignment of a color to each vertex of the graph so that no two adjacent vertices are assigned the same color.
- **Chromatic number**: The chromatic number of a graph is the least number of colors needed for a coloring of this graph. The chromatic number of a graph G is denoted by $\chi(G)$. (Here χ is the Greek letter chi.)
- **Theorem 1: The Four-Color Theorem:** The chromatic number of a planar graph is no greater than four.

Graph Coloring

- ***Example1:*** What are the chromatic numbers of the graphs G and H shown in Figure?



- **Solution:** The chromatic number of G is at least three, because the vertices a, b, and c must be assigned different colors. To see if G can be colored with three colors, assign red to a, blue to b, and green to c. Then, d can (and must) be colored red because it is adjacent to b and c. Furthermore, e can (and must) be colored green because it is adjacent only to vertices colored red and blue, and f can (and must) be colored blue because it is adjacent only to vertices colored red and green. Finally, g can (and must) be colored red because it is adjacent only to vertices colored blue and green. This produces a coloring of G using exactly three colors.

Graph Coloring

- The graph H is made up of the graph G with an edge connecting a and g. Any attempt to color H using three colors must follow the same reasoning as that used to color G, except at the last stage, when all vertices other than g have been colored. Then, because g is adjacent (in H) to vertices colored red, blue, and green, a fourth color, say brown, needs to be used. Hence, H has a chromatic number equal to 4.

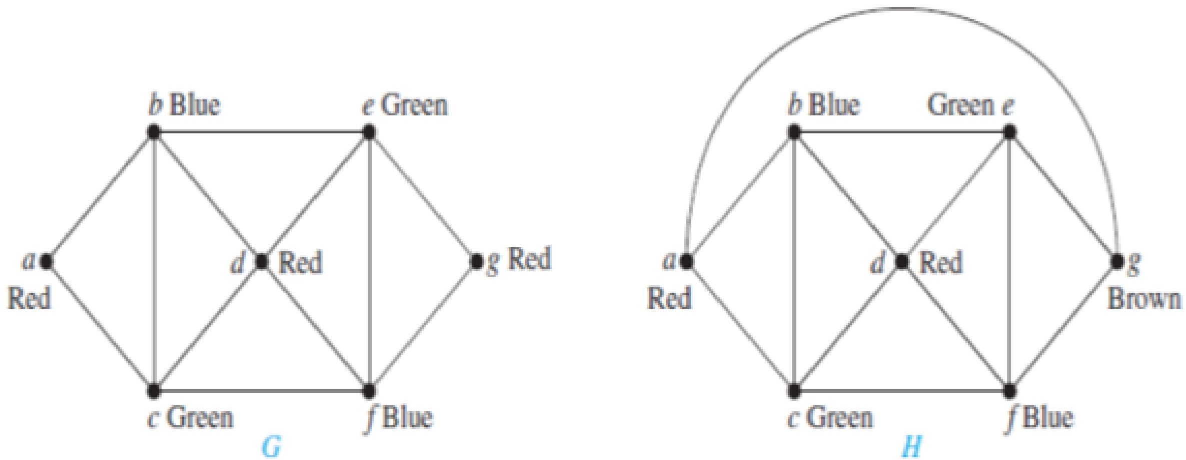


Figure: After Coloring Graph G and H

Graph Coloring

- **Example 2:** ***What is the chromatic number of K_n?***
  - Solution: A coloring of K_n can be constructed using n colors by assigning a different color to each vertex. No two vertices can be assigned the same color, because every two vertices of this graph are adjacent. Hence, the chromatic number of K_n is n. That is, $\chi(K_n) = n$.

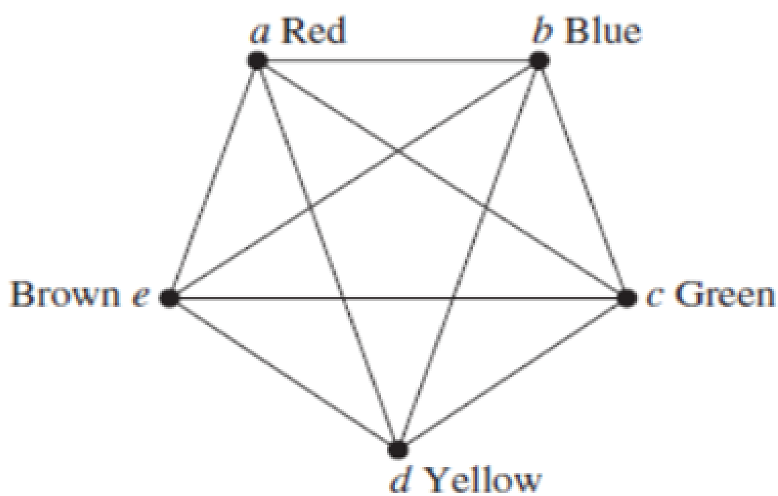


Figure: A Coloring of K_5

Graph Coloring

- **Example 3:** ***What is the chromatic number of the complete bipartite graph $K_{m,n}$, where m and n are positive integers?***
  - Solution: The number of colors needed may seem to depend on m and n. Only two colors are needed, because $K_{m,n}$ is a bipartite graph. Hence, $\chi(K_{m,n}) = 2$. This means that we can color the set of m vertices with one color and the set of n vertices with a second color. Because edges connect only a vertex from the set of m vertices and a vertex from the set of n vertices, no two adjacent vertices have the same color.

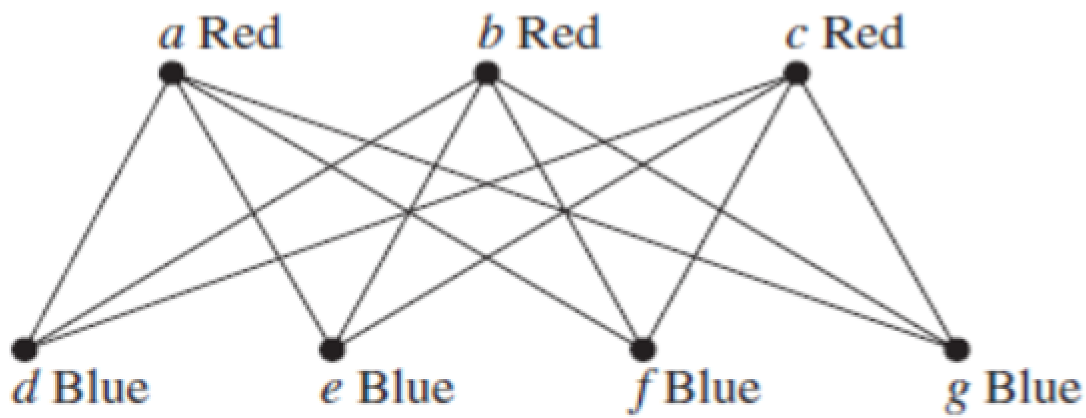


Figure: A Coloring of $K_{3,4}$

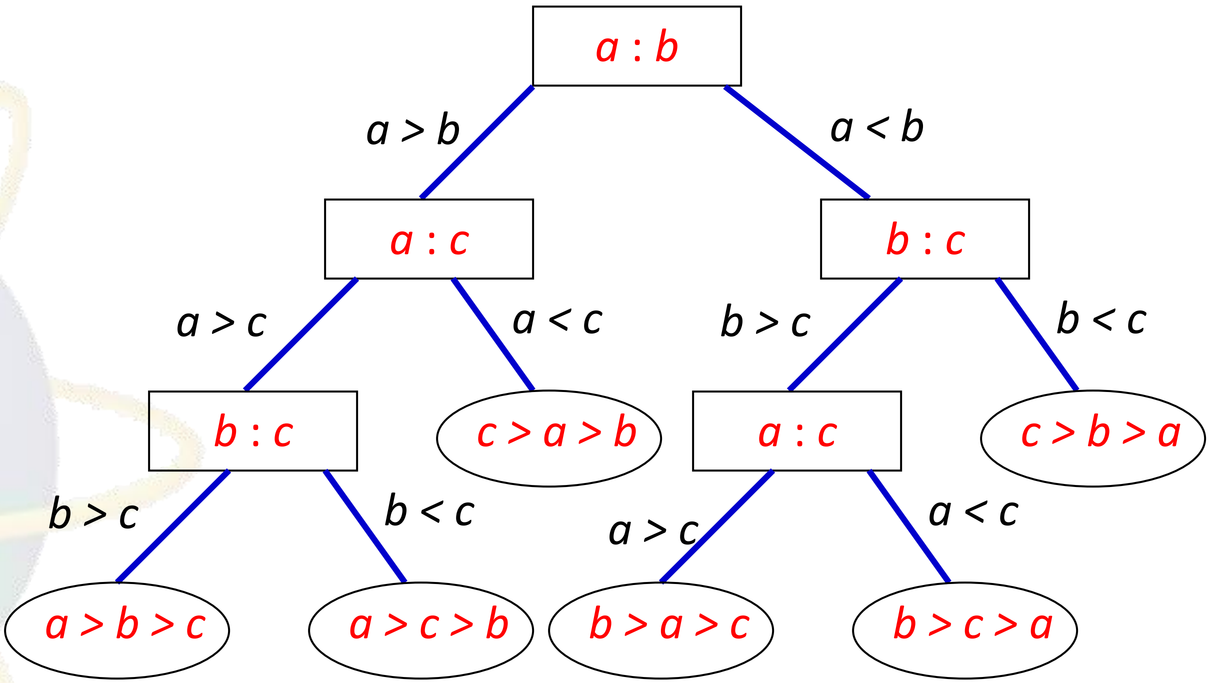
Example of Tree: Decision Trees

- A rooted tree in which each internal vertex corresponds to a decision, with a subtree at these vertices for each possible outcome of decision.

- The possible solutions of the problem correspond to the paths to the leaves of this rooted tree.

EXAMPLE

A Decision tree that orders the elements of the list *a*, *b*, *c*



Tree

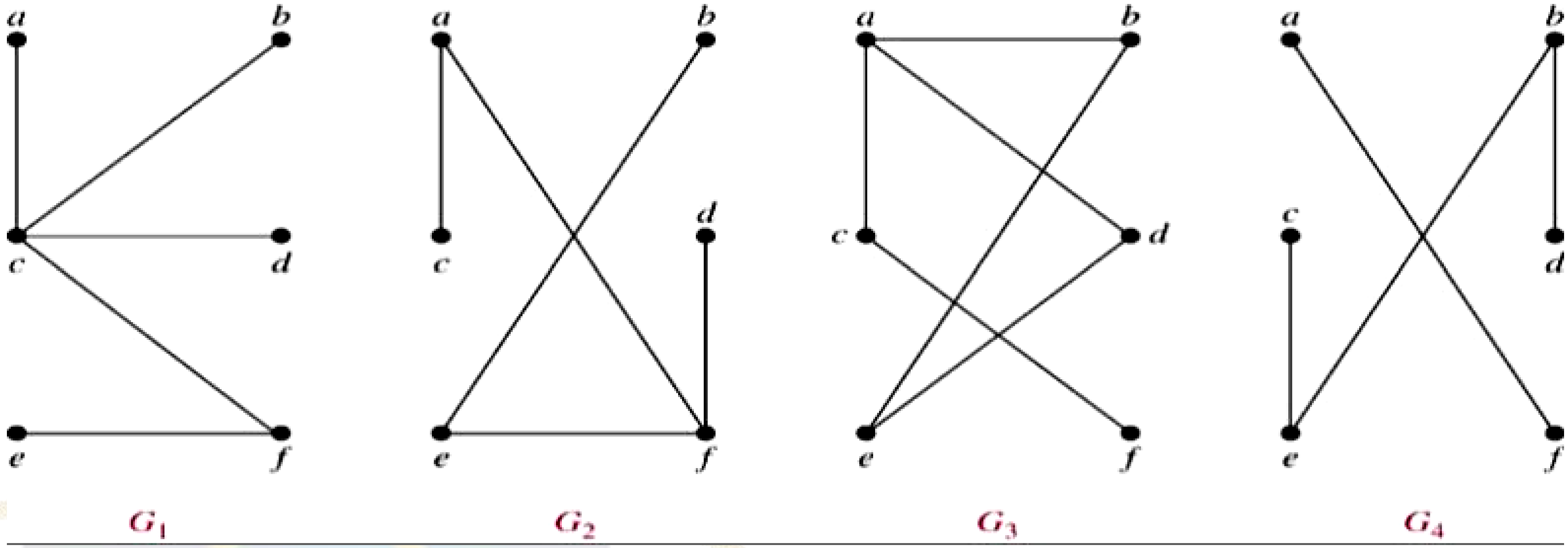
Definition:

- A tree is a connected undirected graph with no simple circuits.
- Since a tree cannot have a circuit, a tree cannot contain multiple edges or loops.
- Therefore, any tree must be a simple graph.

Theorem:

- An undirected graph is a tree if and only if there is a unique simple path between any two of its vertices.
- In general, we use trees to represent hierarchical structures.

Example 1. Which of the graphs are trees?



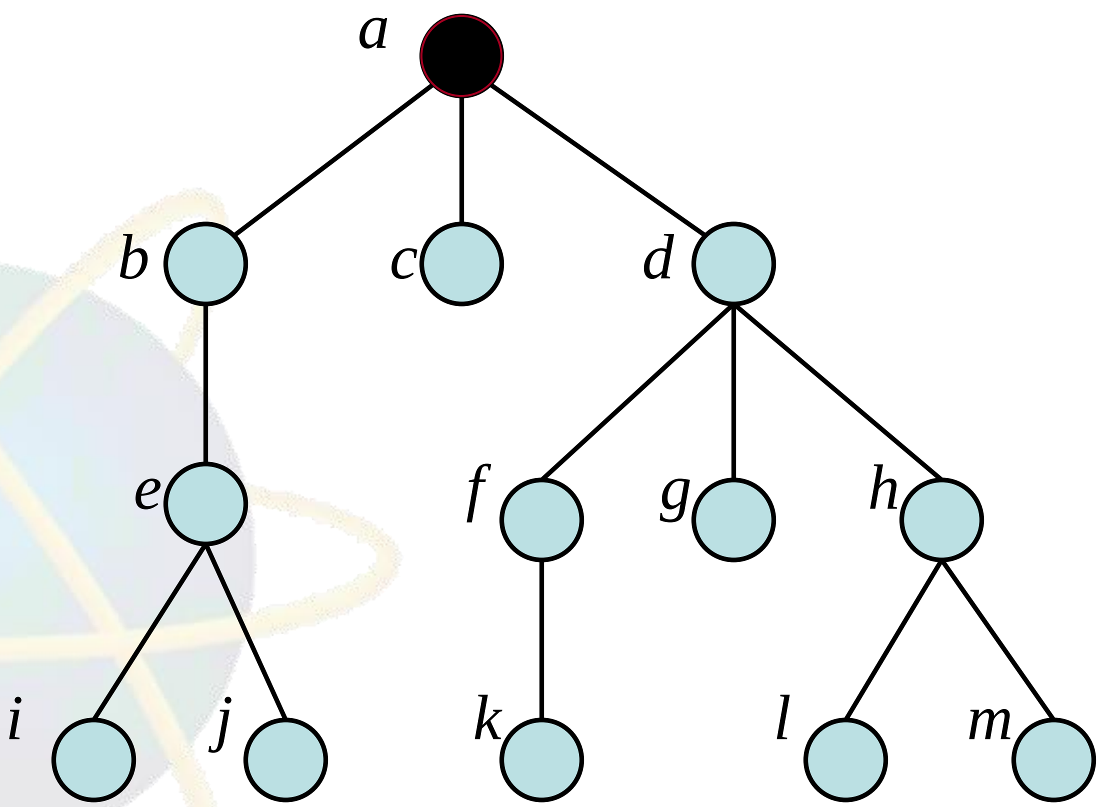
Sol: G_1, G_2

Tree Terminology

- If v is a vertex of tree T other than the root, the **parent** of v is the unique vertex u such that there is a directed edge from u to v.
- When u is the parent of v, v is called the **child** of u.
- If two vertices share the same parent, then they are called **siblings**.

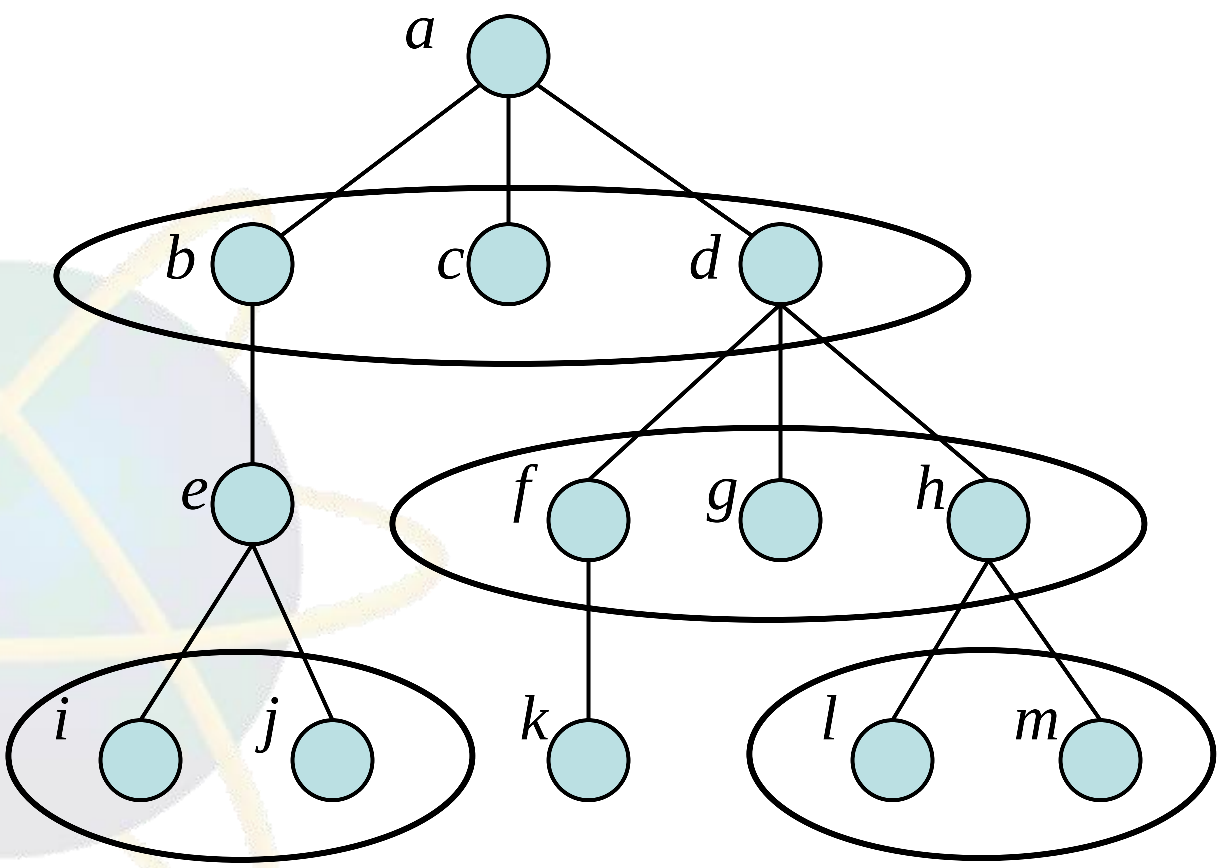
Example

Root



Example

Siblings

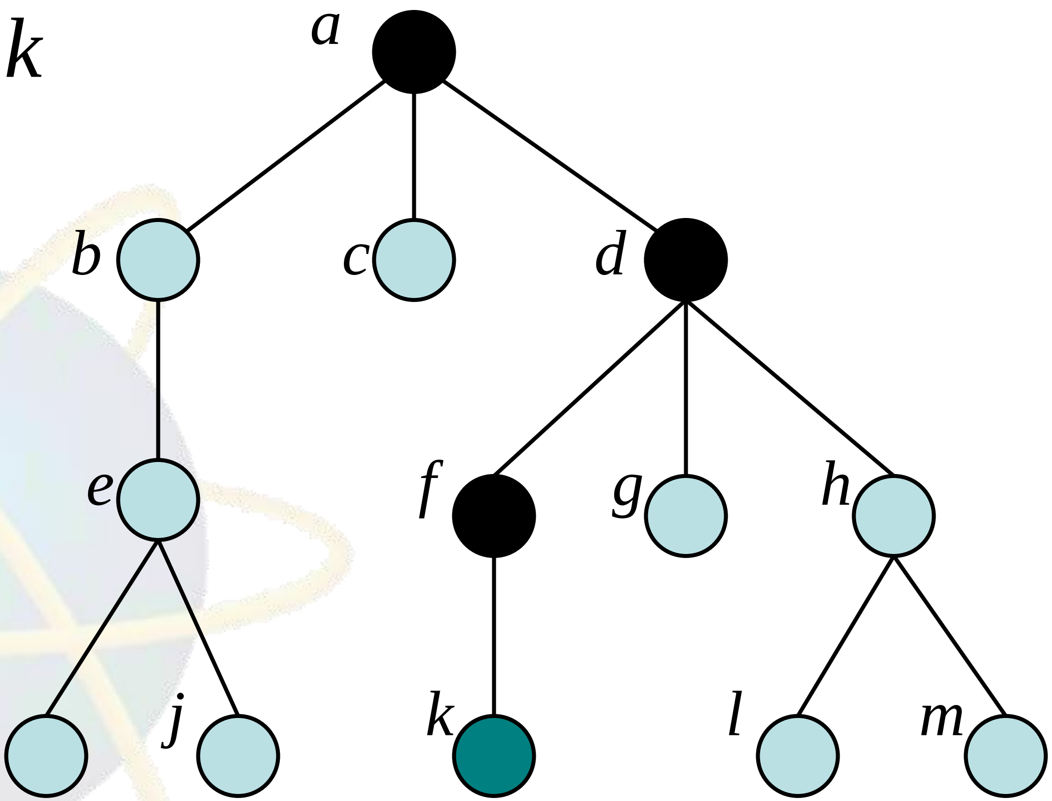


Tree Terminology (Cont.)

- The **ancestors** of a vertex other than the root are the vertices in the path from the root to this vertex, excluding the vertex itself and including the root.
- The **descendants** of a vertex *v* are those vertices that have *v* as an ancestor.

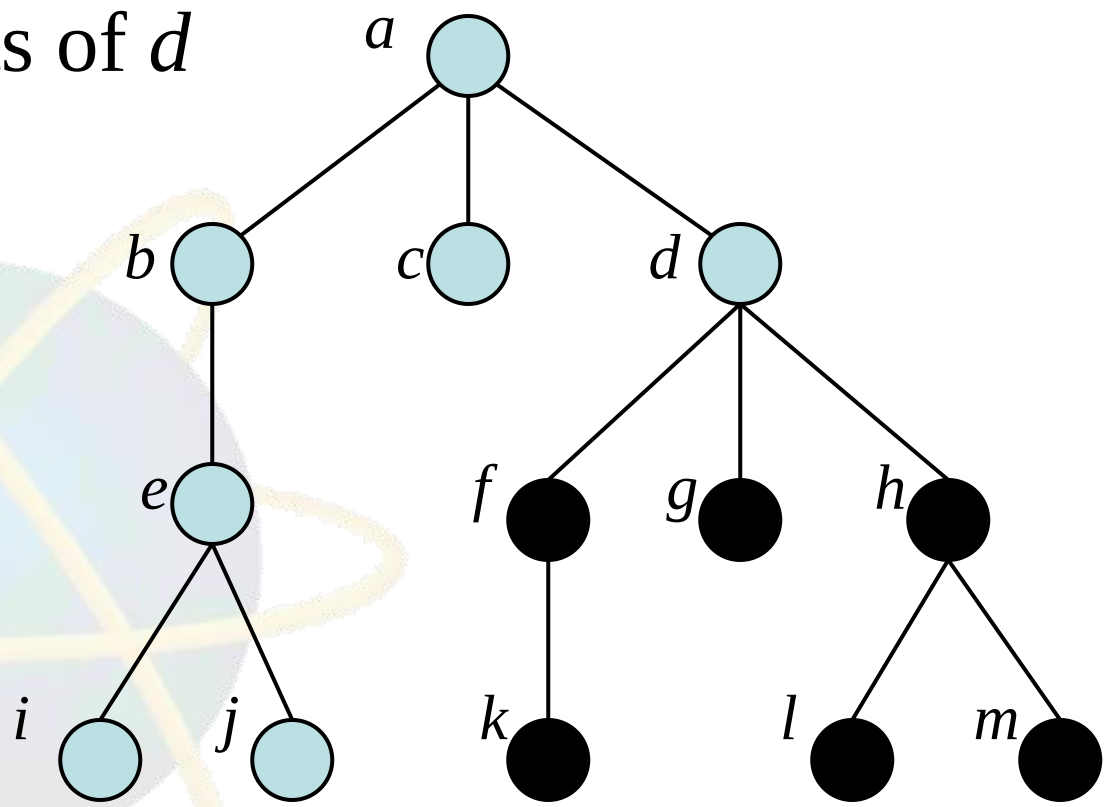
Example

Ancestors of k



Example

Descendants of *d*

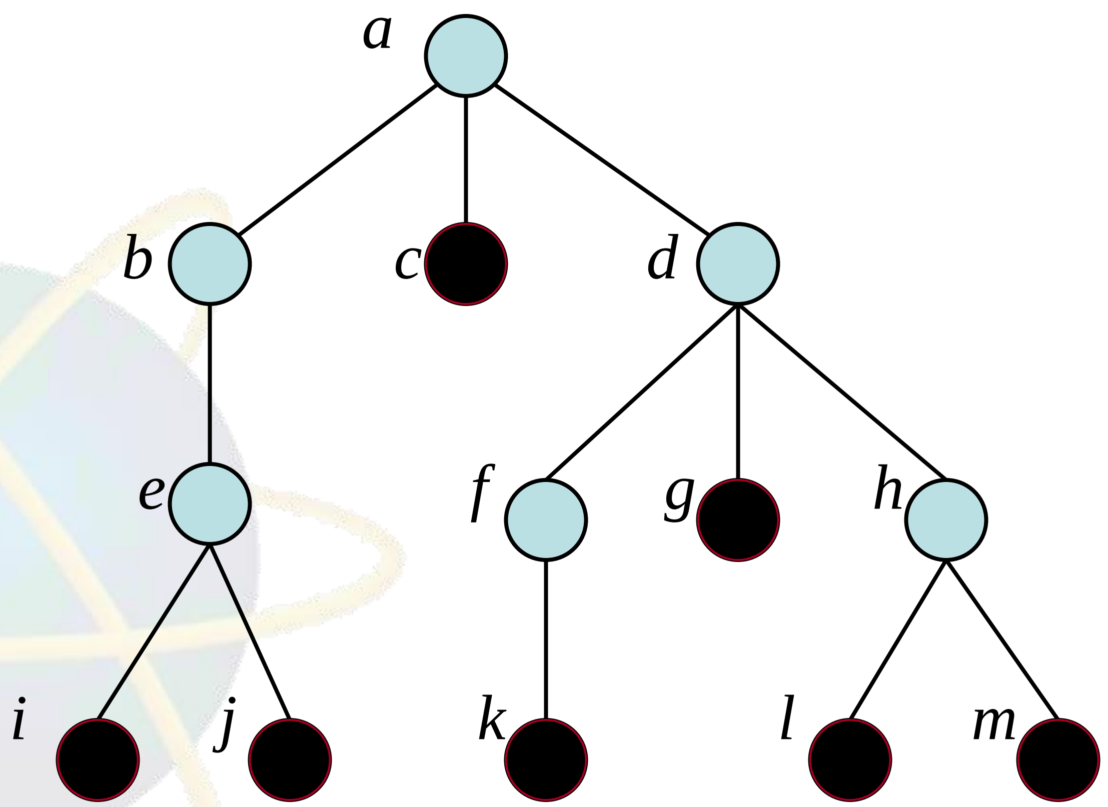


Tree Terminology (Cont.)

- A vertex with no children is called a **leaf**.
- Vertices with children are called ***internal vertices.***

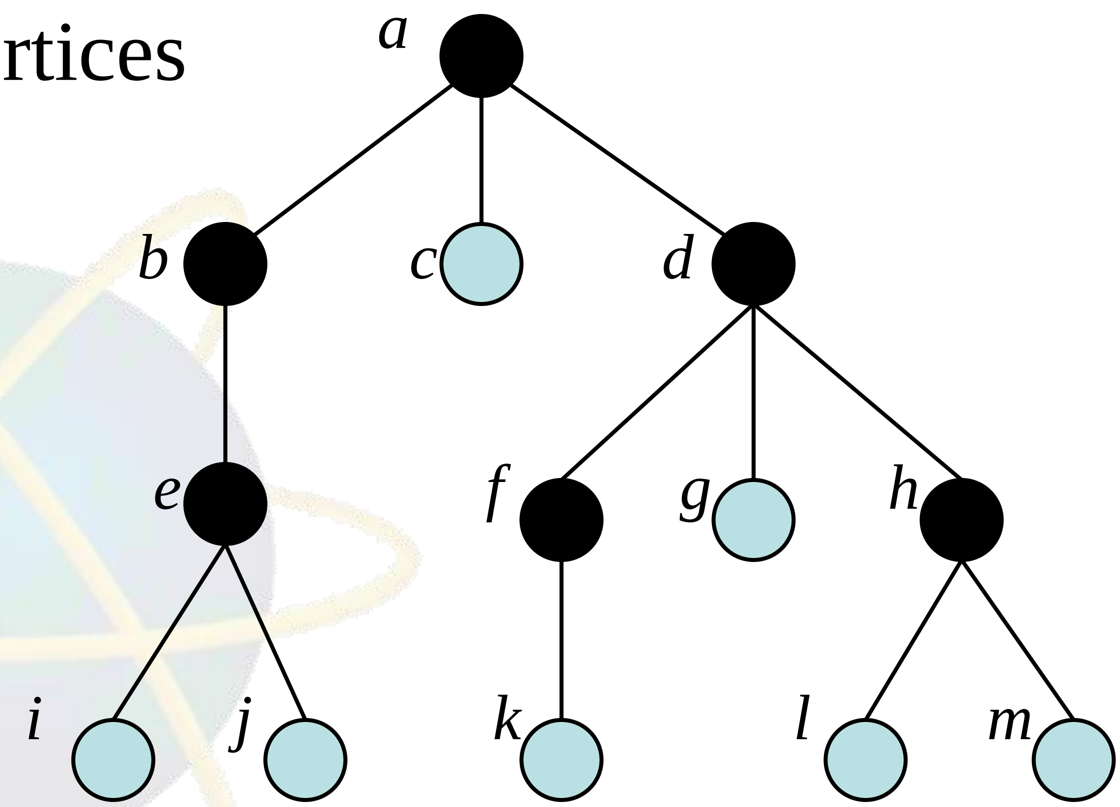
Example

Leaves



Example

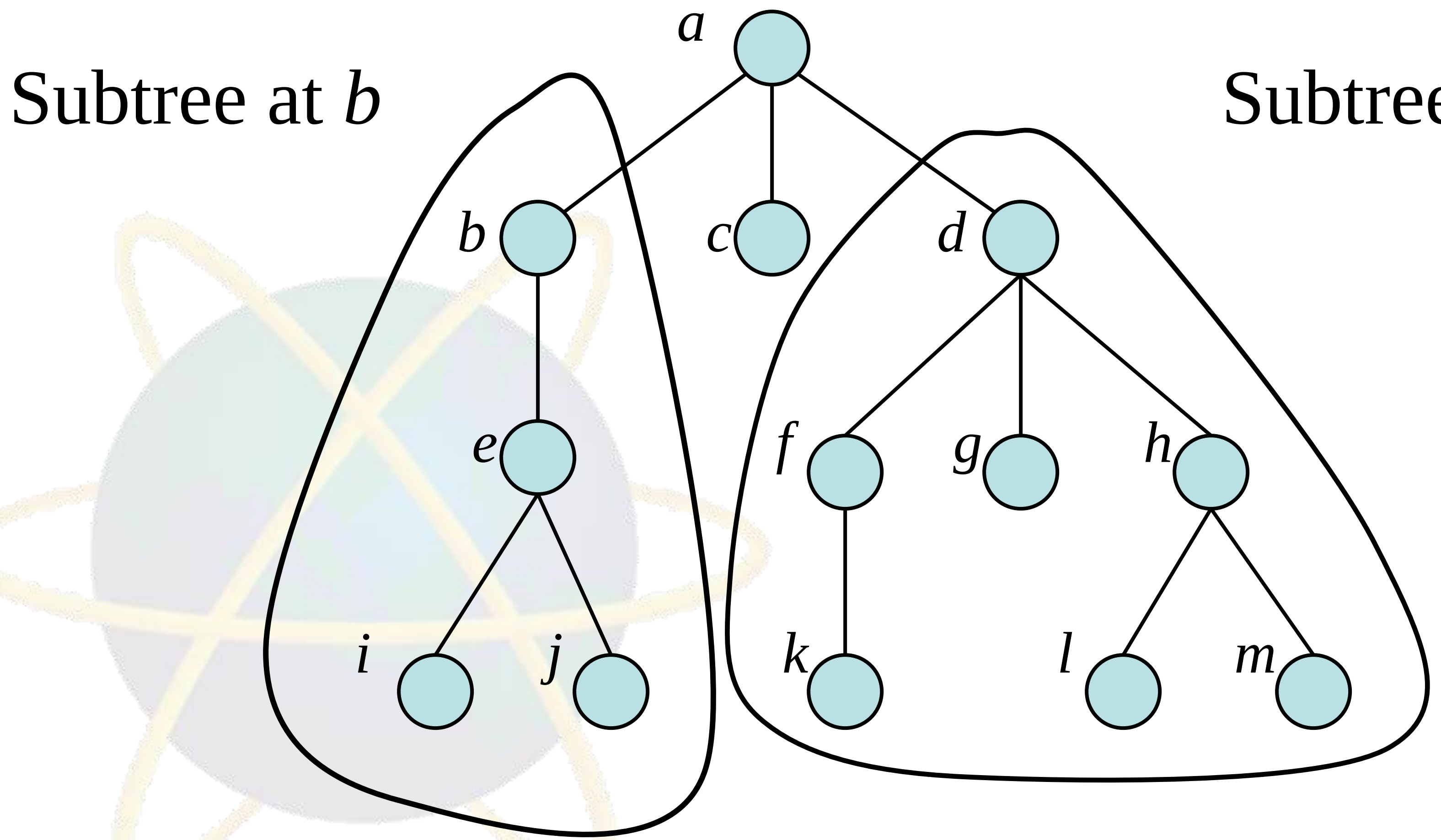
Internal vertices



Tree Terminology (Cont.)

- 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.

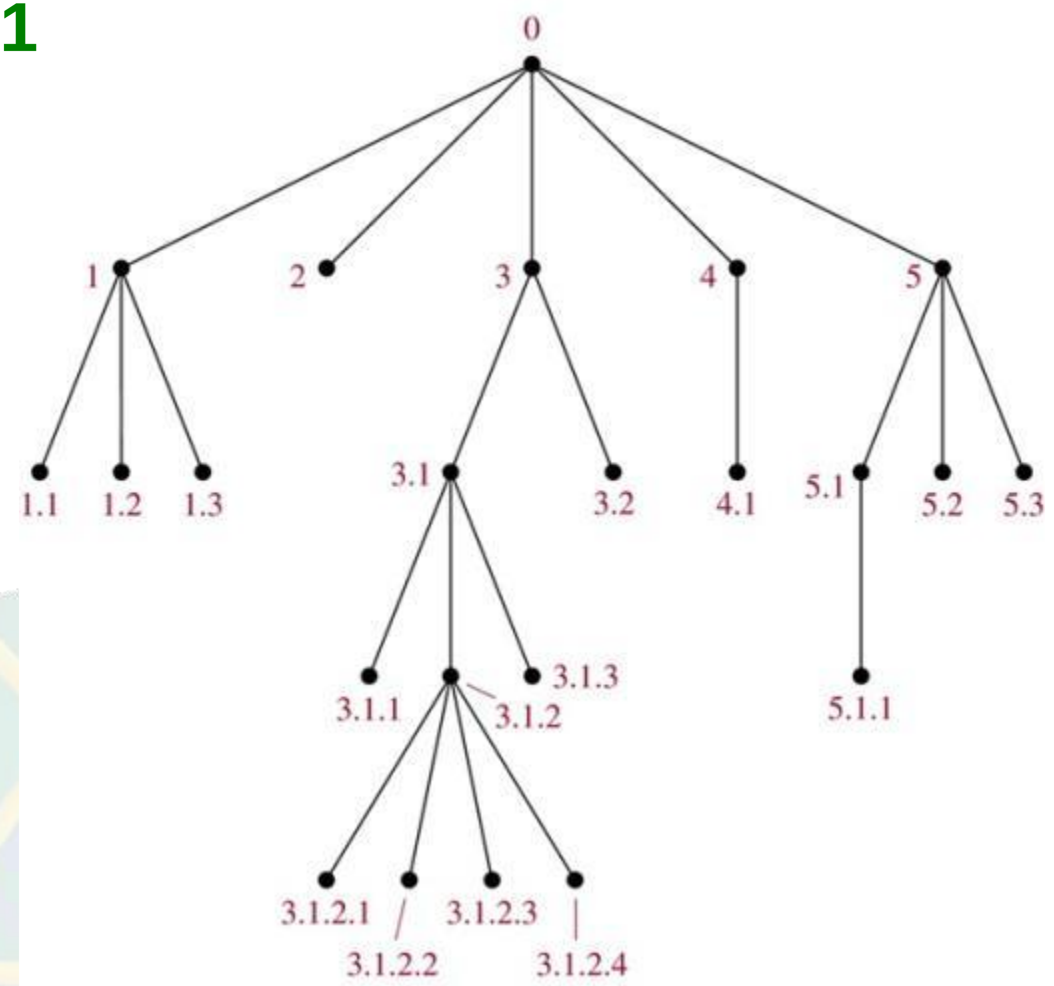
Example



Tree Traversal

- Ordered trees are often used to restore data/info.
- Tree traversal is a procedure for systematically visiting each vertex of an ordered rooted tree to access data.
- Tree traversal algorithm
  - Preorder, inorder and postorder traversal

Example 1



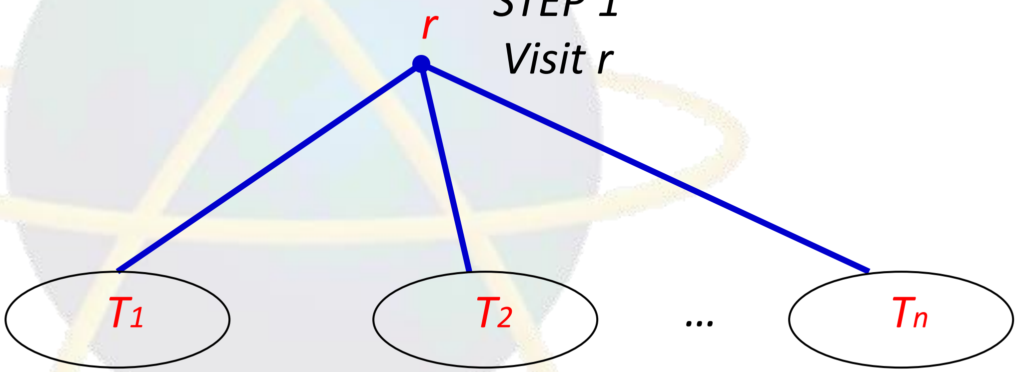
The lexicographic ordering is:

0 < 1<1.1 < 1.2 < 1.3 < 2 < 3 < 3.1 < 3.1.1 < 3.1.2 < 3.1.2.1 < 3.1.2.2 < 3.1.2.3 < 3.1.2.4 < 3.1.3 < 3.2 < 4 < 4.1 < 5 < 5.1 < 5.1.1 < 5.2 < 5.3

Preorder Traversal

- Let T be an ordered rooted tree with root r.
  - If T consists only of r, then r is the *preorder* traversal of T.
  - If $T_1, T_2, \ldots, T_n$ are subtrees at r from left to right in T, then the preorder traversal begins by visiting r, continues by traversing T_1 in preorder, then T_2 in preorder, and so on until T_n is traversed in preorder.

STEP 1
Visit r



STEP 2
Visit T_1 in preorder

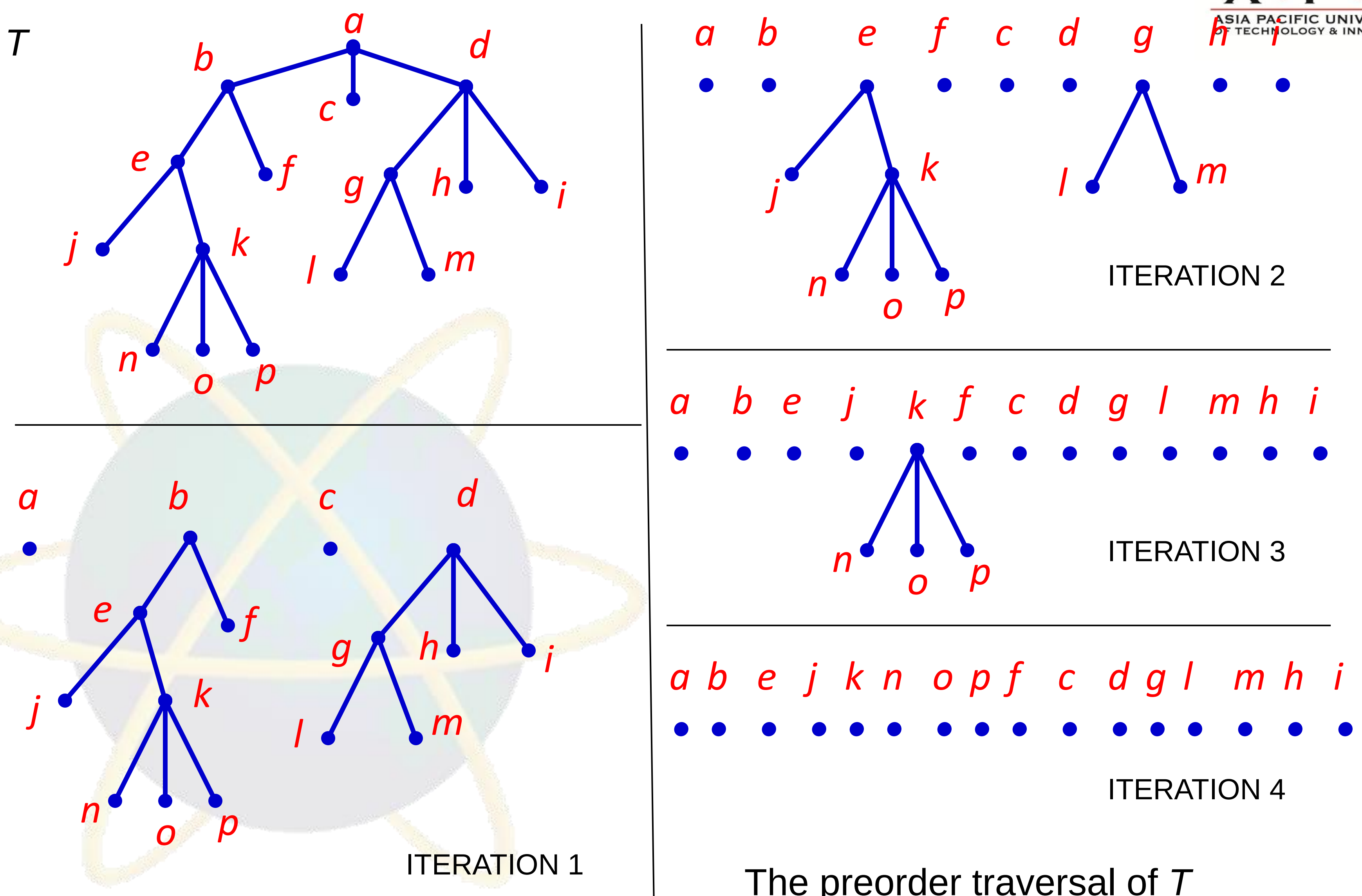
STEP 3
Visit T_2 in preorder

STEP n+1
Visit T_n in preorder

TIPS

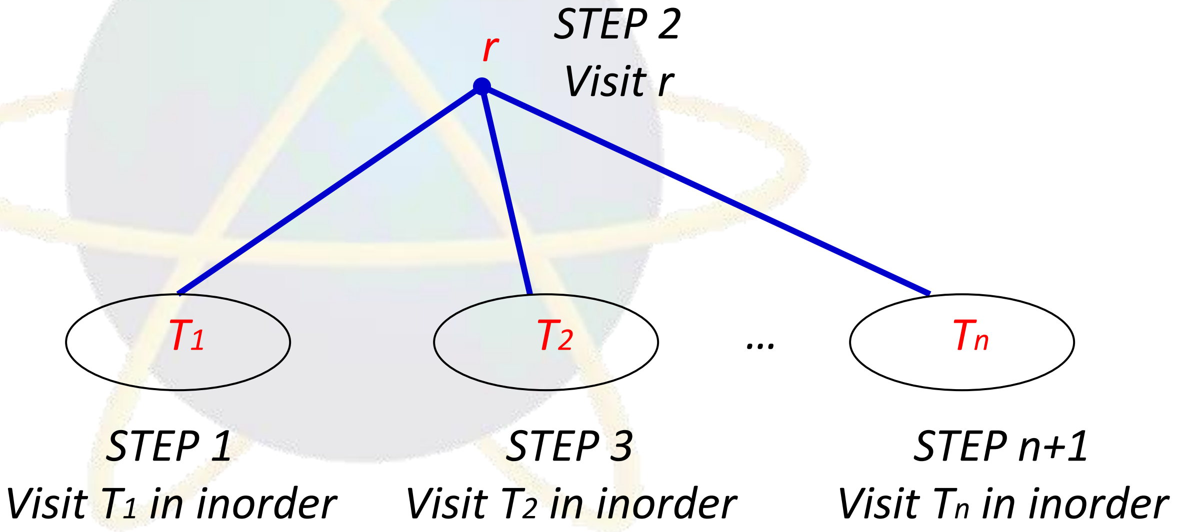
Preorder Traversal: Visit root, visit subtrees left to right

EXAMPLE: Preorder Traversal



Inorder Traversal

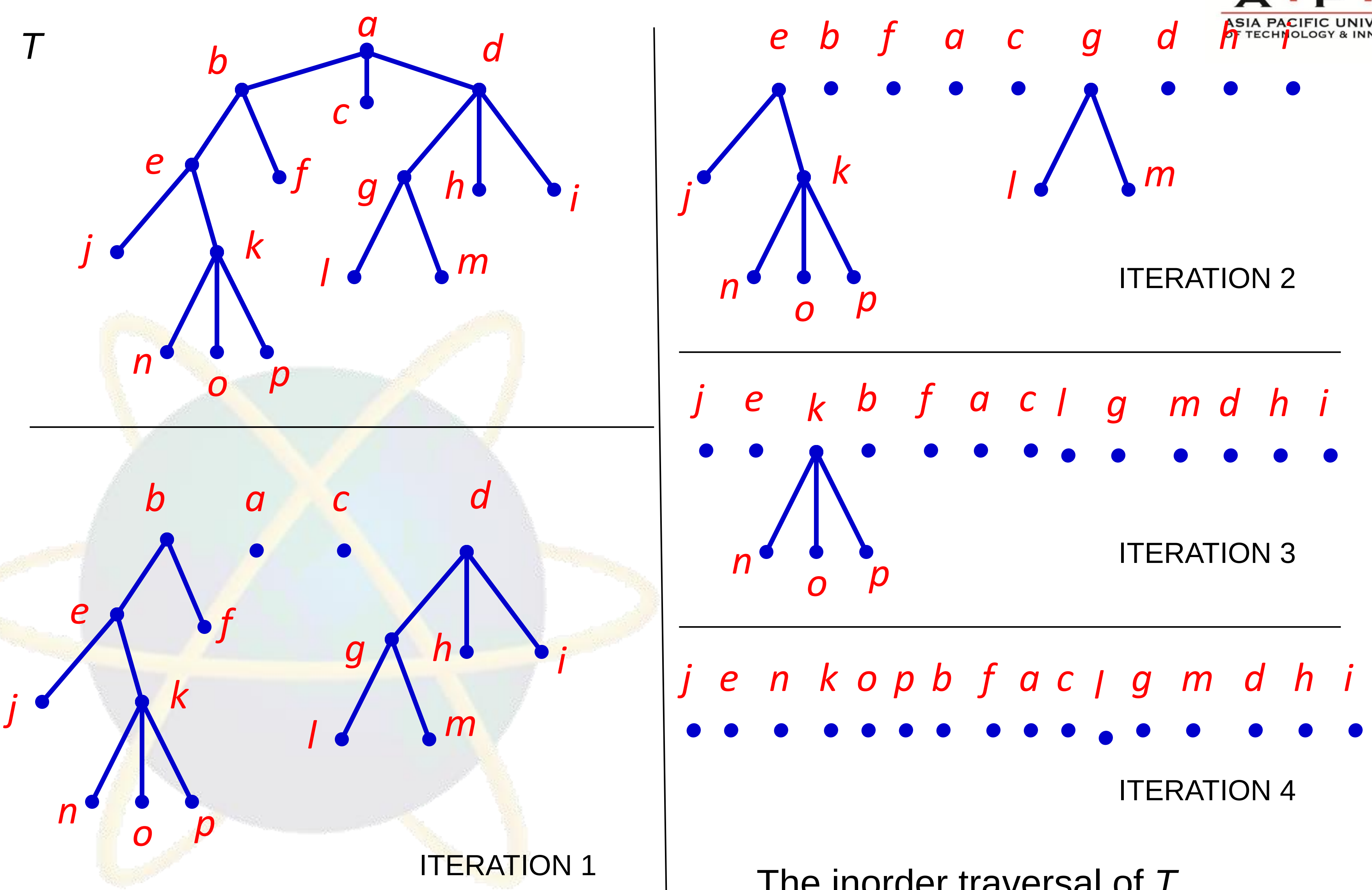
- Let T be an ordered rooted tree with root r.
  - If T consists only of r, then r is the *inorder* traversal of T.
  - If $T_1, T_2, \ldots, T_n$ are subtrees at r from left to right in T, then the inorder traversal begins by traversing T_1 in inorder, then visiting r, continues by traversing T_2 in inorder, and so on until T_n is traversed in inorder.



TIPS

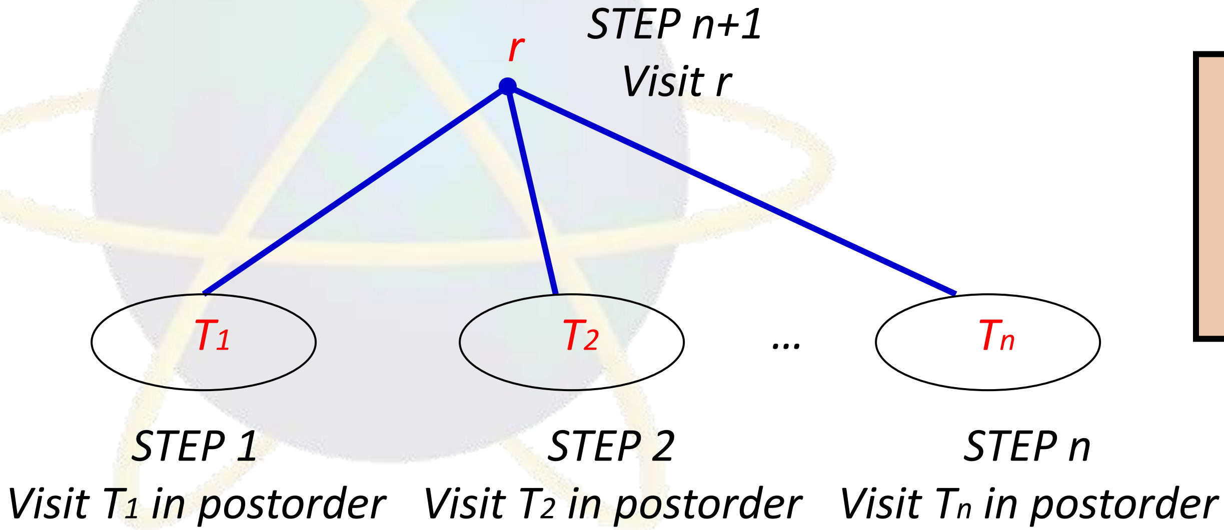
Inorder Traversal: Visit leftmost subtree, Visit root, Visit other subtrees left to right.

EXAMPLE: Inorder Traversal



Postorder Traversal

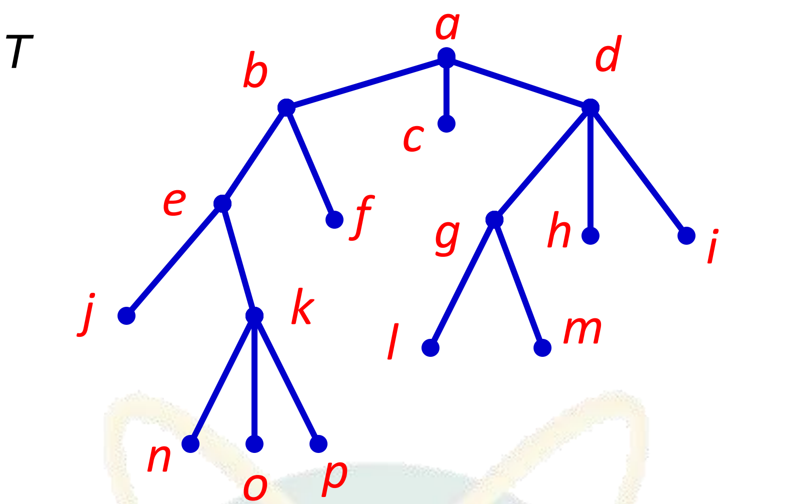
- Let T be an ordered rooted tree with root r.
  - If T consists only of r, then r is the *postorder* traversal of T.
  - If $T_1, T_2, \ldots, T_n$ are subtrees at r from left to right in T, then the preorder traversal begins by traversing T_1 in postorder, then T_2 in postorder, and so on until T_n is traversed in postorder and ends by visiting r.

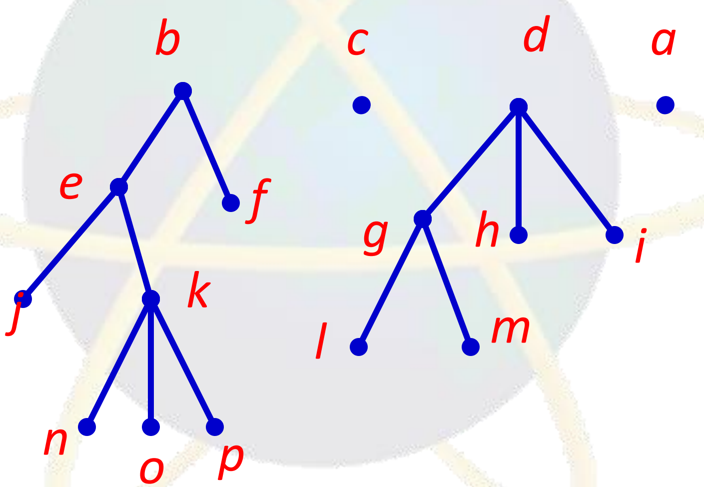


<u>TIPS</u>

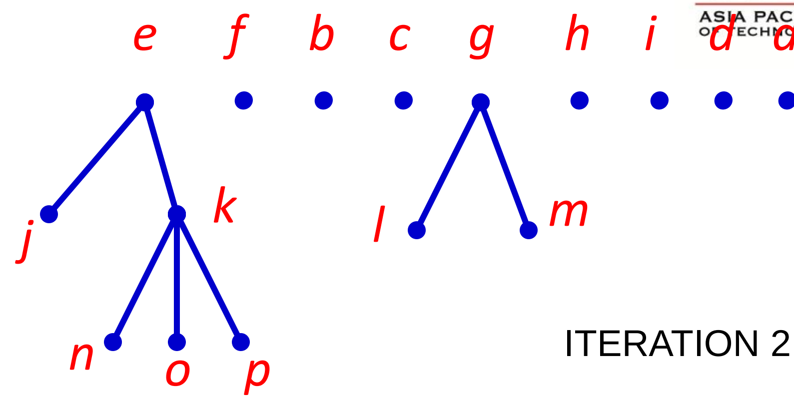
Postorder Traversal: Visit subtrees left to right, Visit root.

EXAMPLE: Postorder Traversal

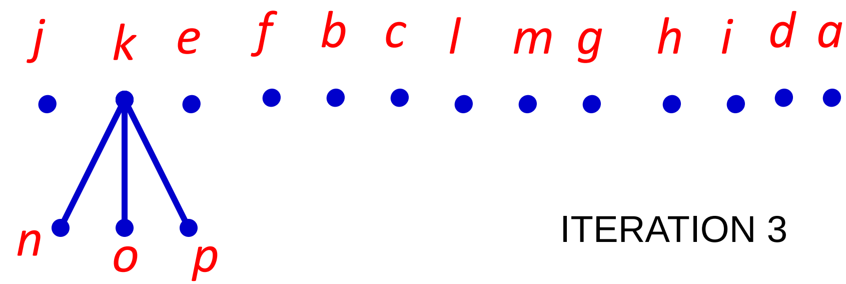




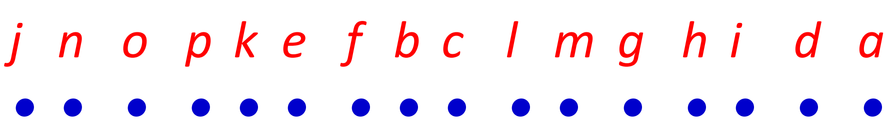
ITERATION 1



ITERATION 2



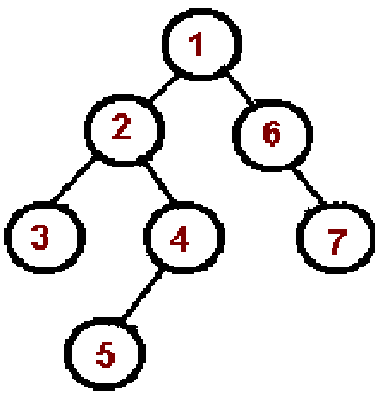
ITERATION 3



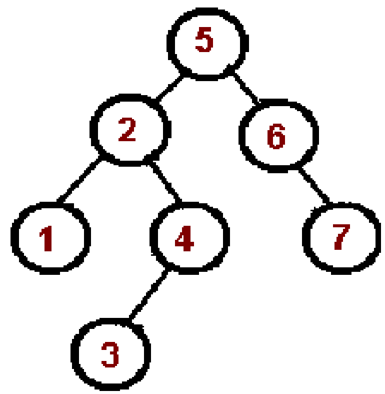
ITERATION 4

The postorder traversal of *T*

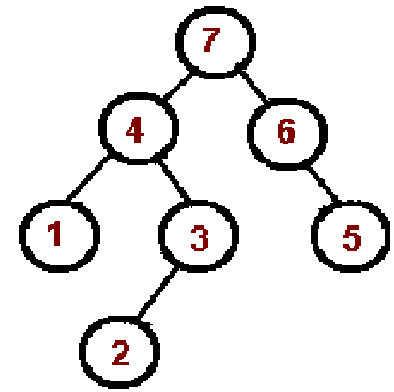
Summary



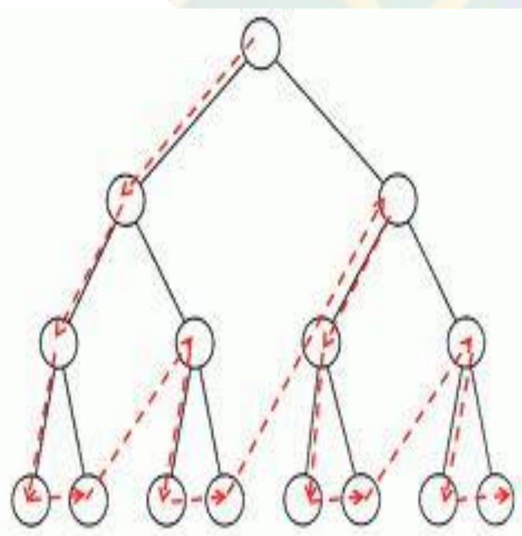
preorder

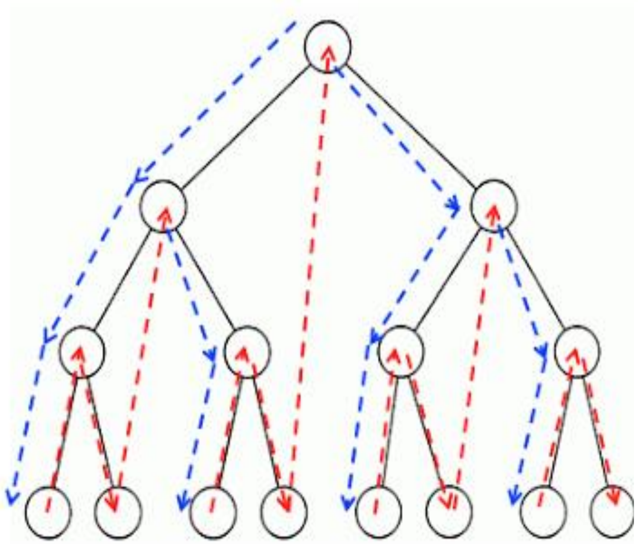


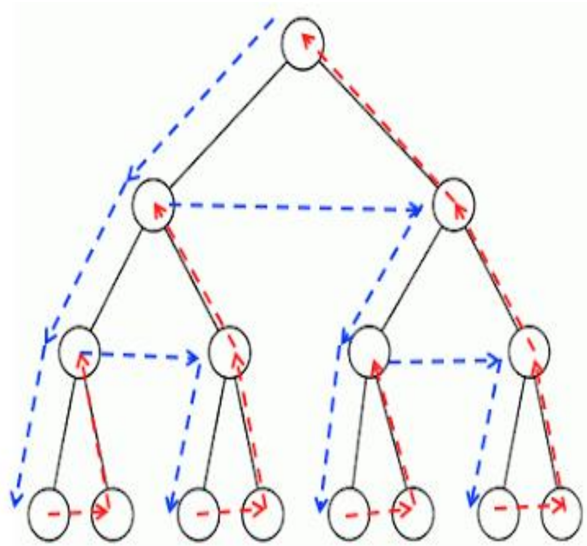
inorder



postorder



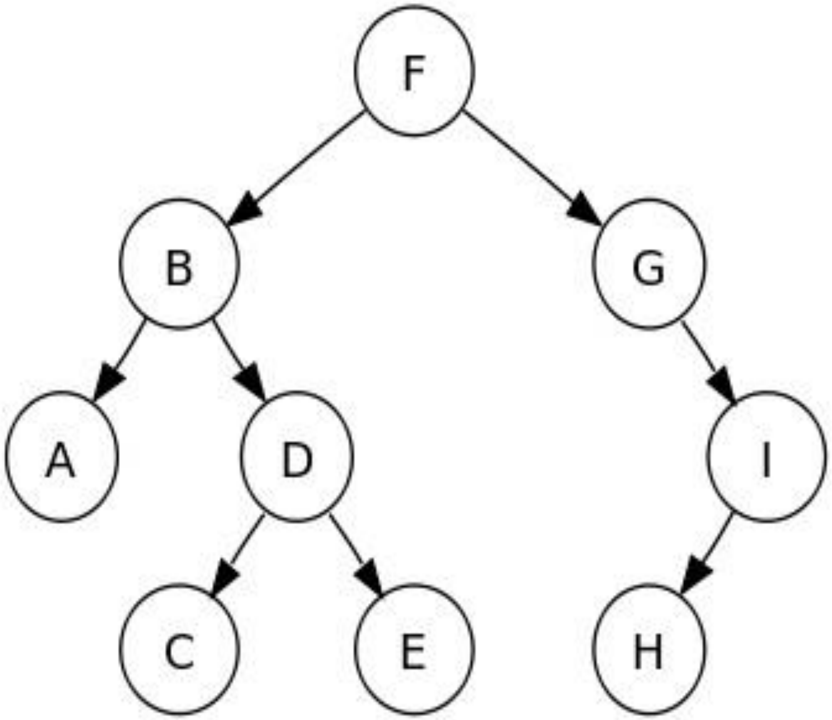




Exercise

In which order does
a) preorder traversal
b) inorder traversal
c) postorder traversal
visit the vertices in the ordered rooted tree?

Binary tree:



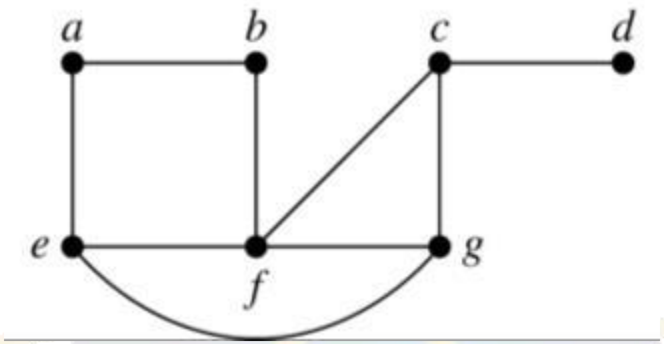
- Preorder traversal sequence: F, B, A, D, C, E, G, I, H (root, left, right)
- Inorder traversal sequence: A, B, C, D, E, F, G, H, I (left, root, right)
- Postorder traversal sequence: A, C, E, D, B, H, I, G, F (left, right, root)

Spanning Trees

Introduction

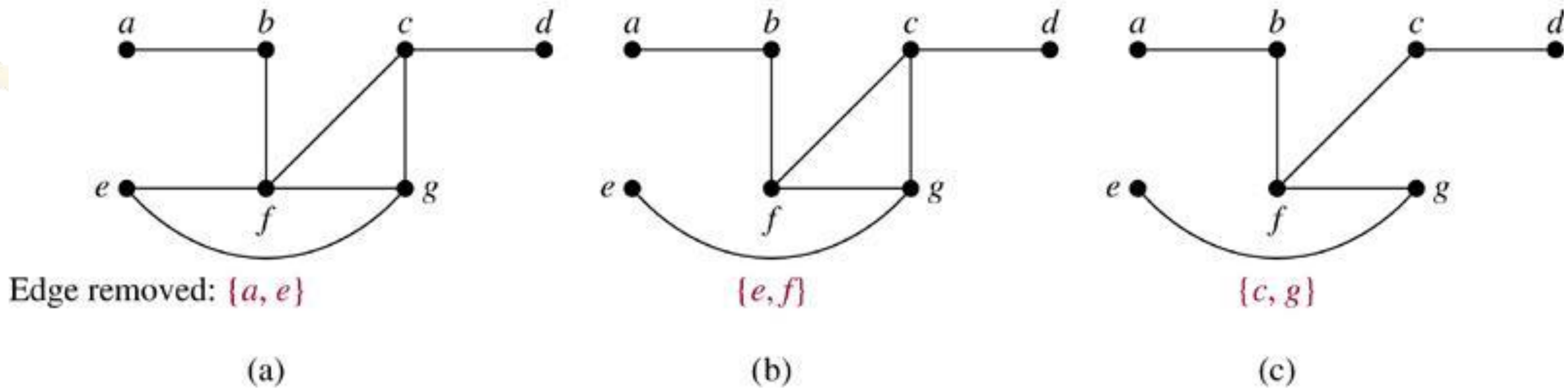
Def. Let G be a simple graph. A spanning tree of G is a subgraph of G that is a tree containing every vertex of G.

Example 1 Find a spanning tree of G.

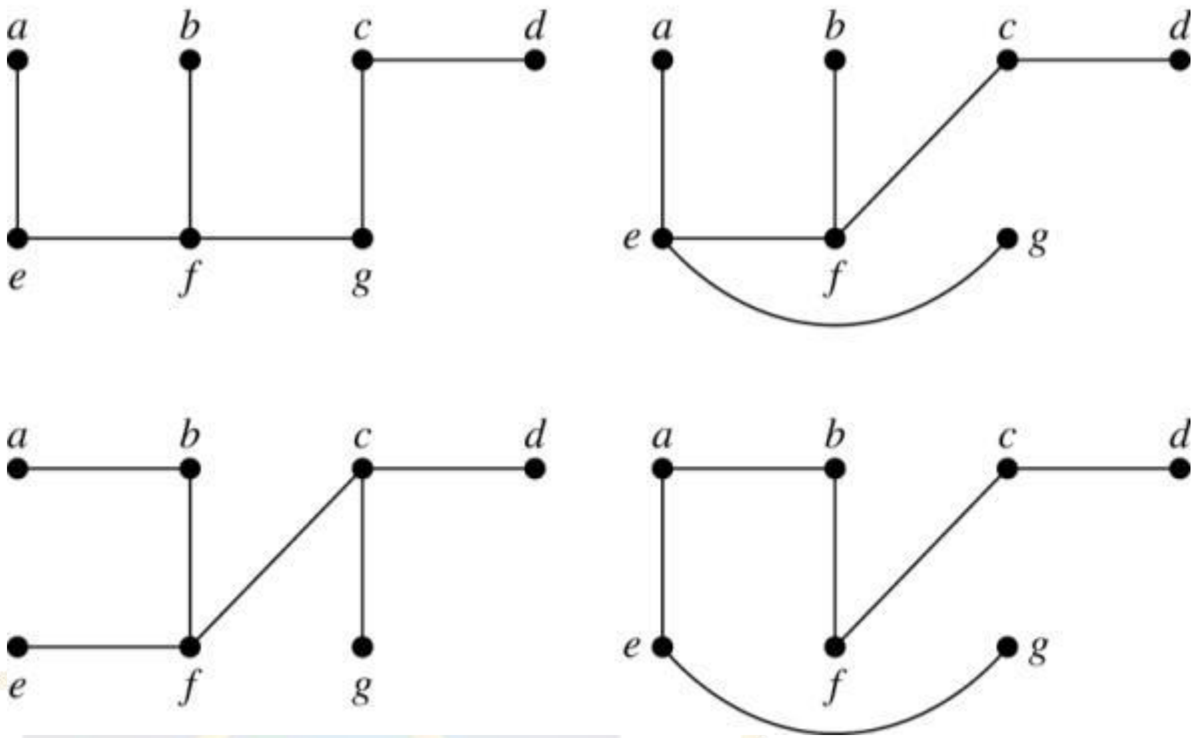


Sol.

Remove an edge from any circuit.
(repeat until no circuit exists)



Four spanning trees of G:



Thm 1 A simple graph is connected if and only if it has a spanning tree.

Spanning Trees

- **Spanning Tree**: Let G be a simple graph. A spanning tree of G is a subgraph of G that is a tree containing every vertex of G.
- A simple graph is connected if and only if it has a spanning tree.

Properties of spanning tree

1. Connected graph G can have more than one spanning tree.
2. All possible spanning trees of graph G have the same number of edges and vertices.
3. A spanning tree does not have any cycle.
4. A complete undirected graph can have maximum n^{n-2} number of spanning trees, where n is the number of nodes. In above graph G, $4^{4-2} = 4^2 \Rightarrow 16$ spanning trees are possible.
5. Spanning tree must include every vertex of graph G.
6. A spanning tree can't be disconnected. That means it is minimally connected.
7. A spanning tree has n vertices and n – 1 edges.

Spanning Trees

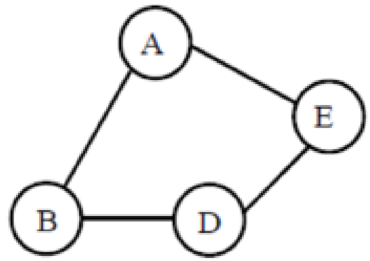
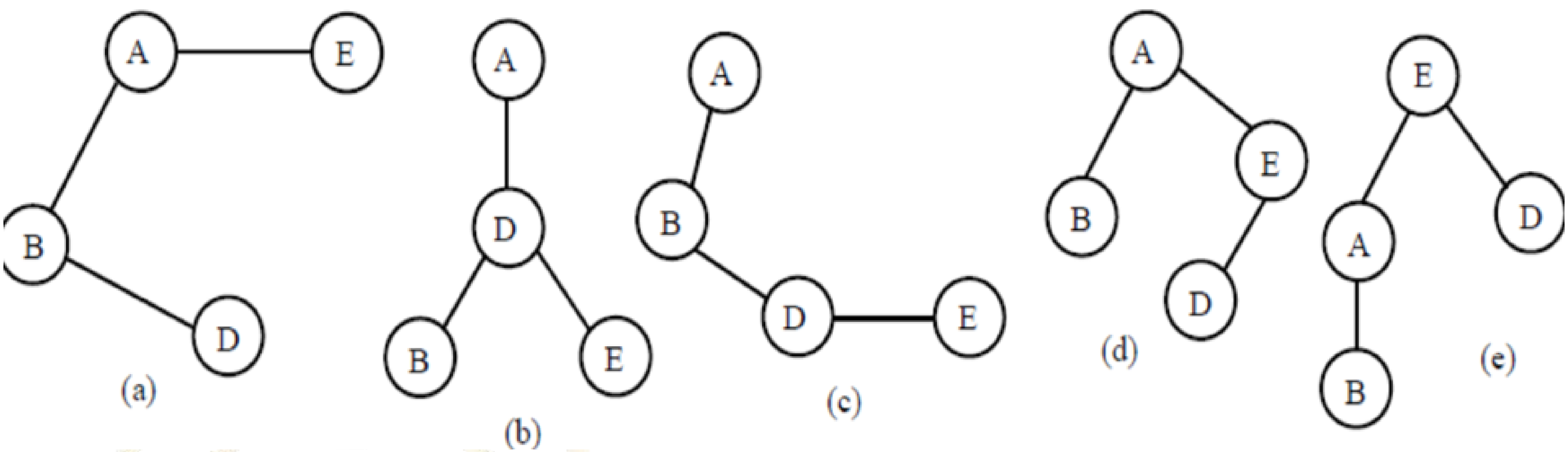


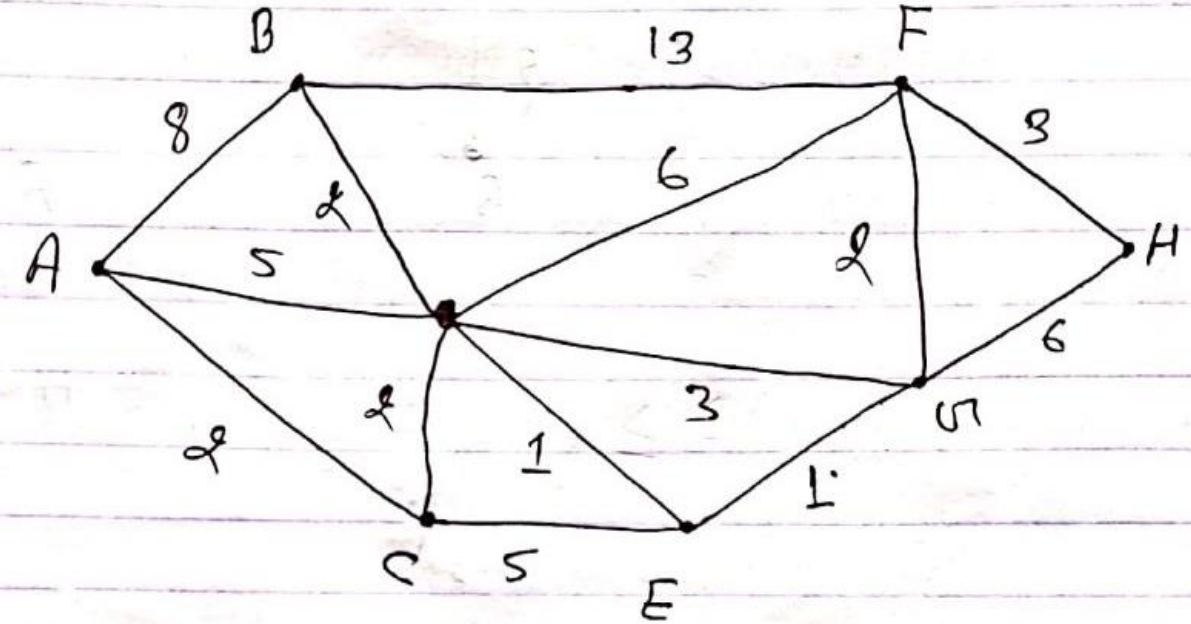
Figure: Graph G

- Find the spanning tree of graph G



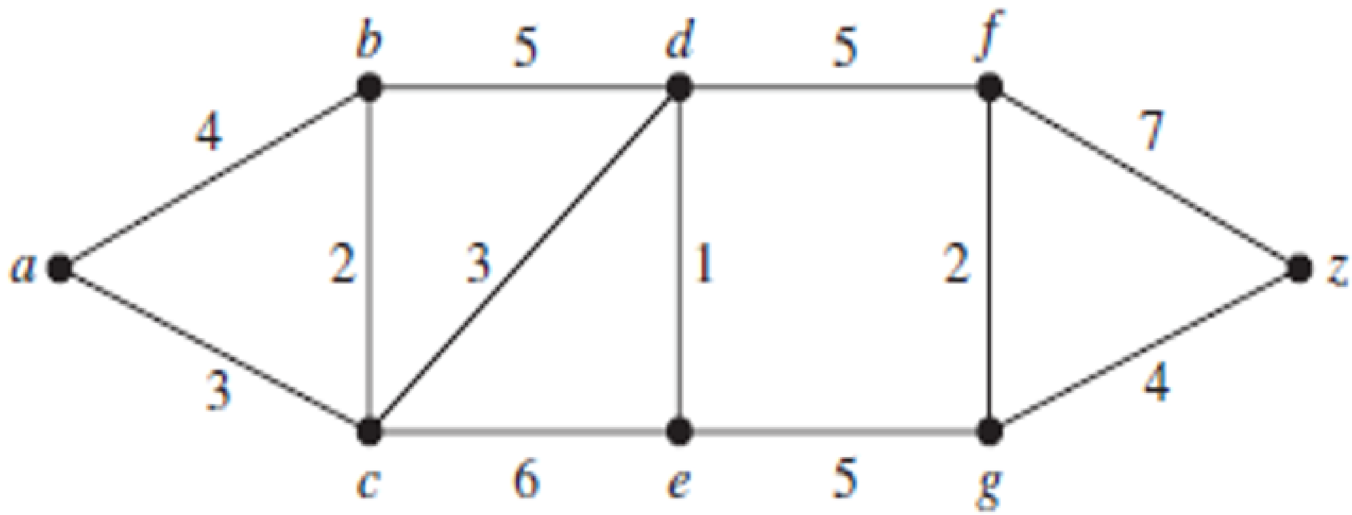
⇒ Shortest path Algorithm : Dijkstra's Algorithm

Example: Use Dijkstra's algorithm to find the length of a shortest path betn the vertices A to H in the weighted graph.



Dijkstra's Algorithm

- ***Class work:*** ***Find a shortest path between a and z in each of the following weighted graph***



Represent Expression by Rooted Tree

- We can represent complicated expression (propositions, sets, arithmetic) using ordered rooted trees.

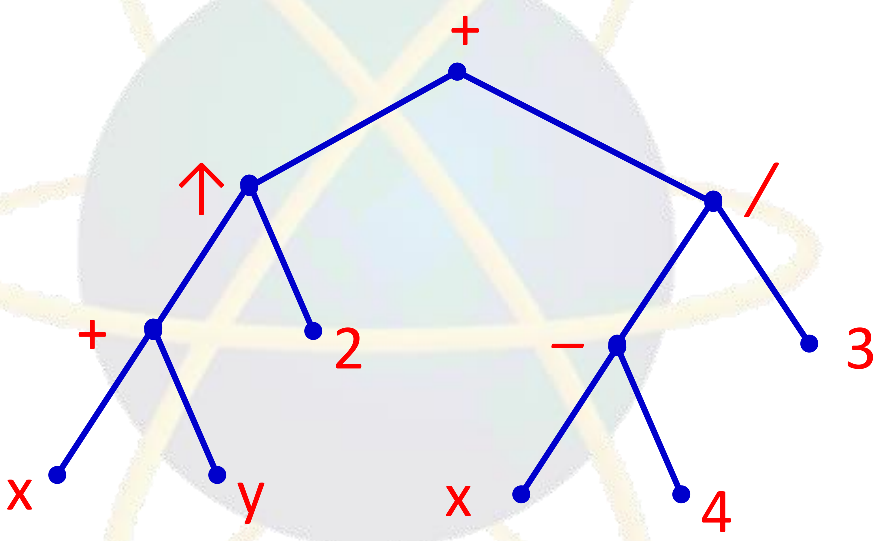
- Binary tree for an arithmetic expression
  – internal nodes: operators
  – leaves: operands

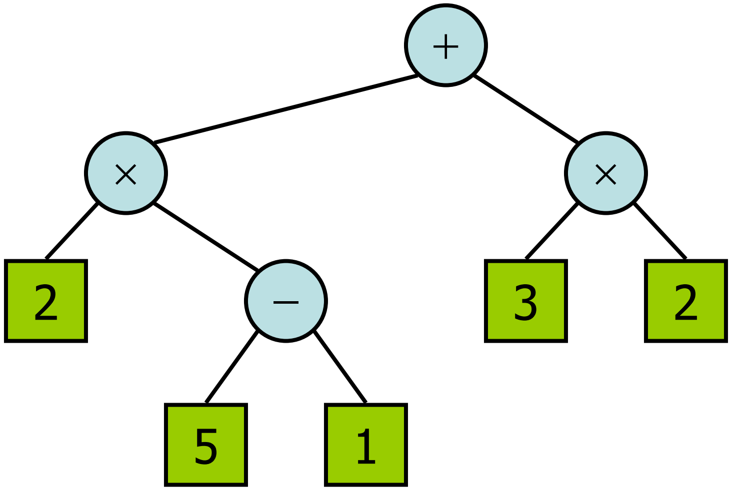
Example

A binary tree representing

i) $((x + y) \uparrow 2) + ((x - 4) / 3)$

ii) $((2 \times (5 - 1)) + (3 \times 2))$





Question and Answer Session

Q & A