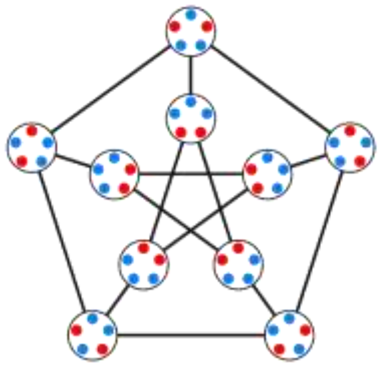
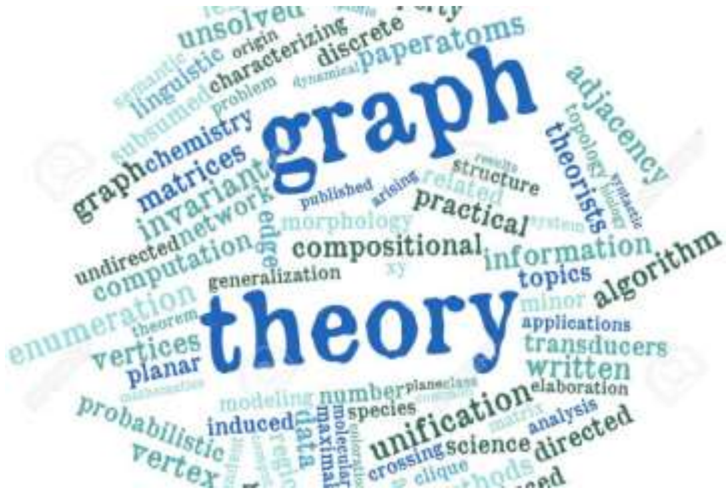


Graph Theory
Basics

Mr. Asim Raza

Email: asim.raza@ucp.edu.pk



Let’s talk more about Graphs

TREES

A fool sees not the same tree that a wise man sees.
William Blake

Summary - Recap

- **Graph**
- **Degree**
  - **Number of Odd-degree vertices are even**
- **Paths**
- **Cycles**
- **Eulerian Graph:** A graph containing Eulerian Path
- **Hamiltonian Graph:** A graph containing Hailtonian Cycle

Trees (Book 1: Page 766)

- A connected graph that contains no simple circuits is called a tree.

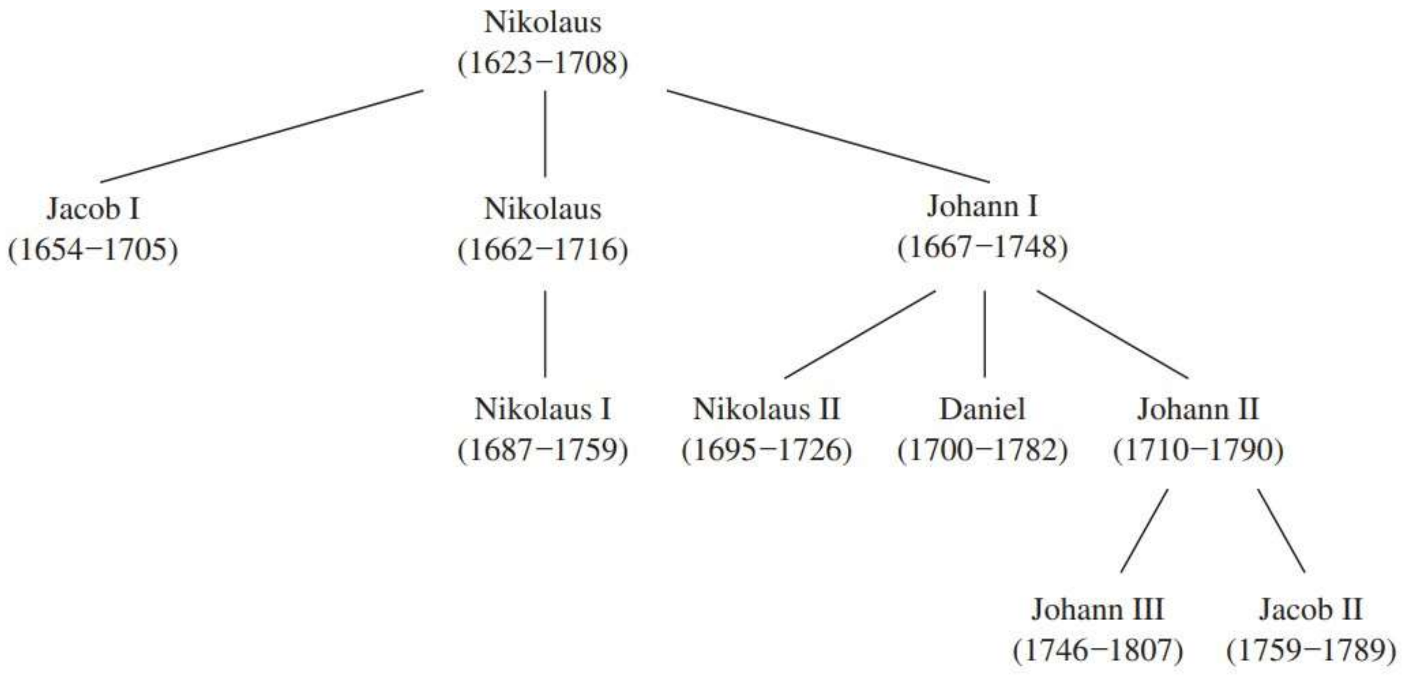


FIGURE 1 **The Bernoulli Family of Mathematicians.**

Applications of Trees (Book 1, Page 766)

- Trees are used to construct efficient algorithms for locating items in a list.
- They can be used in algorithms, such as Huffman coding, that construct efficient codes saving costs in data transmission and storage.
- Trees can be used to study games such as checkers and chess and can help determine winning strategies for playing these games.
- Trees can be used to model procedures carried out using a sequence of decisions. Constructing these models can help determine the computational complexity of algorithms based on a sequence of decisions, such as sorting algorithms.

Trees (Book 1: Page 767)

A *tree* is a connected undirected graph with no simple circuits.

Because a tree cannot have a simple circuit, a tree cannot contain multiple edges or loops. Therefore any tree must be a simple graph.

Example 1: Which of the graphs shown in Figure 2 are trees?

Solution: G1 and G2 are trees, because both are connected graphs with no simple circuits. G3 is not a tree because e, b, a, d, e is a simple circuit in this graph. Finally, G4 is not a tree because it is not connected.

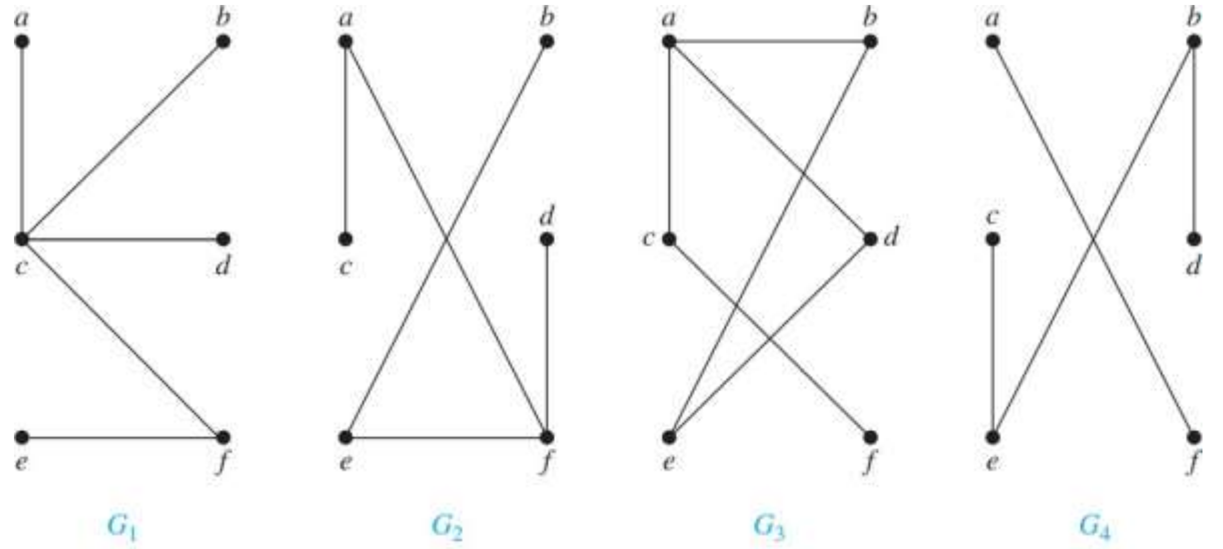


FIGURE 2 **Examples of Trees and Graphs That Are Not Trees.**

Forest (Book 1, Page 767)

Graphs containing no simple circuits that are not necessarily connected are called **forest**.
They have the property that each of their connected components is a tree as shown in Figure 3.

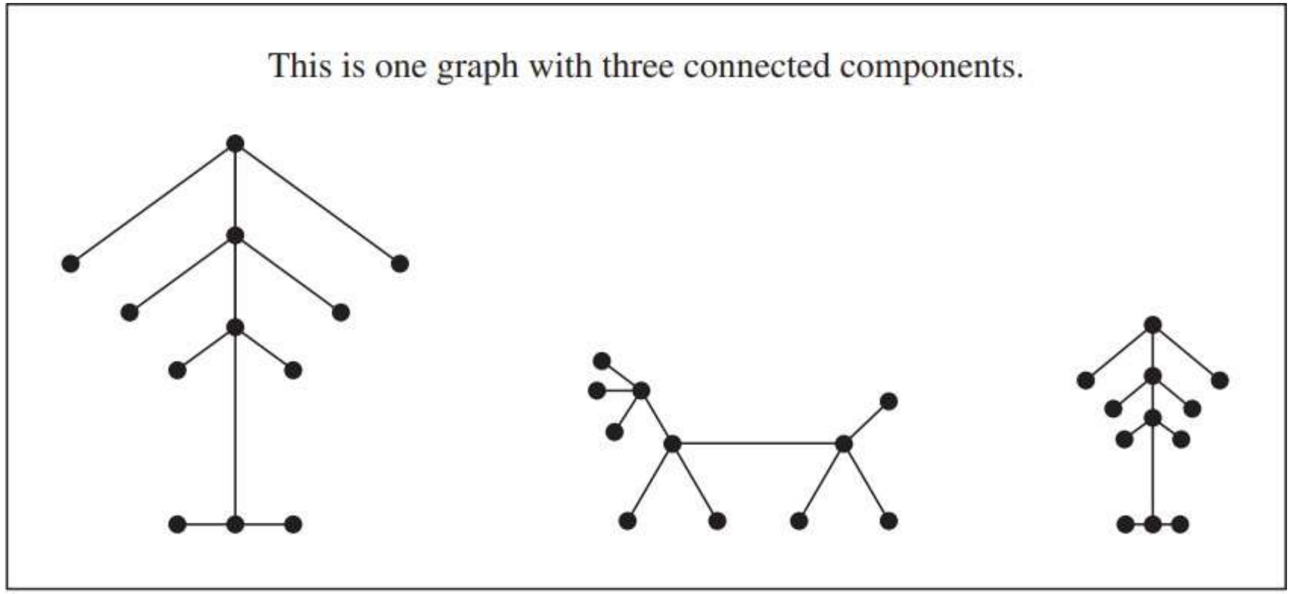


FIGURE 3 **Example of a Forest.**

Tree (Book 1, Page 767)

THEOREM 1 An undirected graph is a tree if and only if there is a unique simple path between any two of its vertices.

Rooted Trees (Book 1, Page 768)

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.

In many applications of trees, a particular vertex of a tree is designated as the **root**. Once we specify a root, we can assign a direction to each edge as follows. Because there is a unique path from the root to each vertex of the graph (by Theorem 1), we direct each edge away from the root. Thus, a tree together with its root produces a **directed graph** called a **rooted tree**.

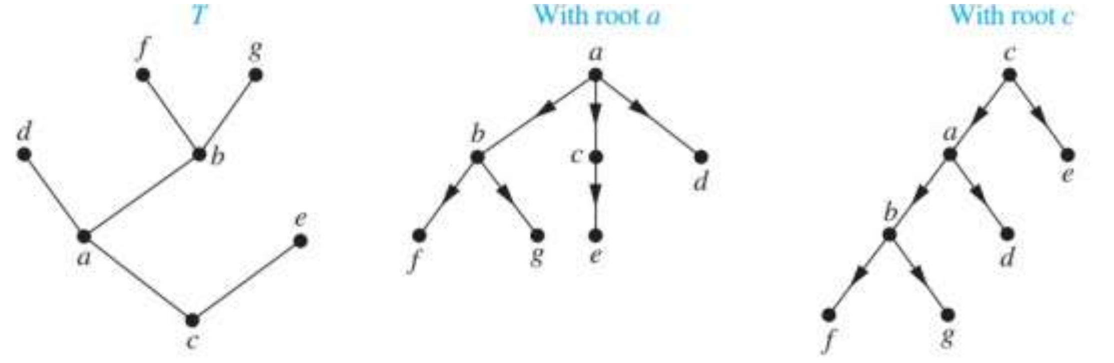


FIGURE 4 A Tree and Rooted Trees Formed by Designating Two Different Roots.

Terminology of Trees (Book 1, Page 768)

- The terminology for trees has botanical and genealogical.
- Let T be a tree with root v_0. Suppose that x, y, and z are vertices in T and that $(v_0, v_1, \dots, v_n)$ is a simple path in T. Then

1. v_{n-1} is the **parent** of v_n.
2. $v_0, \dots, v_{n-1}$ are **ancestors** of v_n.
3. v_n is a **child** of v_{n-1}.
4. If x is an **ancestor** of y, y is a **descendent** of x.
5. If x and y are children of z, x and y are **siblings**.
6. If x has no children, x is a terminal **vertex** (or a **leaf**).
7. If x is not a terminal vertex and has children, x is an **internal** (or **branch**) **vertex**.

Terminology of Trees

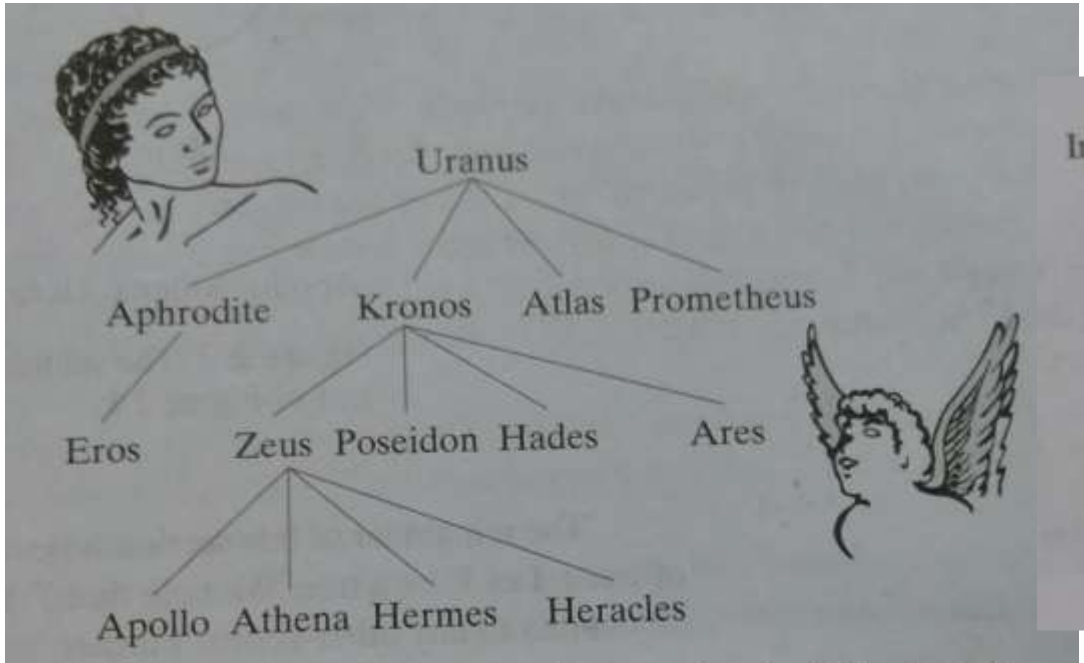


Figure 2.1 A portion of the family tree of ancient Greek gods.

In the rooted tree of Figure 2.1,

(a) The parent of Eros is Aphrodite.

(b) The ancestors of Hermes are Zeus, Kronos, and Uranus.

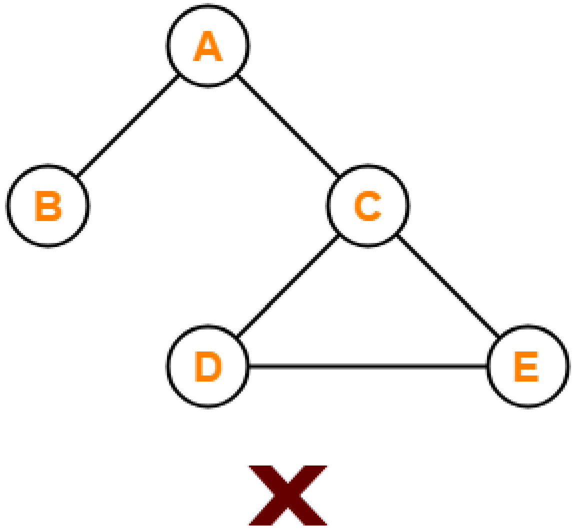
(c) The children of Zeus are Apollo, Athena, Hermes, and Heracles.

(d) The descendants of Kronos are Zeus, Poseidon, Hades, Ares, Apollo, Athena, Hermes, and Heracles.

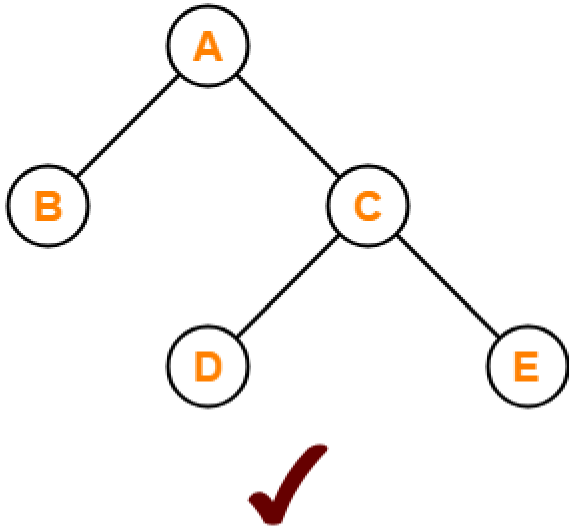
(e) Aphrodite and Prometheus are siblings.

(f) The terminal vertices are Eros, Apollo, Athena, Hermes, Heracles, Poseidon, Hades, Ares, Atlas, and Prometheus.

Trees



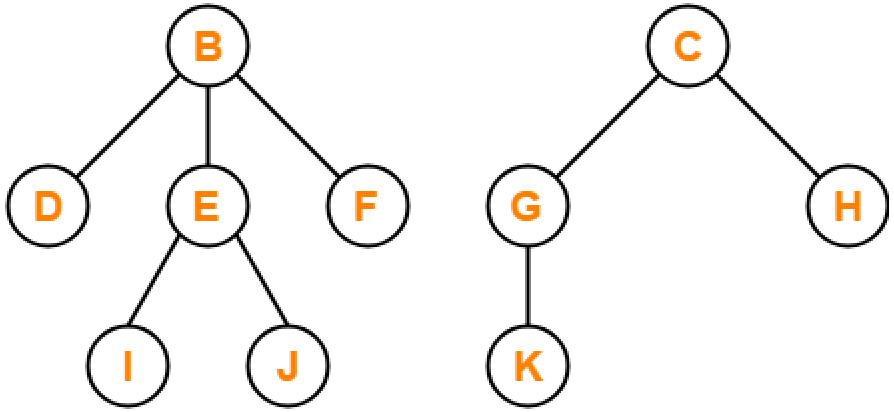
This graph is not a Tree



This graph is a Tree

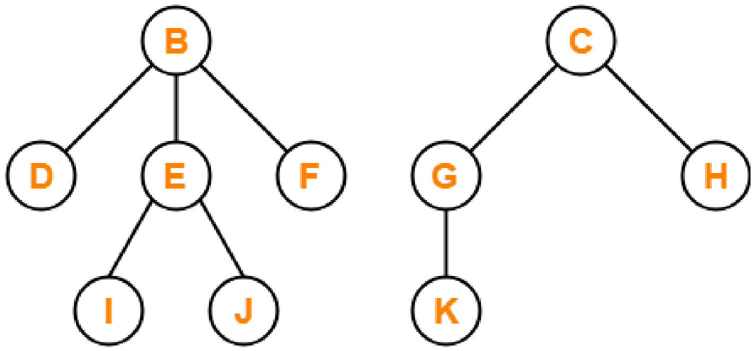
Trees - Definitions

- A connected graph containing no cycles as its sub-graphs.
- **Connectedness:** Graph cannot have ***too few*** edges
- **No Cycles:** Graph cannot have ***too many*** edges



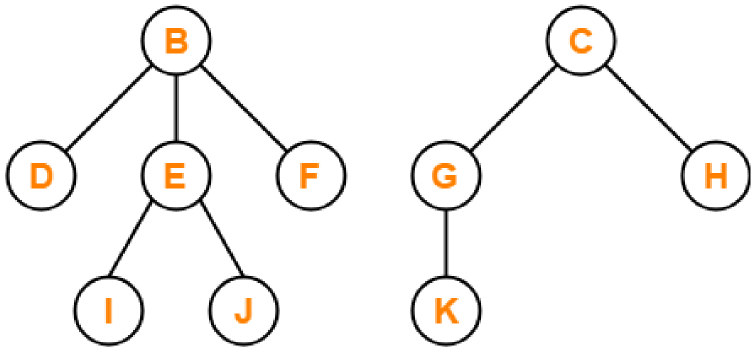
Trees - Definitions

- A connected graph containing no cycles as its sub-graphs.
- **Connectedness:** Graph cannot have ***too few*** edges
  - Removing any edge will make graph disconnected
- **No Cycles:** Graph cannot have ***too many*** edges
  - Adding any edge will create cycle in the graph



Trees - Definitions

- A connected graph containing no cycles as its sub-graphs.
- **Connectedness:** Graph cannot have ***too few*** edges
  - Removing any edge will make graph disconnected
- **No Cycles:** Graph cannot have ***too many*** e



Theorem 8.1.1 *(a) A graph G is a tree if and only if it is connected, but deleting any of its edges results in a disconnected graph.*

(b) A graph G is a tree if and only if it contains no cycles, but adding any new edge creates a cycle.

Trees - Definitions

Consider a connected graph G on n nodes, and an edge e of G. If we delete e, the remaining graph may or may not remain connected. If it is disconnected, then we call e a *cut-edge.* Part (a) of Theorem 8.1.1 implies that every edge of a tree is a cut-edge.

Cut-edge is also called **Bridge**

Theorem 8.1.1 *(a) A graph G is a tree if and only if it is connected, but deleting any of its edges results in a disconnected graph.*

(b) A graph G is a tree if and only if it contains no cycles, but adding any new edge creates a cycle.

Trees - Definitions

If we find an edge that is not a cut-edge, delete it. Go on deleting edges until a graph is obtained that is still connected, but deleting any edge from it leaves a disconnected graph. By part (a) of Theorem 8.1.1, this is a tree, with the same node set as G. A subgraph of G with the same node set that is a tree is called a *spanning tree* of G. The edge deletion process above can, of course, be carried out in many ways, so a connected graph can have many different spanning trees.

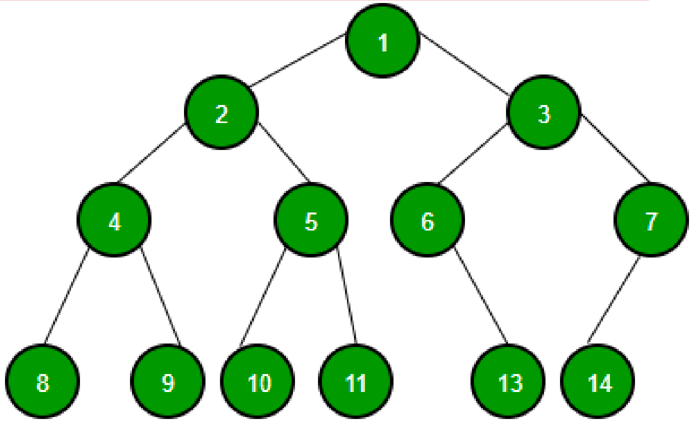
Theorem 8.1.1 *(a) A graph G is a tree if and only if it is connected, but deleting any of its edges results in a disconnected graph.*

(b) A graph G is a tree if and only if it contains no cycles, but adding any new edge creates a cycle.

Rooted Trees

Rooted trees. Often, we use trees that have a special node, which we call the *root*.

We can take any tree, select any of its nodes, and call it a root. A tree with a specified root is called a *rooted tree*.

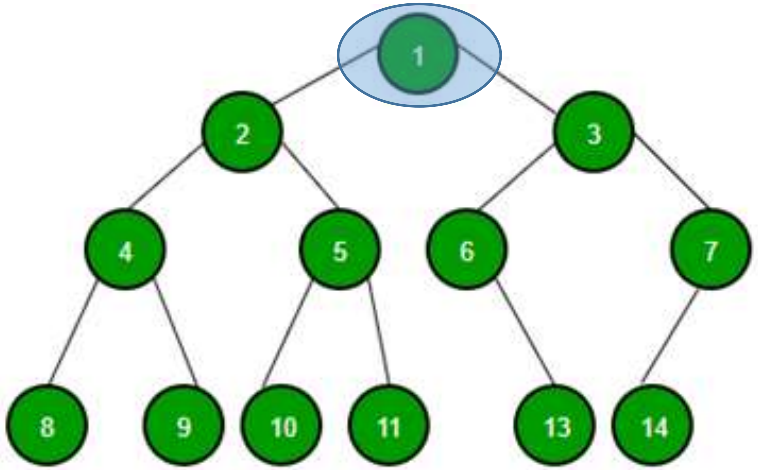


Rooted Trees

Rooted trees. Often, we use trees that have a special node, which we call the *root.*

We can take any tree, select any of its nodes, and call it a root. A tree with a specified root is called a *rooted tree.*

- We may define Parent nodes and Children nodes
- Nodes with degree-one are called Leaf nodes
- **A child node have only one parent node.** If have more than one parents ⇒ there is cycle, hence not a tree graph



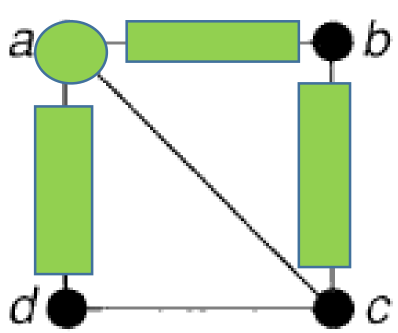
Trees – Summary of Properties

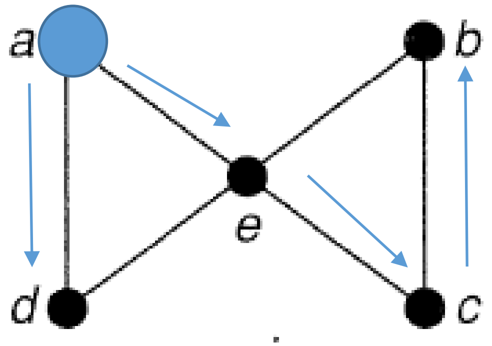
THEOREM. *Let T be a graph with n vertices. Then the following statements are equivalent:*

- (i) *T is a tree;*
- (ii) *T contains no cycles, and has $n-1$ edges;*
- (iii) *T is connected, and has $n-1$ edges;*
- (iv) *T is connected, and each edge is a bridge;*
- (v) *any two vertices of T are connected by exactly one path;*
- (vi) *T contains no cycles, but the addition of any new edge creates exactly one cycle.*

Trees - Exercise

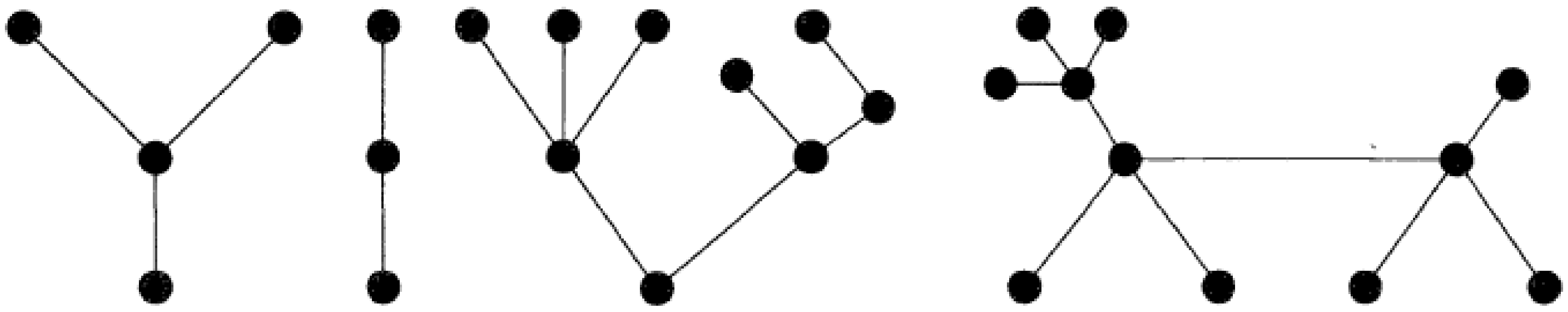
Draw all the spanning trees in the graph





How to Grow Trees – *(Section 8.2 of textbook)*

Theorem 8.2.1 *Every tree with at least two nodes has at least two nodes of degree* 1.



How to Grow Trees – *(Section 8.2 of textbook)*

Theorem 8.2.1 *Every tree with at least two nodes has at least two nodes of degree* 1.

- Join different trees with exactly one edge between them, to expand the tree

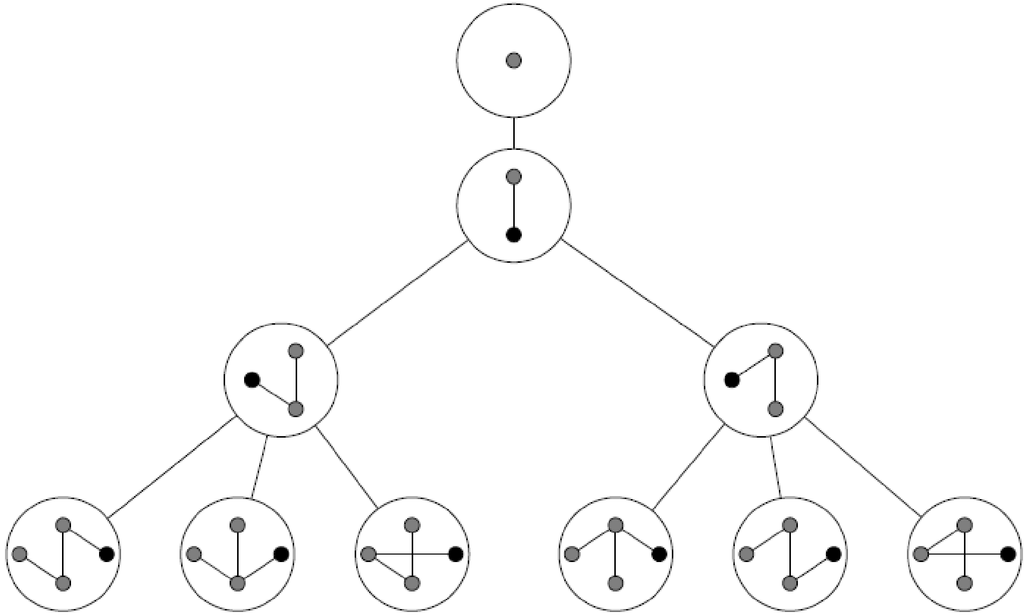


FIGURE 8.2. The descent tree of trees

Let’s Count Trees

- How many trees are there on n nodes?

Let’s Count Trees

- How many trees are there on n nodes?

- Counting **<u>Labelled</u>** or **<u>Unlabeled</u>** Trees?

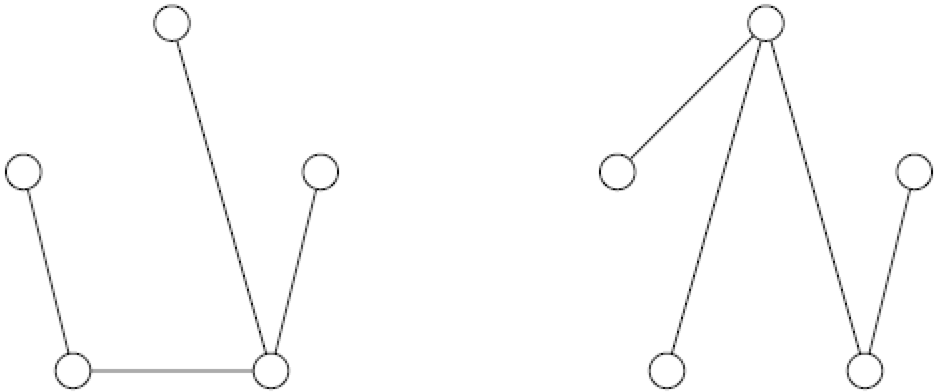


FIGURE 8.3. Are these trees the same?

Let’s Count Trees

- How many trees are there on n nodes?

Counting Labelled Trees !

Theorem 8.3.1 (Cayley's Theorem) *The number of labeled trees on n nodes is n^{n-2}.*

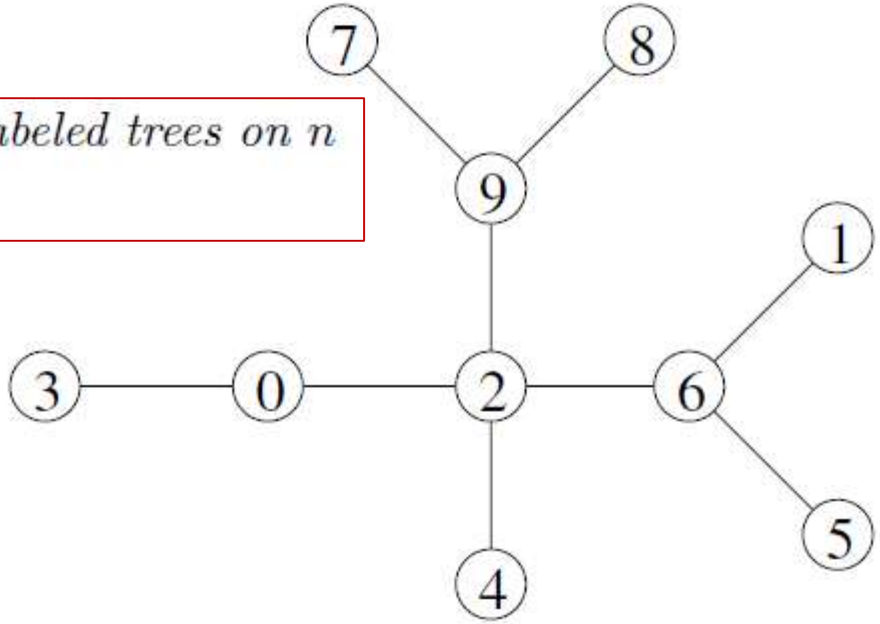


FIGURE 8.4. A labeled tree

Counting Labelled Trees

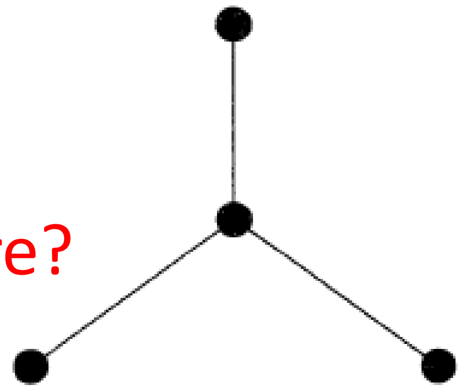
Counting Labelled Trees

- On 4 nodes, there will be $4^{4-2} = 4^2$ $= 16$ labelled trees.

Draw the rest

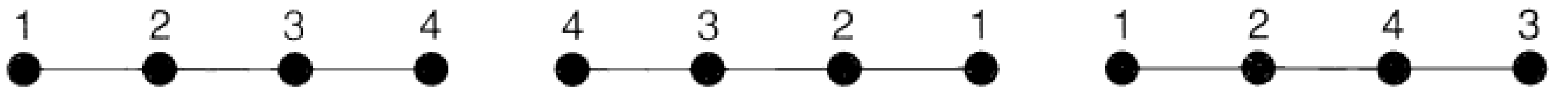
How many unlabeled trees will be there?

Draw all unlabeled trees.



Counting Labelled Trees

THEOREM (Cayley, 1889). *There are n^{n-2} distinct labelled trees on n vertices.*

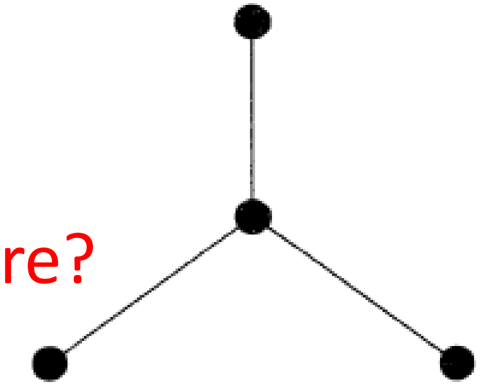


On 4 nodes, there will be $4^{4-2} = 4^2$
$= 16$ labelled trees.

Draw the rest

How many unlabeled trees will be there?

Draw all unlabeled trees.



Counting Labelled Trees

THEOREM (Cayley, 1889). *There are n^{n-2} distinct labelled trees on n vertices.*

COROLLARY. *The number of spanning trees of K_n is n^{n-2}.*

Draw all spanning trees for K_3 and K_4

Trees – Examples

Binary Trees

Depth First (DFS) and Breadth First (BFS) Trees

Red-Black Trees

AVL Trees

Subtree (Book 1, Page769)

- If a is a vertex in a tree, 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.

In the rooted tree T (with root a) shown in Figure 5, find the parent of c, the children of g, the siblings of h, all ancestors of e, all descendants of b, all internal vertices, and all leaves. What is the subtree rooted at g?

Solution: The parent of c is b. The children of g are h, i, and j. The siblings of h are i and j. The ancestors 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. The leaves are d, e, f, i, k, l, and m. The subtree rooted at g is shown in Figure 6.

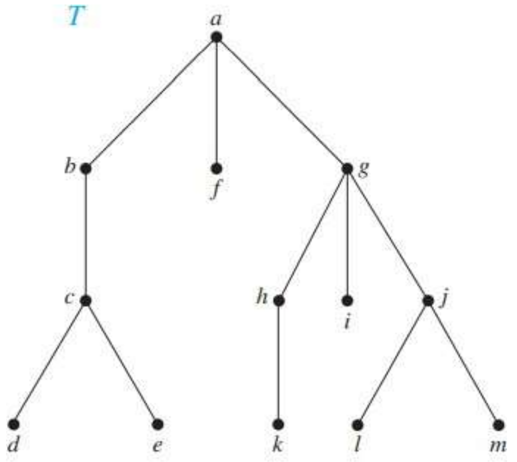


FIGURE 5 **A Rooted Tree T.**

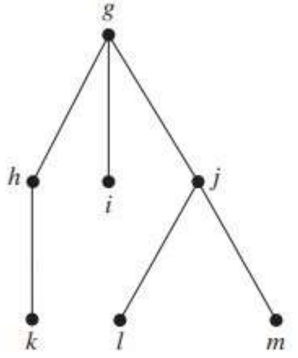


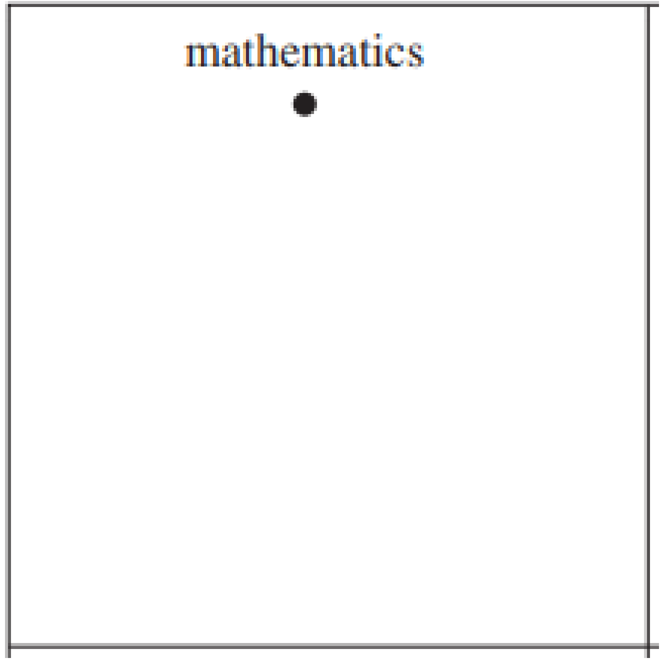
FIGURE 6 **The Subtree Rooted at g.**

Binary Search Trees (Book 1, Page 778)

- Searching for items in a list is one of the most important tasks that arises in computer science.
- To implement a searching algorithm that finds items efficiently when the items are totally ordered.
- A **binary search tree**, which is a binary tree in which each child of a vertex is designated as a right or left child, no vertex has more than one right child or left child, and each vertex is labeled with a key, which is one of the items.
- Furthermore, vertices are assigned keys so that the key of a vertex is both larger than the keys of all vertices in its left subtree and smaller than the keys of all vertices in its right subtree

Binary Search Trees (Book 1, Page 779)

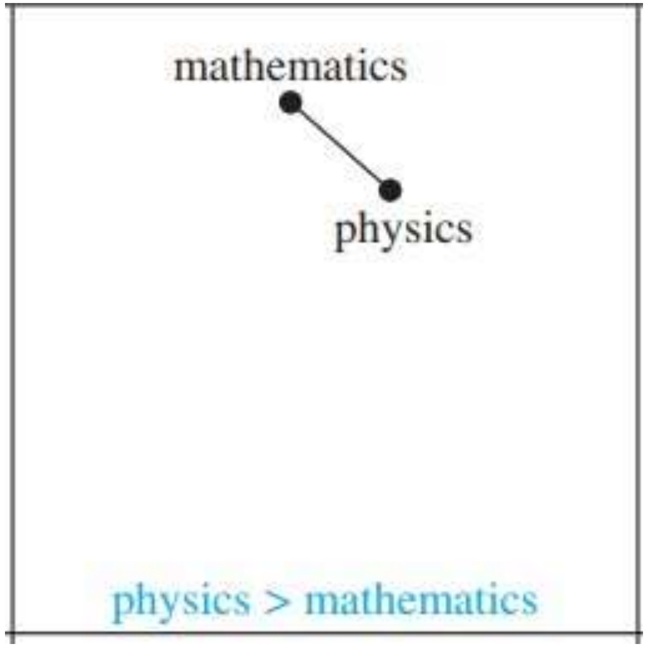
- Form a binary search tree for the words mathematics, physics, geography, zoology, meteorology, geology, psychology, and chemistry (using alphabetical order).



https://sites.google.com/view/adeel-arif/teaching/discrete-structures-f22

Binary Search Trees (Book 1, Page 779)

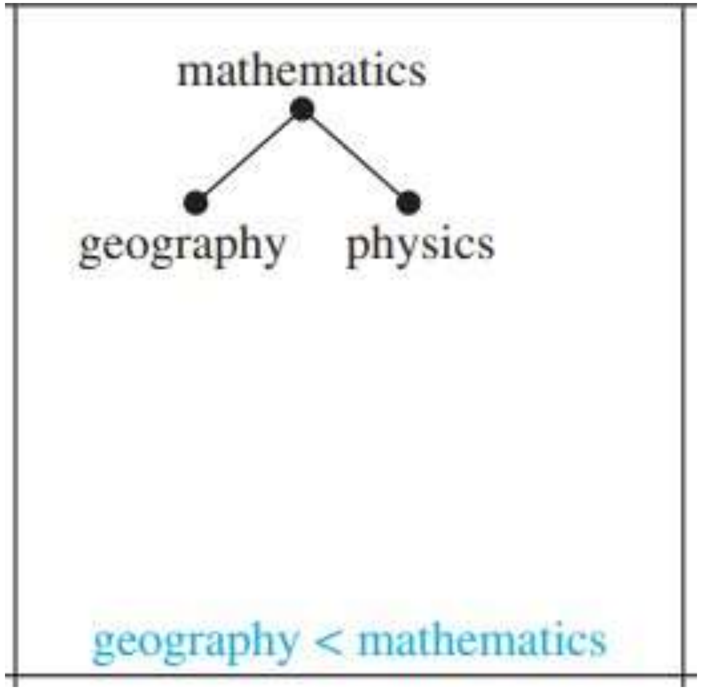
- Form a binary search tree for the words mathematics, physics, geography, zoology, meteorology, geology, psychology, and chemistry (using alphabetical order).



https://sites.google.com/view/adeel-arif/teaching/discrete-structures-f22

Binary Search Trees (Book 1, Page 779)

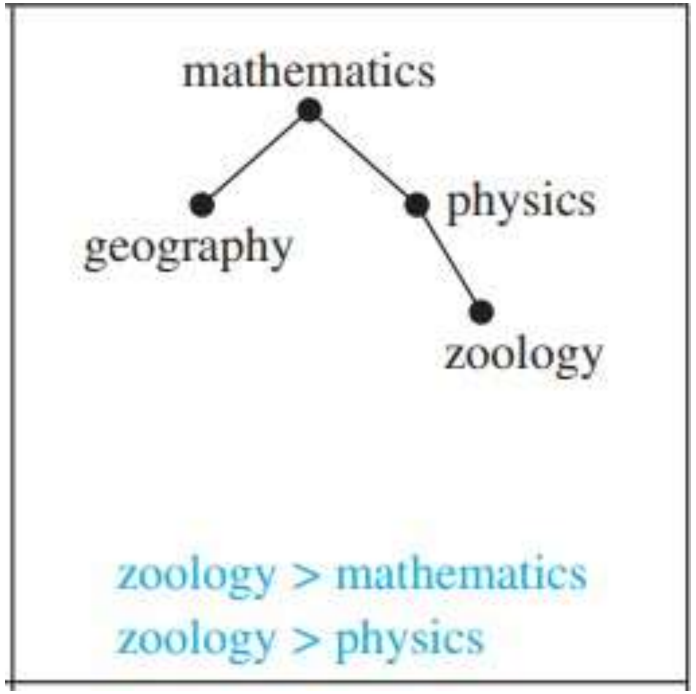
- Form a binary search tree for the words mathematics, physics, geography, zoology, meteorology, geology, psychology, and chemistry (using alphabetical order).



https://sites.google.com/view/adeel-arif/teaching/discrete-structures-f22

Binary Search Trees (Book 1, Page 779)

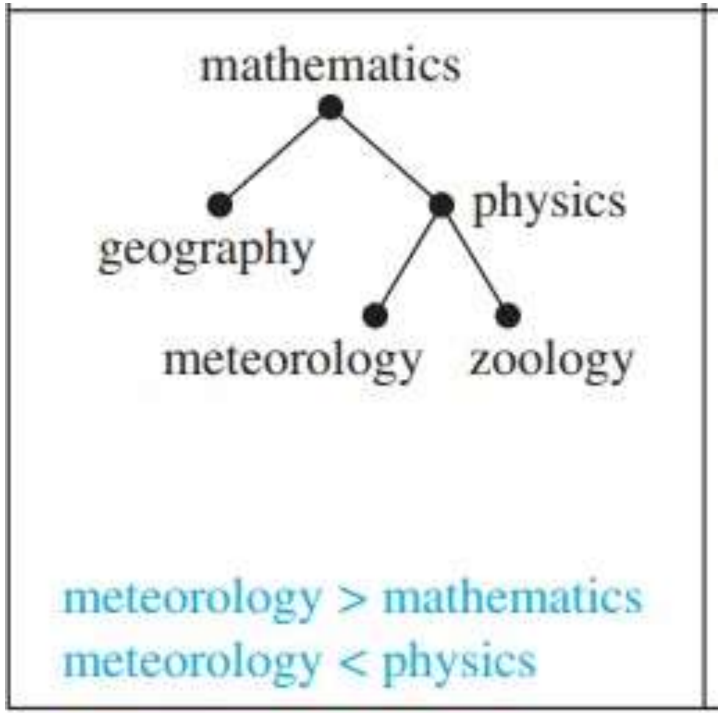
- Form a binary search tree for the words mathematics, physics, geography, zoology, meteorology, geology, psychology, and chemistry (using alphabetical order).



https://sites.google.com/view/adeel-arif/teaching/discrete-structures-f22

Binary Search Trees (Book 1, Page 779)

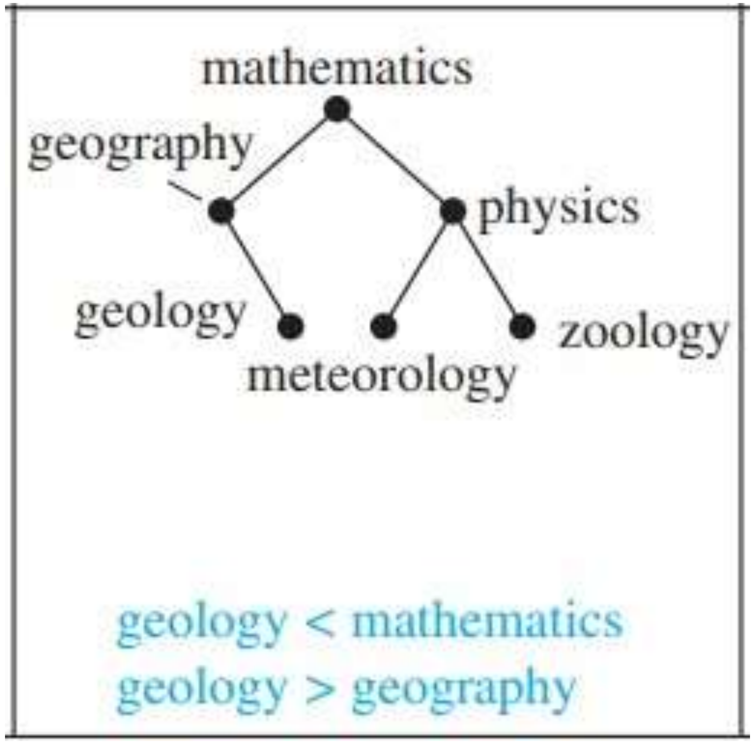
- Form a binary search tree for the words mathematics, physics, geography, zoology, meteorology, geology, psychology, and chemistry (using alphabetical order).



https://sites.google.com/view/adeel-arif/teaching/discrete-structures-f22

Binary Search Trees (Book 1, Page 779)

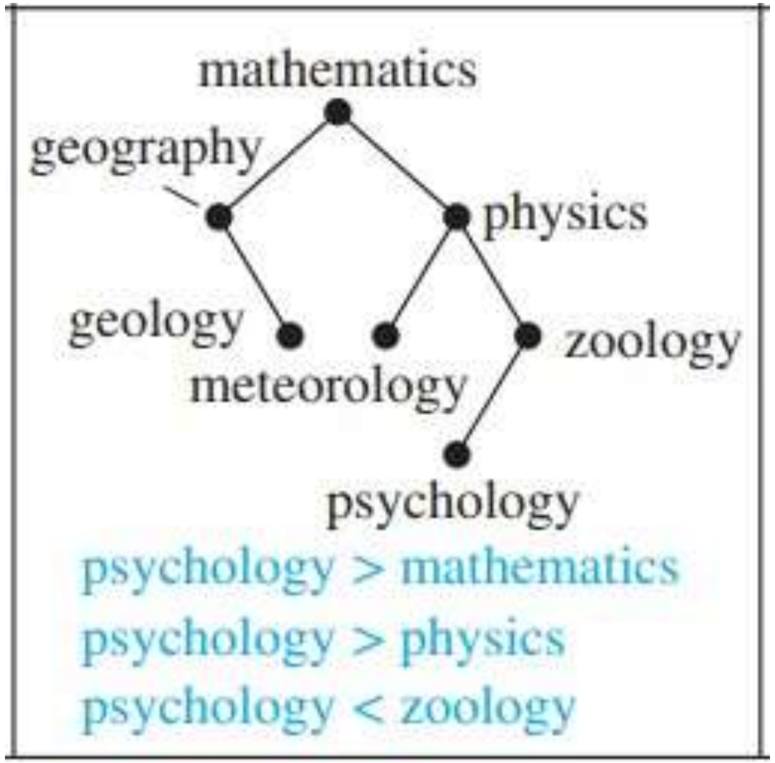
- Form a binary search tree for the words mathematics, physics, geography, zoology, meteorology, geology, psychology, and chemistry (using alphabetical order).



https://sites.google.com/view/adeel-arif/teaching/discrete-structures-f22

Binary Search Trees (Book 1, Page 779)

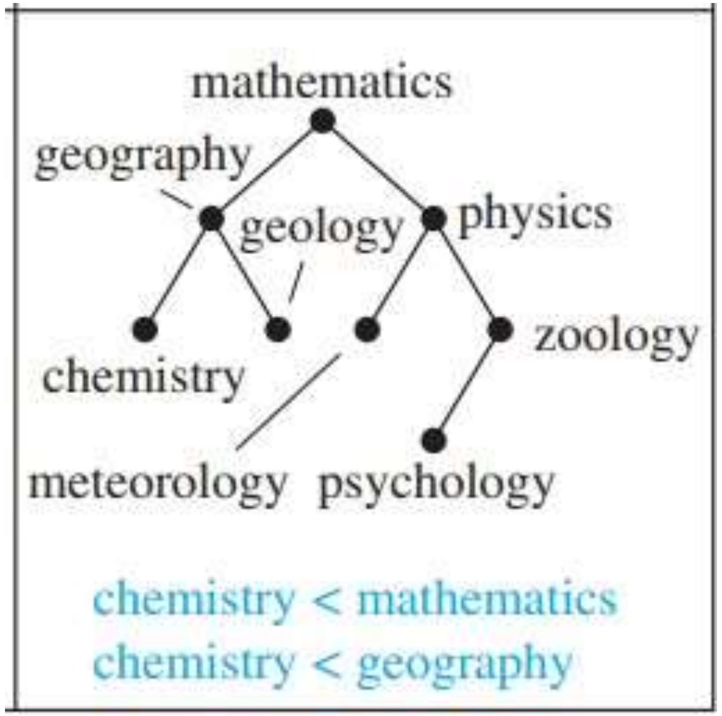
- Form a binary search tree for the words mathematics, physics, geography, zoology, meteorology, geology, psychology, and chemistry (using alphabetical order).



https://sites.google.com/view/adeel-arif/teaching/discrete-structures-f22

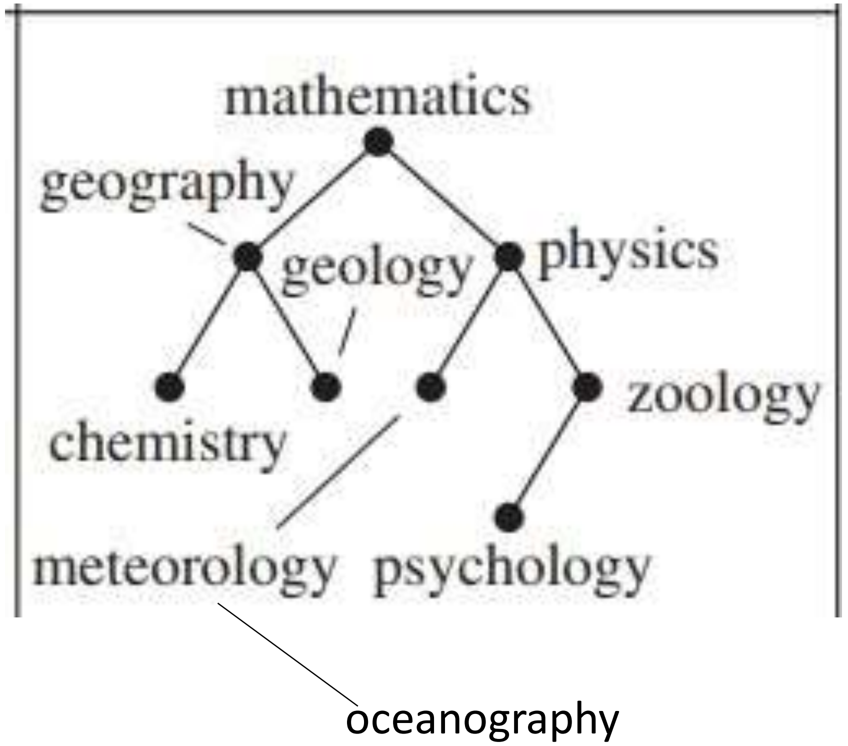
Binary Search Trees (Book 1, Page 779)

- Form a binary search tree for the words mathematics, physics, geography, zoology, meteorology, geology, psychology, and chemistry (using alphabetical order).



Binary Search Trees (Book 1, Page 780)

- Use Algorithm 1 to insert the word oceanography into the binary search tree in Previous Example.



Prefix Codes (Book 1, Page 783)

- To encode letters so that the bit string for a letter never occurs as the first part of the bit string for another letter. Codes with this property are called **prefix codes**.
- Example: the encoding of e as 0, a as 10, and t as 11 is a prefix code. A word can be recovered from the unique bit string that encodes its letters

  The string 10110 is the encoding of ate.

Prefix Codes (Book 1, Page 783)

- A prefix code can be represented using a binary tree, where the characters are the labels of the leaves in the tree.

- The tree in Figure 5 represents the encoding of e by 0, a by 10, t by 110, n by 1110, and s by 1111.

- Consider the word encoded by 11111011100 using the code in Figure 5.

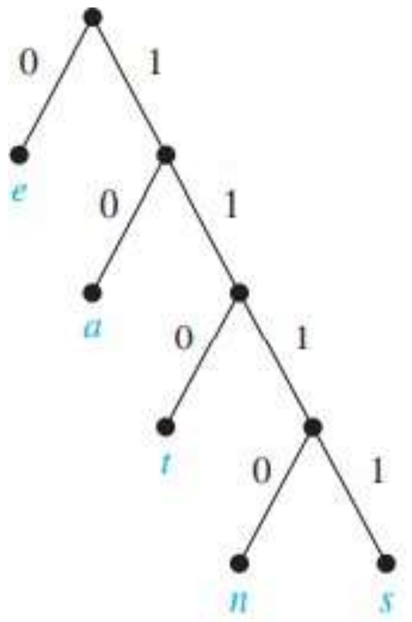


FIGURE 5 **A Binary Tree with a Prefix Code.**

Prefix Codes (Book 1, Page 783)

- A prefix code can be represented using a binary tree, where the characters are the labels of the leaves in the tree.
- The tree in Figure 5 represents the encoding of e by 0, a by 10, t by 110, n by 1110, and s by 1111.
- Consider the word encoded by 11111011100 using the code in Figure 5. **sane**

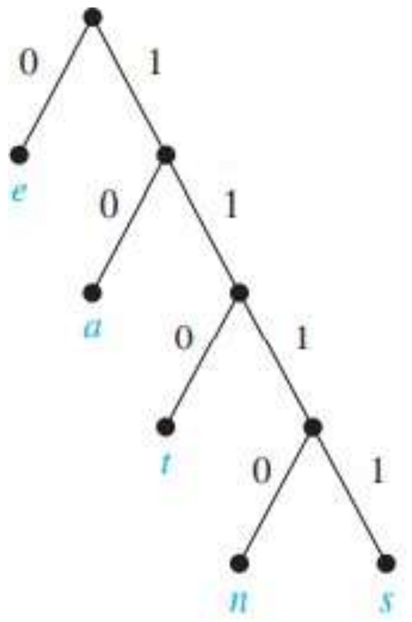


FIGURE 5 **A Binary Tree with a Prefix Code.**

Tree Traversal (Book 1, Page 794)

- Ordered rooted trees are often used to store information.
- We need procedures for visiting each vertex of an ordered rooted tree to access data.
- Procedures for systematically visiting every vertex of an ordered rooted tree are called **traversal algorithms**.
- There are three of the most commonly used such algorithms:
    1. Preorder traversal,
    2. Inorder traversal, and
    3. Postorder traversal.

Each of these algorithms can be defined recursively.

Preorder Traversal (Book 1, Page 794)

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

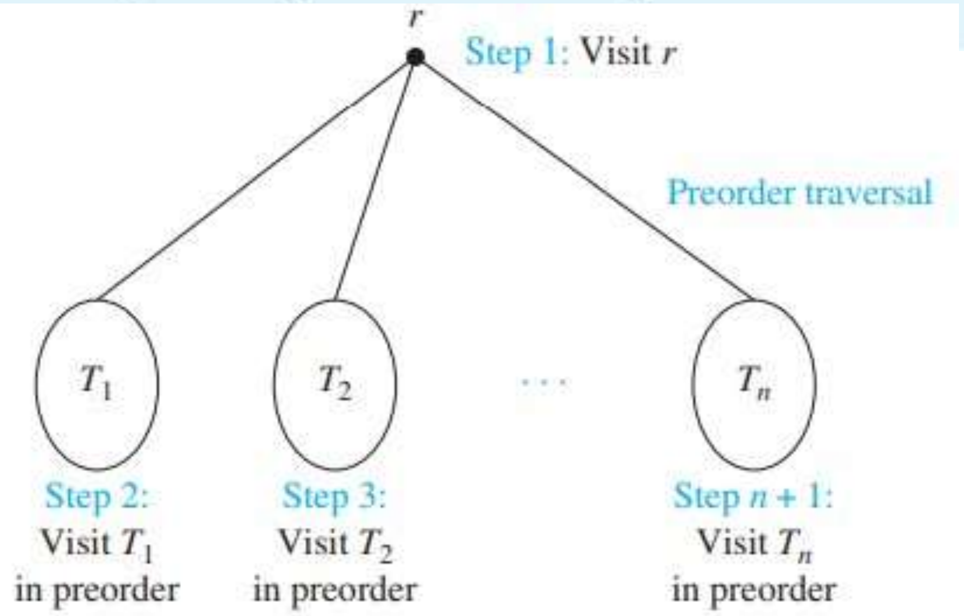


FIGURE 2 Preorder Traversal.

https://sites.google.com/view/adeel-arif/teaching/discrete-structures-f22

Preorder Traversal (Book 1, Page 794)

In which order does a preorder traversal visit the vertices in the ordered rooted tree T shown in Figure 3?

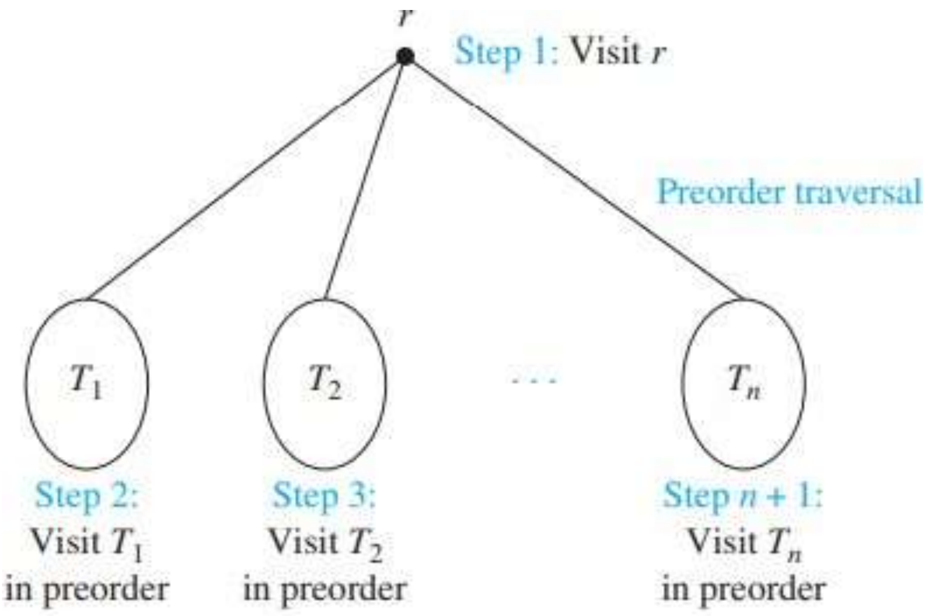


FIGURE 2 Preorder Traversal.

Preorder traversal: Visit root, visit subtrees left to right

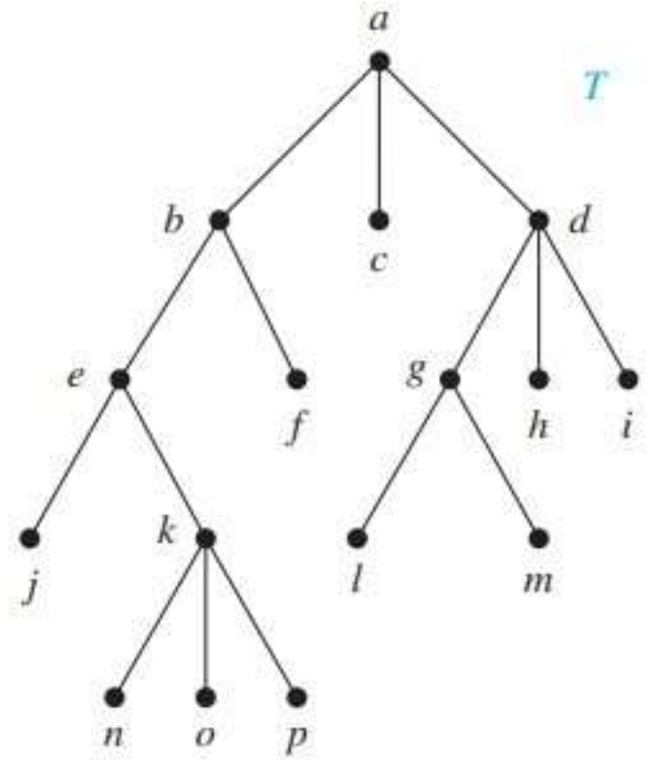
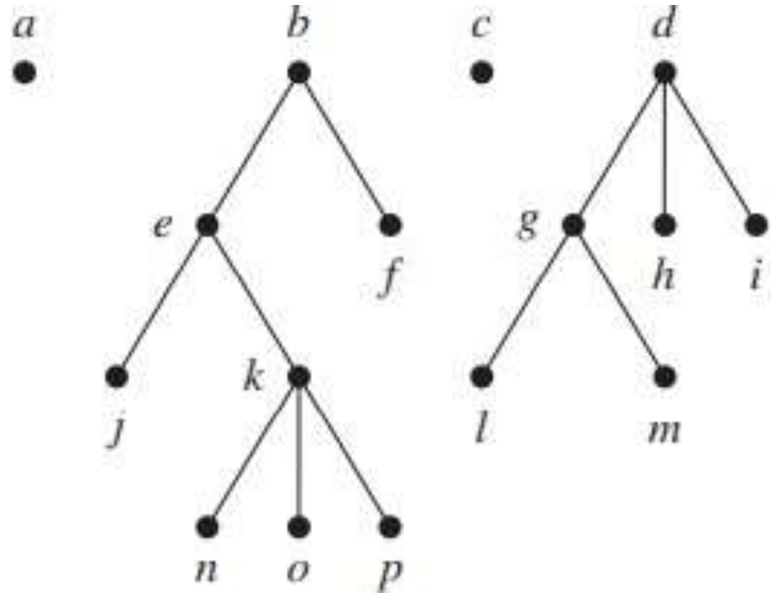


FIGURE 3 The Ordered Rooted Tree T.

Preorder Traversal (Book 1, Page 795)

In which order does a preorder traversal visit the vertices in the ordered rooted tree T shown in Figure 3?



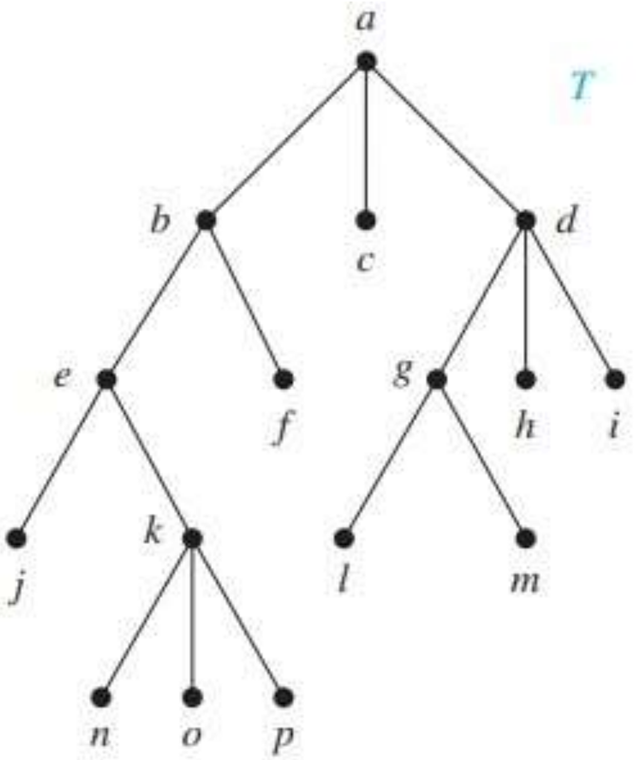
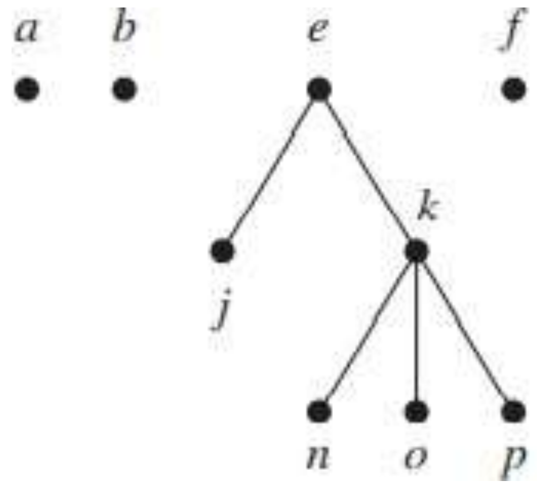


FIGURE 3 The Ordered Rooted Tree *T*.

Preorder Traversal (Book 1, Page 795)

In which order does a preorder traversal visit the vertices in the ordered rooted tree T shown in Figure 3?



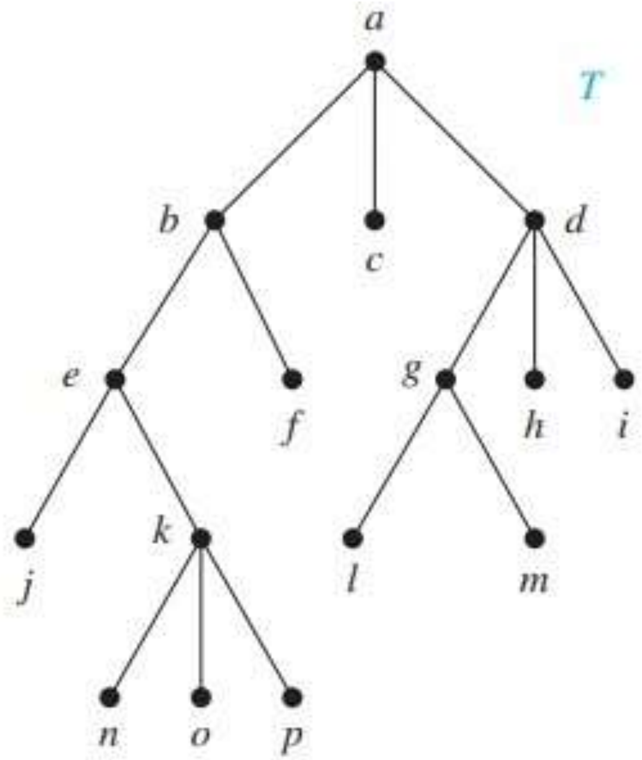
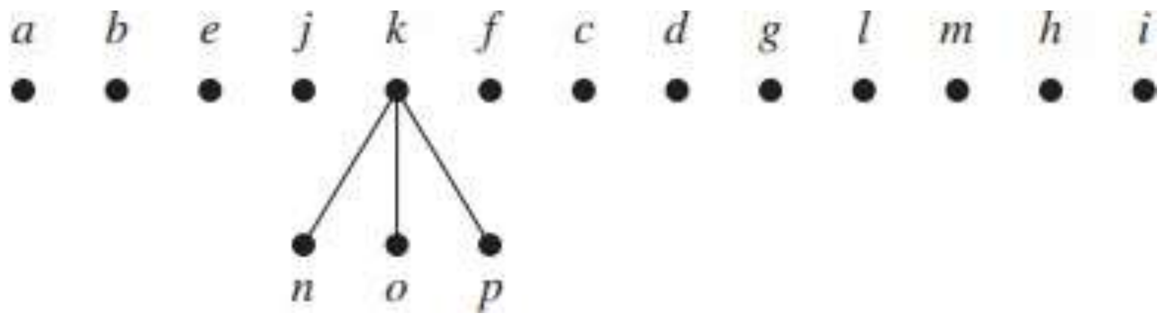


FIGURE 3 The Ordered Rooted Tree T.

https://sites.google.com/view/adeel-arif/teaching/discrete-structures-f22

Preorder Traversal (Book 1, Page 795)

In which order does a preorder traversal visit the vertices in the ordered rooted tree T shown in Figure 3?



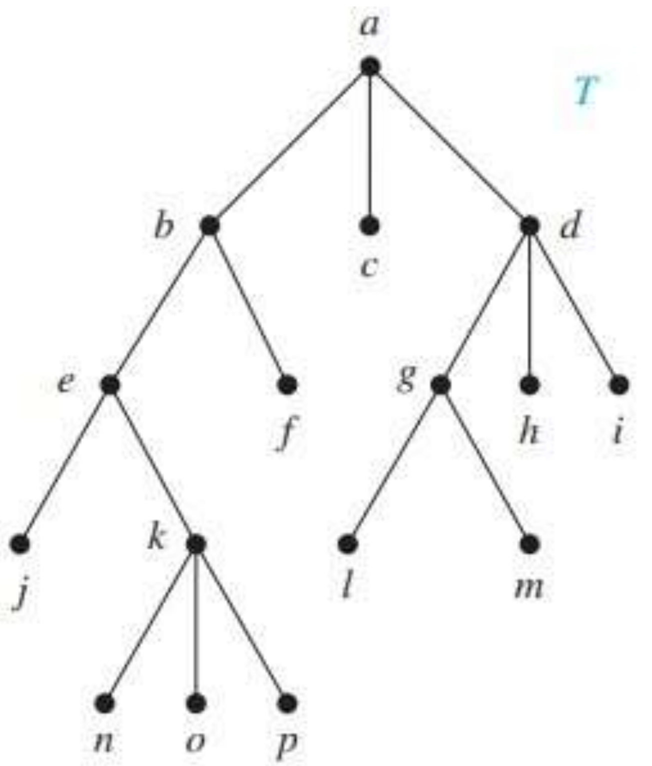
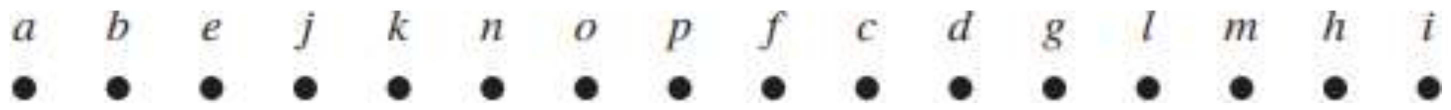


FIGURE 3 The Ordered Rooted Tree *T*.

https://sites.google.com/view/adeel-arif/teaching/discrete-structures-f22

Preorder Traversal (Book 1, Page 795)

In which order does a preorder traversal visit the vertices in the ordered rooted tree T shown in Figure 3?



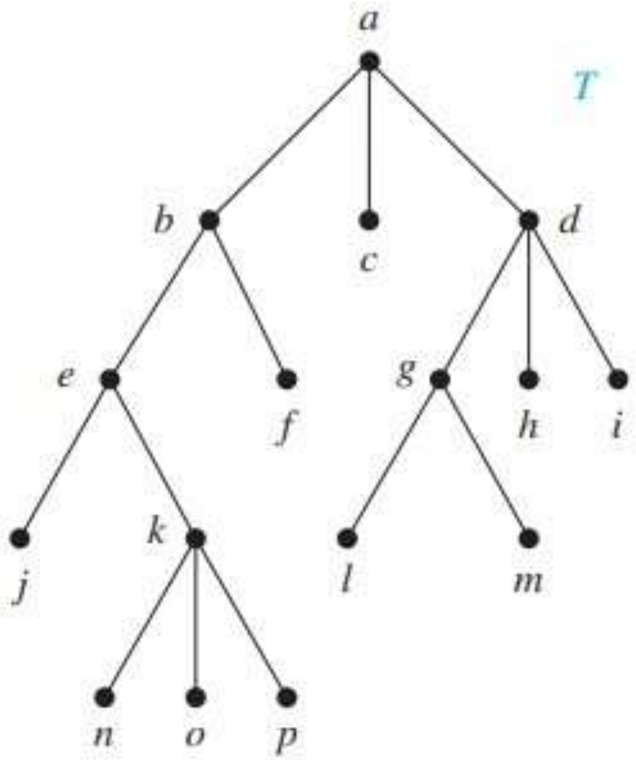


FIGURE 3 The Ordered Rooted Tree *T*.

https://sites.google.com/view/adeel-arif/teaching/discrete-structures-f22

Preorder Traversal (Book 1, Page 796)

Let T be an ordered rooted tree with root r. If T consists only of r, then r is the *inorder traversal* of T. Otherwise, suppose that $T_1, T_2, \ldots, T_n$ are the subtrees at r from left to right. The *inorder traversal* begins by traversing T_1 in inorder, then visiting r. It continues by traversing T_2 in inorder, then T_3 in inorder, $\ldots$, and finally T_n in inorder.

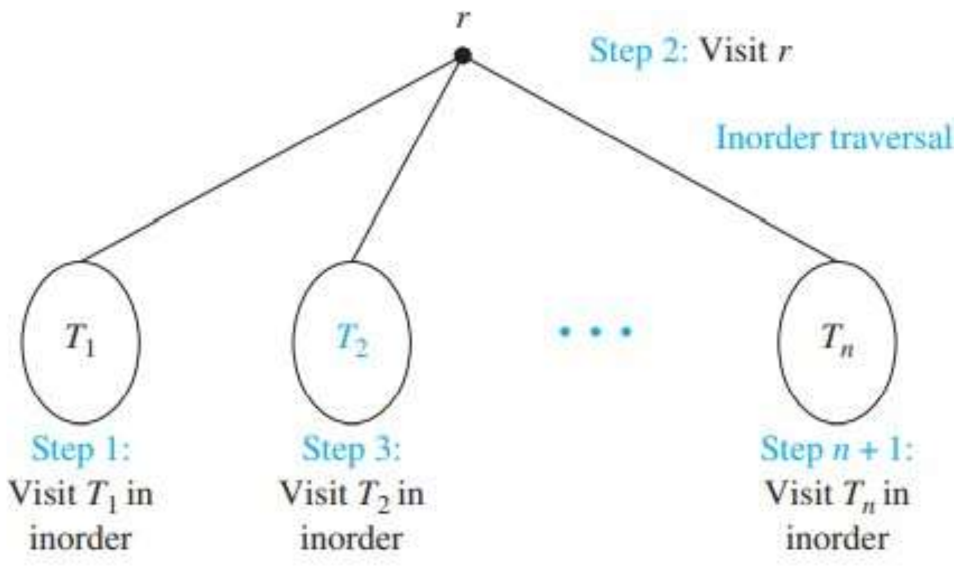


FIGURE 5 Inorder Traversal.

Preorder Traversal (Book 1, Page 796)

In which order does an inorder traversal visit the vertices of the ordered rooted tree T in Figure 3?

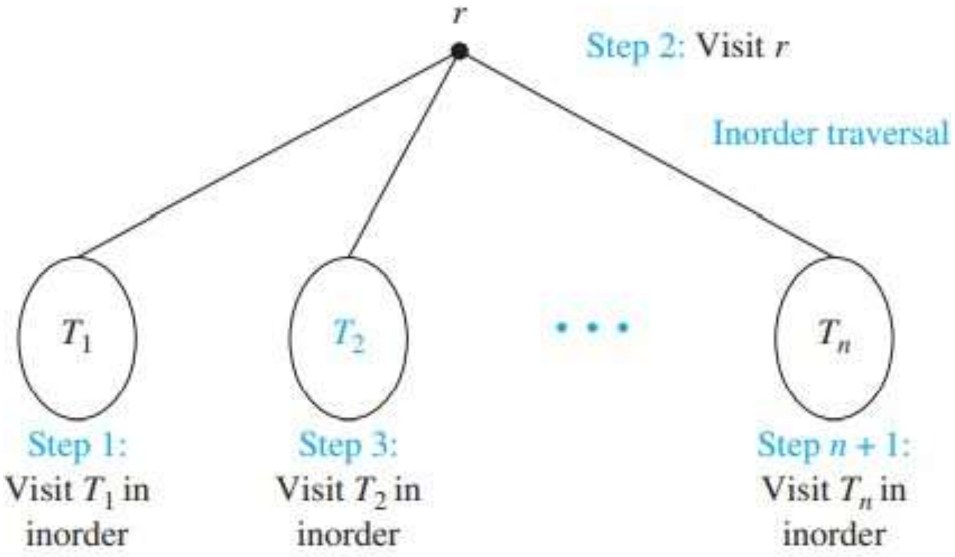
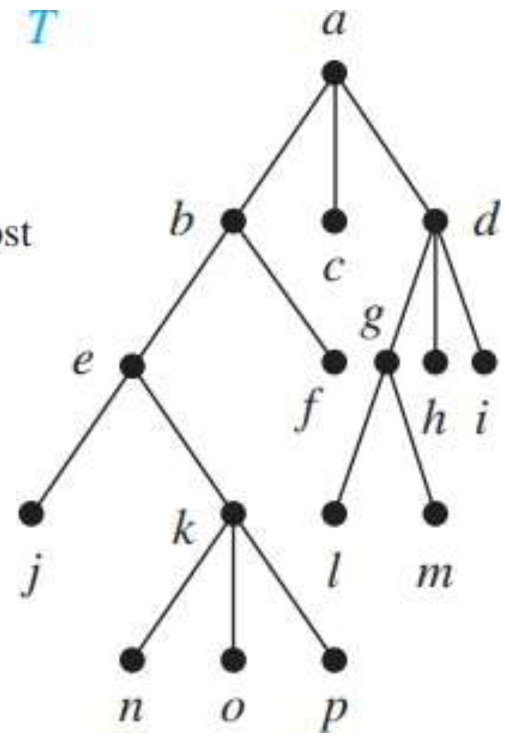


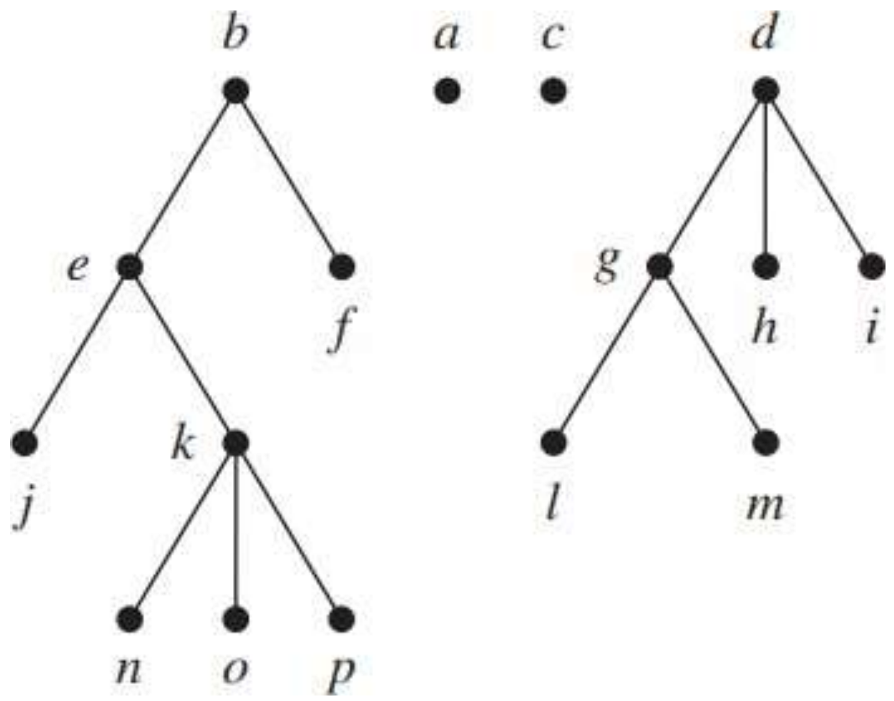
FIGURE 5 **Inorder Traversal.**

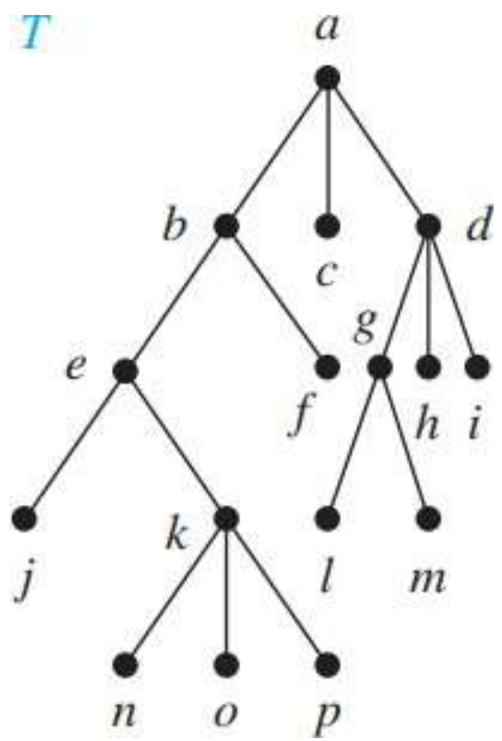
Inorder traversal: Visit leftmost subtree, visit root, visit other subtrees left to right



Preorder Traversal (Book 1, Page 796)

In which order does an inorder traversal visit the vertices of the ordered rooted tree T in Figure 3?

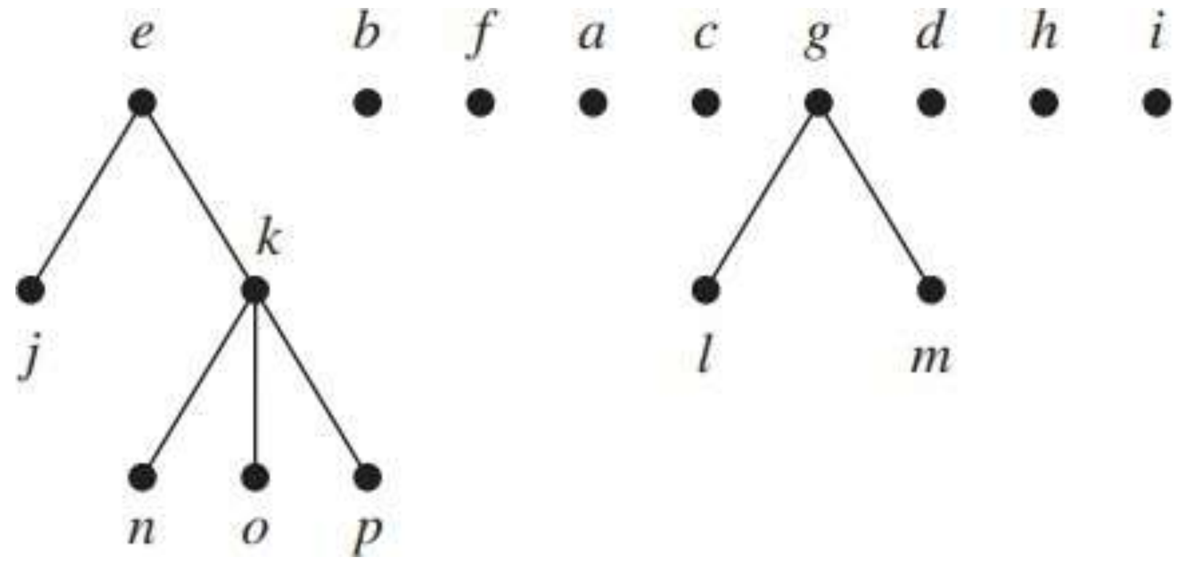


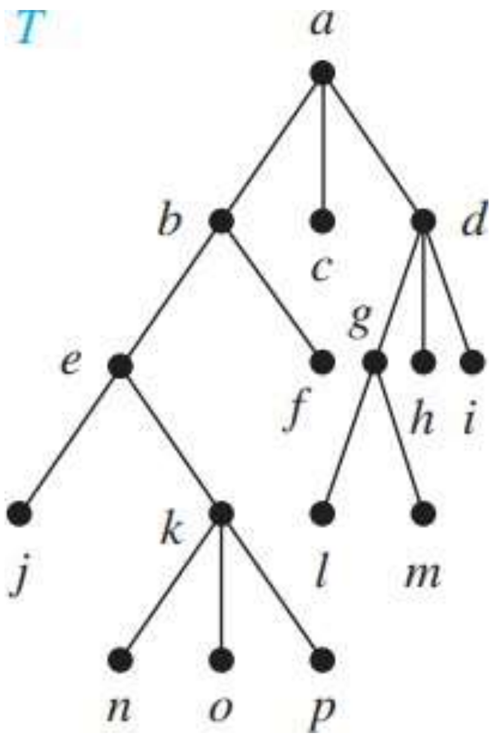


https://sites.google.com/view/adeel-arif/teaching/discrete-structures-f22

Preorder Traversal (Book 1, Page 796)

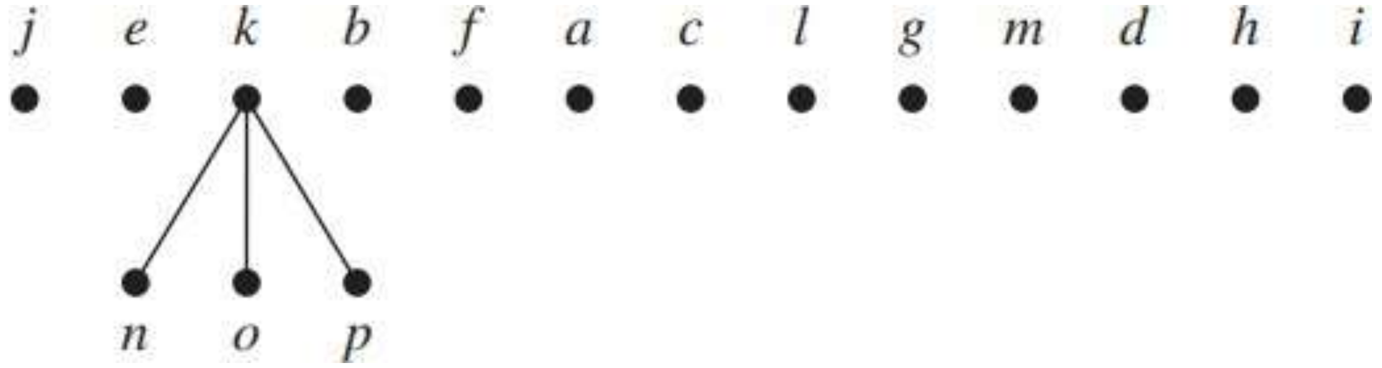
In which order does an inorder traversal visit the vertices of the ordered rooted tree T in Figure 3?

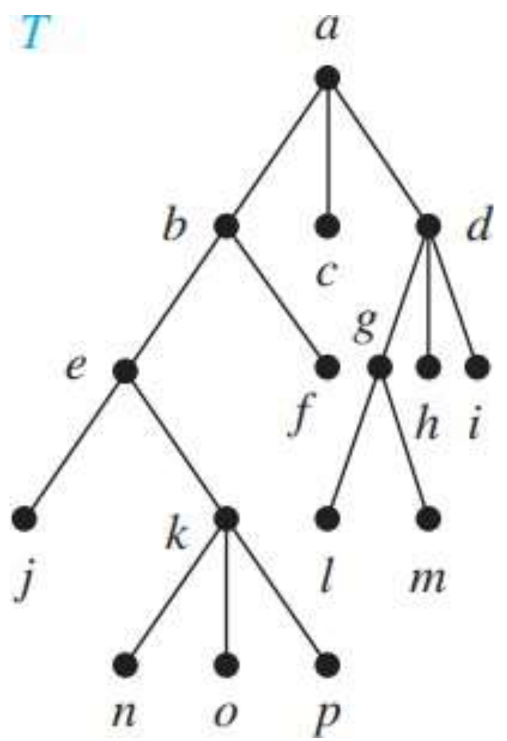




Preorder Traversal (Book 1, Page 796)

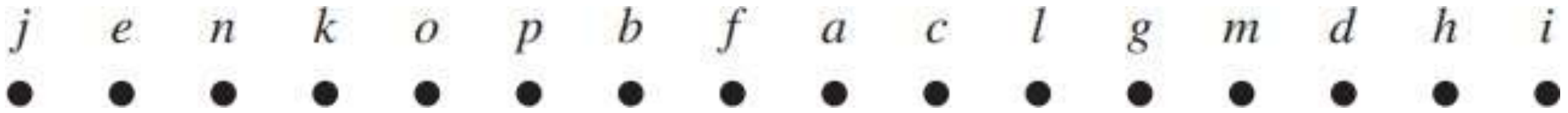
In which order does an inorder traversal visit the vertices of the ordered rooted tree T in Figure 3?

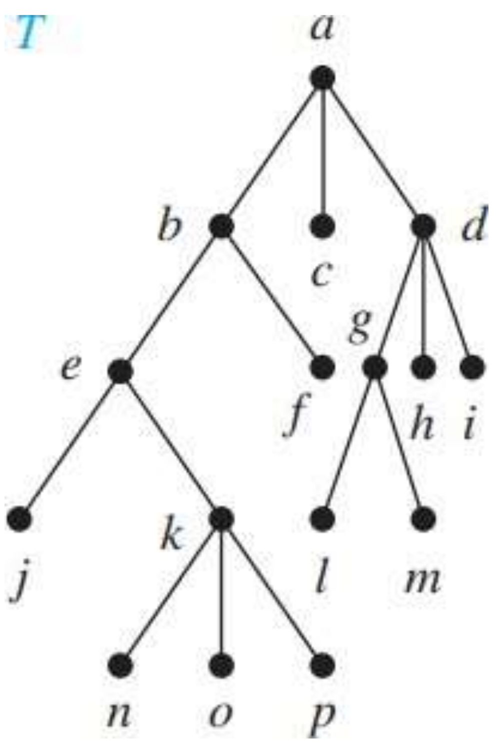




Preorder Traversal (Book 1, Page 797)

In which order does an inorder traversal visit the vertices of the ordered rooted tree T in Figure 3?





Postorder Traversal (Book 1, Page 797)

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

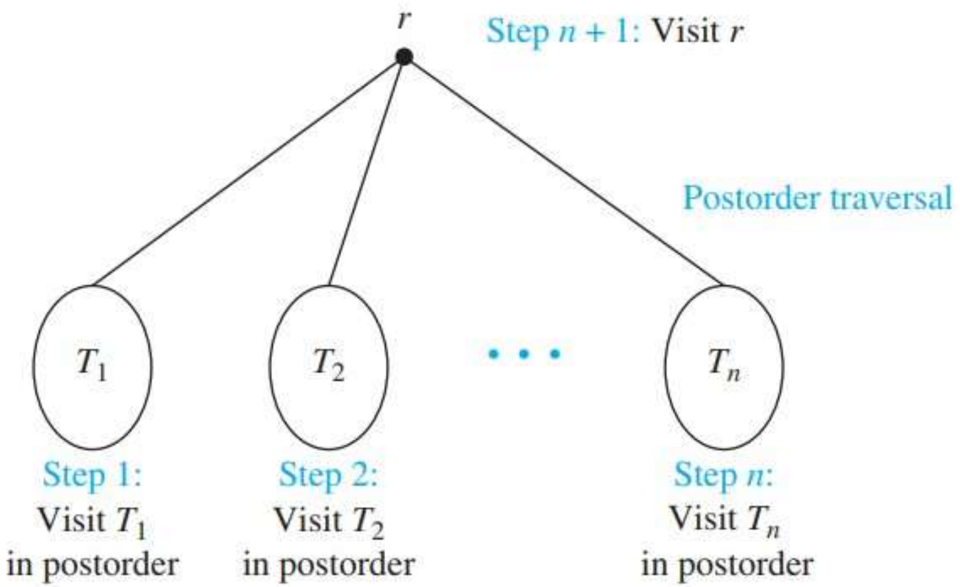


FIGURE 7 Postorder Traversal.

https://sites.google.com/view/adeel-arif/teaching/discrete-structures-f22

Postorder Traversal (Book 1, Page 798)

In which order does a postorder traversal visit the vertices of the ordered rooted tree T shown in Figure 3?

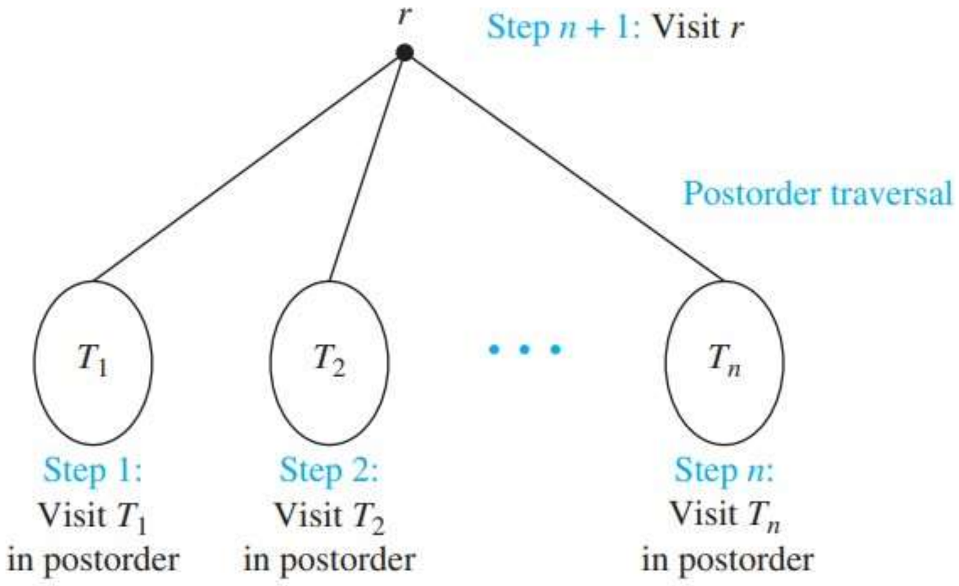
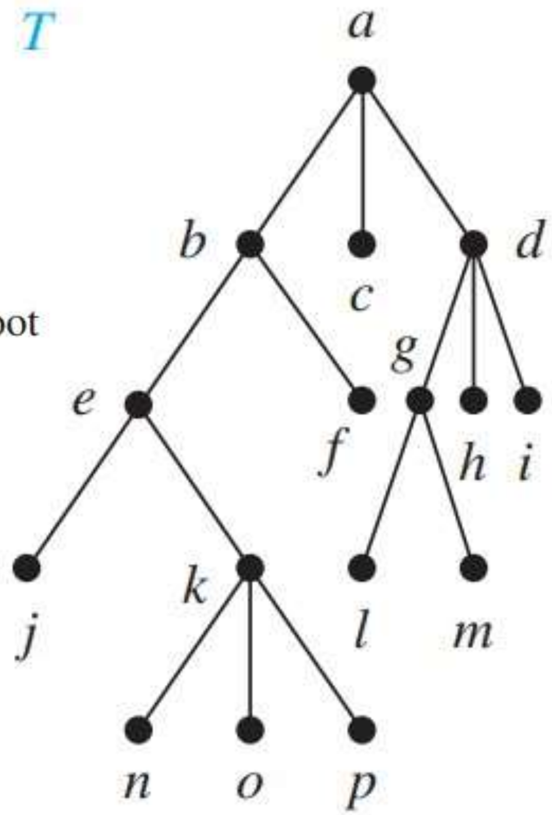


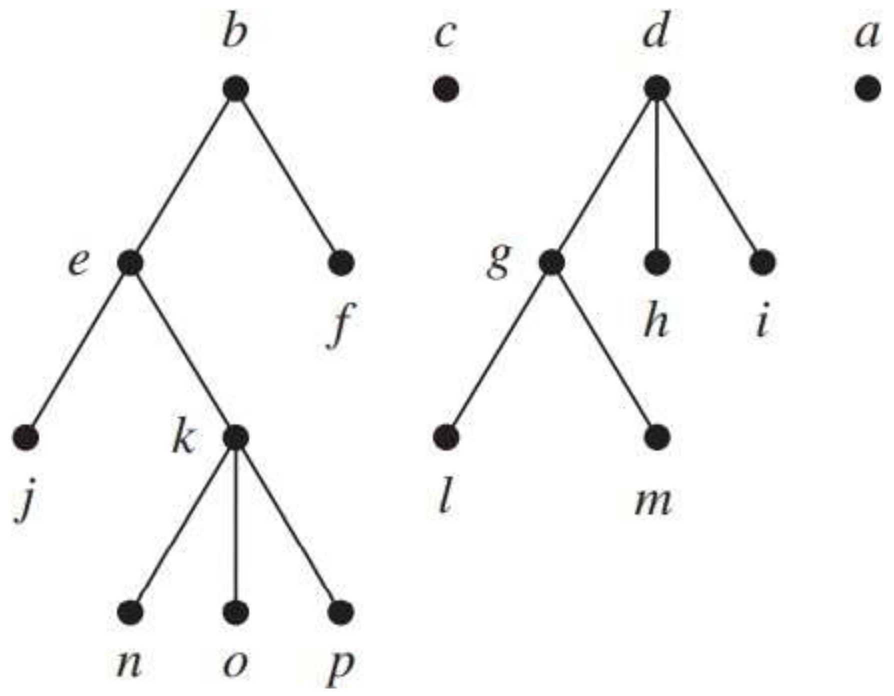
FIGURE 7 Postorder Traversal.

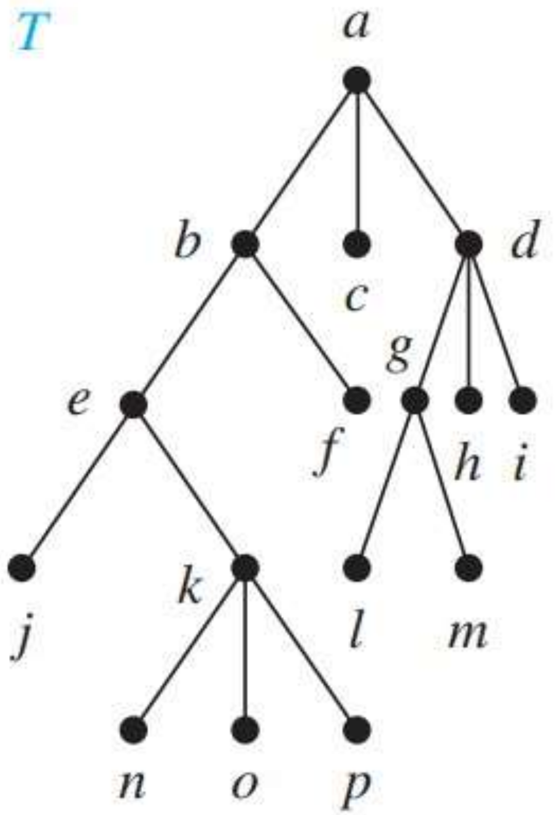
Postorder traversal: Visit subtrees left to right; visit root



Postorder Traversal (Book 1, Page 799)

In which order does a postorder traversal visit the vertices of the ordered rooted tree T shown in Figure 3?

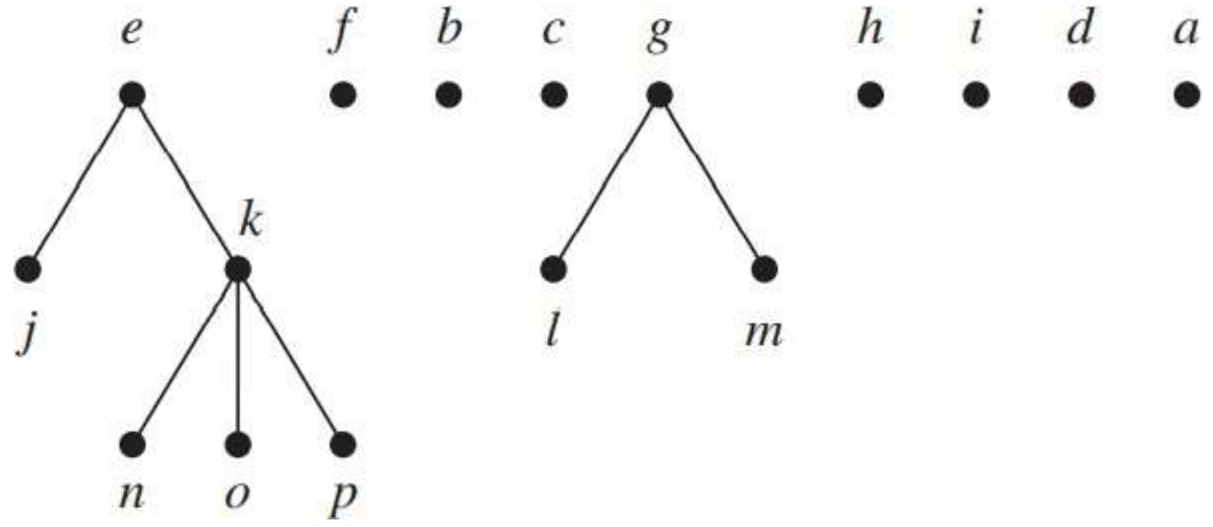


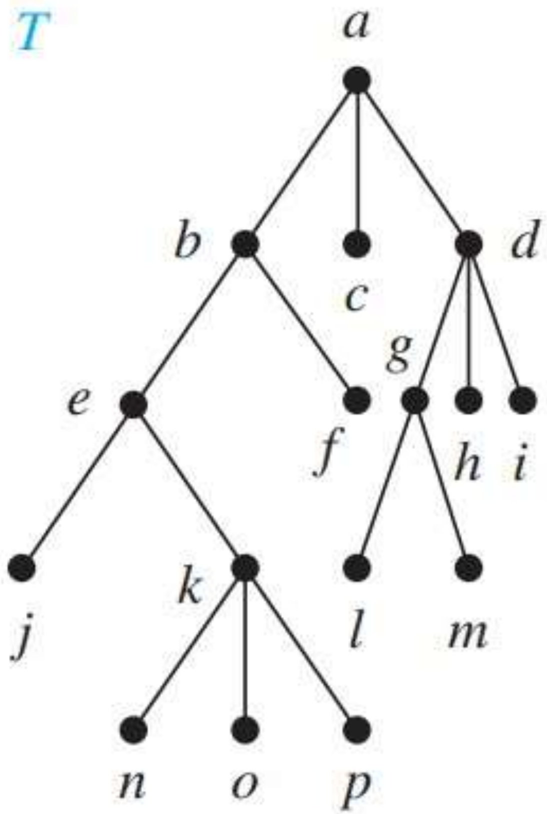


https://sites.google.com/view/adeel-arif/teaching/discrete-structures-f22

Postorder Traversal (Book 1, Page 799)

In which order does a postorder traversal visit the vertices of the ordered rooted tree T shown in Figure 3?

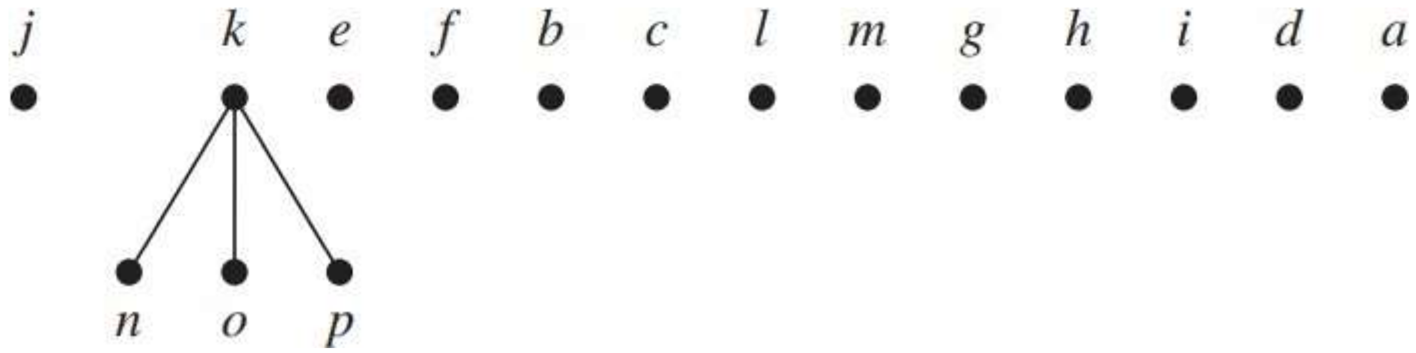


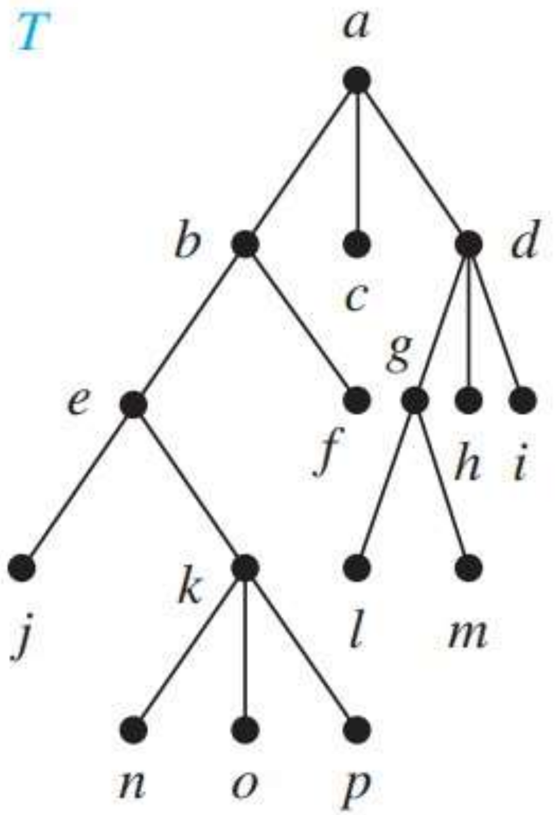


https://sites.google.com/view/adeel-arif/teaching/discrete-structures-f22

Postorder Traversal (Book 1, Page 799)

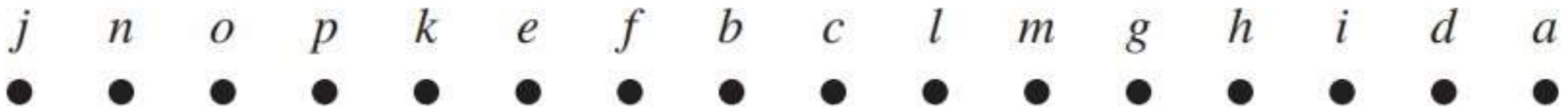
In which order does a postorder traversal visit the vertices of the ordered rooted tree T shown in Figure 3?

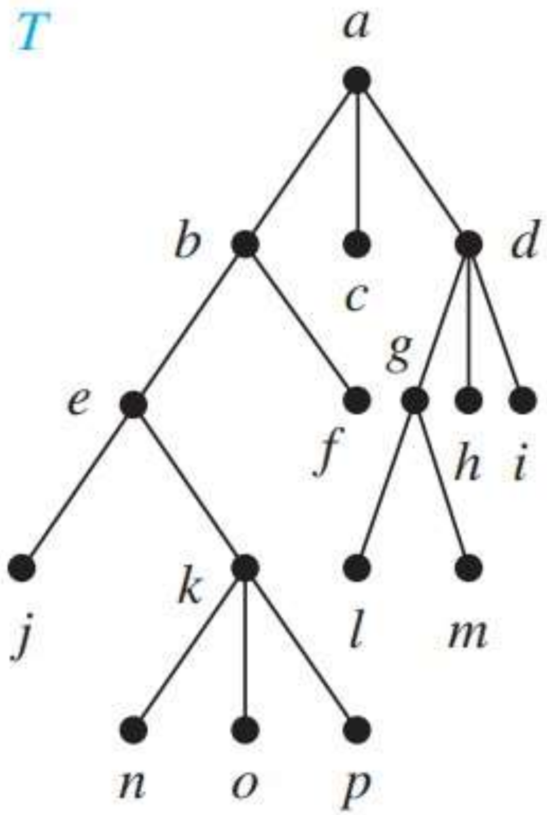




Postorder Traversal (Book 1, Page 799)

In which order does a postorder traversal visit the vertices of the ordered rooted tree T shown in Figure 3?





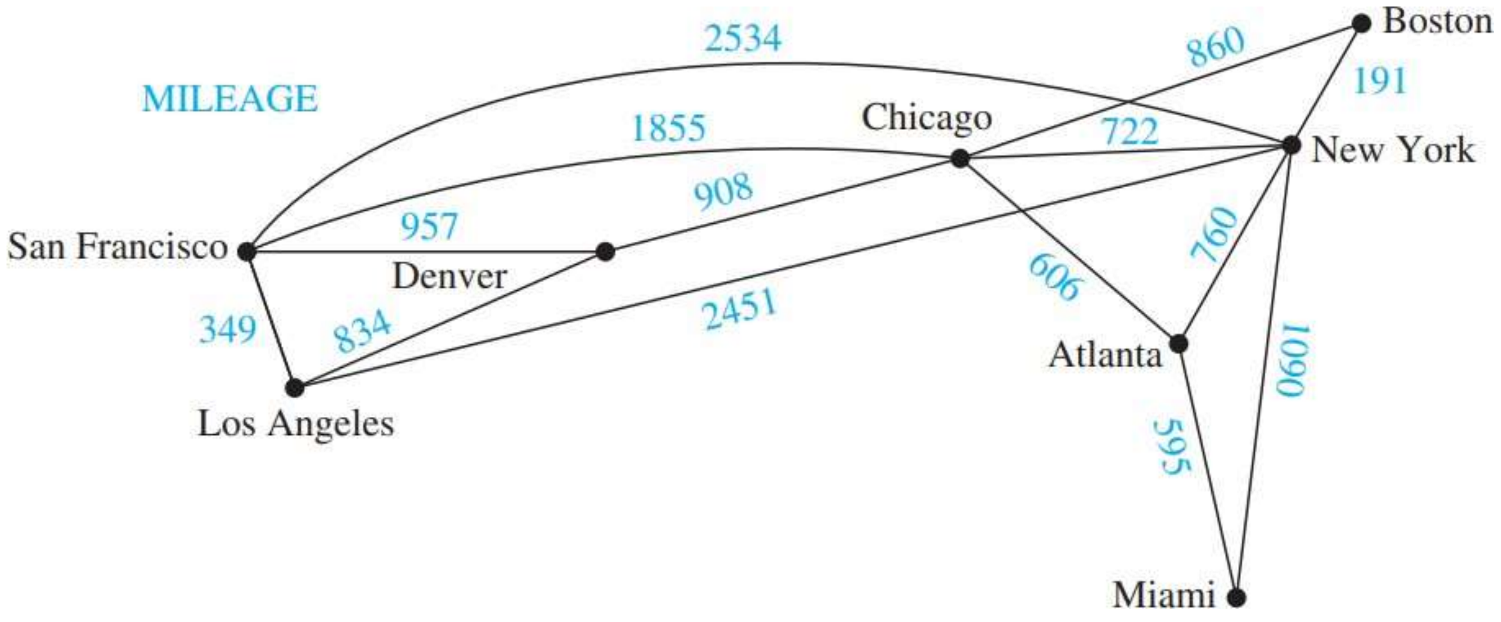
Shortest-Path Problems (Book 1, Page 728)

- Many problems can be modeled using graphs with weights assigned to their edges.
- How an airline system can be modeled. We set up the basic graph model by representing cities by vertices and flights by edges.

1. Problems involving distances can be modeled by assigning distances between the cities to the edge.
2. Problems involving flight time can be modeled by assigning flight times to edges.
3. Problems involving fares can be modeled by assigning fares to the edges.

Weighted Graphs Modeling an Airline System

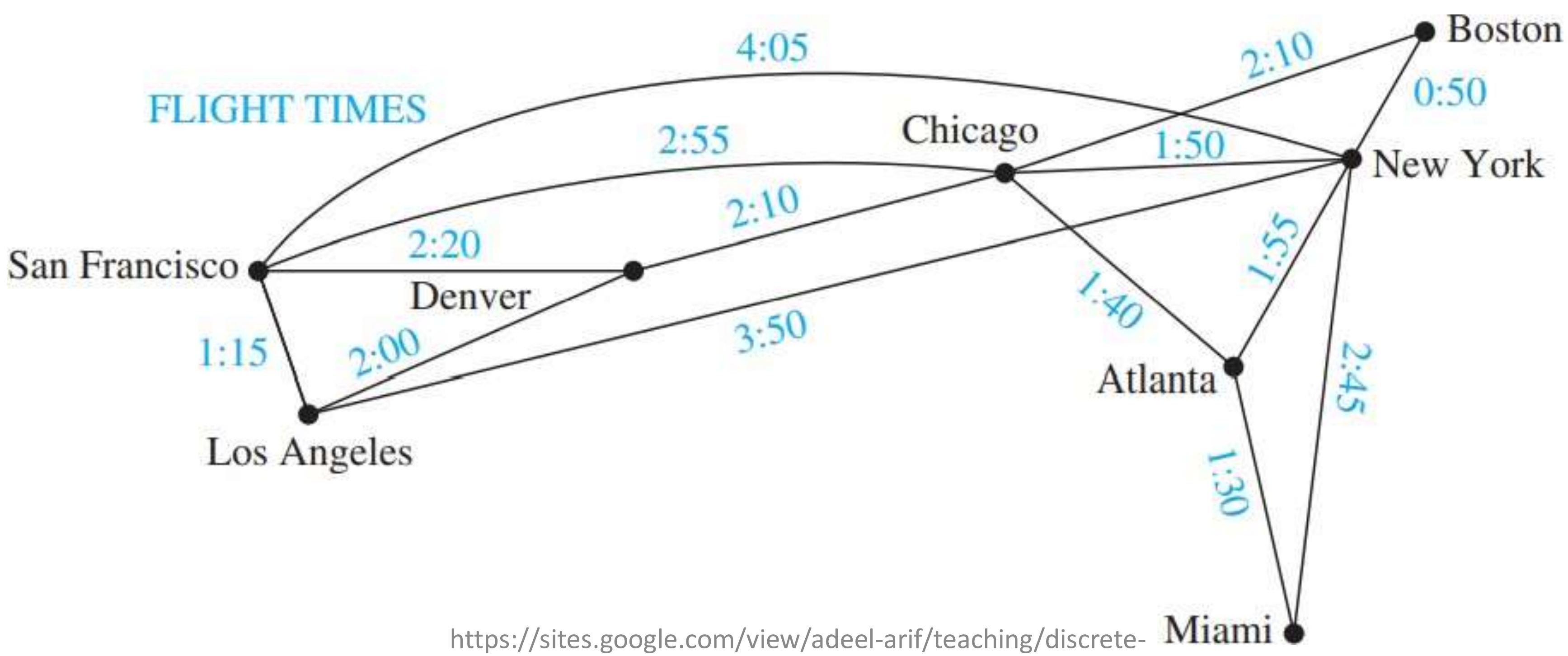
- Problems involving distances can be modeled by assigning distances between the cities to the edge.



https://sites.google.com/view/adeel-arif/teaching/discrete-structures-f22

Weighted Graphs Modeling an Airline System

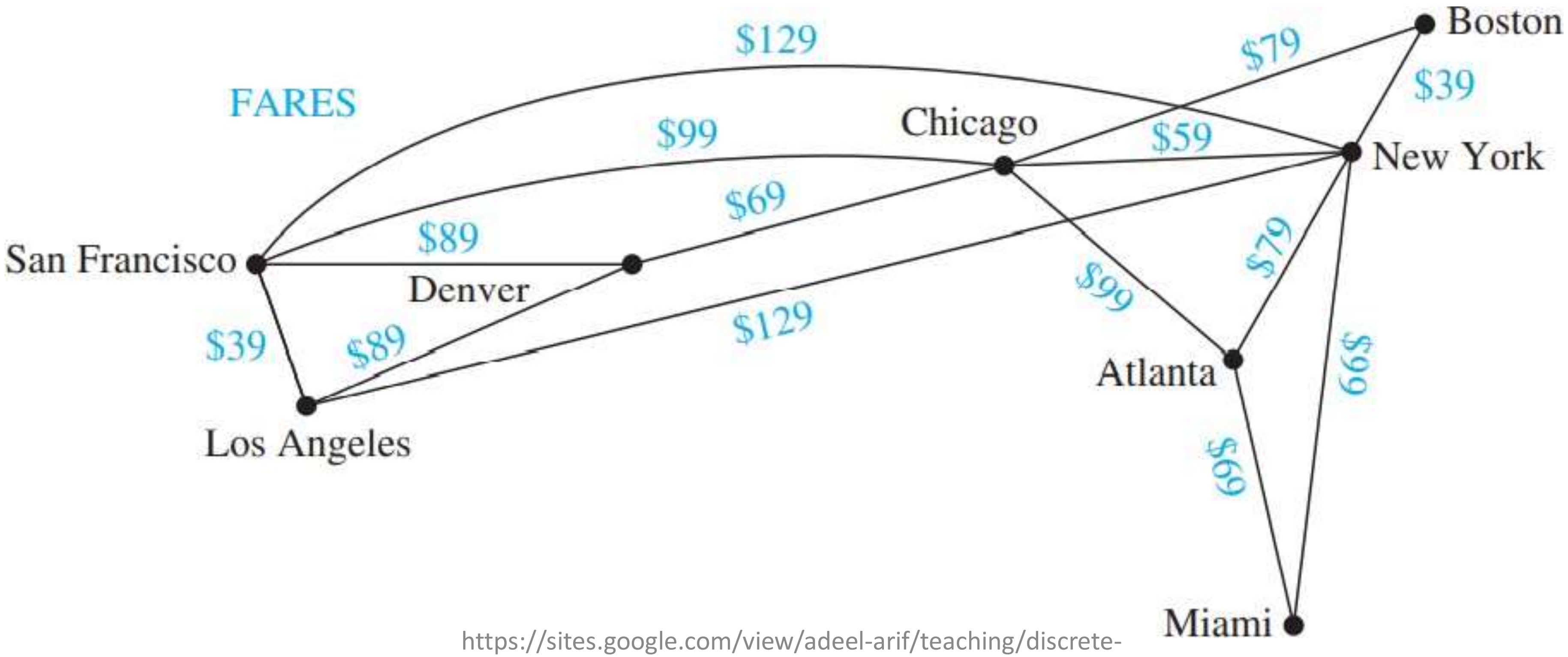
1. Problems involving flight time can be modeled by assigning flight times to edges.



https://sites.google.com/view/adeel-arif/teaching/discrete-structures-f22

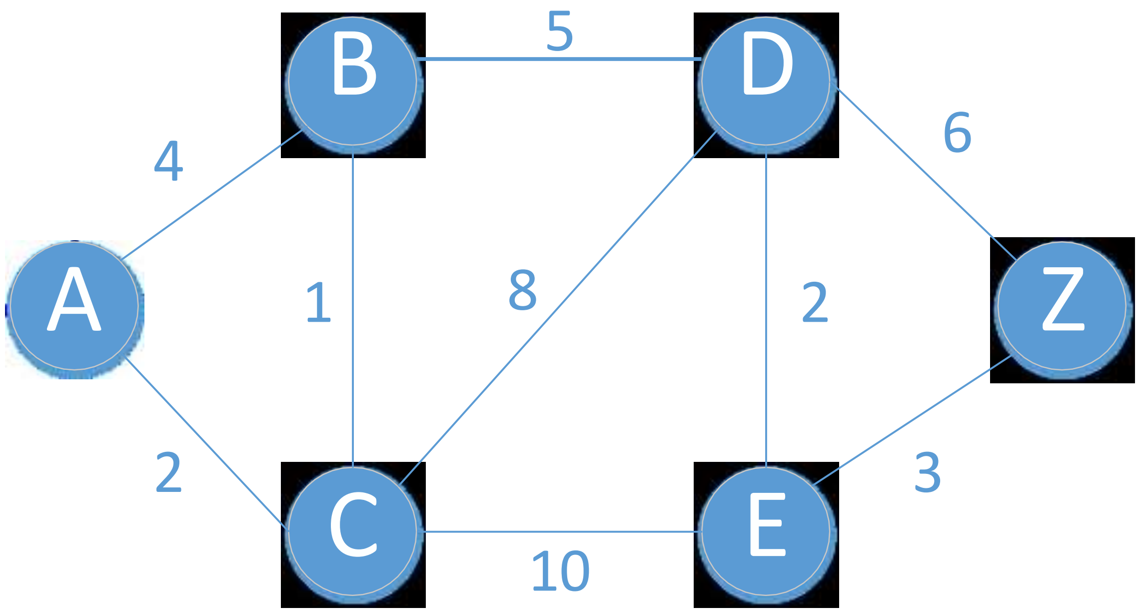
Weighted Graphs Modeling an Airline System

- Problems involving fares can be modeled by assigning fares to the edges.



https://sites.google.com/view/adeel-arif/teaching/discrete-structures-f22

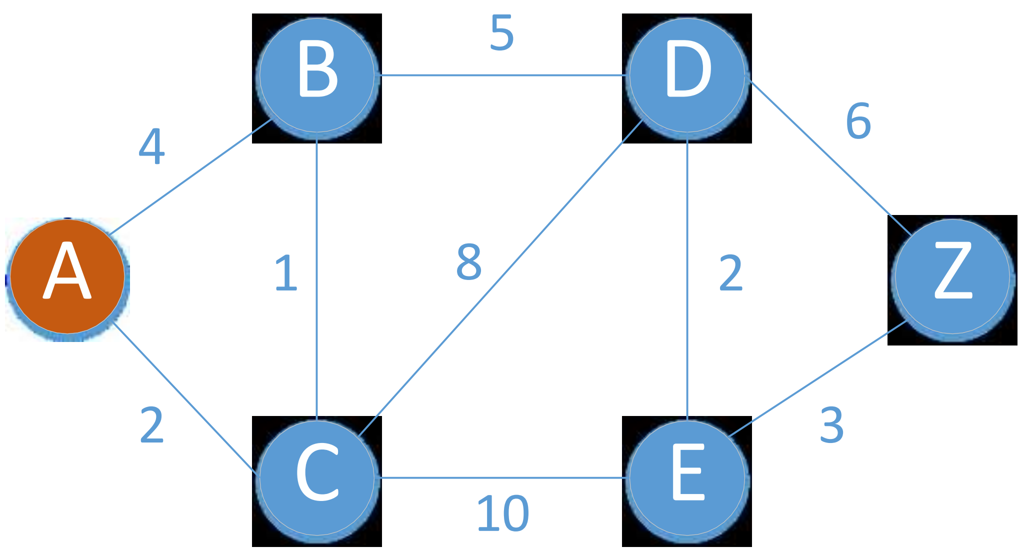
Find a shortest path from a to z. Using Dijkstra’s Algorithm (Book 1, Page 733)



Node	A	B	C	D	E	Z
Mini Distance	0	∞	∞	∞	∞	∞
Previous Node						

https://sites.google.com/view/adeel-arif/teaching/discrete-structures-f22

Find a shortest path from a to z. Using Dijkstra's Algorithm (Book 1, Page 733)



Node	A	B	C	D	E	Z
Mini Distance	0	∞	∞	∞	∞	∞
Previous Node						

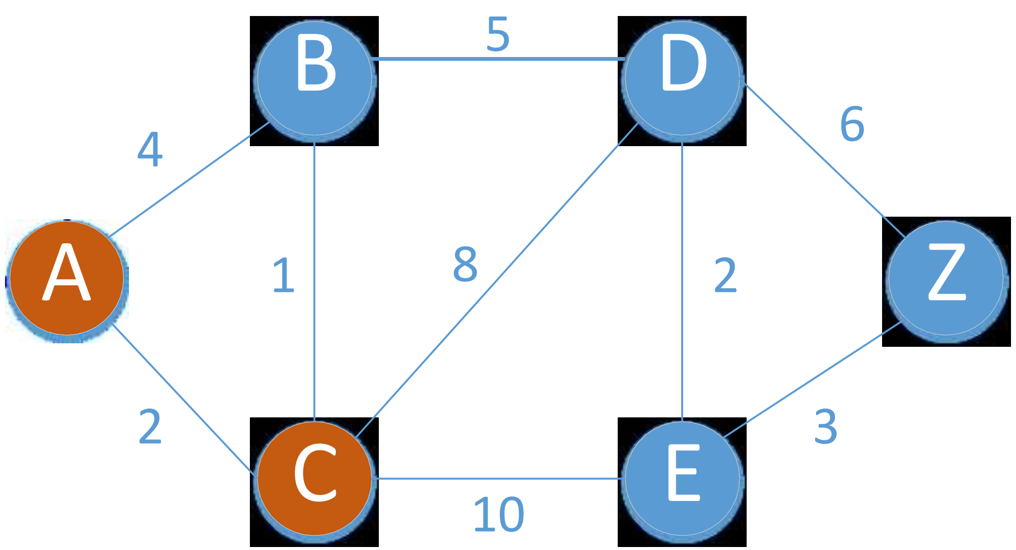
Node	A	B	C	D	E	Z
Mini Distance	0	4	2	∞	∞	∞
Previous Node		A	A			

L(b) = min{old L(b), L(a)+w(ab)}= {∞,0+4}= 4

L(c) = min{old L(c), L(a)+w(ac)}= {∞,0+2}= 2

https://sites.google.com/view/adeel-arif/teaching/discrete-structures-f22

Find a shortest path from a to z. Using Dijkstra’s Algorithm (Book 1, Page 733)

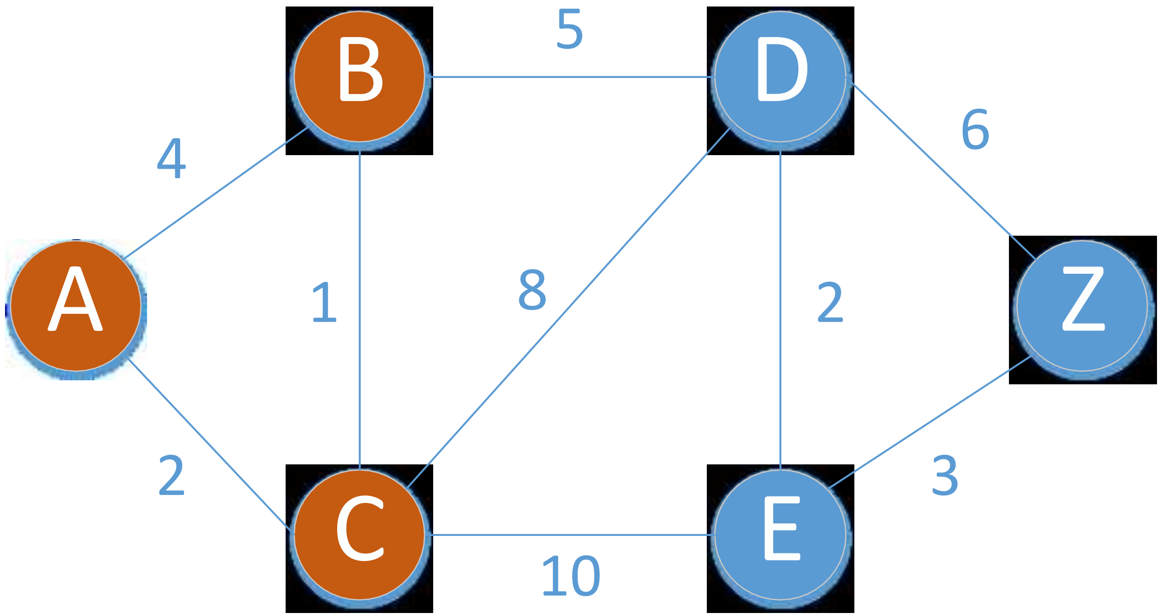


Node	A	B	C	D	E	Z
Mini Distance	0	4	2	∞	∞	∞
Previous Node		A	A			

Node	A	B	C	D	E	Z
Mini Distance	0	3	2	10	12	∞
Previous Node		C	A	C	C	

L(b) = min{old L(b), L(c)+w(cb)}= {4,2+1}=3
L(d) = min{old L(d), L(c)+w(cd)}= {∞,2+8}=10
L(e) = min{old L(e), L(c)+w(ce)}= {∞,2+10}=12

Find a shortest path from a to z. Using Dijkstra’s Algorithm (Book 1, Page 733)

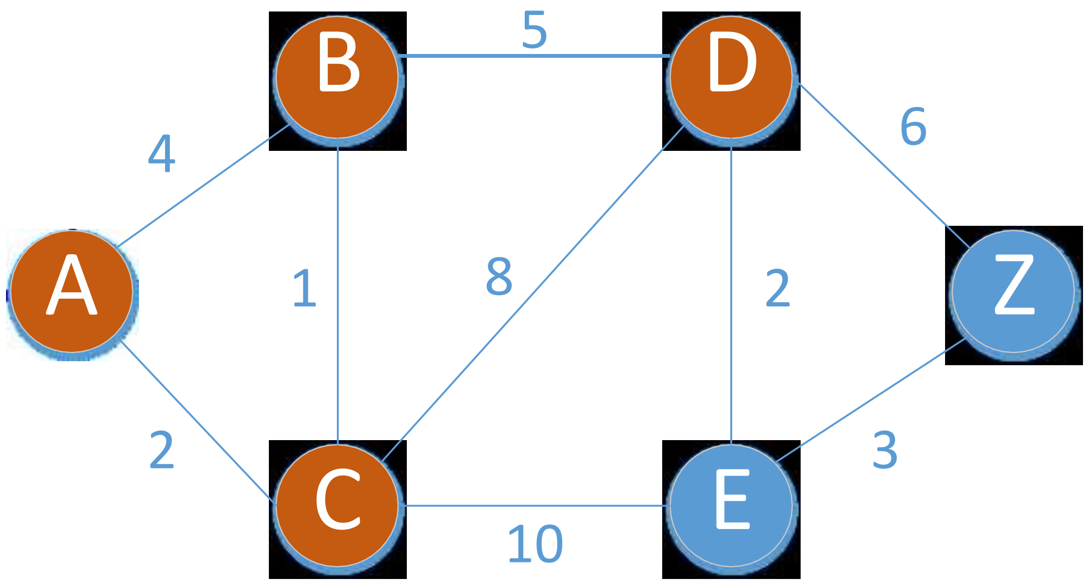


Node	A	B	C	D	E	Z
Mini Distance	0	3	2	10	12	∞
Previous Node		C	A	C	C	

Node	A	B	C	D	E	Z
Mini Distance	0	3	2	8	12	∞
Previous Node		C	A	B	C	

$L(d) = \min\{\text{old } L(d), L(b)+w(bd)\} = \{10, 3+5\} = 8$

Find a shortest path from a to z. Using Dijkstra’s Algorithm (Book 1, Page 733)



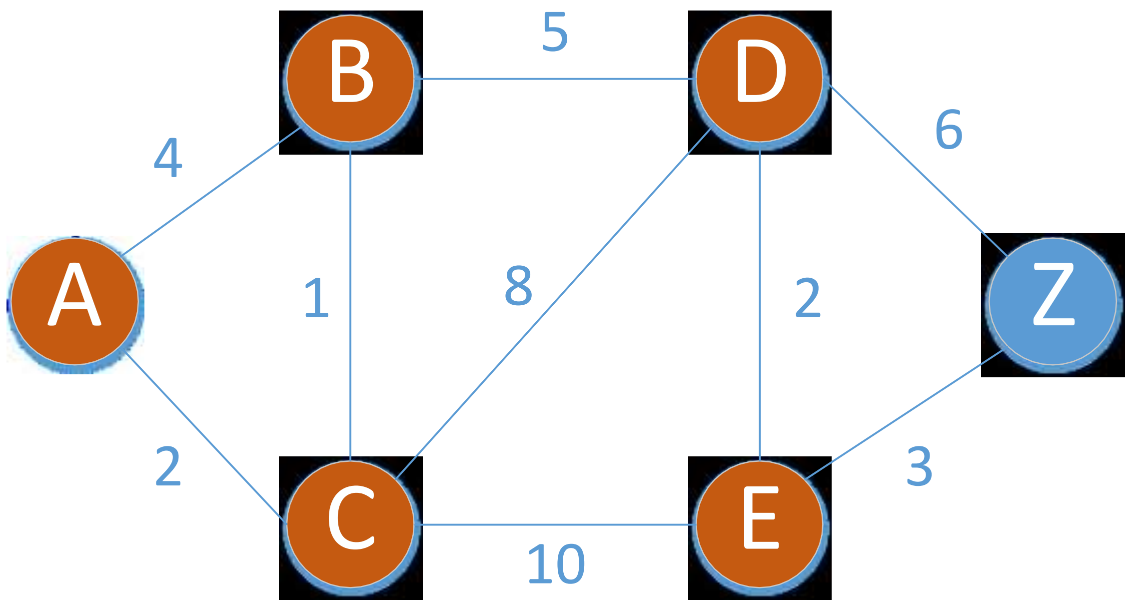
Node	A	B	C	D	E	Z
Mini Distance	0	3	2	8	12	∞
Previous Node		C	A	B	C	

Node	A	B	C	D	E	Z
Mini Distance	0	3	2	8	10	14
Previous Node		C	A	B	D	D

L(e) = min{old L(e), L(d)+w(de)}= {12,8+2}=10
L(z) = min{old L(z), L(d)+w(dz)}= {∞,8+6}=14

https://sites.google.com/view/adeel-arif/teaching/discrete-structures-f22

Find a shortest path from a to z. Using Dijkstra’s Algorithm (Book 1, Page 733)



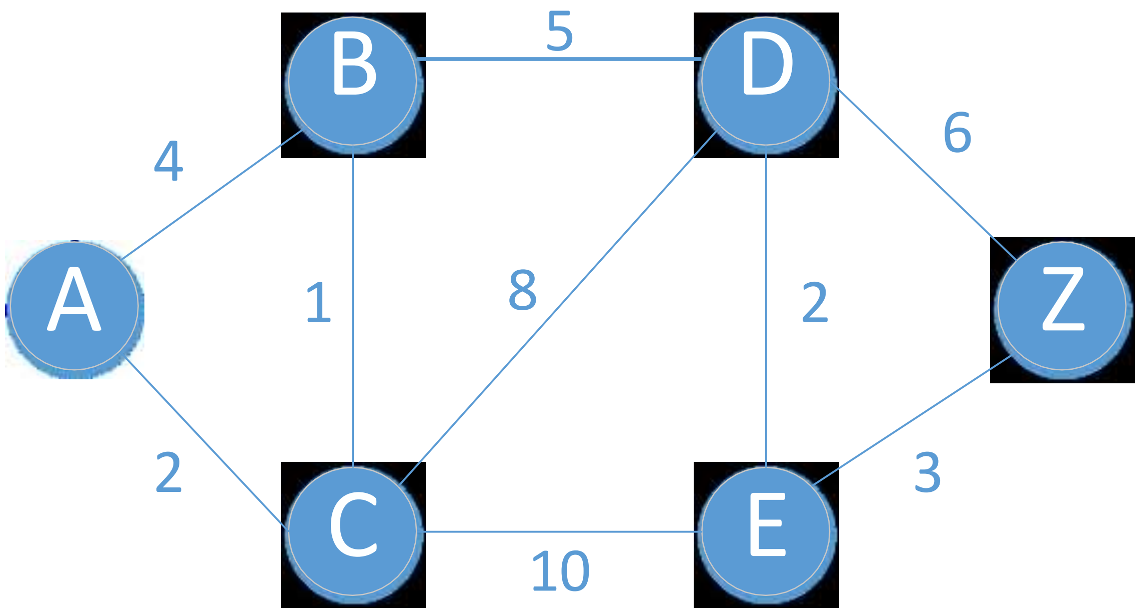
Node	A	B	C	D	E	Z
Mini Distance	0	3	2	8	10	14
Previous Node		C	A	B	D	D

Node	A	B	C	D	E	Z
Mini Distance	0	3	2	8	10	13
Previous Node		C	A	B	D	E

L(z) = min{old L(z), L(e)+w(ez)}= {14,10+3}=13

https://sites.google.com/view/adeel-arif/teaching/discrete-structures-f22

Find a shortest path from a to z. Using Dijkstra's Algorithm (Book 1, Page 733)



Node	A	B	C	D	E	Z
Mini Distance	0	3	2	8	10	13
Previous Node		C	A	B	D	E

The shortest path from a to z is a,c,b,d,e,z with length 13.

https://sites.google.com/view/adeel-arif/teaching/discrete-structures-f22

Consider the graph G in Fig. 7-14. Find two paths from v_1 to v_6.

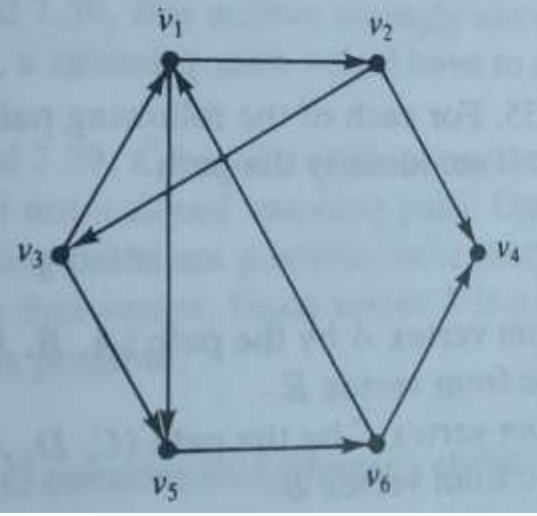


Fig. 7-14

7.31 Find two simple paths from v_1 to v_6 in the graph G of Problem 7.30.

I A simple path is a path where all vertices are distinct. There are only two simple paths from v_1 to v_6 in Fig. 7-14:

$$(v_1, v_5, v_6) \quad \text{and} \quad (v_1, v_2, v_3, v_5, v_6)$$

Cayley's Formula – How to Store the Trees

Theorem 8.3.1 (Cayley's Theorem) *The number of labeled trees on n nodes is n^{n-2}.*

- How to store this tree in computer?
- We want to store the tree so it should use least memory

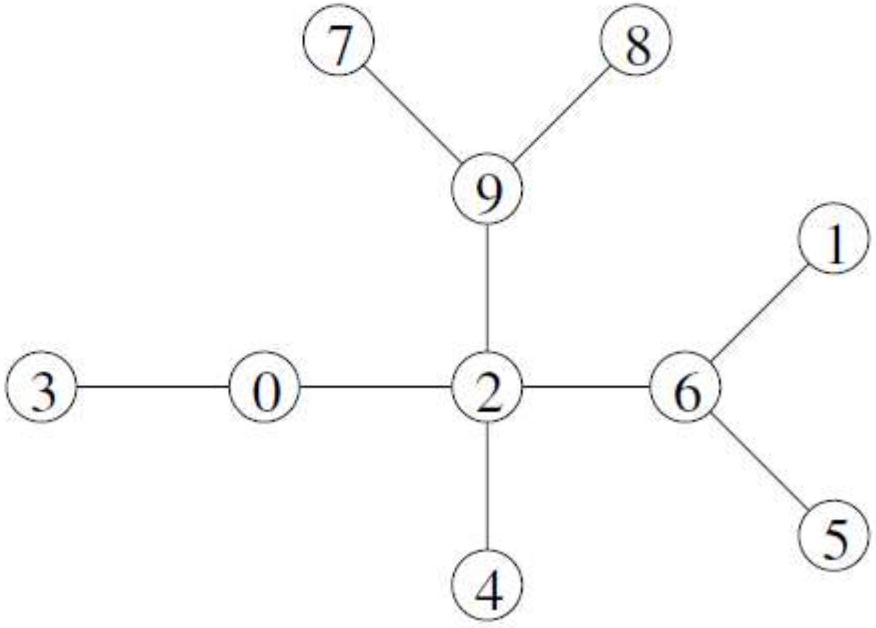


FIGURE 8.4. A labeled tree

How to Store the Trees – 1: Adjacency Matrix

- How to store this tree in computer?
- We want to store the tree so it should use least memory
- One solution could be to store the adjacency matrix … not an optimal way --- **consuming n^2 bits**

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

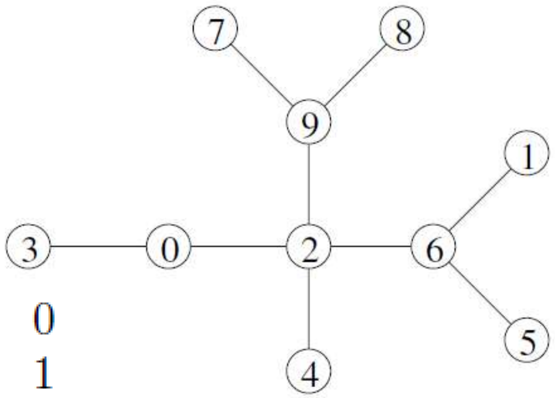


FIGURE 8.4. A labeled tree

How to Store the Trees – 1: Adjacency Matrix

- Adjacency Matrix

Cols

Rows

$$
\begin{array}{c|cccccccccc}
 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 0 \\
\hline
1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
2 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 1 \\
3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
4 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
5 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
6 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
7 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
8 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
9 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\
0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{array}
$$

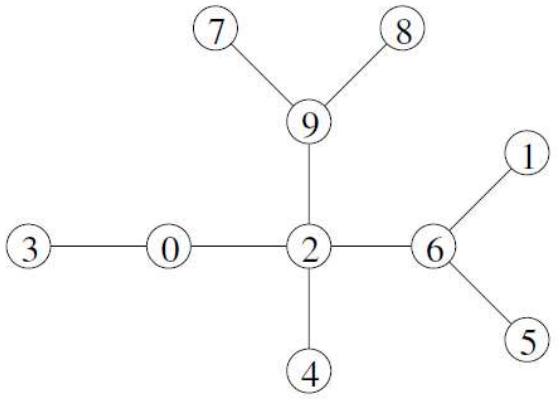


FIGURE 8.4. A labeled tree

One solution could be to store the adjacency matrix …

not an optimal way ---

consuming n^2 bits