CS215 DISCRETE MATH

Dr. Qi Wang

Department of Computer Science and Engineering
Office: Room413, CoE South Tower
Email: wangqi@sustech.edu.cn

Euler's Formula

- **Theorem** (Euler's Formula) Let G be a connected planar simple graph with e edges and v vertices. Let r be the number of regions in a planar representation of G. Then $r = e - v + 2$.

Euler's Formula

- **Theorem** (Euler's Formula) Let G be a connected planar simple graph with e edges and v vertices. Let r be the number of regions in a planar representation of G. Then $r = e - v + 2$.

- **Definition** The *degree* of a region is defined to be the number of edges on the boundary of this region. When an edge occurs twice on the boundary, it contributes two to the degree.

Corollaries

- **Corollary 1** If G is a connected planar simple graph with e edges and v vertices, where $v \geq 3$, then $e \leq 3v - 6$.

  Proof The degree of every region is at least 3.

  - G is simple
  - $v \geq 3$

  The sum of the degrees of the regions is exactly twice the number of edges in the graph.

$$2e = \sum_{\text{all regions } R} \deg(R) \geq 3r$$

By Euler's formula, the proof is completed.

- **Corollary 2** If G is a connected planar simple graph, then G has a vertex of degree not exceeding 5.

  Proof

  (By contradiction)

  By Corollary 1 and the Handshaking Theorem.

Corollaries

- **Corollary 2** If G is a connected planar simple graph, then G has a vertex of degree not exceeding 5.

  Proof

  (By contradiction)

  By Corollary 1 and the Handshaking Theorem.

  Corollary 3 In a connected planar simple graph has e edges and v vertices with $v \geq 3$ and no circuits of length three, then $e \leq 2v - 4$.

  Proof similar to that of Corollary 1.

Examples

- Show that K_5 is nonplanar.

  Using Corollary 1

Examples

- Show that K_5 is nonplanar.

  Using Corollary 1

  Show that $K_{3,3}$ is nonplanar.

  Using Corollary 3

Examples

- Show that K_5 is nonplanar.

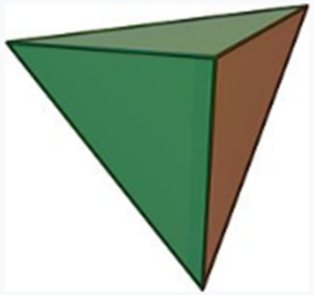
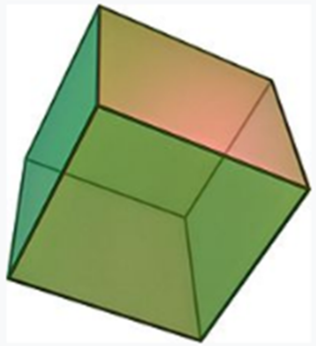
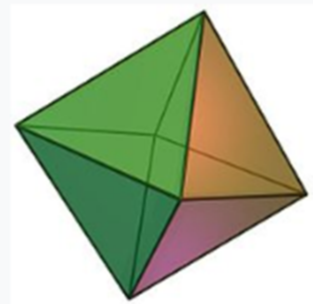
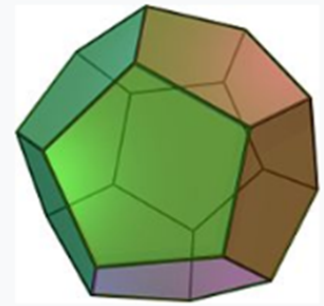
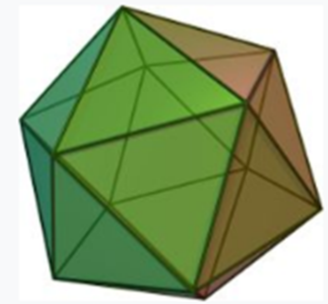
  Using Corollary 1

  Show that $K_{3,3}$ is nonplanar.

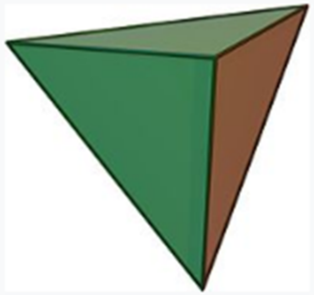
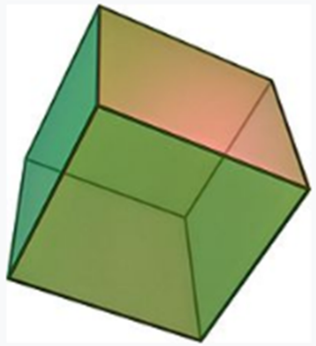
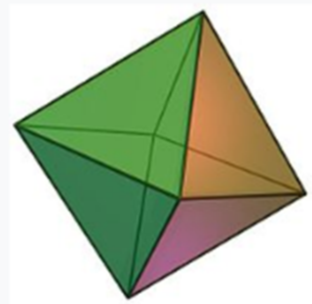
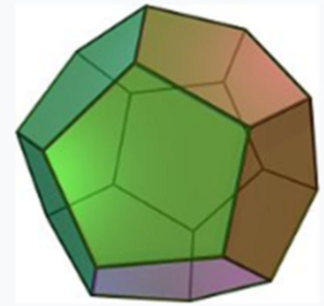
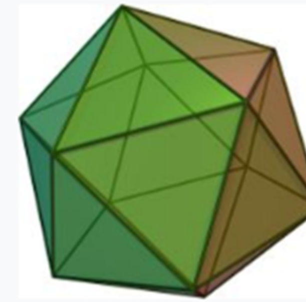
  Using Corollary 3

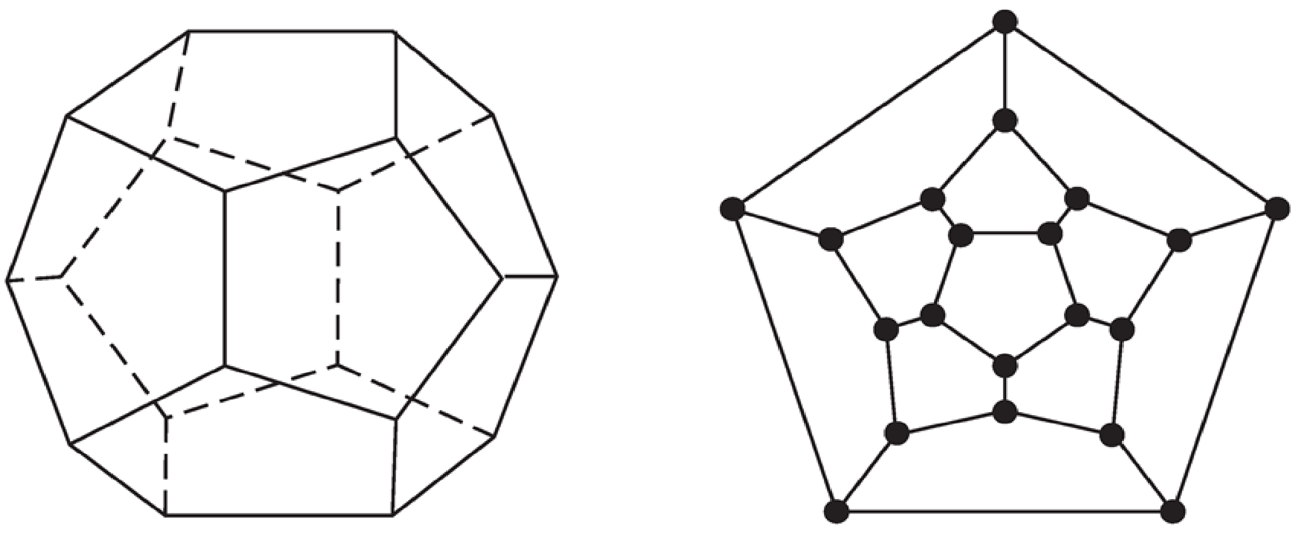
  Corollary 2 is used in the proof of Five Color Theorem.

Only 5 Platonic Solids

				
Tetrahedron {3, 3}	Cube {4, 3}	Octahedron {3, 4}	Dodecahedron {5, 3}	Icosahedron {3, 5}

Only 5 Platonic Solids

				
Tetrahedron {3, 3}	Cube {4, 3}	Octahedron {3, 4}	Dodecahedron {5, 3}	Icosahedron {3, 5}



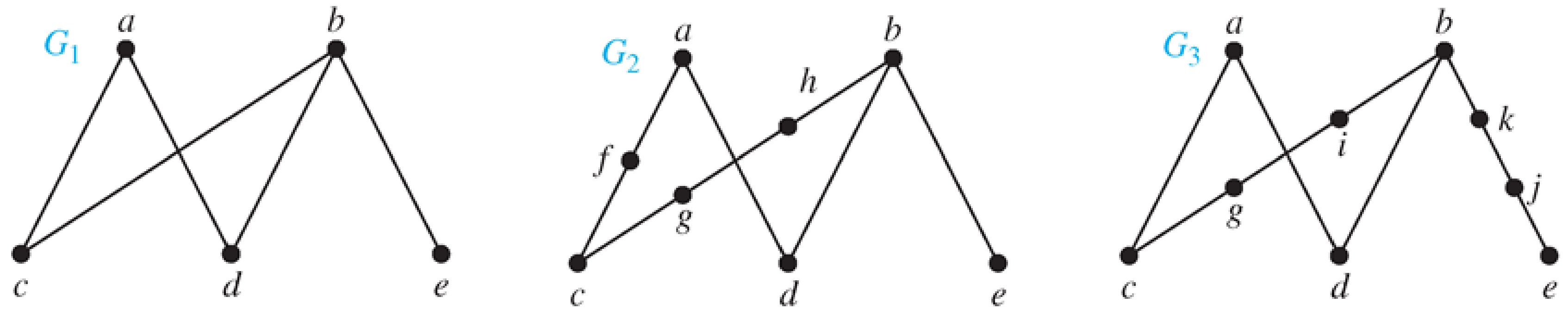
(a) (b)

Kuratowski's Theorem

- **Definition** If a graph is planar, so will be any graph obtained by removing an edge $\{u, v\}$ and adding a new vertex w together with edges $\{u, w\}$ and $\{w, v\}$. Such an operation is called an *elementary subdivision*. The graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ are called *homomorphic* if they can be obtained from the same graph by a sequence of elementary subdivisions.

Kuratowski's Theorem

- **Definition** If a graph is planar, so will be any graph obtained by removing an edge $\{u, v\}$ and adding a new vertex w together with edges $\{u, w\}$ and $\{w, v\}$. Such an operation is called an *elementary subdivision*. The graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ are called *homomorphic* if they can be obtained from the same graph by a sequence of elementary subdivisions.

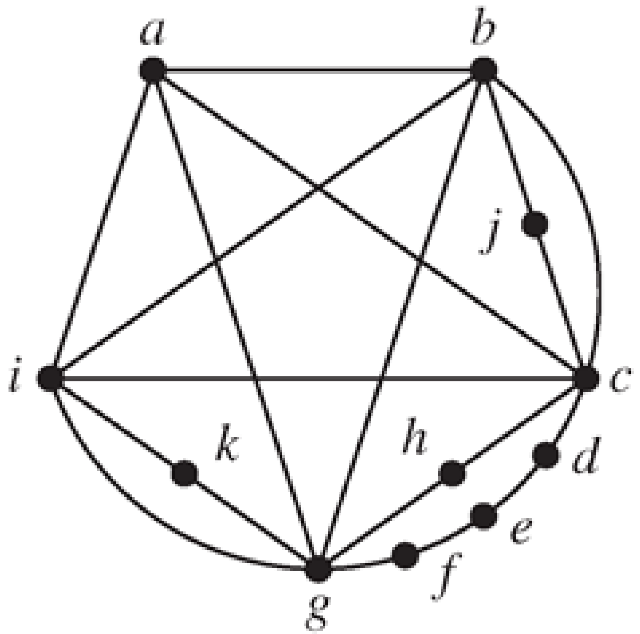


Kuratowski's Theorem

- **Definition** If a graph is planar, so will be any graph obtained by removing an edge $\{u, v\}$ and adding a new vertex w together with edges $\{u, w\}$ and $\{w, v\}$. Such an operation is called an *elementary subdivision*. The graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ are called *homomorphic* if they can be obtained from the same graph by a sequence of elementary subdivisions.

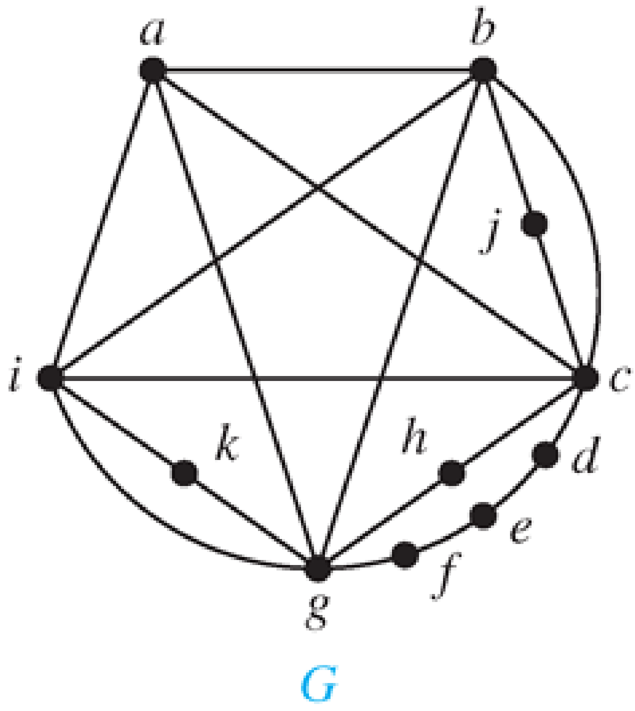
  Theorem A graph is nonplanar if and only if it contains a subgraph homomorphic to $K_{3,3}$ or K_5.

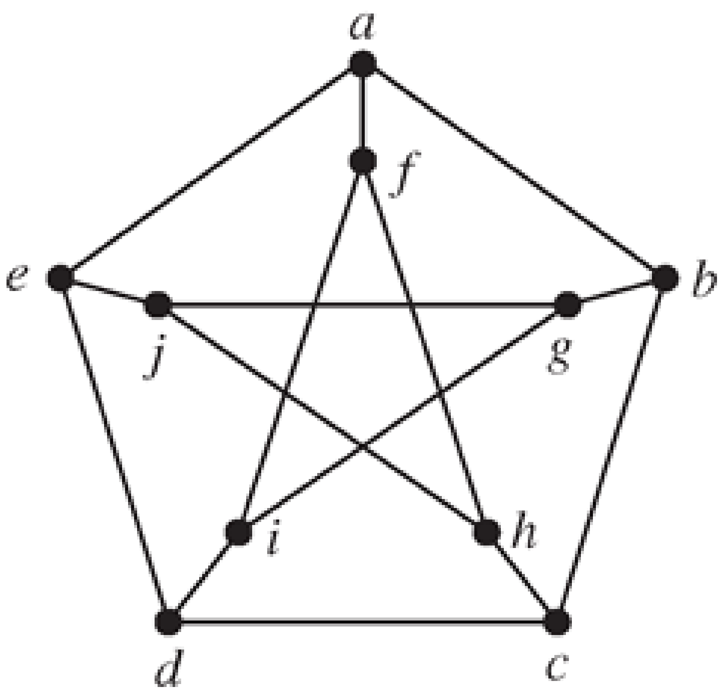
Examples



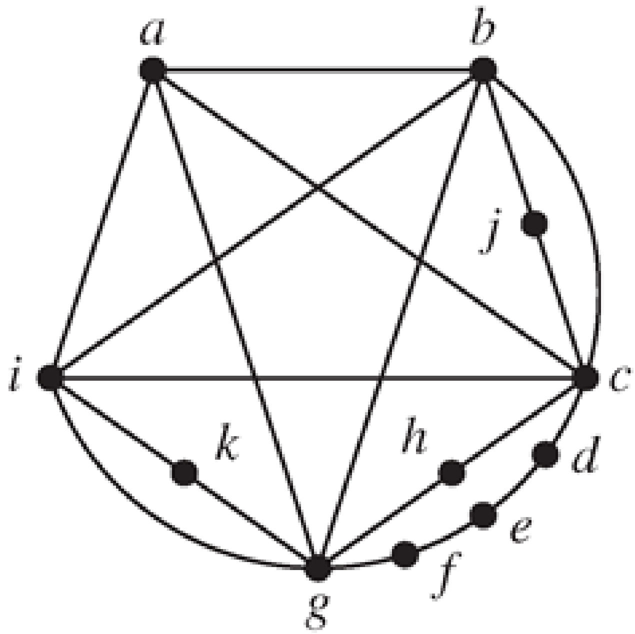
G

Examples

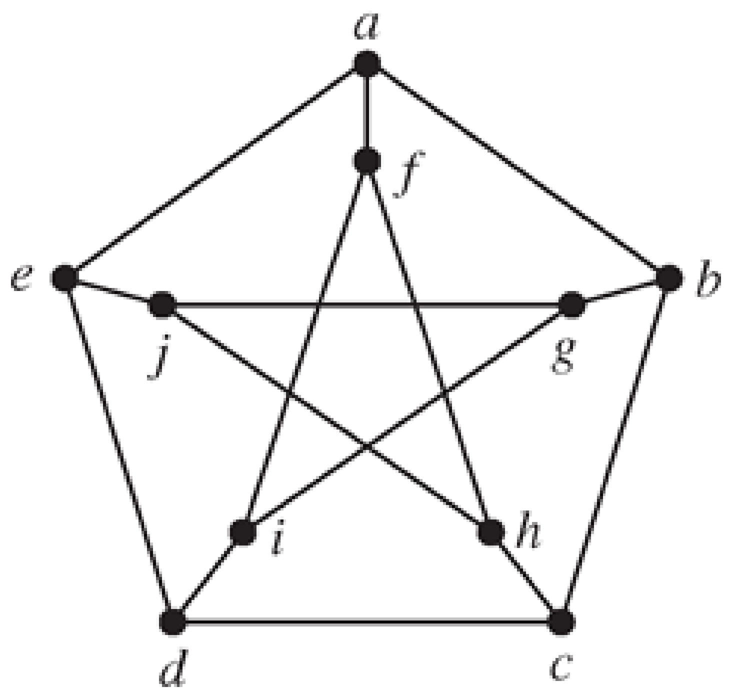


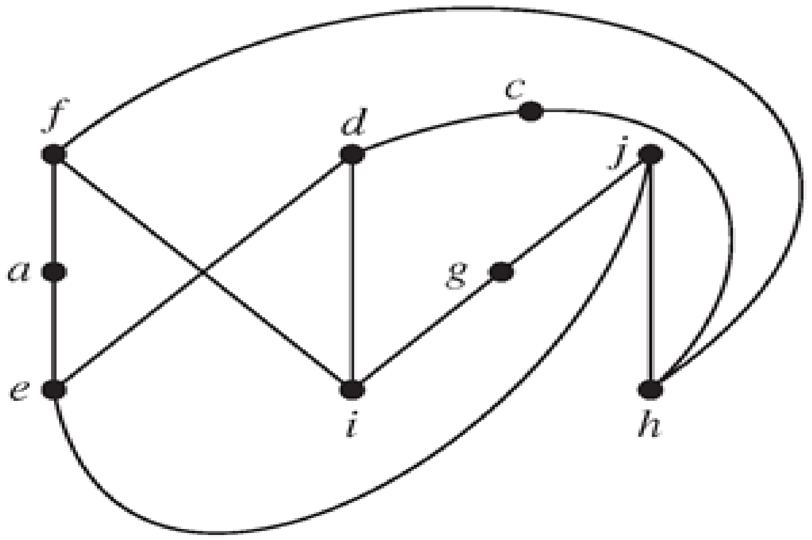


Examples



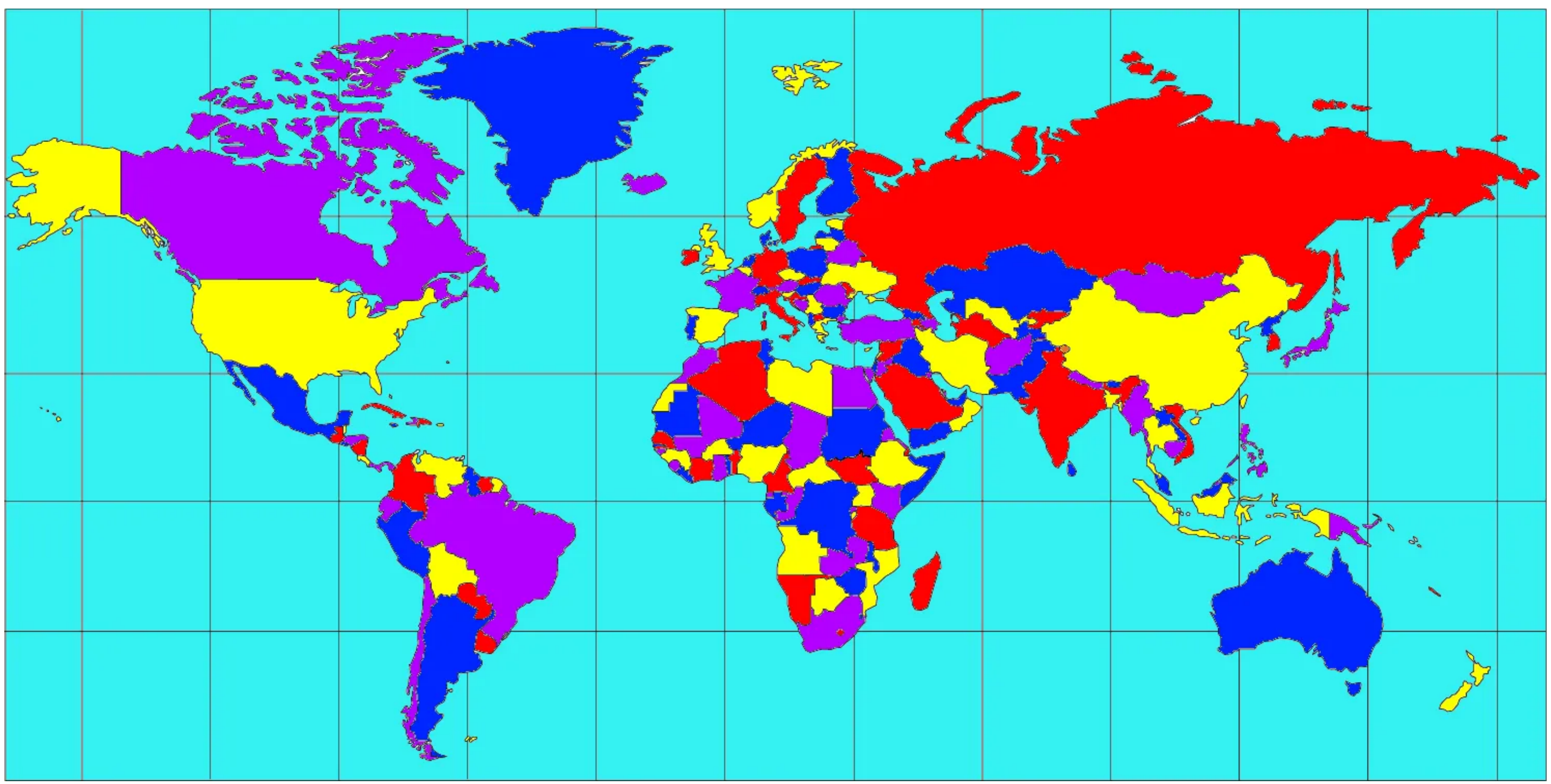
G





Graph Coloring

- **Four-color theorem** Given any separation of a plane into contiguous regions, producing a figure called a *map*, no more than four colors are required to color the regions of the map so that no two adjacent regions have the same color.



- **Four-color theorem**
  - ◇ first proposed by Francis Guthrie in 1852
  - ◇ his brother Frederick Guthrie told Augustus De Morgan
  - ◇ De Morgan wrote to William Hamilton
  - ◇ Alfred Kempe proved it incorrectly in 1879
  - ◇ Percy Heawood found an error in 1890 and proved the *five-color theorem*
  - ◇ Finally, Kenneth Appel and Wolfgang Haken proved it with case by case analysis by computer in 1976 (*the first computer-aided proof*)
  - ◇ Kempe's incorrect proof serves as a basis

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.

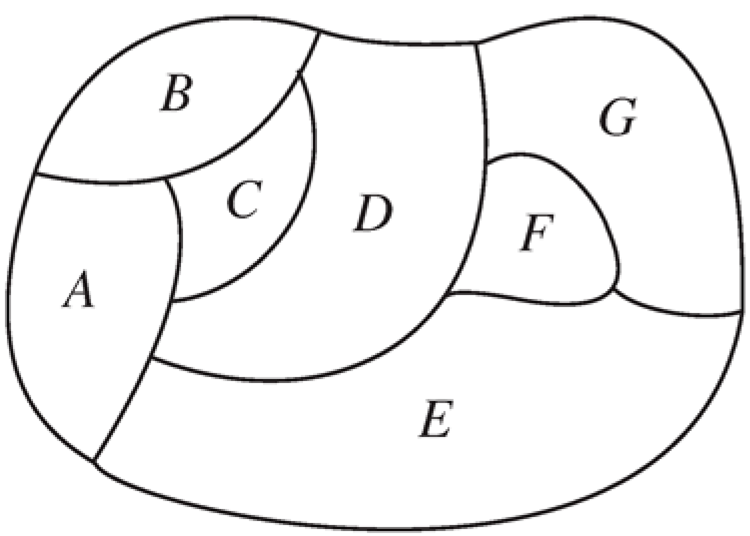
Graph Coloring

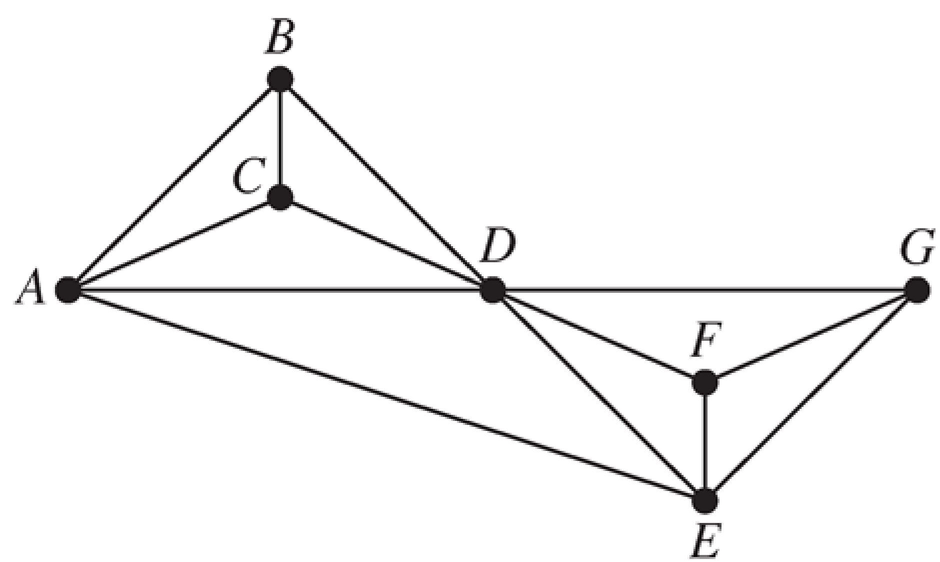
- A *coloring* of a simple graph is the assignment of a color to each vertex of the graph so that no two adjacent vertices are assigned the same color.

  The *chromatic number* of a graph is the least number of colors needed for a coloring of this graph, denoted by $\chi(G)$.

- 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.

  The *chromatic number* of a graph is the least number of colors needed for a coloring of this graph, denoted by $\chi(G)$.



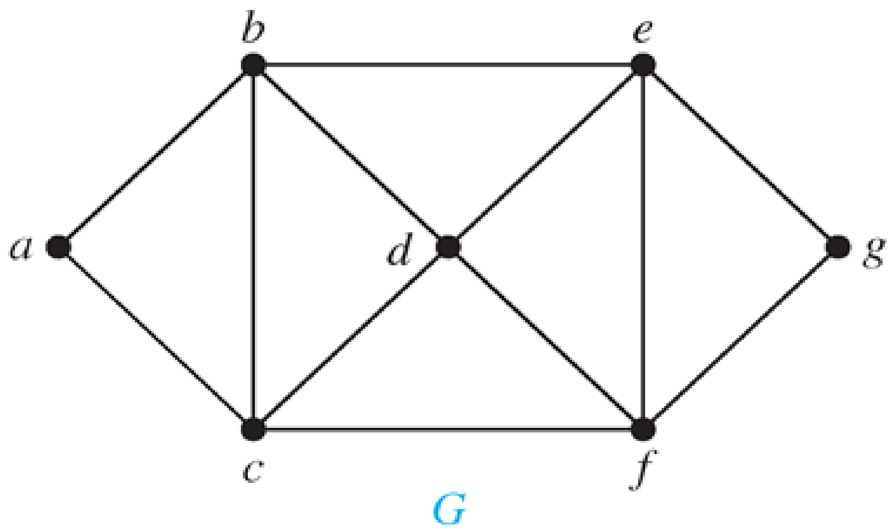


Graph Coloring

- **Theorem** (Four Color Theorem) The chromatic number of a planar graph is no greater than four.

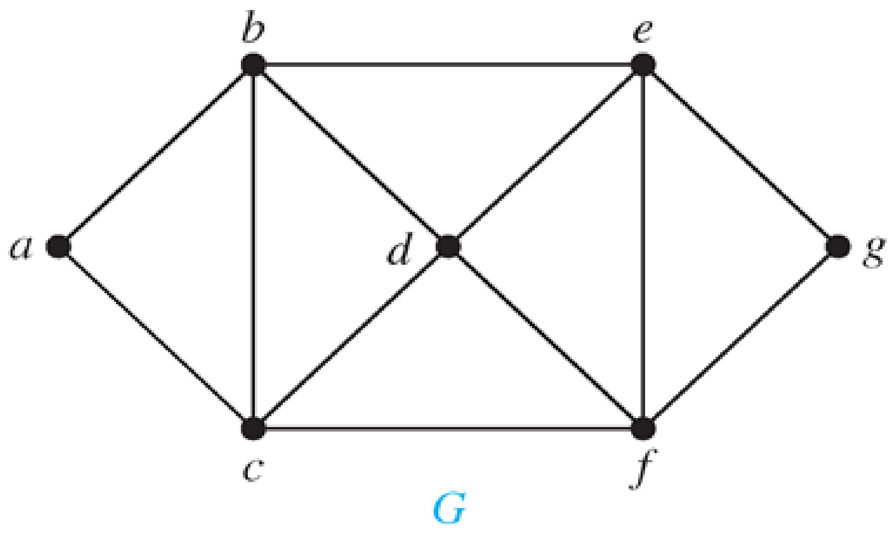
Graph Coloring

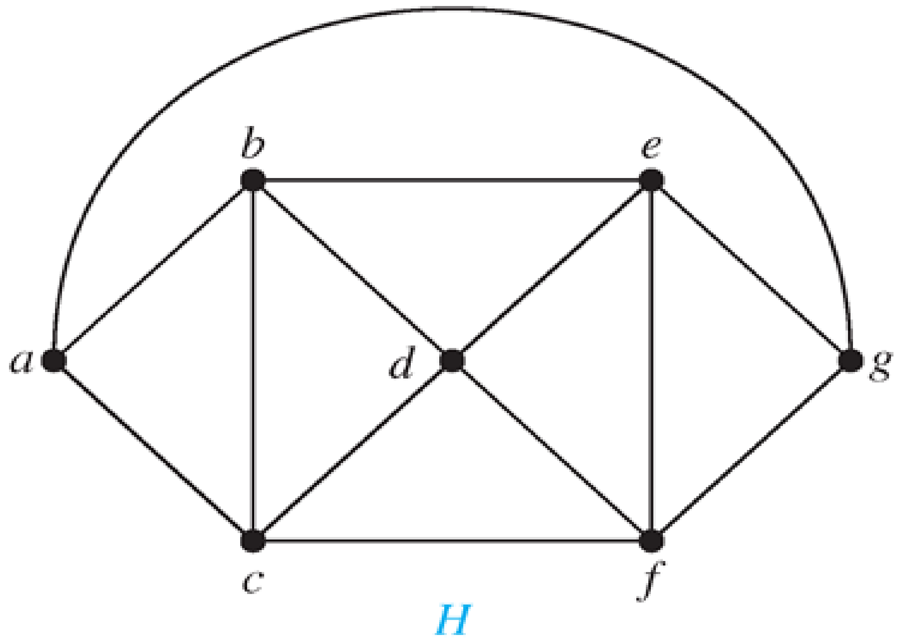
- **Theorem** (Four Color Theorem) The chromatic number of a planar graph is no greater than four.



Graph Coloring

- **Theorem** (Four Color Theorem) The chromatic number of a planar graph is no greater than four.





Graph Coloring

- **Theorem** (Six Color Theorem) The chromatic number of a planar graph is no greater than six.

Graph Coloring

- **Theorem** (Six Color Theorem) The chromatic number of a planar graph is no greater than six.

  Proof (by induction on the number of vertices)
  w.l.o.g., assume that the graph is connected.

- **Theorem** (Six Color Theorem) The chromatic number of a planar graph is no greater than six.

  Proof (by induction on the number of vertices)
  w.l.o.g., assume that the graph is connected.

  Basic step: For one single vertex, pick an arbitrary color.

- **Theorem** (Six Color Theorem) The chromatic number of a planar graph is no greater than six.

  Proof (by induction on the number of vertices)
  w.l.o.g., assume that the graph is connected.

  Basic step: For one single vertex, pick an arbitrary color.
  Inductive hypothesis: Assume that every planar graph with $k \geq 1$ or fewer vertices can be 6-colored.

- **Theorem** (Six Color Theorem) The chromatic number of a planar graph is no greater than six.

  Proof (by induction on the number of vertices)
  w.l.o.g., assume that the graph is connected.

  Basic step: For one single vertex, pick an arbitrary color.
  Inductive hypothesis: Assume that every planar graph with $k \geq 1$ or fewer vertices can be 6-colored.
  Inductive step: Consider a planar graph with $k+1$ vertices.

- **Theorem** (Six Color Theorem) The chromatic number of a planar graph is no greater than six.

  Proof (by induction on the number of vertices)
  w.l.o.g., assume that the graph is connected.

  Basic step: For one single vertex, pick an arbitrary color.
  Inductive hypothesis: Assume that every planar graph with $k \geq 1$ or fewer vertices can be 6-colored.
  Inductive step: Consider a planar graph with $k+1$ vertices. Recall Corollary 2 (the graph has a vertex of degree 5 or fewer). Remove this vertex, by i.h., we can color the remaining graph with 6 colors. Put the vertex back in. Since there are at most 5 colors adjacent, so we have at least one color left.

Graph Coloring

- **Theorem** (Five Color Theorem) The chromatic number of a planar graph is no greater than five.

- **Theorem** (Five Color Theorem) The chromatic number of a planar graph is no greater than five.

  Proof (by induction on the number of vertices)
  w.l.o.g., assume that the graph is connected.

- **Theorem** (Five Color Theorem) The chromatic number of a planar graph is no greater than five.

  Proof (by induction on the number of vertices)
  w.l.o.g., assume that the graph is connected.

  Basic step: For one single vertex, pick an arbitrary color.

Graph Coloring

- **Theorem** (Five Color Theorem) The chromatic number of a planar graph is no greater than five.

  Proof (by induction on the number of vertices)
  w.l.o.g., assume that the graph is connected.

  Basic step: For one single vertex, pick an arbitrary color.
  Inductive hypothesis: Assume that every planar graph with $k \geq 1$ or fewer vertices can be 5-colored.

Graph Coloring

- **Theorem** (Five Color Theorem) The chromatic number of a planar graph is no greater than five.

  Proof (by induction on the number of vertices)
  w.l.o.g., assume that the graph is connected.

  Basic step: For one single vertex, pick an arbitrary color.
  Inductive hypothesis: Assume that every planar graph with $k \geq 1$ or fewer vertices can be 5-colored.
  Inductive step: Consider a planar graph with $k+1$ vertices.

- **Theorem** (Five Color Theorem) The chromatic number of a planar graph is no greater than five.

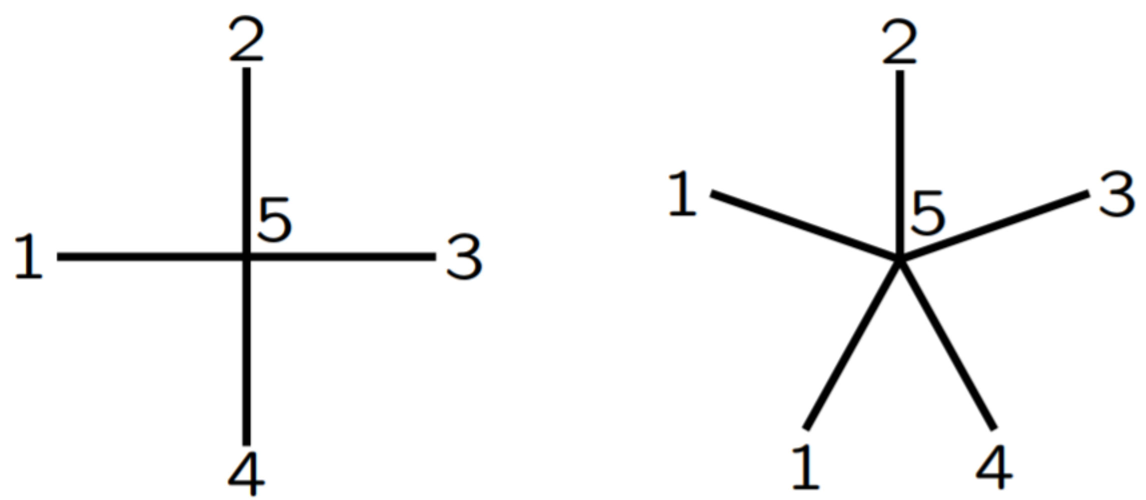
  Proof (by induction on the number of vertices)
  w.l.o.g., assume that the graph is connected.

  Basic step: For one single vertex, pick an arbitrary color.
  Inductive hypothesis: Assume that every planar graph with $k \geq 1$ or fewer vertices can be 5-colored.
  Inductive step: Consider a planar graph with $k+1$ vertices. Recall Corollary 2 (the graph has a vertex of degree 5 or fewer). Remove this vertex, by i.h., we can color the remaining graph with 5 colors. Put the vertex back in.

- **Theorem** (Five Color Theorem) The chromatic number of a planar graph is no greater than five.

  Proof (by induction on the number of vertices)
  w.l.o.g., assume that the graph is connected.

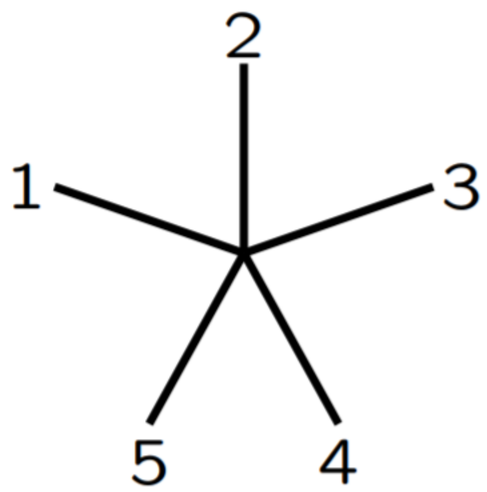
  If the vertex has degree less than 5, or if it has degree 5 and only ≤ 4 colors are used for vertices connected to it, we can pick an available color for it.



- **Theorem** (Five Color Theorem) The chromatic number of a planar graph is no greater than five.

  Proof (by induction on the number of vertices)

  If the vertex has degree 5, and all 5 colors are connected to it, we label the vertices adjacent to the "special" vertex (degree 5) 1 to 5 (in order).



- **Theorem** (Five Color Theorem) The chromatic number of a planar graph is no greater than five.

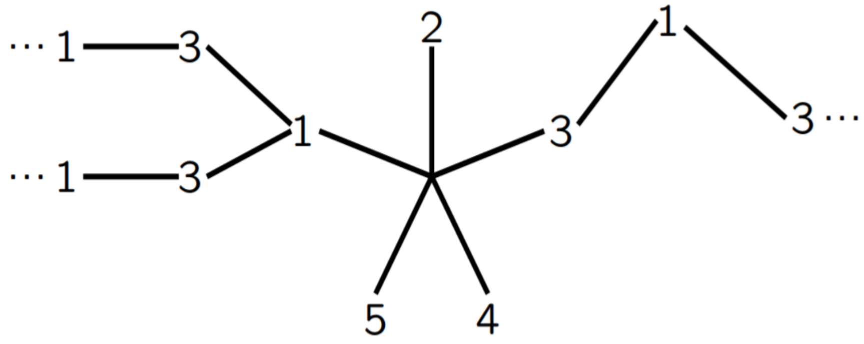
  Proof (by induction on the number of vertices)

  We make a subgraph out of all the vertices colored 1 or 3. If the adjacent vertex colored 1 and the adjacent vertex colored 3 are not connected by a path in the subgraph.

- **Theorem** (Five Color Theorem) The chromatic number of a planar graph is no greater than five.

  Proof (by induction on the number of vertices)

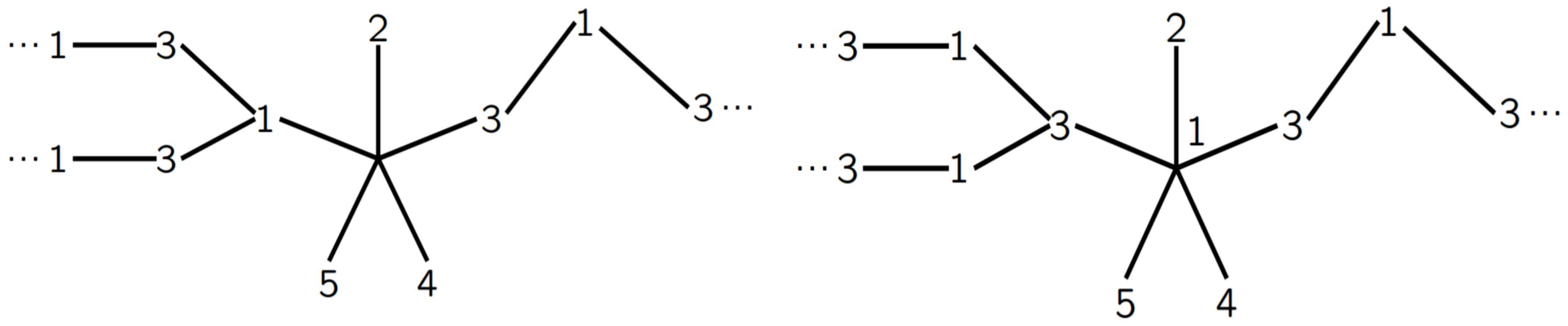
  We make a subgraph out of all the vertices colored 1 or 3. If the adjacent vertex colored 1 and the adjacent vertex colored 3 are not connected by a path in the subgraph.



- **Theorem** (Five Color Theorem) The chromatic number of a planar graph is no greater than five.

  Proof (by induction on the number of vertices)

  We make a subgraph out of all the vertices colored 1 or 3. If the adjacent vertex colored 1 and the adjacent vertex colored 3 are not connected by a path in the subgraph.



- **Theorem** (Five Color Theorem) The chromatic number of a planar graph is no greater than five.

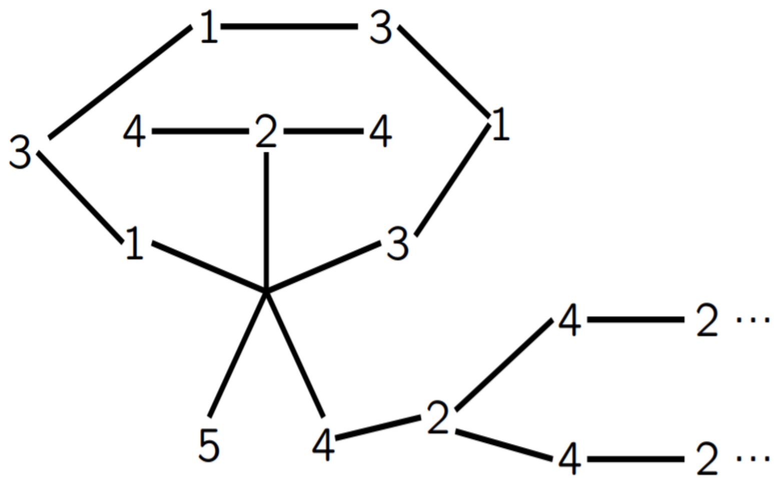
  Proof (by induction on the number of vertices)

  On the other hand, if the vertices colored 1 and 3 are connected via a path in the subgraph, we do the the same for the vertices colored 2 and 4. Note that this will be a disconnected pair of subgraphs, separated by a path connecting the vertices colored 1 and 3 (Why?)

- **Theorem** (Five Color Theorem) The chromatic number of a planar graph is no greater than five.

  Proof (by induction on the number of vertices)

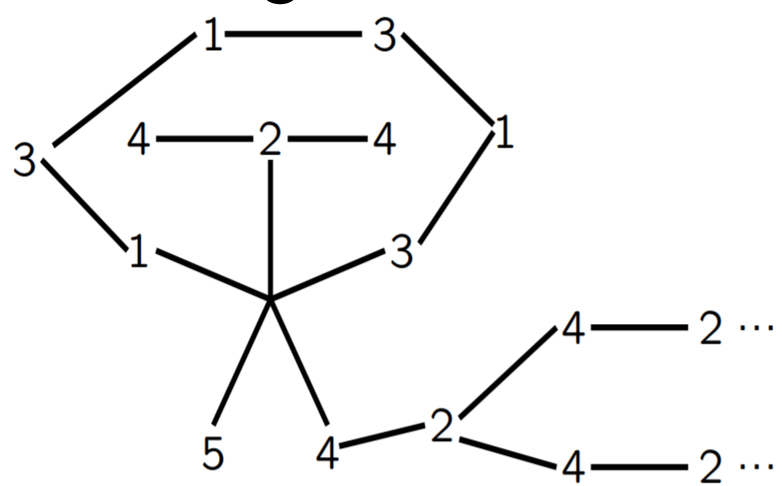
  On the other hand, if the vertices colored 1 and 3 are connected via a path in the subgraph, we do the the same for the vertices colored 2 and 4. Note that this will be a disconnected pair of subgraphs, separated by a path connecting the vertices colored 1 and 3 (Why?)

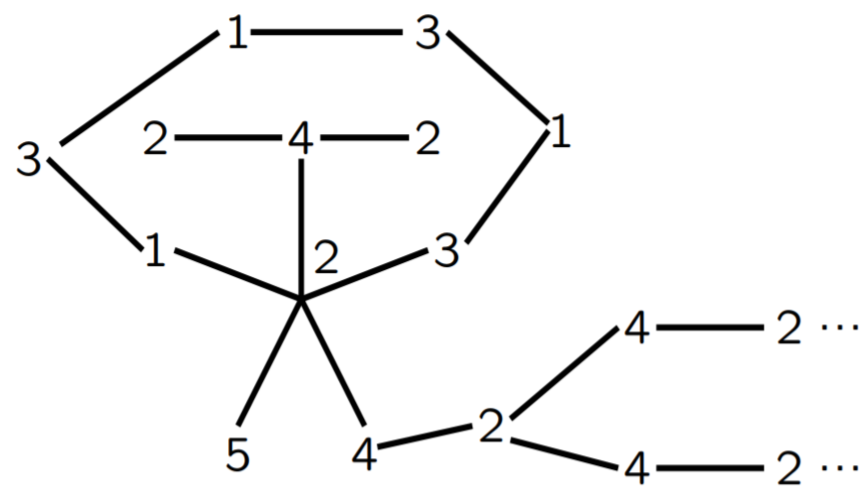


- **Theorem** (Five Color Theorem) The chromatic number of a planar graph is no greater than five.

  Proof (by induction on the number of vertices)

  On the other hand, if the vertices colored 1 and 3 are connected via a path in the subgraph, we do the the same for the vertices colored 2 and 4. Note that this will be a disconnected pair of subgraphs, separated by a path connecting the vertices colored 1 and 3 (Why?)



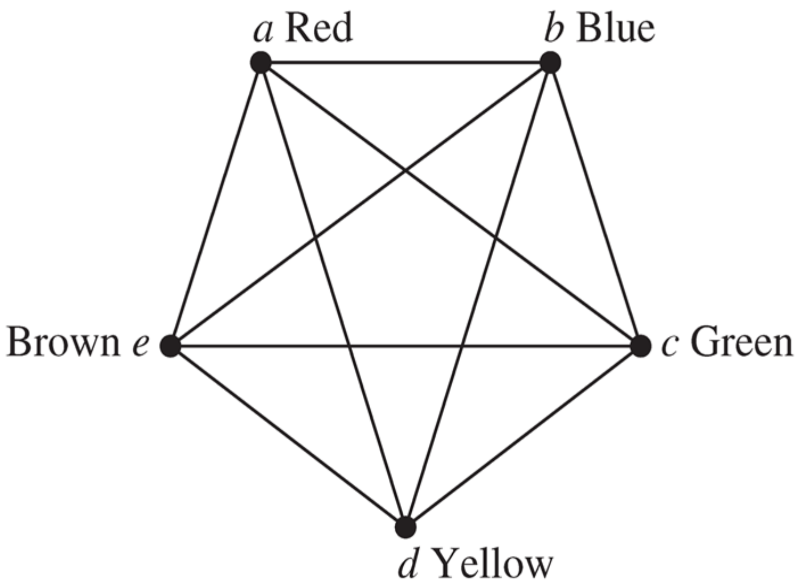


Examples

- What is the chromatic number of K_n, $K_{m,n}$, C_n?

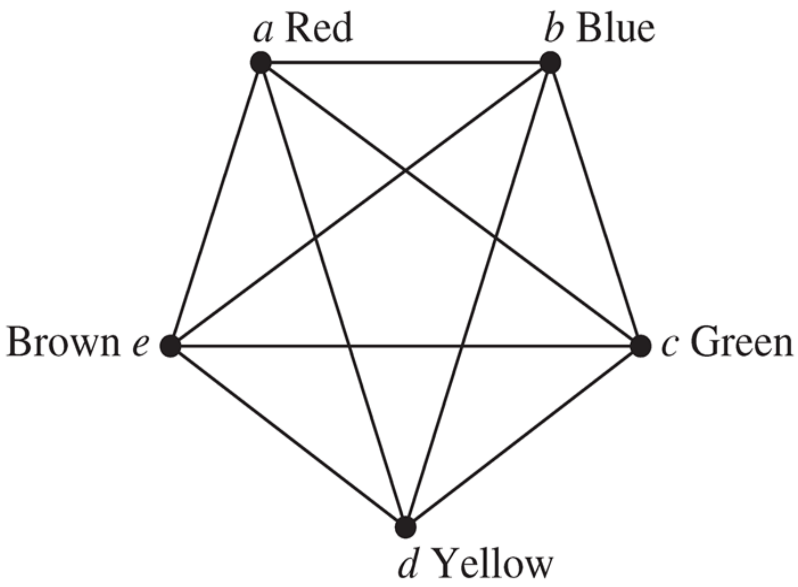
Examples

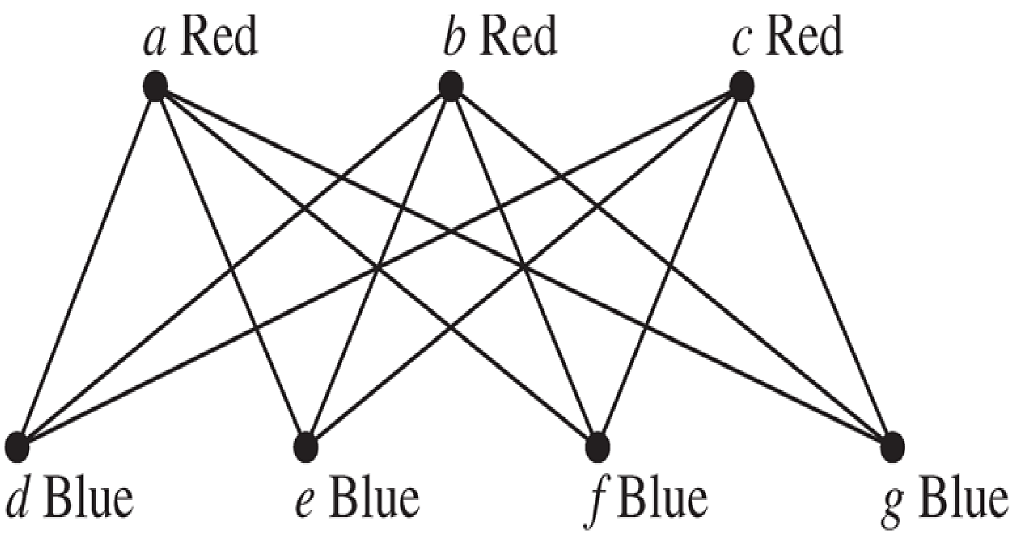
- What is the chromatic number of K_n, $K_{m,n}$, C_n?



Examples

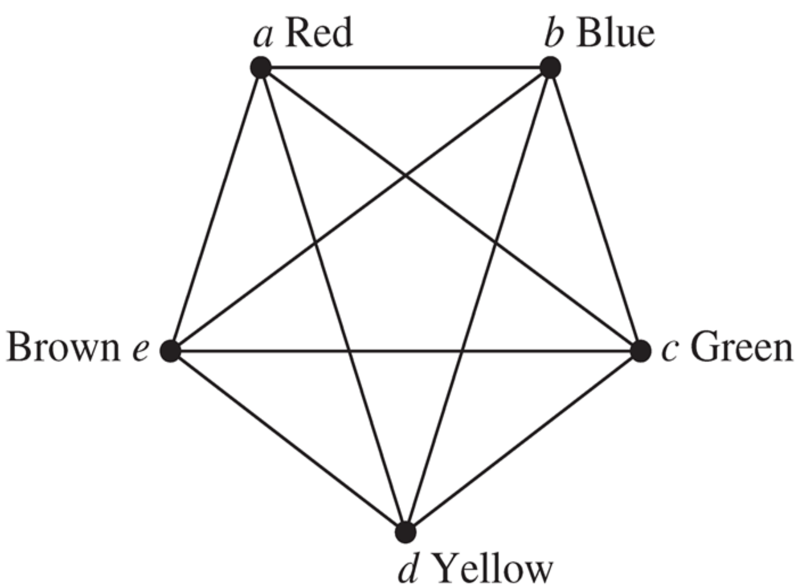
- What is the chromatic number of K_n, $K_{m,n}$, C_n?

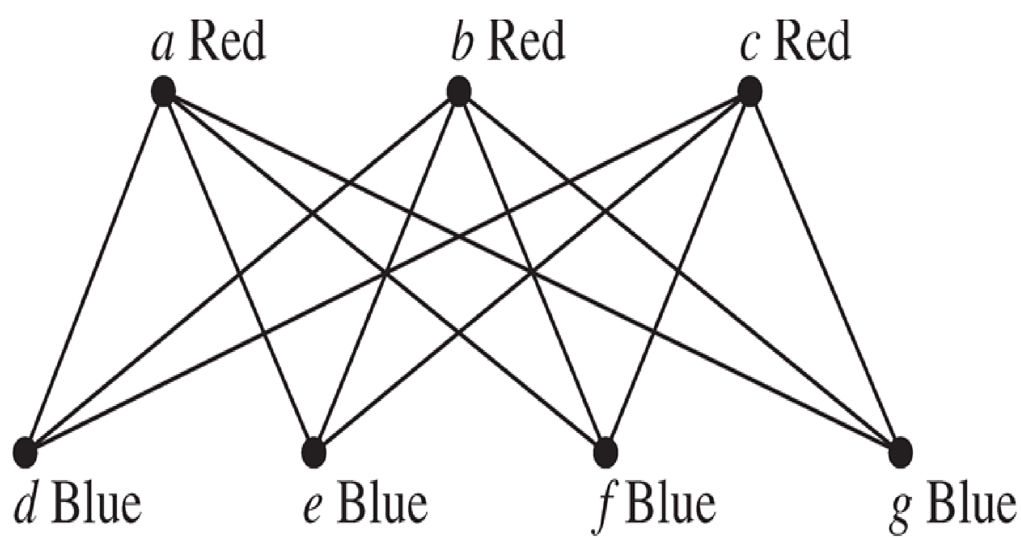


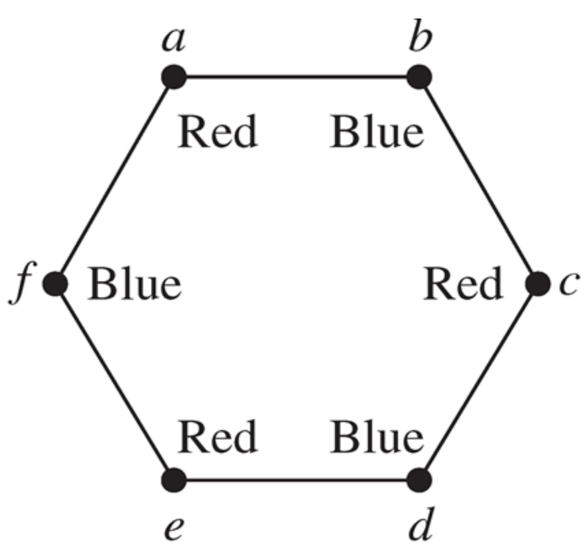


Examples

- What is the chromatic number of K_n, $K_{m,n}$, C_n?

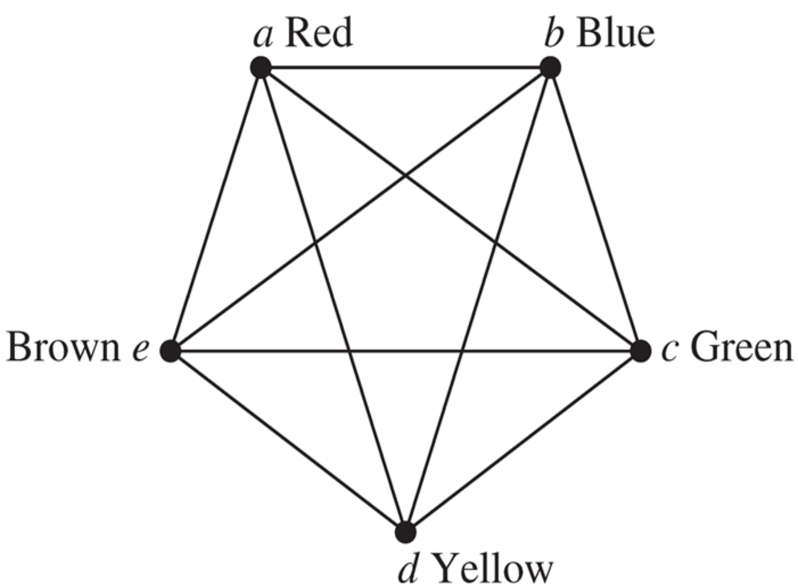


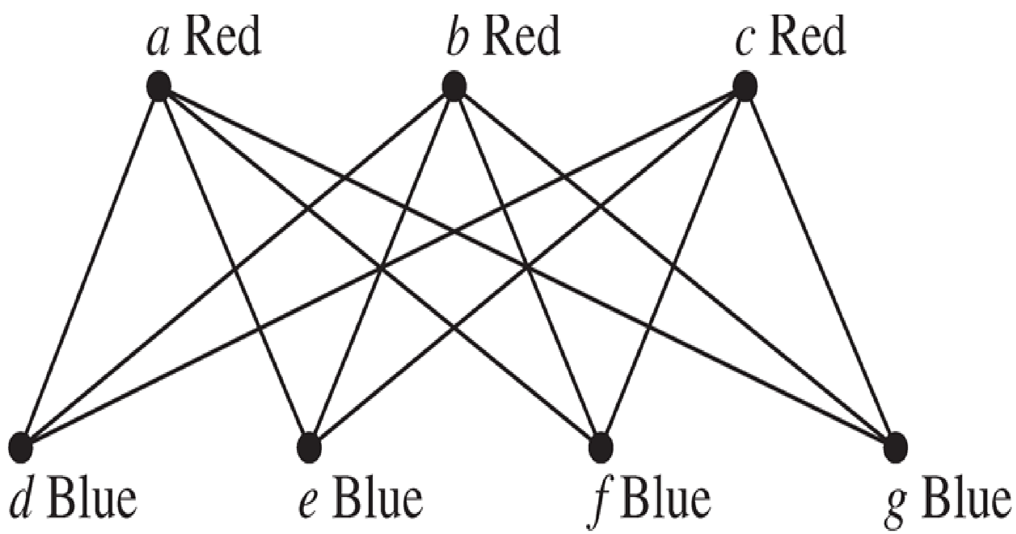


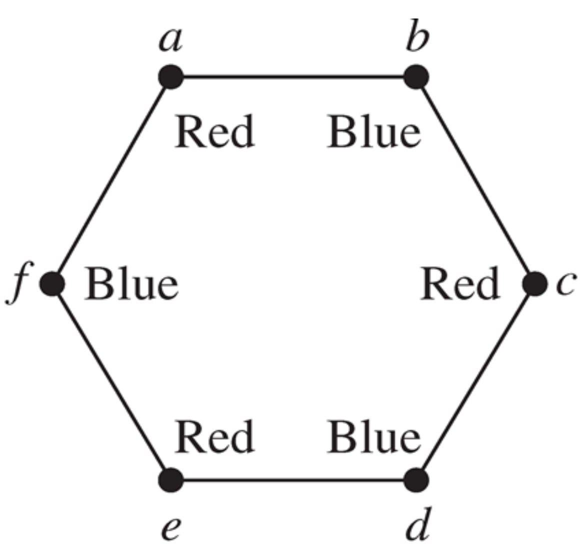


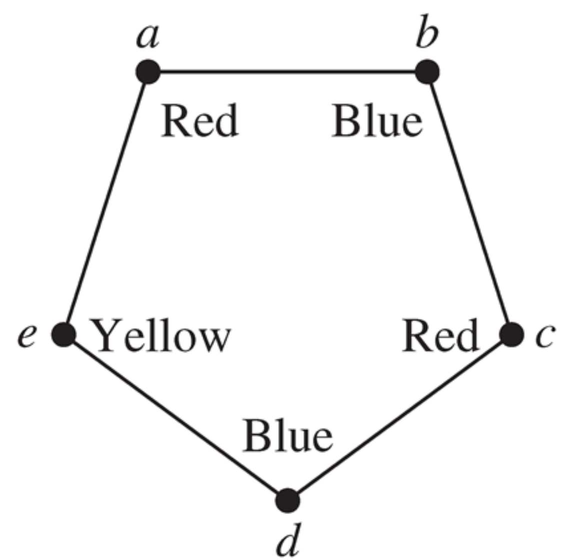
Examples

- What is the chromatic number of K_n, $K_{m,n}$, C_n?





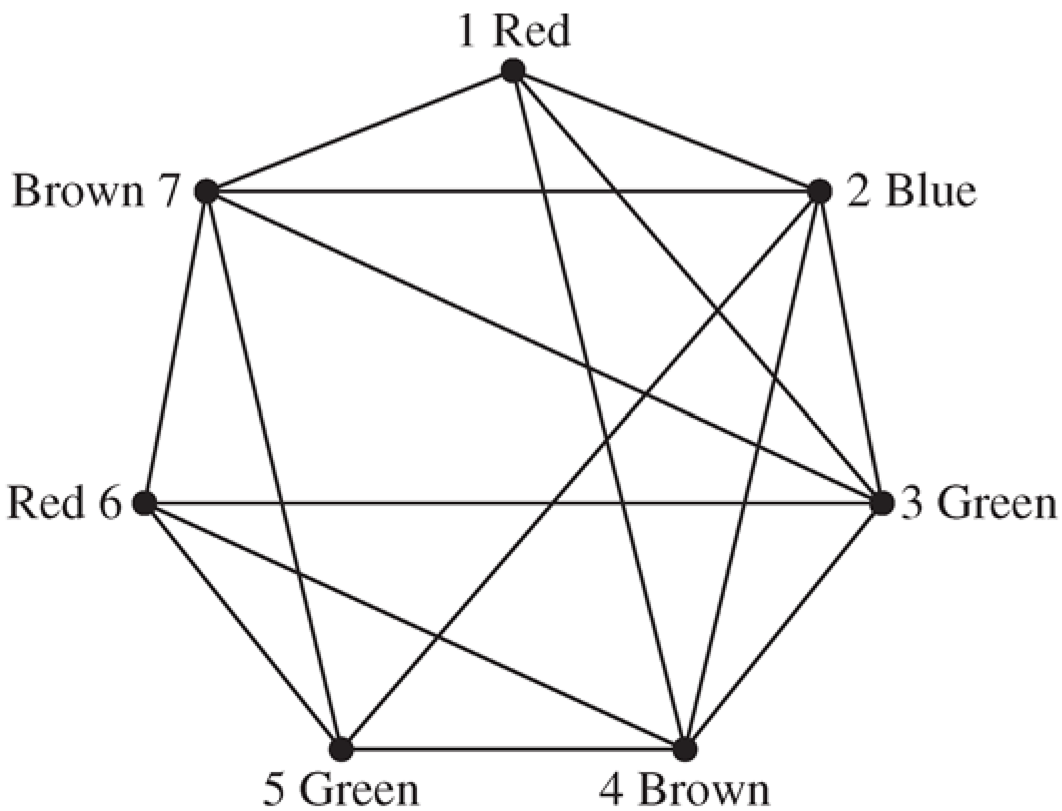




Applications of Graph Coloring

- Scheduling Final Exams

Vertices represent courses, and there is an edge between two vertices if there is a common student in the courses.



Time Period	Courses
I	1, 6
II	2
III	3, 5
IV	4, 7

Applications of Graph Coloring

- Channel Assignments

  Television channels 2 through 13 are assigned to stations in North America so that no two stations within 150 miles can operate on the same channel . How can the assignment of channels be modeled by graph coloring?

Applications of Graph Coloring

- Channel Assignments

  Television channels 2 through 13 are assigned to stations in North America so that no two stations within 150 miles can operate on the same channel . How can the assignment of channels be modeled by graph coloring?

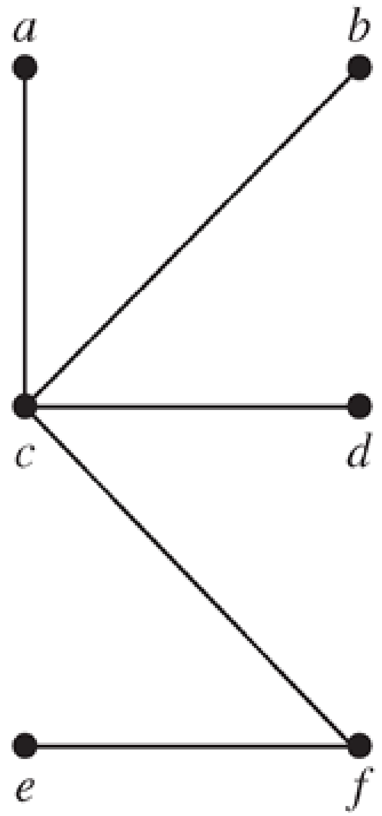
Graph Coloring $\in$ NPC

Trees

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

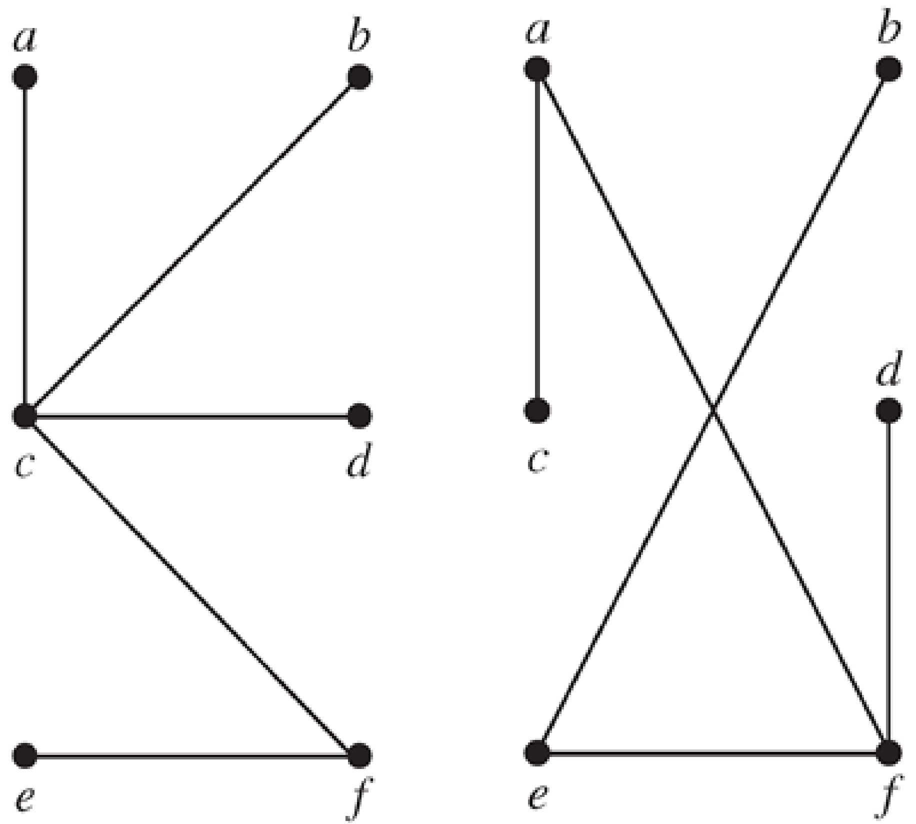
Trees

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

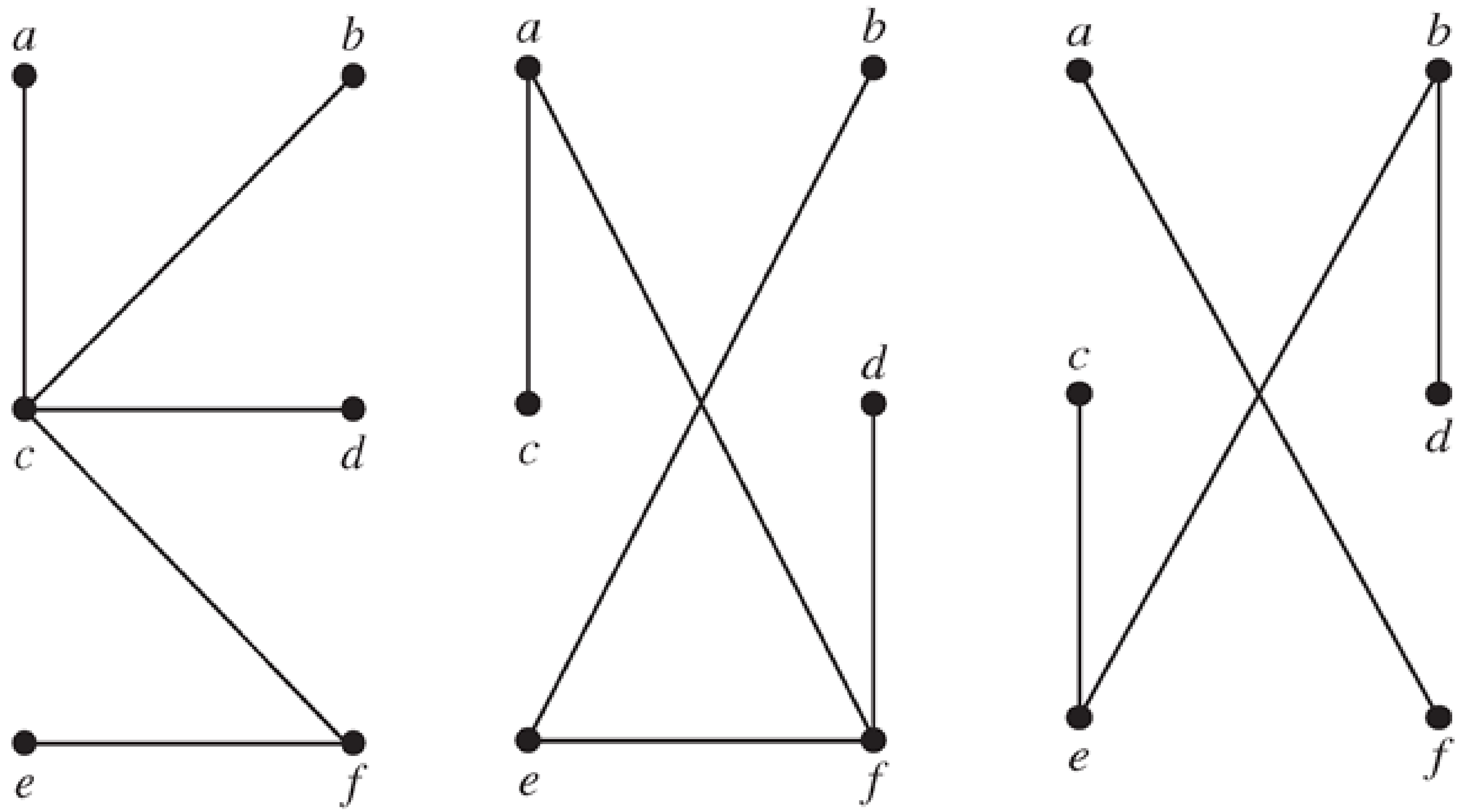


Trees

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

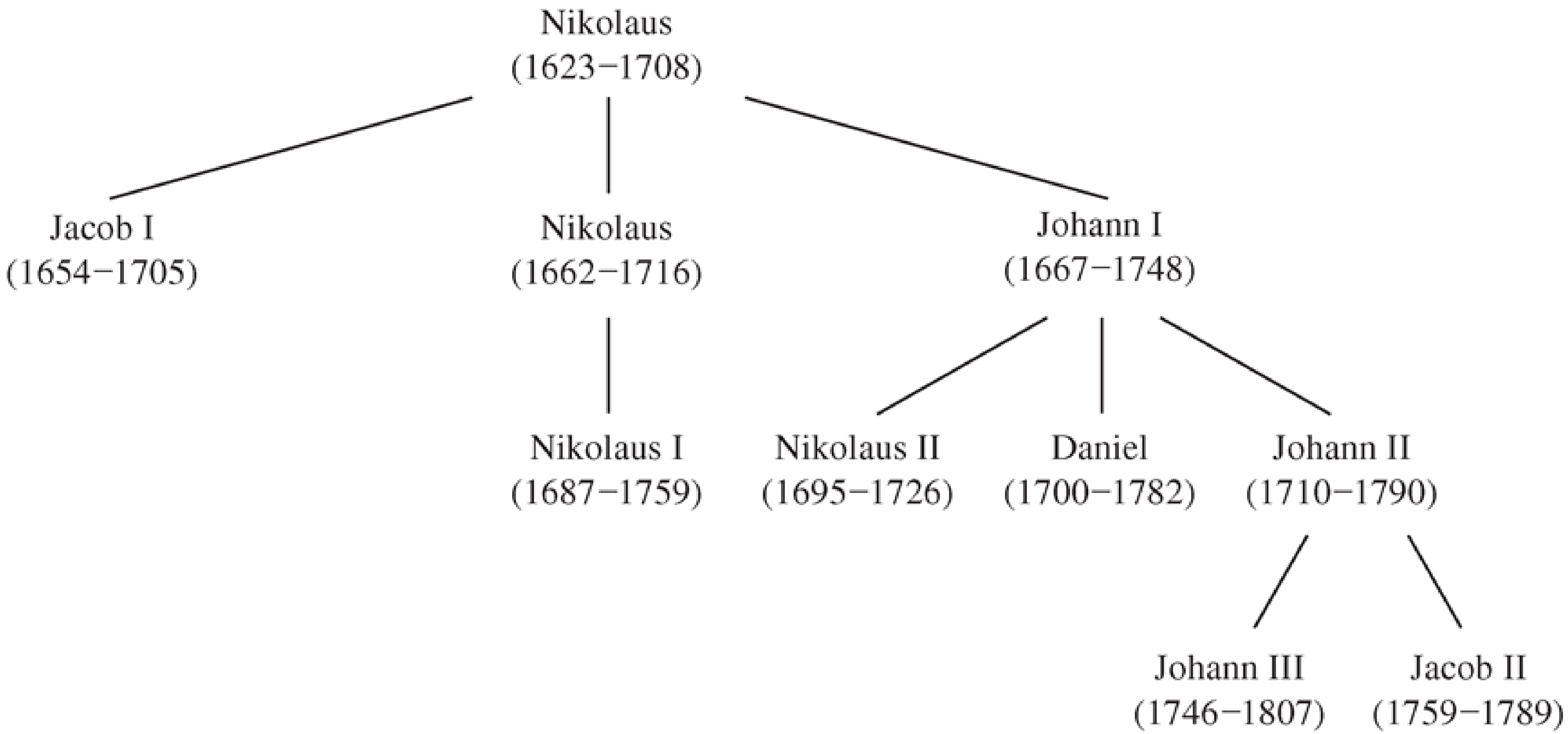


- **Definition** A *tree* is a connected undirected graph with no simple circuits.



Trees

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



Trees

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

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

  Proof

Trees

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

Proof

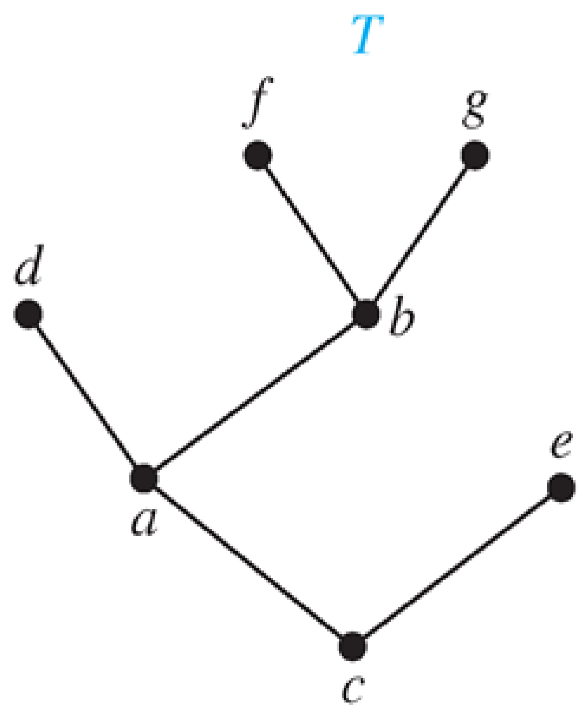
Two properties of tree: connected, no circuit

Rooted Trees

- **Definition** 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.

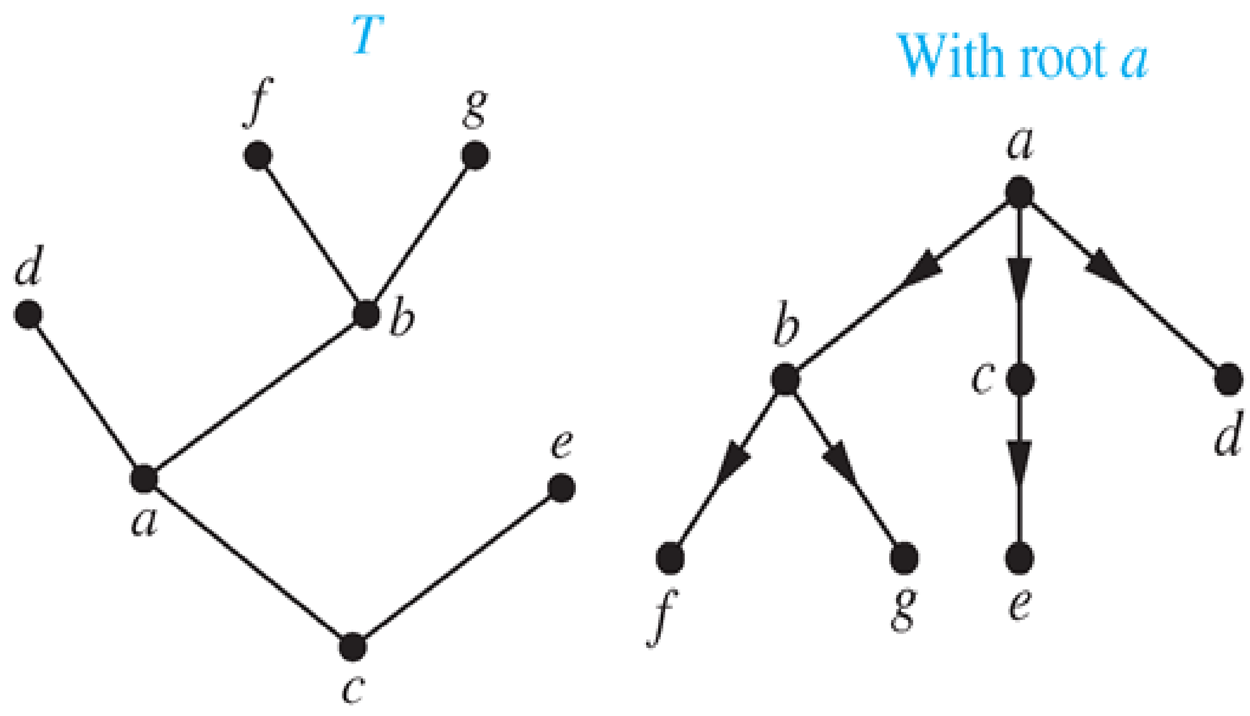
Rooted Trees

- **Definition** 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.



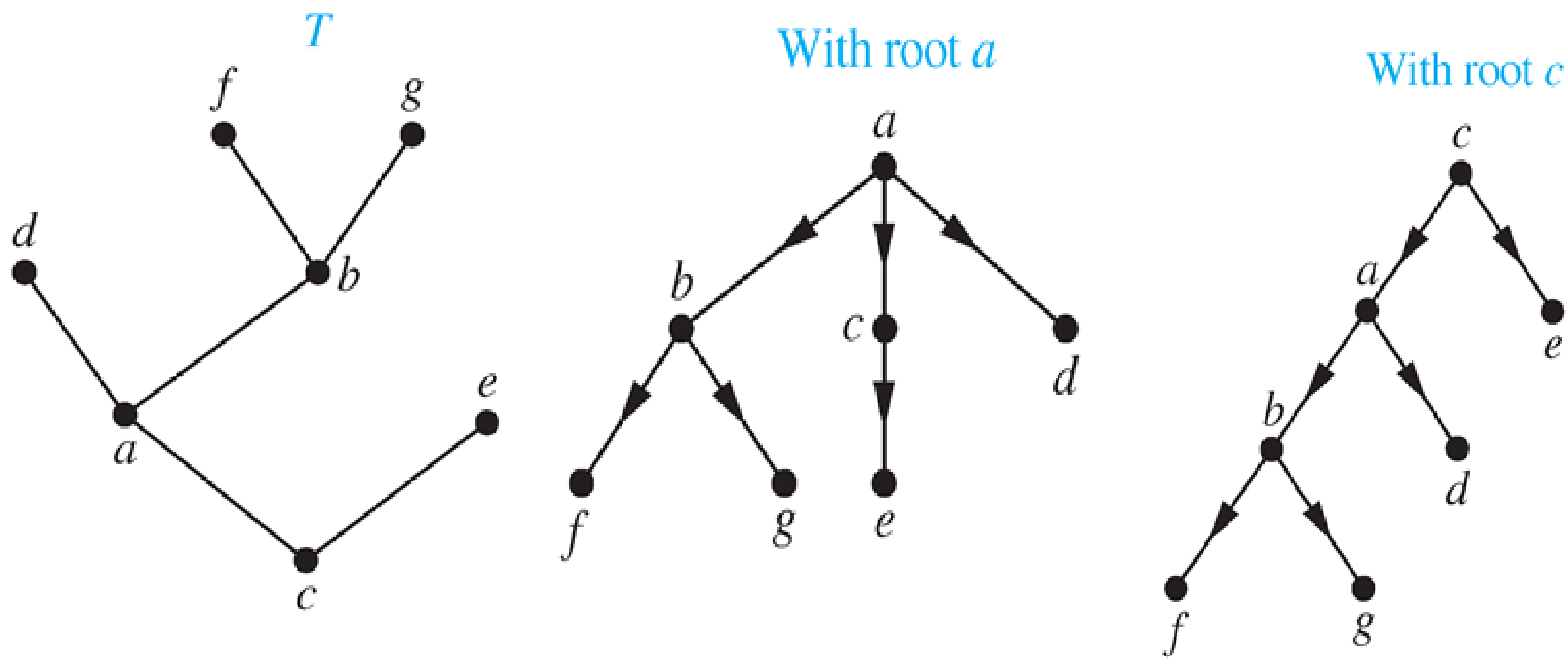
Rooted Trees

- **Definition** 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.



Rooted Trees

- **Definition** 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.



Rooted Trees

- *parent, child, sibling*

Rooted Trees

- *parent, child, sibling*

  ancestor, descendant

Rooted Trees

- *parent, child, sibling*

  ancestor, descendant

  leaf, internal vertex

Rooted Trees

- *parent, child, sibling*

  ancestor, descendant

  leaf, internal vertex

  subtree with a as its root: consists of a and its descendants and all edges incident to these descendants

m-Ary Trees

- **Definition** 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. In particular, an m-ary tree with $m = 2$ is called a *binary tree*.

m-Ary Trees

- **Definition** 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. In particular, an m-ary tree with $m = 2$ is called a *binary tree*.

  Definition A binary tree is an *ordered rooted tree* where the children of each internal vertex are ordered. In a binary tree, the first child is called the *left child*, and the second child is called the *right child*.

m-Ary Trees

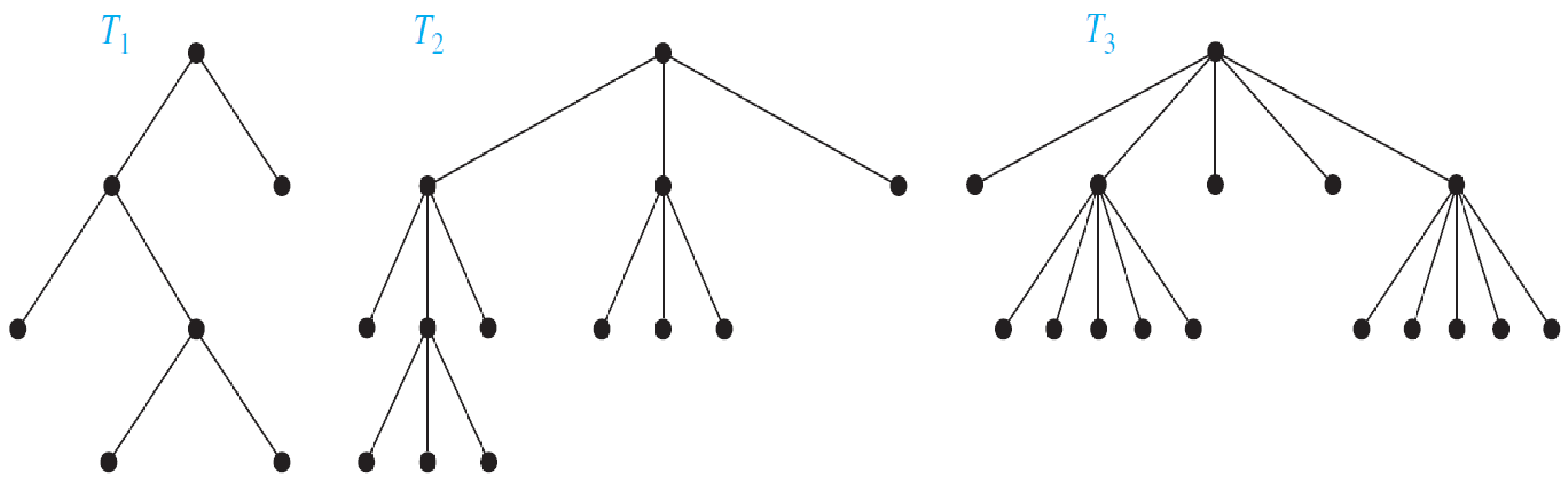
- **Definition** A rooted tree is called an *m-ary tree* if every internal vertex has no more than m children. The tree is called a *full m-ary tree* if every internal vertex has exactly m children. In particular, an m-ary tree with $m = 2$ is called a *binary tree*.

  Definition A binary tree is an *ordered rooted tree* where the children of each internal vertex are ordered. In a binary tree, the first child is called the *left child*, and the second child is called the *right child*.

  left subtree, *right subtree*

Full *m*-Ary Trees

- **Definition** 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. In particular, an m-ary tree with $m = 2$ is called a *binary tree*.



Counting Vertices in a Full m-Ary Trees

- **Theorem** A full m-ary tree with i internal vertices has $n = mi + 1$ vertices.

Counting Vertices in a Full m-Ary Trees

- **Theorem** A full m-ary tree with i internal vertices has $n = mi + 1$ vertices.

  Theorem A full m-ary tree with

  (i) n vertices
  (ii) i internal vertices
  (iii) ℓ leaves

Counting Vertices in a Full m-Ary Trees

- **Theorem** A full m-ary tree with i internal vertices has $n = mi + 1$ vertices.

  Theorem A full m-ary tree with

  (i) n vertices
  (ii) i internal vertices
  (iii) ℓ leaves

  (i) n vertices, $i = (n-1)/m$, $\ell = [(m-1)n+1)]/m$
  (ii) i internal vertices, $n = mi+1$, $\ell = (m-1)i+1$
  (iii) ℓ leaves, $n = (m\ell-1)/(m-1)$, $i = (\ell-1)/(m-1)$

Counting Vertices in a Full m-Ary Trees

- **Theorem** A full m-ary tree with i internal vertices has $n = mi + 1$ vertices.

  Theorem A full m-ary tree with

  (i) n vertices
  (ii) i internal vertices
  (iii) ℓ leaves

  (i) n vertices, $i = (n-1)/m$, $\ell = [(m-1)n+1)]/m$
  (ii) i internal vertices, $n = mi + 1$, $\ell = (m-1)i + 1$
  (iii) ℓ leaves, $n = (m\ell - 1)/(m-1)$, $i = (\ell - 1)/(m-1)$

  using $n = mi + 1$ and $n = i + \ell$

Level and Height

- The *level* of a vertex v in a rooted tree is the length of the unique path from the root to this vertex.

Level and Height

- The *level* of a vertex v in a rooted tree is the length of the unique path from the root to this vertex.

  The *height* of a rooted tree is the maximum of the levels of the vertices.

Level and Height

- The *level* of a vertex v in a rooted tree is the length of the unique path from the root to this vertex.

  The *height* of a rooted tree is the maximum of the levels of the vertices.

  Definition A rooted m-ary tree of height h is *balanced* if all leaves are at levels h or $h-1$. (differ no greater than 1)

The Number of Leaves

- **Theorem** There are at most m^h leaves in an m-ary tree of height h.

The Number of Leaves

- **Theorem** There are at most m^h leaves in an m-ary tree of height h.

  Proof (by induction)

The Number of Leaves

- **Theorem** There are at most m^h leaves in an m-ary tree of height h.

  Proof (by induction)

  Corollary If an m-ary tree of height h has ℓ leaves, then $h \geq \lceil \log_m \ell \rceil$. If the m-ary tree is full and balanced, then $h = \lceil \log_m \ell \rceil$.

The Number of Leaves

- **Theorem** There are at most m^h leaves in an m-ary tree of height h.

  Proof (by induction)

  Corollary If an m-ary tree of height h has ℓ leaves, then $h \geq \lceil \log_m \ell \rceil$. If the m-ary tree is full and balanced, then $h = \lceil \log_m \ell \rceil$.

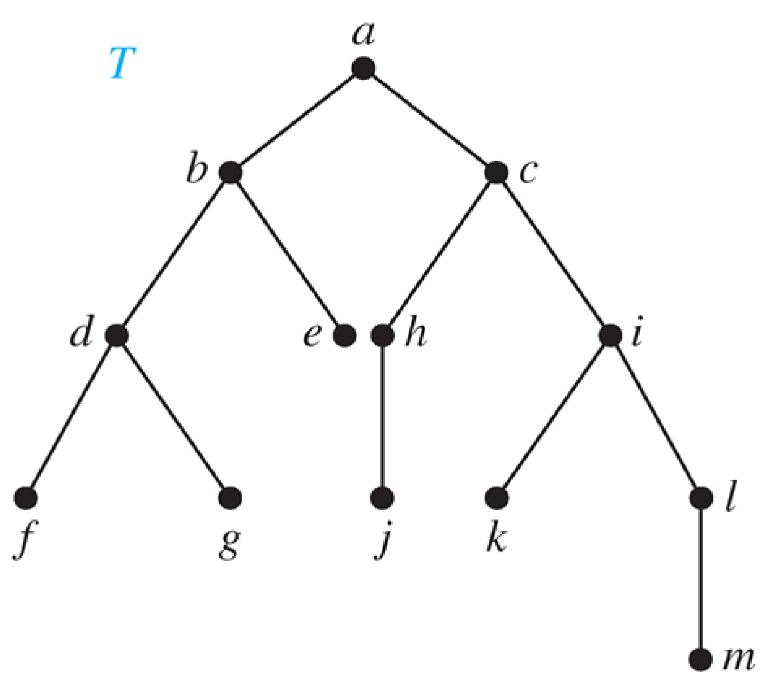
  Proof

Binary Trees

- **Definition** A *binary tree* is an ordered rooted tree where each internal tree has two children, the first is called the *left child* and the second is the *right child*. The tree rooted at the left child of a vertex is called the *left subtree* of this vertex, and the tree rooted at the right child of a vertex is called the *right subtree* of this vertex.

Binary Trees

- **Definition** A *binary tree* is an ordered rooted tree where each internal tree has two children, the first is called the *left child* and the second is the *right child*. The tree rooted at the left child of a vertex is called the *left subtree* of this vertex, and the tree rooted at the right child of a vertex is called the *right subtree* of this vertex.



Tree Traversal

- The procedures for systematically visiting every vertex of an ordered tree are called *traversals*.

Tree Traversal

- The procedures for systematically visiting every vertex of an ordered tree are called *traversals*.

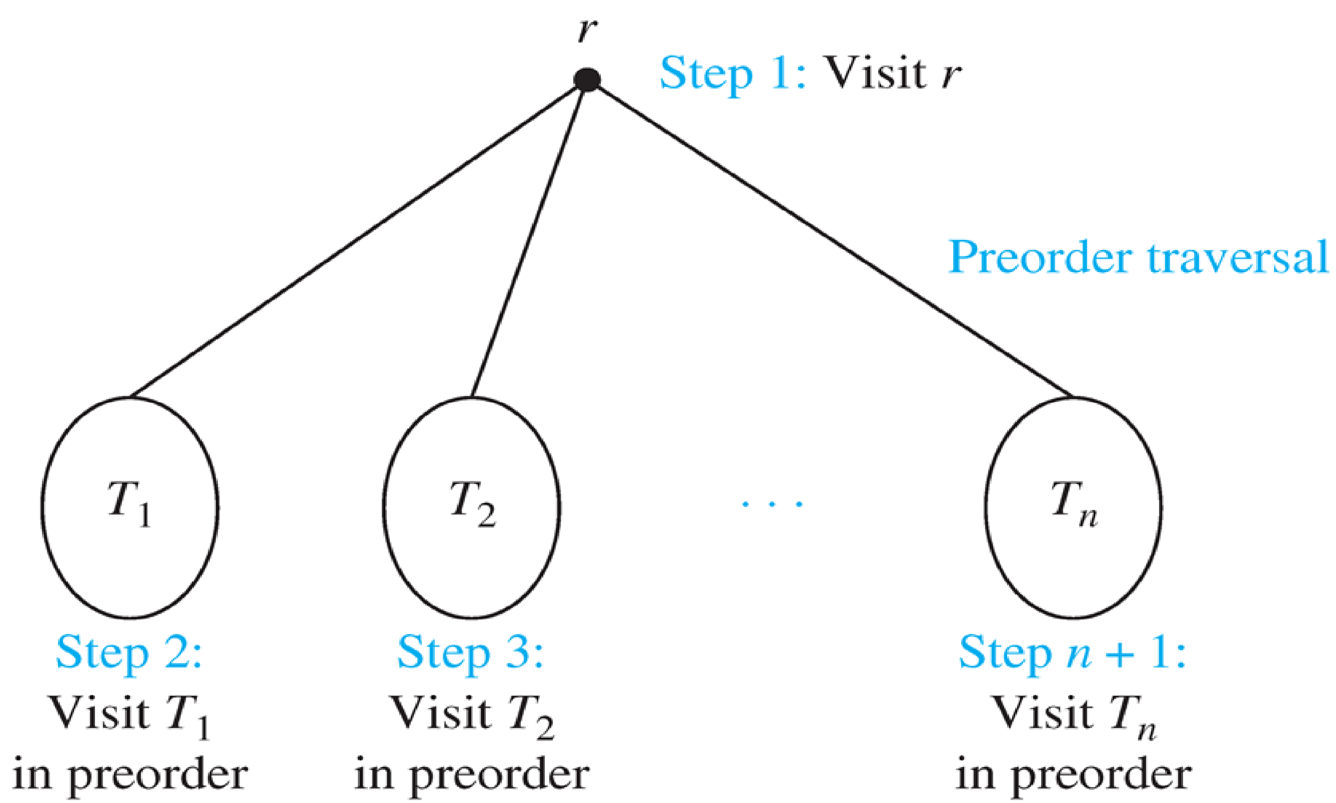
  The three most commonly used traversals are *preorder traversal*, *inorder traversal*, *postorder traversal*.

Preorder Traversal

- **Definition** 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 of r from left to right in T. The *preorder traversal* begins by visiting r, and continues by traversing T_1 in preorder, then T_2 in preorder, and so on, until T_n is traversed in preorder.

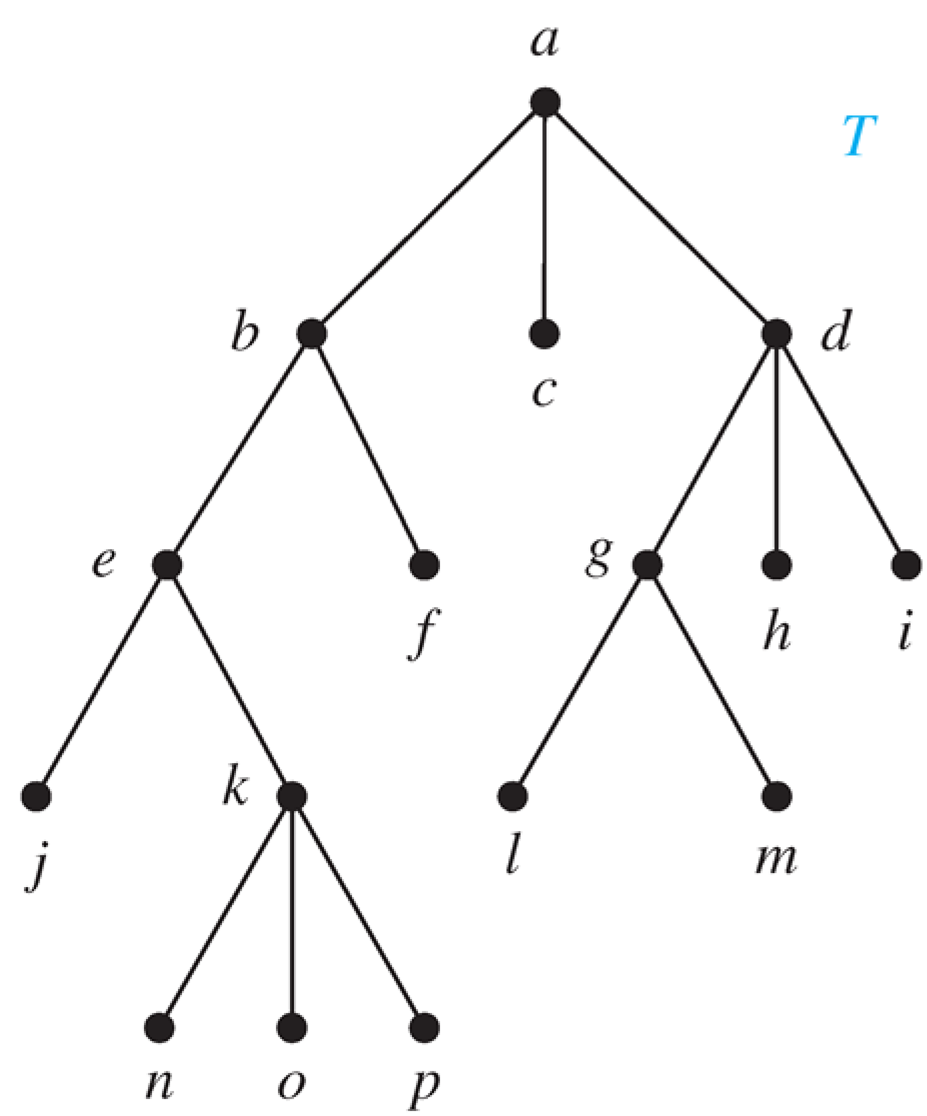
Preorder Traversal

- **Definition** 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 of r from left to right in T. The *preorder traversal* begins by visiting r, and continues by traversing T_1 in preorder, then T_2 in preorder, and so on, until T_n is traversed in preorder.



Preorder Traversal

- **Example**



Preorder Traversal

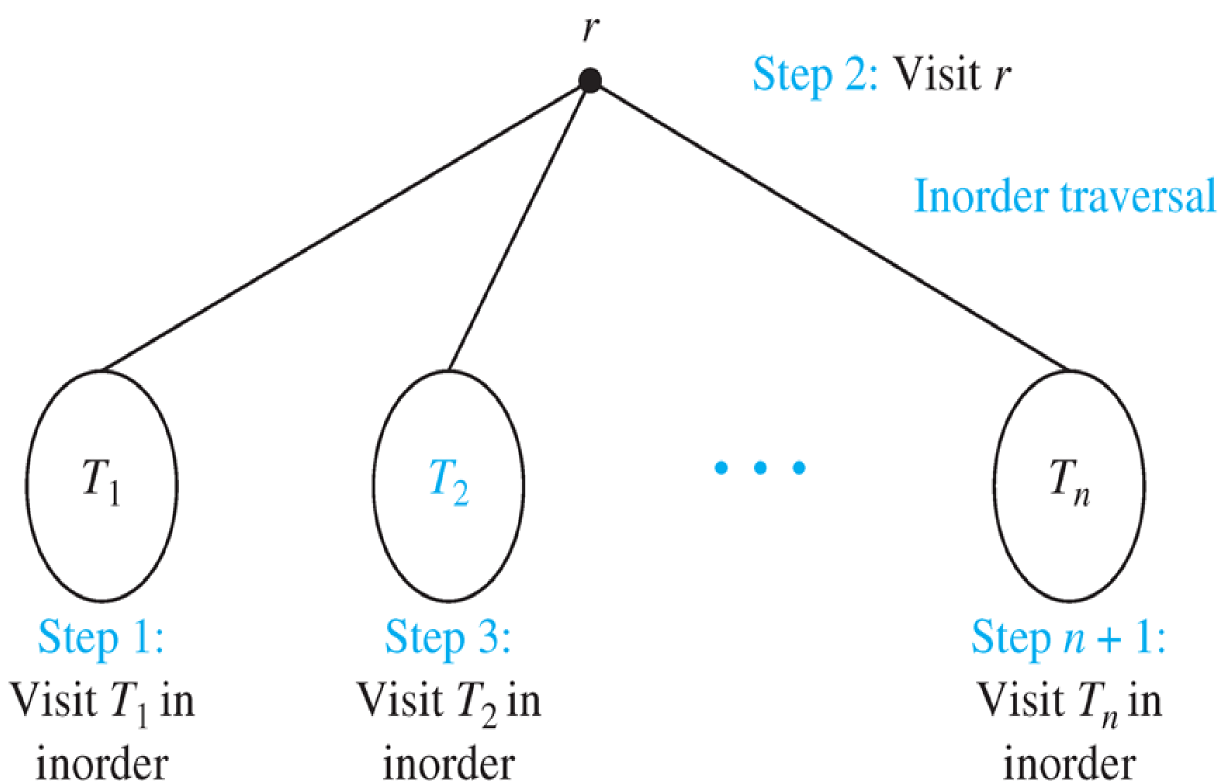
procedure *preorder* (T: ordered rooted tree)
r := root of T
list r
for each child c of r from left to right
    $T(c)$:= subtree with c as root
    preorder($T(c)$)

Inorder Traversal

- **Definition** 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 of r from left to right in T. The *inorder traversal* begins by traversing T_1 in inorder, then visiting r, and continues by traversing T_2 in inorder, and so on, until T_n is traversed in inorder.

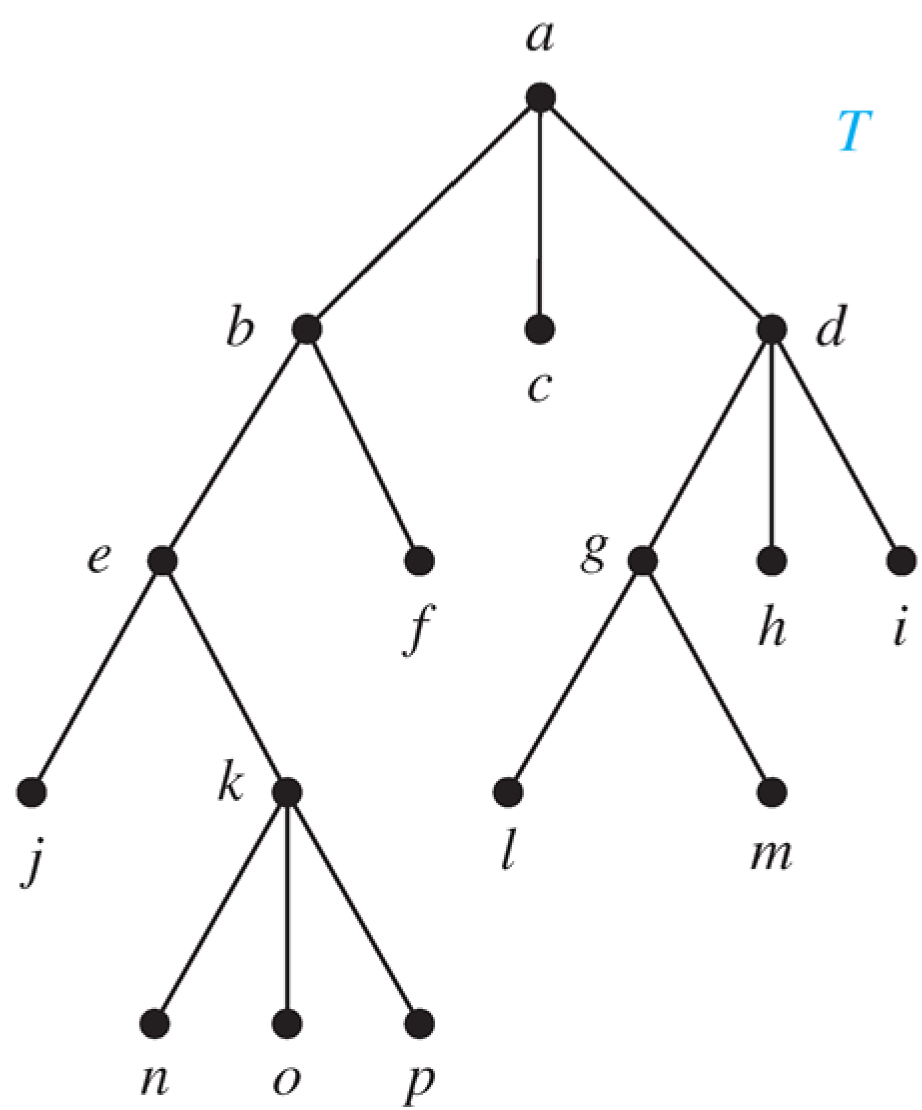
Inorder Traversal

- **Definition** 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 of r from left to right in T. The *inorder traversal* begins by traversing T_1 in inorder, then visiting r, and continues by traversing T_2 in inorder, and so on, until T_n is traversed in inorder.



Inorder Traversal

- **Example**



Inorder Traversal

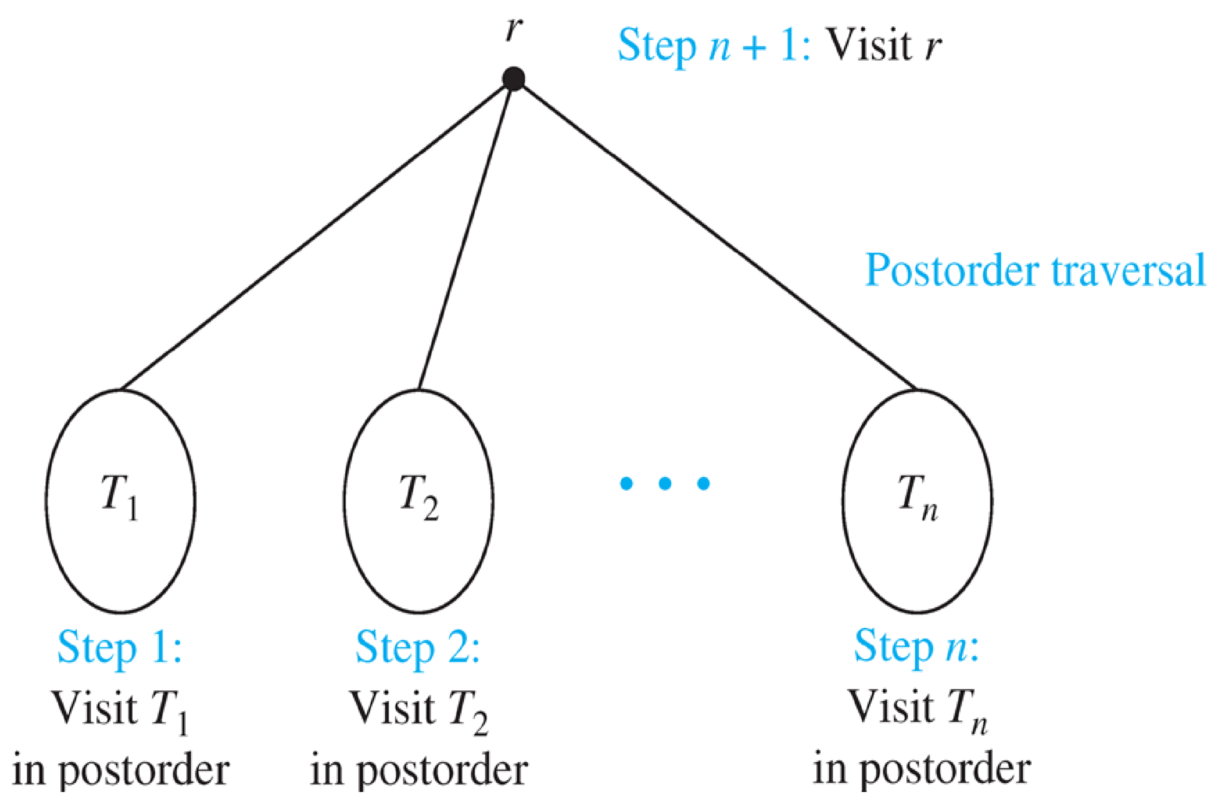
```
procedure inorder (T: ordered rooted tree)
r := root of T
if r is a leaf then list r
else
    l := first child of r from left to right
    T(l) := subtree with l as its root
    inorder(T(l))
    list(r)
    for each child c of r from left to right
        T(c) := subtree with c as root
        inorder(T(c))
```

Postorder Traversal

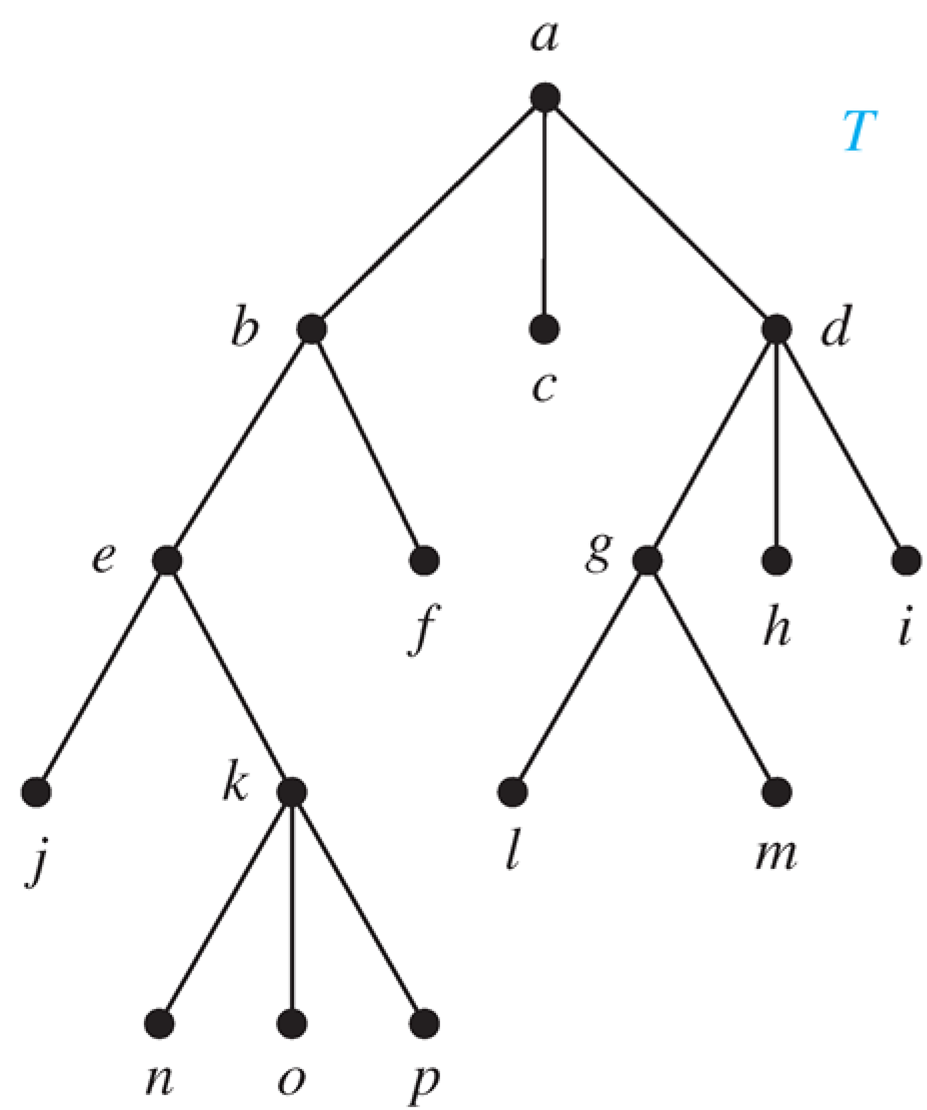
- **Definition** Let T be an ordered rooted tree with root r. If T consists only of r, then r is the *postorder traversal* of T. Otherwise, suppose that $T_1, T_2, \ldots, T_n$ are the subtrees of r from left to right in T. The *postorder traversal* begins by traversing T_1 in postorder, then T_2 in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

Postorder Traversal

- **Definition** 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 of r from left to right in T. The *postorder traversal* begins by traversing T_1 in postorder, then T_2 in postorder, and so on, after T_n is traversed in postorder, r is visited.



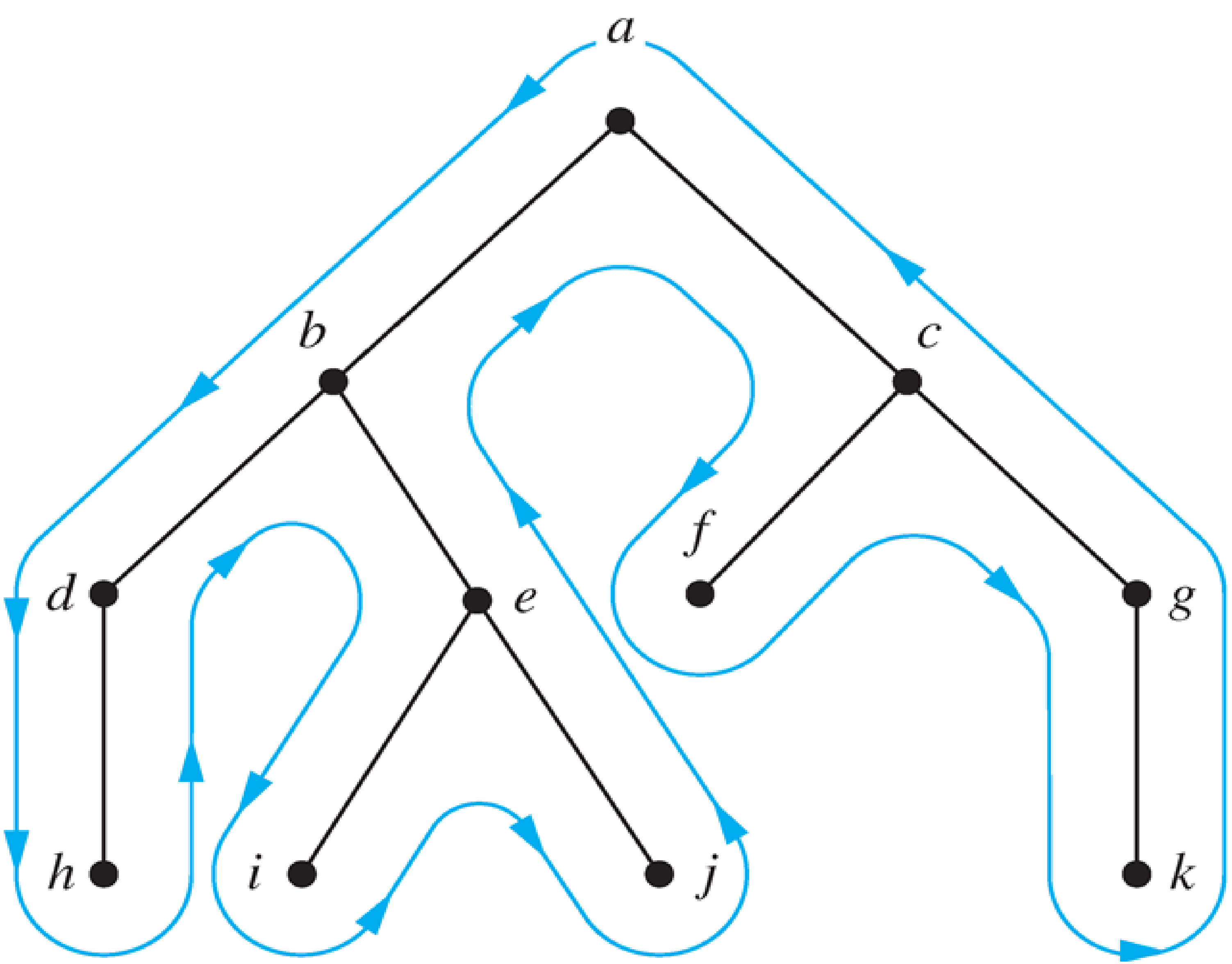
- **Example**



Postorder Traversal

procedure *postordered* (T: ordered rooted tree)
r := root of T
for each child c of r from left to right
    $T(c)$:= subtree with c as root
    postorder($T(c)$)
list r

Preorder, Inorder, Postorder Traversal



Expression Trees

- Complex expressions can be represented using ordered rooted trees

Expression Trees

- Complex expressions can be represented using ordered rooted trees

Example

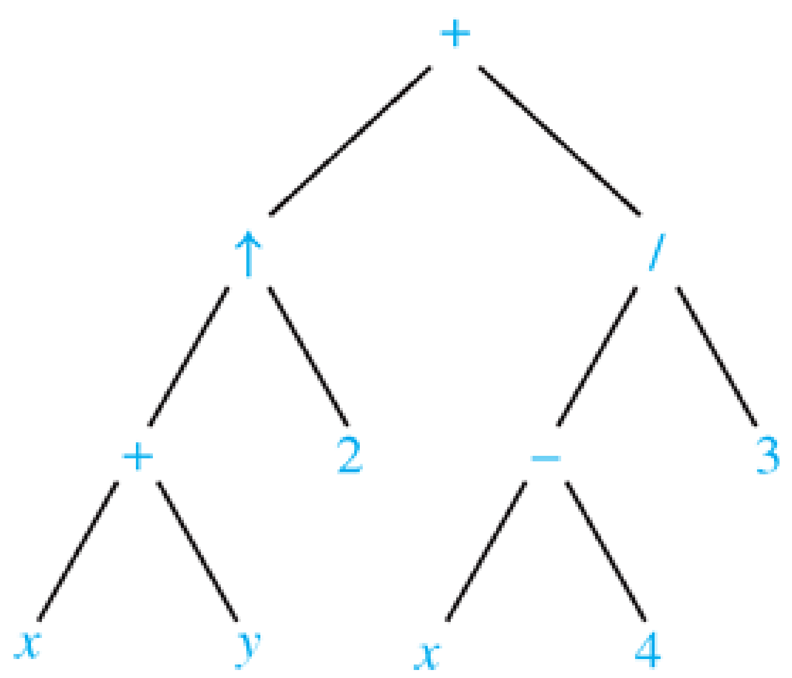
consider the expression $((x + y) \uparrow 2) + ((x - 4)/3)$

Expression Trees

- Complex expressions can be represented using ordered rooted trees

Example

consider the expression $((x + y) \uparrow 2) + ((x - 4)/3)$



Infix Notation

- An inorder traversal of the tree representing an expression produces the original expression when parentheses are included except for unary operation.

Infix Notation

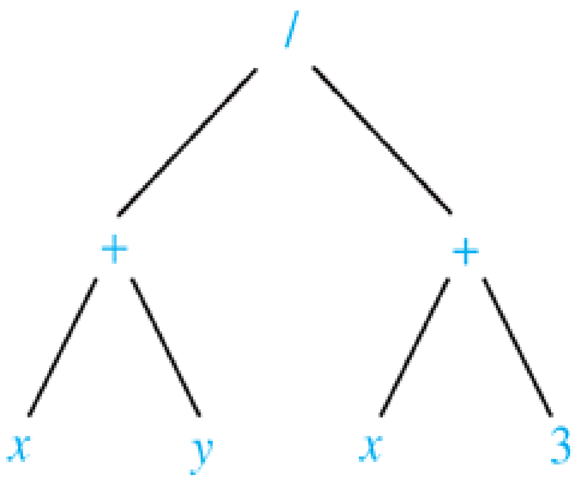
- An inorder traversal of the tree representing an expression produces the original expression when parentheses are included except for unary operation.

  Why parentheses are needed?

Infix Notation

- An inorder traversal of the tree representing an expression produces the original expression when parentheses are included except for unary operation.

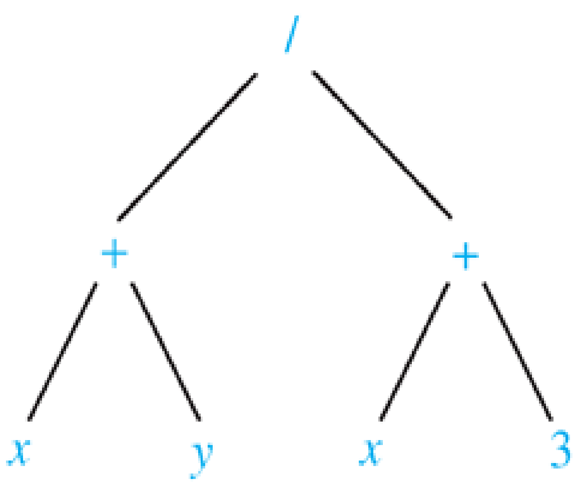
  Why parentheses are needed?

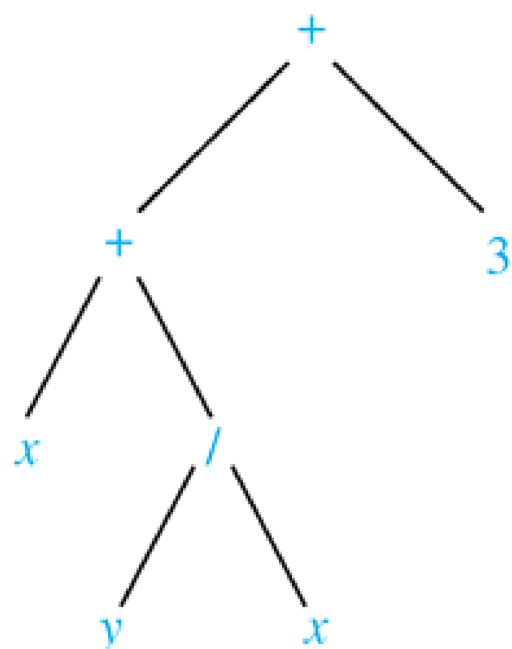


Infix Notation

- An inorder traversal of the tree representing an expression produces the original expression when parentheses are included except for unary operation.

  Why parentheses are needed?

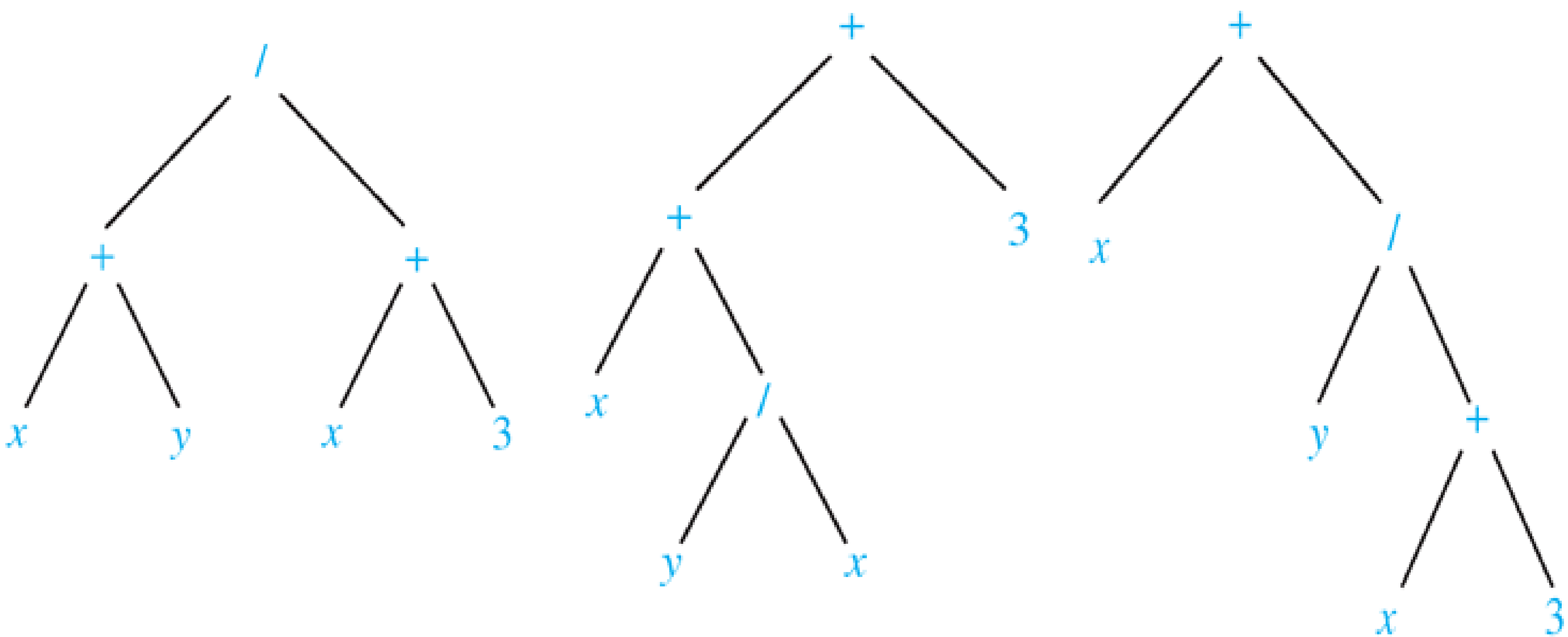




Infix Notation

- An inorder traversal of the tree representing an expression produces the original expression when parentheses are included except for unary operation.

  Why parentheses are needed?



Prefix Notation

- The preorder traversal of expression trees leads to the *prefix form* of the expression (*Polish notation*).

Prefix Notation

- The preorder traversal of expression trees leads to the *prefix form* of the expression (*Polish notation*).

  Operators precede their operands in the prefix notation. Parentheses are not needed as the representation is unambiguous.

Prefix Notation

- The preorder traversal of expression trees leads to the *prefix form* of the expression (*Polish notation*).

  Operators precede their operands in the prefix notation. Parentheses are not needed as the representation is unambiguous.

  Prefix expressions are evaluated by working from right to left. When we encounter an operator, we perform the operation with the two operands to the right.

Prefix Notation

- **Example**

  $+ \ - \ * \, 2 \ 3 \ 5 \ / \ \uparrow \ 2 \ 3 \ 4$

Prefix Notation

- **Example**

  $+ \ - \ * \ 2 \ 3 \ 5 \ / \ \uparrow \ 2 \ 3 \ 4$

$$+ \quad - \quad * \quad 2 \quad 3 \quad 5 \quad / \quad \underbrace{\uparrow \quad 2 \quad 3}_{2 \uparrow 3 = 8} \quad 4$$

$$+ \quad - \quad * \quad 2 \quad 3 \quad 5 \quad \underbrace{/ \quad 8 \quad 4}_{8 / 4 = 2}$$

$$+ \quad - \quad \underbrace{* \quad 2 \quad 3}_{2 * 3 = 6} \quad 5 \quad 2$$

$$+ \quad \underbrace{- \quad 6 \quad 5}_{6 - 5 = 1} \quad 2$$

$$\underbrace{+ \quad 1 \quad 2}_{1 + 2 = 3}$$

Postfix Notation

- The postorder traversal of expression trees leads to the *postfix form* of the expression (*reverse Polish notation*).

Postfix Notation

- The postorder traversal of expression trees leads to the *postfix form* of the expression (*reverse Polish notation*).

  Operators follow their operands in the postfix notation. Parentheses are not needed as the representation is unambiguous.

Postfix Notation

- The postorder traversal of expression trees leads to the *postfix form* of the expression (*reverse Polish notation*).

  Operators follow their operands in the postfix notation. Parentheses are not needed as the representation is unambiguous.

  Postfix expressions are evaluated by working from left to right. When we encounter an operator, we perform the operation with the two operands to the left.

Postfix Notation

- **Example**

  $7\ 2\ 3\ *\ -\ 4\ \uparrow\ 9\ 3\ /\ +$

Postfix Notation

- **Example**

  $7\ 2\ 3\ *\ -\ 4\ \uparrow\ 9\ 3\ /\ +$

$$7\quad \underbrace{2\quad 3\quad *}_{2 * 3 = 6}\quad -\quad 4\quad \uparrow\quad 9\quad 3\quad /\quad +$$

$$\underbrace{7\quad 6\quad -}_{7 - 6 = 1}\quad 4\quad \uparrow\quad 9\quad 3\quad /\quad +$$

$$\underbrace{1\quad 4\quad \uparrow}_{1^4 = 1}\quad 9\quad 3\quad /\quad +$$

$$1\quad \underbrace{9\quad 3\quad /}_{9 / 3 = 3}\quad +$$

$$\underbrace{1\quad 3\quad +}_{1 + 3 = 4}$$

Next Lecture

- tree II ...