Discrete mathematics course
Chapter 3: Trees

Anh Tuan GIANG

ICTLab, ICT Department
University of Science and Technology of Hanoi-USTH

April 17, 2017

Trees definition

Definition 1.1.

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

Trees definition

Definition 1.1.

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

Example: which of the graphs shown in Figure below are trees?

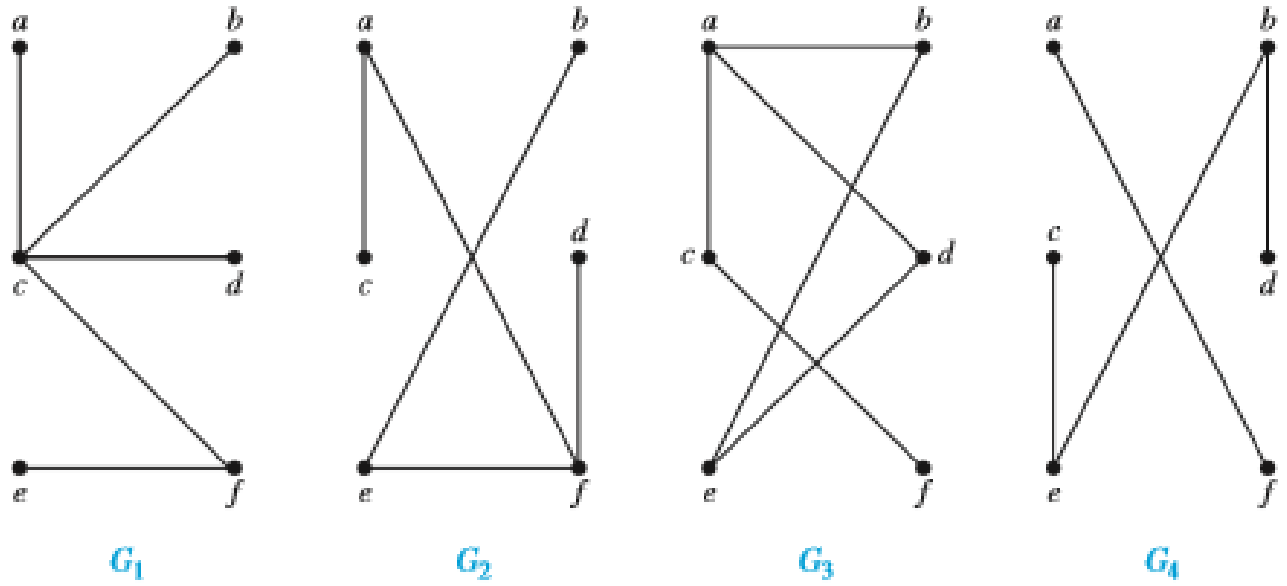


FIGURE 2 Examples of Trees and Graphs That Are Not Trees.

continue...

Theorem 1.2.

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

continue...

Theorem 1.2.

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

Definition 1.3.

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.

continue...

Definition 1.4.

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

continue...

Definition 1.4.

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

Example: are the rooted trees in Figure below full $m-ary$ trees for some positive integer m?

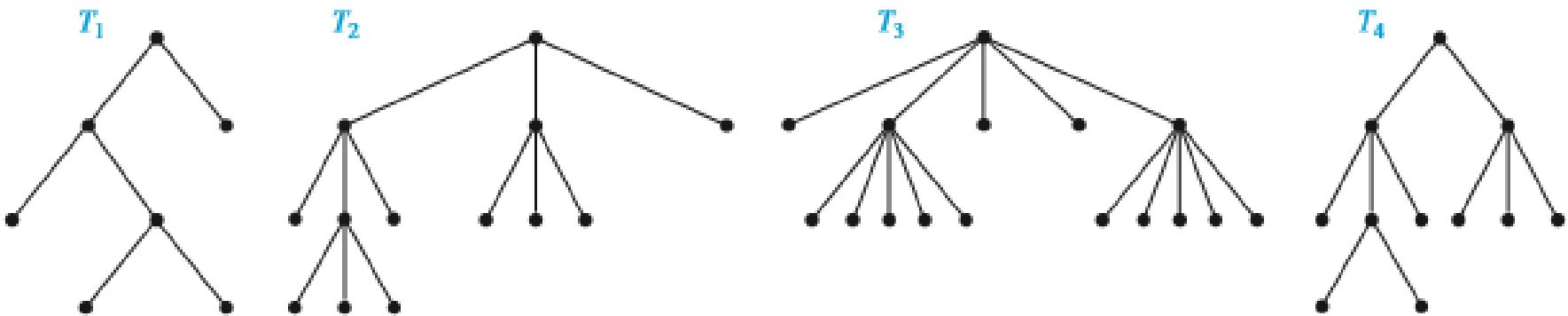


FIGURE 7 Four Rooted Trees.

Trees as a model

- Chemical representation (C_4H_{10})?

Trees as a model

- Chemical representation (C_4H_{10})?
- Representing organization?

Trees as a model

- Chemical representation (C_4H_{10})?
- Representing organization?
- Computer file system?

Trees properties

Theorem 1.5.

A tree with n vertices has $n - 1$ edges.

Trees properties

Theorem 1.5.

A tree with n vertices has $n - 1$ edges.

Theorem 1.6 (Counting vertices in full $m - ary$ trees).

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

Trees properties

Theorem 1.7.

A full m-ary tree with

i *n vertices has* $i = (n-1)/m$ *internal vertices and* $l = [(m1)n+1]/m$ *leaves*

ii *i internal vertices has* $n = mi+1$ *vertices and* $l = (m1)i+1$ *leaves*

iii *l leaves has* $n = (ml1)/(m1)$ *vertices and* $i = (l1)/(m1)$ *internal vertices.*

Trees properties

Theorem 1.7.

A full m-ary tree with

i. *n vertices has* $i = (n-1)/m$ *internal vertices and* $l = [(m1)n + 1]/m$ *leaves*

ii. *i internal vertices has* $n = mi + 1$ *vertices and* $l = (m1)i + 1$ *leaves*

iii. *l leaves has* $n = (ml1)/(m1)$ *vertices and* $i = (l1)/(m1)$ *internal vertices.*

Example: suppose that someone starts a chain letter. Each person who receives the letter is asked to send it on to four other people. Some people do this, but others do not send any letters. How many people have seen the letter, including the first person, if no one receives more than one letter and if the chain letter ends after there have been 100 people who read it but did not send it out? How many people sent out the letter?

Trees properties

Theorem 1.8 (A bound for the number of leaves in an $m-ary$ tree).

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

Trees properties

Theorem 1.8 (A bound for the number of leaves in an $m-ary$ tree).

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

Corollary 1.9.

If an $m-ary$ tree of height h has l leaves, then $h \geq \lceil \log_m l \rceil$. If the $m-ary$ tree is full and balanced, then $h = \lceil \log_m l \rceil$. (We are using the ceiling function here. Recall that $\lceil x \rceil$ is the smallest integer greater than or equal to x.)

Binary search trees

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

Binary search trees

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

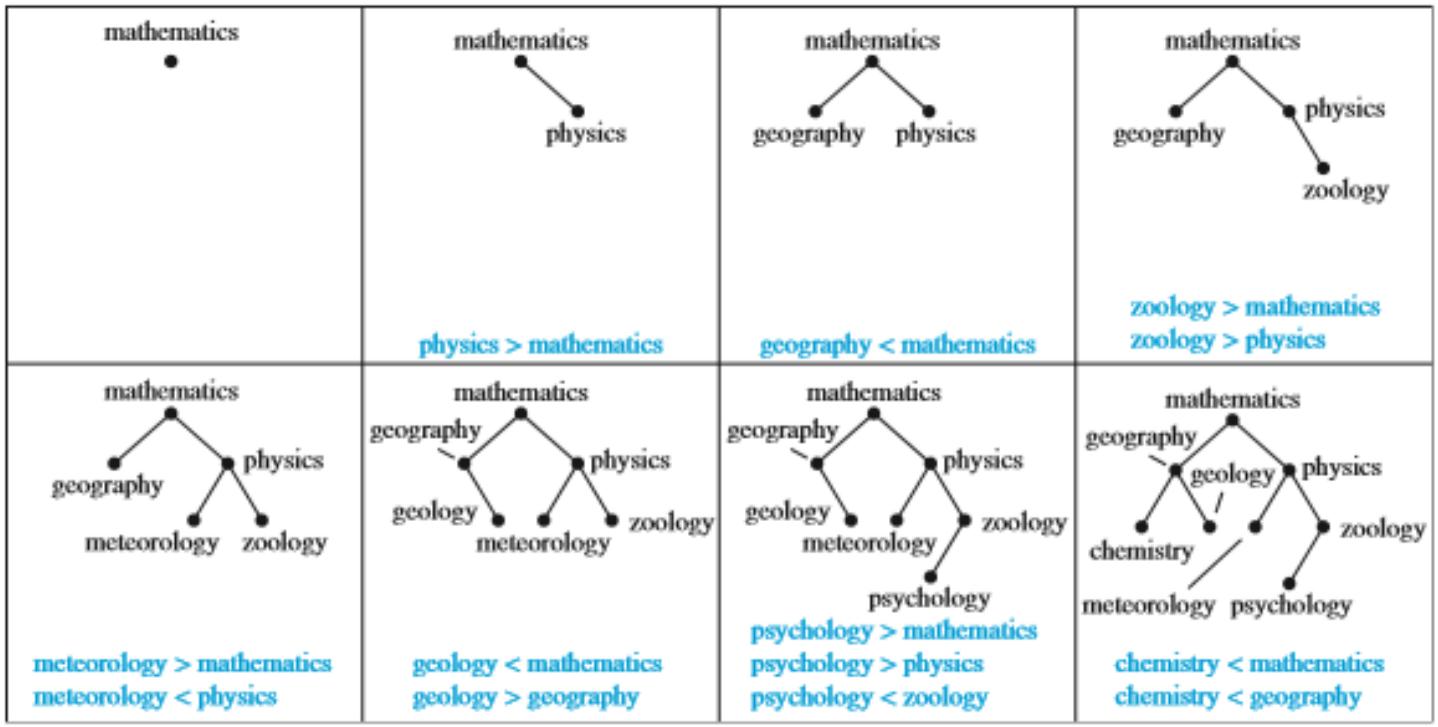


FIGURE 1 **Constructing a Binary Search Tree.**

Decision trees

Eaxmple: Suppose there are seven coins, all with the same weight, and a counter feit coin that weighs less than the others. How many weighings are necessary using a balance scale to determine which of the eight coins is the counterfeit one? Give an algorithm for finding this counterfeit coin.

Decision trees

Eaxmple: Suppose there are seven coins, all with the same weight, and a counter feit coin that weighs less than the others. How many weighings are necessary using a balance scale to determine which of the eight coins is the counterfeit one? Give an algorithm for finding this counterfeit coin.

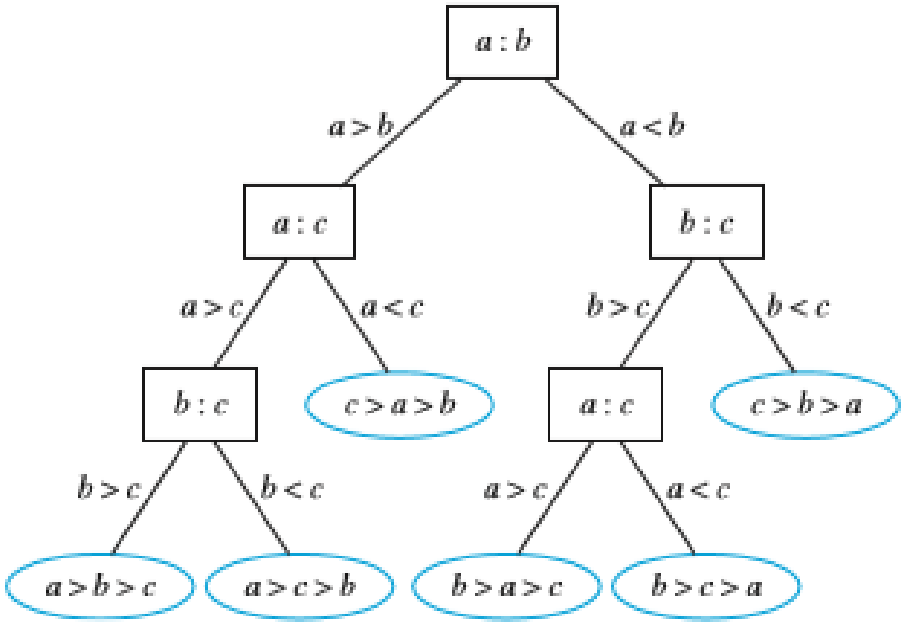


FIGURE 4 A Decision Tree for Sorting Three Distinct Elements.

Prefix codes

Eaxmple: Consider the problem of using bit strings to encode the letters of the English alphabet (where no distinction is made between lowercase and uppercase letters). We can represent each letter with a bit string of length five, because there are only 26 letters and there are 32 bit strings of length five.The total number of bits used to encode data is five times the number of characters in the text when each character is encoded with five bits. Is it possible to find a coding scheme of these letters such that, when data are coded, fewer bits are used? We can save memory and reduce transmittal time if this can be done.

Prefix codes

- Find decoded word for 0101 if $e = 0, a = 1, t = 01$

Prefix codes

- Find decoded word for 0101 if $e = 0, a = 1, t = 01$

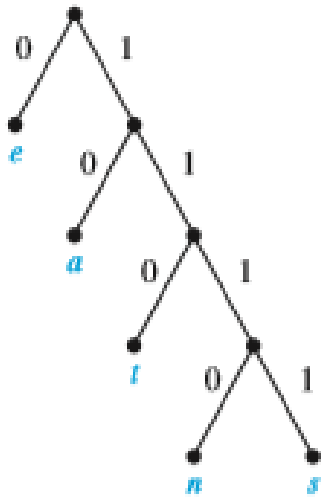


FIGURE 5 A Binary Tree with a Prefix Code.

Prefix codes

- Find decoded word for 0101 if $e = 0, a = 1, t = 01$

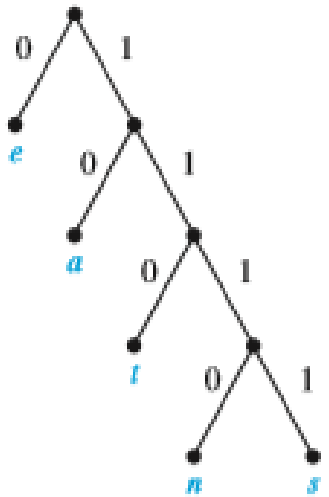


FIGURE 5 A Binary Tree with a Prefix Code.

- Now using prefix codes to decode 11111011100

Universal Address Systems

- Ordered rooted trees are often used to store information.

Universal Address Systems

- Ordered rooted trees are often used to store information.
- We need procedures for visiting each vertex of an ordered rooted tree to access data.

Universal Address Systems

- Ordered rooted trees are often used to store information.
- We need procedures for visiting each vertex of an ordered rooted tree to access data.
- We will describe one way we can totally order the vertices of an ordered rooted tree.
  - Label the root with the integer 0. Then label its k children (at level 1) from left to right with $1, 2, 3, ..., k$.

Universal Address Systems

- Ordered rooted trees are often used to store information.
- We need procedures for visiting each vertex of an ordered rooted tree to access data.
- We will describe one way we can totally order the vertices of an ordered rooted tree.
  - Label the root with the integer 0. Then label its k children (at level 1) from left to right with $1, 2, 3, ..., k$.
  - For each vertex v at level n with label A, label its k_v children, as they are drawn from left to right, with $A.1, A.2, ..., A.k_v$.

Universal Address Systems of an Ordered Rooted Tree

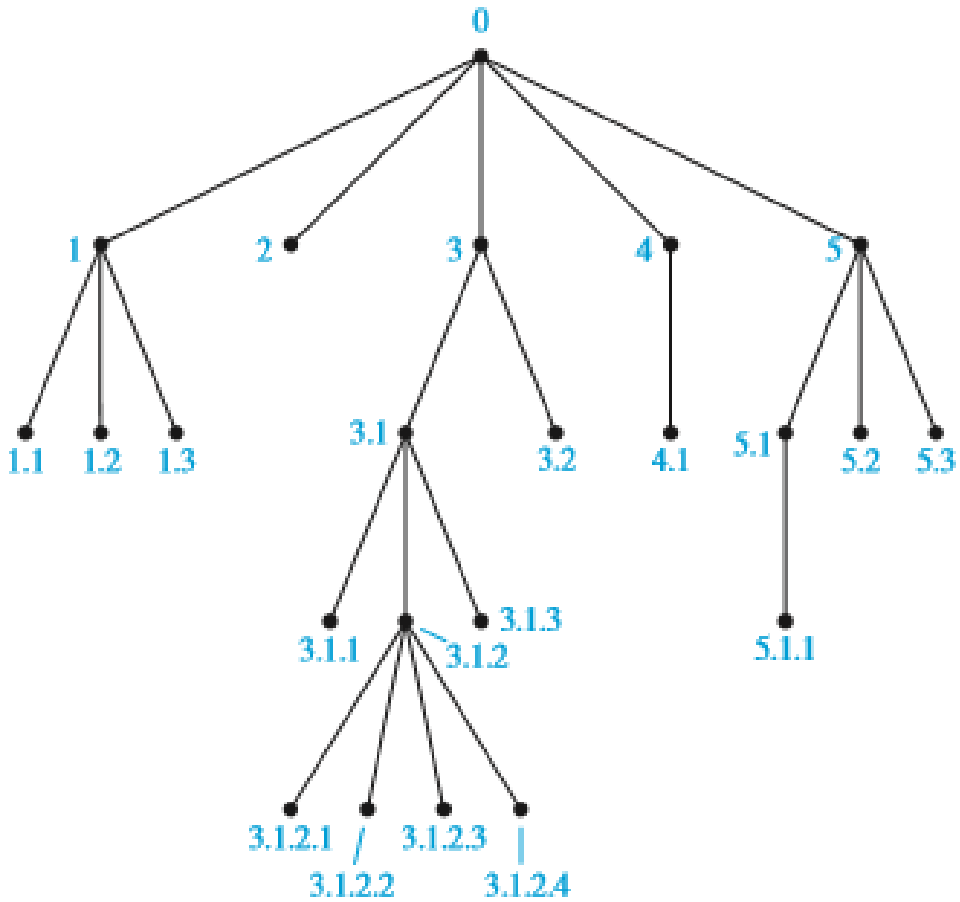


FIGURE 1 The Universal Address System of an Ordered Rooted Tree.

Traversal algorithm

Definition 3.1 (Pre-order traversal).

Let T be an ordered rooted tree with root r. If T consists only of r, then r is the pre-order traversal of T. Otherwise, suppose that $T_1, T_2, ..., T_n$ 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.

Traversal algorithm

Definition 3.1 (Pre-order traversal).

Let T be an ordered rooted tree with root r. If T consists only of r, then r is the pre-order traversal of T. Otherwise, suppose that $T_1, T_2, ..., T_n$ 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.

Definition 3.2 (In-order 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. Otherwise, suppose that $T_1, T_2, ..., T_n$ 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,..., and finally T_n in inorder.

Traversal algorithm

Definition 3.3 (Post-order 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. Otherwise, suppose that $T_1, T_2, ..., T_n$ are the subtrees at r from left to right. The post order traversal begins by traversing T_1 in postorder, then T_2 in postorder,..., then T_n in postorder, and ends by visiting r

Traversal algorithm

Definition 3.3 (Post-order 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. Otherwise, suppose that $T_1, T_2, ..., T_n$ are the subtrees at r from left to right. The post order traversal begins by traversing T_1 in postorder, then T_2 in postorder,..., then T_n in postorder, and ends by visiting r

Question: give the sequence of traversal for pre, in, post-order of the Tree below?

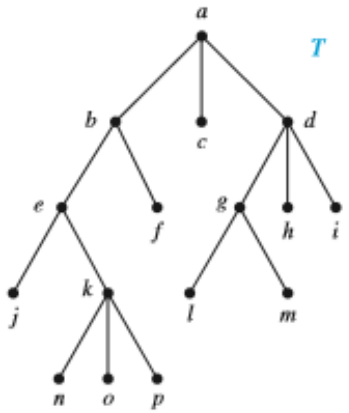


FIGURE 3 The Ordered Rooted Tree T.

Expression representation

Definition 3.4 (Prefix notation).

We obtain the prefix form for a expression by traversing the binary tree that represents it in preorder.

Expression representation

Definition 3.4 (Prefix notation).

We obtain the prefix form for a expression by traversing the binary tree that represents it in preorder.

Definition 3.5 (Infix notation).

An inorder traversal of the binary tree representing an expression produces the original expression with the elements and operations in the same order as they originally occurred.

Expression representation

Definition 3.4 (Prefix notation).

We obtain the prefix form for a expression by traversing the binary tree that represents it in preorder.

Definition 3.5 (Infix notation).

An inorder traversal of the binary tree representing an expression produces the original expression with the elements and operations in the same order as they originally occurred.

Definition 3.6 (Postfix notation).

We obtain the postfix form of an expression by traversing its binary tree in postorder.

Expression representation

Question: give the infix, prefix, postfix notation of

$$(x+y)^2 + \frac{x-4}{3}$$

Definition

Definition 4.1.

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.

Definition

Definition 4.1.

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: find a spanning tree of the simple graph G shown in Figure below.

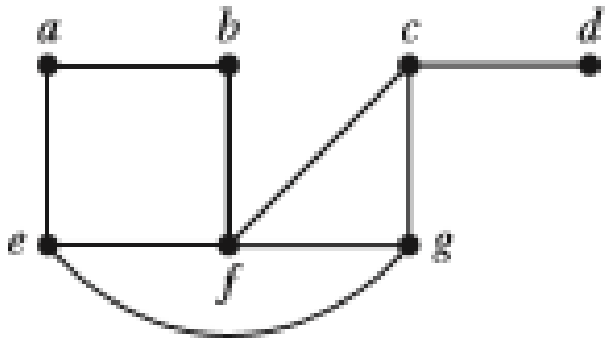


FIGURE 2 The Simple Graph G.

Definition

Theorem 4.2.

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

Definition

Theorem 4.2.

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

Example: IP Multicasting

Definition

Theorem 4.2.

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

Example: IP Multicasting

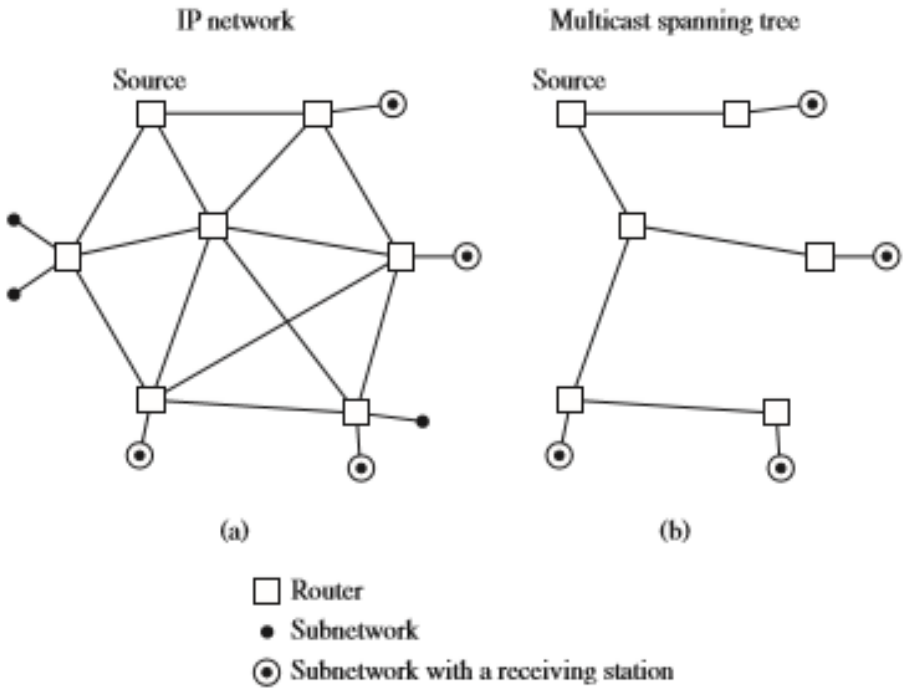


FIGURE 5 **A Multicast Spanning Tree.**

Construct spanning trees

- We can build a spanning tree for a connected simple graph using **depth-first search**. We will form a rooted tree, and the spanning tree will be the underlying undirected graph of this rooted tree. Arbitrarily choose a vertex of the graph as the root. Form a path starting at this vertex by successively adding vertices and edges, where each new edge is incident with the last vertex in the path and a vertex not already in the path. Continue adding vertices and edges to this path as long as possible. If the path goes through all vertices of the graph, the tree consisting of this path is a spanning tree. However, if the path does not go through all vertices, more vertices and edges must be added. Move back to the next to last vertex in the path, and, if possible, form a new path starting at this vertex passing through vertices that were not already visited. If this cannot be done, move back another vertex in the path, that is, two vertices back in the path,and try again.

Depth-first search

Example: using Depth-first search to construct spanning tree for graph below:

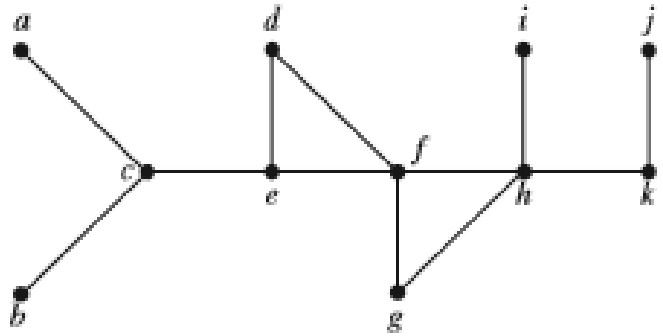


FIGURE 6 The Graph G.

Depth-first search

Example: using Depth-first search to construct spanning tree for graph below:

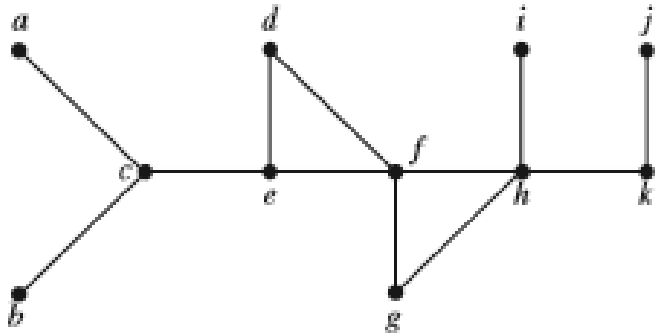


FIGURE 6 The Graph G.

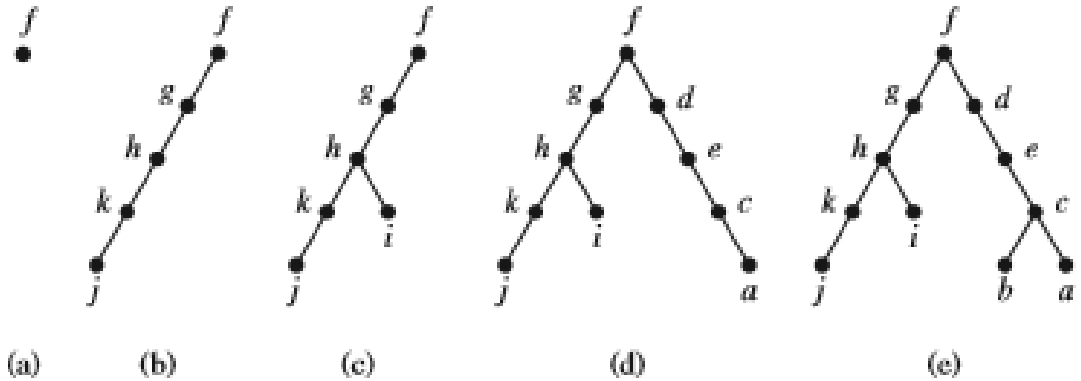


FIGURE 7 Depth-First Search of G.

Construct spanning trees

- We can also produce a spanning tree of a simple graph by the use of **breadth-first search**. Again, a rooted tree will be constructed, and the underlying undirected graph of this rooted tree forms the spanning tree. Arbitrarily choose a root from the vertices of the graph. Then add all edges incident to this vertex. The new vertices added at this stage become the vertices at level 1 in the spanning tree. Arbitrarily order them. Next, for each vertex at level 1, visited in order, add each edge incident to this vertex to the tree as long as it does not produce a simple circuit. Arbitrarily order the children of each vertex at level 1. This produces the vertices at level 2 in the tree. Follow the same procedure until all the vertices in the tree have been added. The procedure ends because there are only a finite number of edges in the graph.

Breadth-first search

Example: using Breadth-first search to construct spanning tree for graph below:

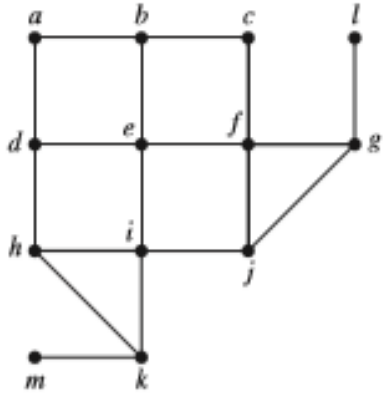


FIGURE 9 **A Graph *G*.**

Breadth-first search

Example: using Breadth-first search to construct spanning tree for graph below:

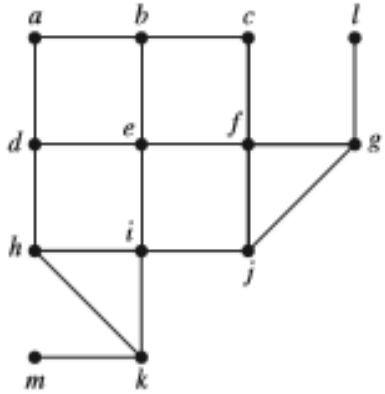


FIGURE 9 A Graph G.

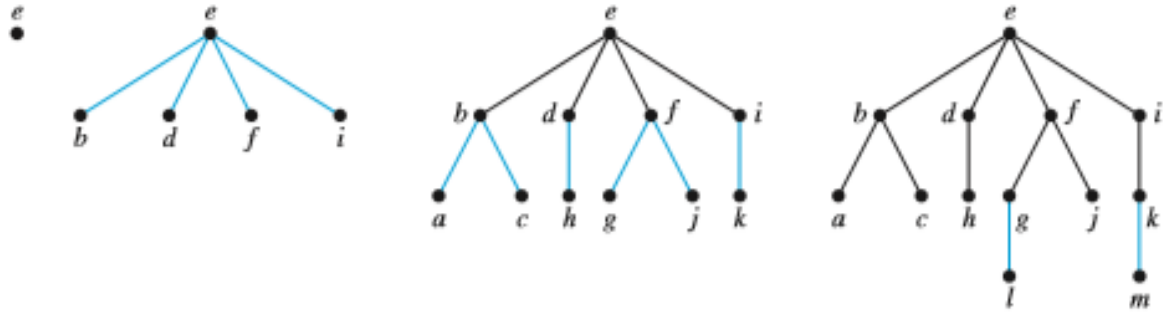


FIGURE 10 Breadth-First Search of G.

Backtracking application

Graph Colorings: how can backtracking be used to decide whether a graph can be colored using n colors?

Backtracking application

Graph Colorings: how can backtracking be used to decide whether a graph can be colored using n colors?

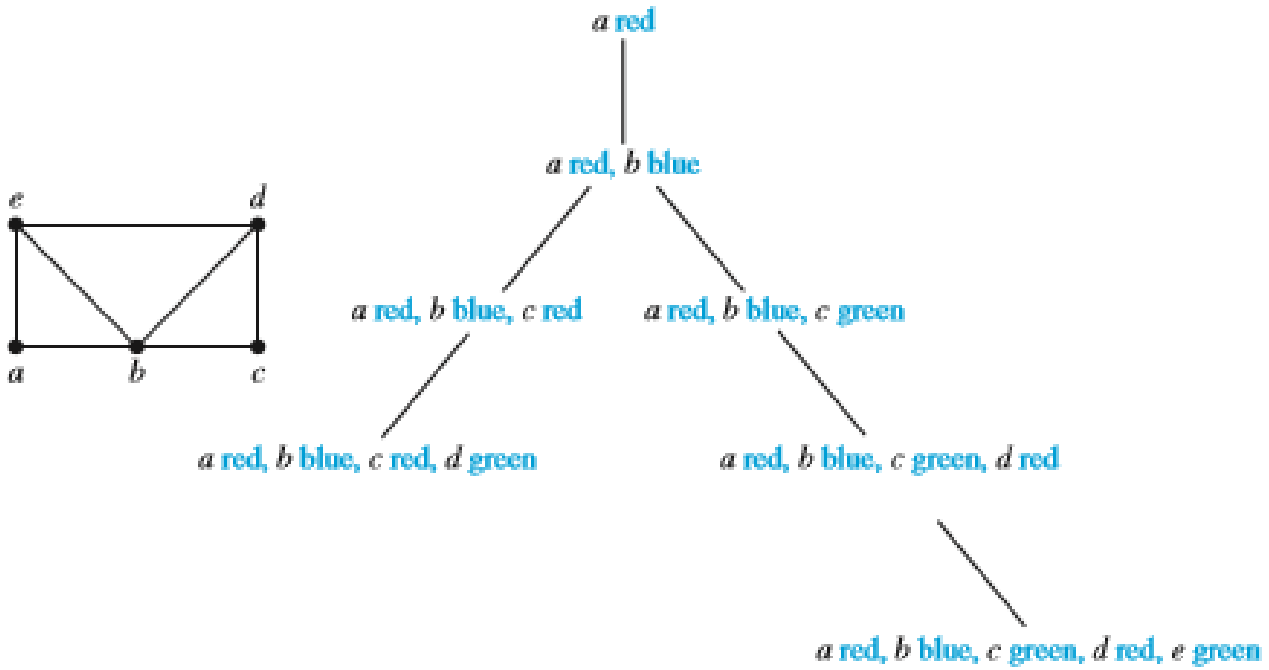


FIGURE 11 Coloring a Graph Using Backtracking.

Definition

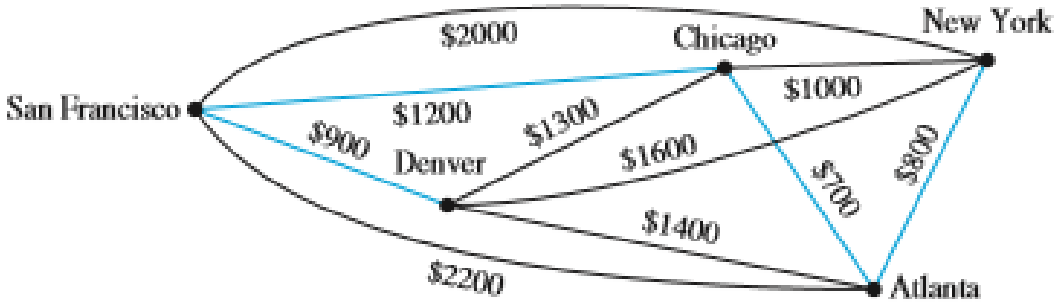
Definition 5.1.

A minimum spanning tree in a connected weighted graph is a spanning tree that has the smallest possible sum of weights of its edges.

Definition

Definition 5.1.

A minimum spanning tree in a connected weighted graph is a spanning tree that has the smallest possible sum of weights of its edges.



Choice	Edge	Cost
1	{Chicago, Atlanta}	$ 700
2	{Atlanta, New York}	$ 800
3	{Chicago, San Francisco}	$1200
4	{San Francisco, Denver}	$ 900
	Total:	$3600

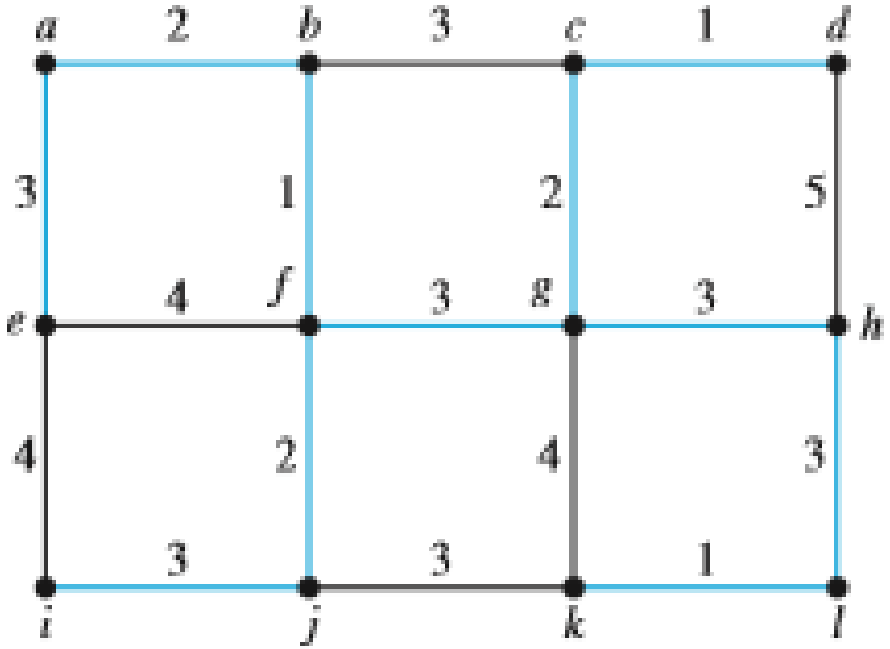
FIGURE 2 **A Minimum Spanning Tree for the Weighted Graph in Figure 1.**

Prim Algorithm

ALGORITHM 1 Prim's Algorithm.

procedure *Prim*(G: weighted connected undirected graph with n vertices)
T := a minimum-weight edge
for $i := 1$ **to** $n - 2$
    e := an edge of minimum weight incident to a vertex in T and not forming a simple circuit in T if added to T
    $T := T$ with e added
return T {T is a minimum spanning tree of G}

Prim Algorithm Example



(a)

Choice	Edge	Weight
1	{b, f}	1
2	{a, b}	2
3	{f, j}	2
4	{a, e}	3
5	{i, j}	3
6	{f, g}	3
7	{c, g}	2
8	{c, d}	1
9	{g, h}	3
10	{h, l}	3
11	{k, l}	1
	Total:	24

(b)

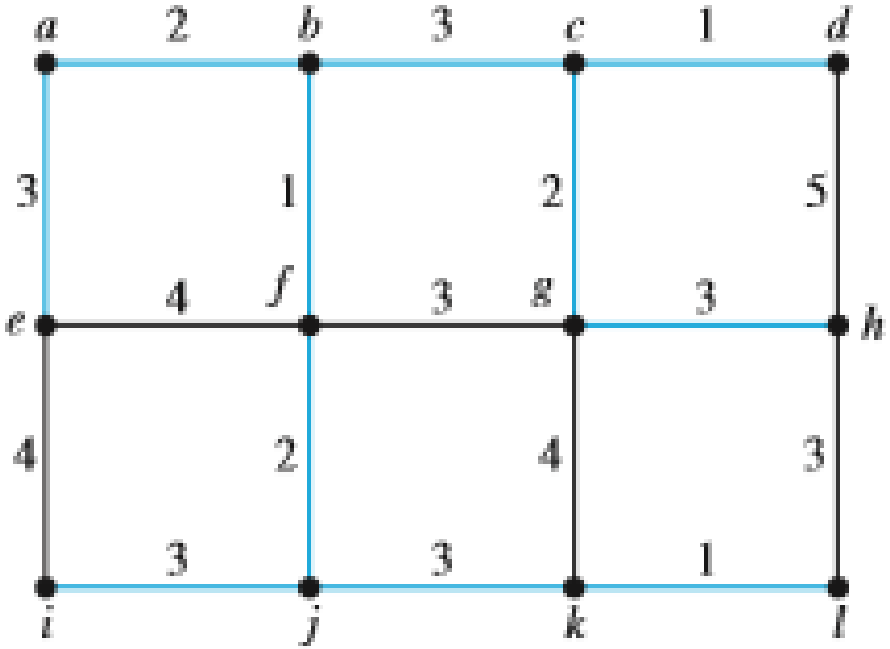
FIGURE 4 **A Minimum Spanning Tree Produced Using Prim's Algorithm.**

Kruskal Algorithm

ALGORITHM 2 Kruskal's Algorithm.

procedure *Kruskal*(G: weighted connected undirected graph with n vertices)
$T :=$ empty graph
for $i := 1$ **to** $n - 1$
    $e :=$ any edge in G with smallest weight that does not form a simple circuit when added to T
    $T := T$ with e added
return T {T is a minimum spanning tree of G}

Kruskal Algorithm Example



(a)

Choice	Edge	Weight
1	{c, d}	1
2	{k, l}	1
3	{b, f}	1
4	{c, g}	2
5	{a, b}	2
6	{f, j}	2
7	{b, c}	3
8	{j, k}	3
9	{g, h}	3
10	{i, j}	3
11	{a, e}	3
	Total:	24

(b)

FIGURE 5 A Minimum Spanning Tree Produced by Kruskal's Algorithm.