

**CS 441 Discrete Mathematics for CS**  
**Lecture 26**

**Graphs**

**Milos Hauskrecht**  
[milos@cs.pitt.edu](mailto:milos@cs.pitt.edu)  
5329 Sennott Square

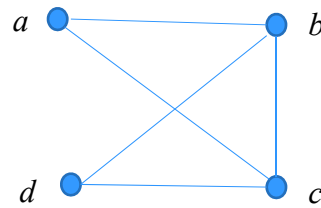
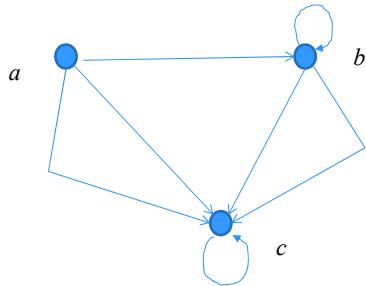
**Final exam**

- **Saturday, April 26, 2014 at 10:00-11:50am**
- **The same classroom as lectures**
- **The exam is:**
  - **Closed book**
  - **cumulative**
- **Please bring your own calculators**

## Graphs: basics

### Basic types of graphs:

- Directed graphs
- Undirected graphs

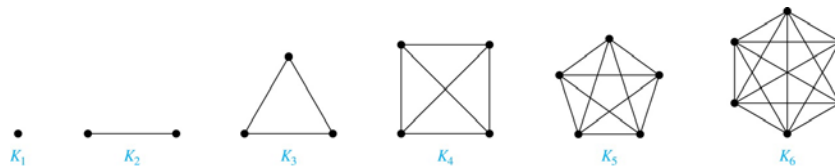


CS 441 Discrete mathematics for CS

M. Hauskrecht

## Complete graphs

A complete graph on  $n$  vertices, denoted by  $K_n$ , is the simple graph that contains exactly one edge between each pair of distinct vertices.

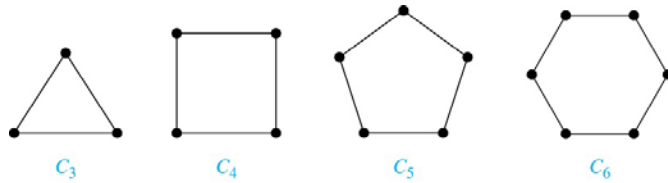


CS 441 Discrete mathematics for CS

M. Hauskrecht

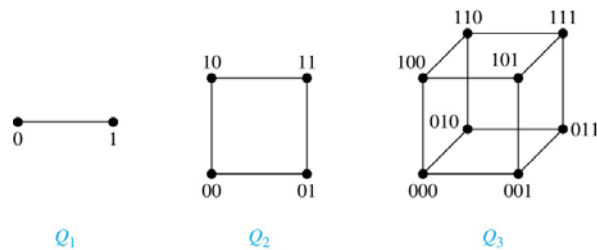
## A cycle

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



## N-dimensional hypercube

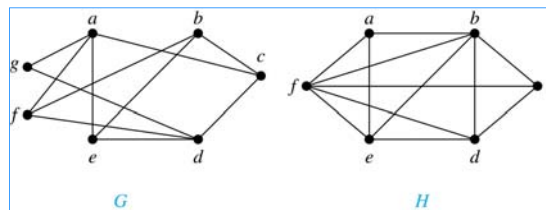
An  $n$ -dimensional hypercube, or  $n$ -cube,  $Q_n$ , is a graph with  $2^n$  vertices representing all bit strings of length  $n$ , where there is an edge between two vertices that differ in exactly one bit position.



## Bipartite graphs

**Definition:** A simple graph  $G$  is **bipartite** if  $V$  can be partitioned into two disjoint subsets  $V_1$  and  $V_2$  such that every edge connects a vertex in  $V_1$  and a vertex in  $V_2$ . In other words, there are no edges which connect two vertices in  $V_1$  or in  $V_2$ .

**Note:** An equivalent definition of a bipartite graph is a graph where it is possible to color the vertices red or blue so that no two adjacent vertices are the same color.



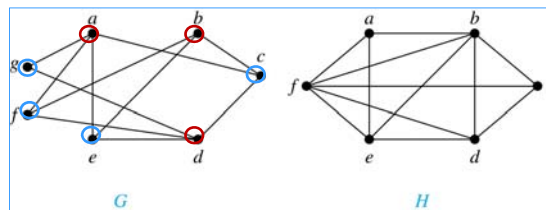
CS 441 Discrete mathematics for CS

M. Hauskrecht

## Bipartite graphs

**Definition:** A simple graph  $G$  is **bipartite** if  $V$  can be partitioned into two disjoint subsets  $V_1$  and  $V_2$  such that every edge connects a vertex in  $V_1$  and a vertex in  $V_2$ . In other words, there are no edges which connect two vertices in  $V_1$  or in  $V_2$ .

**Note:** An equivalent definition of a bipartite graph is a graph where it is possible to color the vertices red or blue so that no two adjacent vertices are the same color.



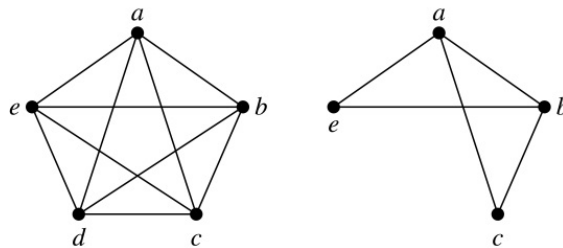
CS 441 Discrete mathematics for CS

M. Hauskrecht

## Subgraphs

**Definition:** A *subgraph of a graph*  $G = (V, E)$  is a graph  $(W, F)$ , where  $W \subset V$  and  $F \subset E$ . A subgraph  $H$  of  $G$  is a proper subgraph of  $G$  if  $H \neq G$ .

**Example:**  $K_5$  and one of its subgraphs.



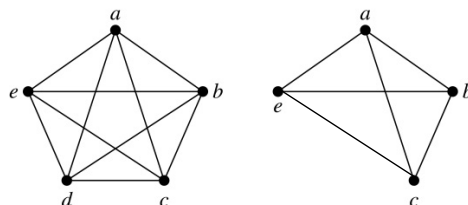
CS 441 Discrete mathematics for CS

M. Hauskrecht

## Subgraphs

**Definition:** Let  $G = (V, E)$  be a simple graph. The *subgraph induced* by a subset  $W$  of the vertex set  $V$  is the graph  $(W, F)$ , where the edge set  $F$  contains an edge in  $E$  if and only if both endpoints are in  $W$ .

**Example:**  $K_5$  and the subgraph induced by  $W = \{a, b, c, e\}$ .



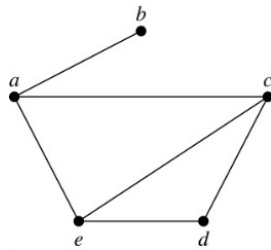
CS 441 Discrete mathematics for CS

M. Hauskrecht

## Representation of graphs

**Definition:** An *adjacency list* can be used to represent a graph with no multiple edges by specifying the vertices that are adjacent to each vertex of the graph.

**Example:**



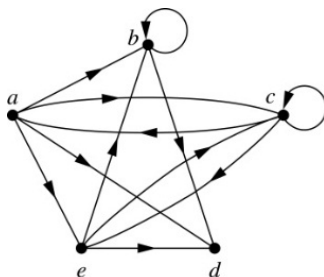
**TABLE 1** An Adjacency List for a Simple Graph.

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

## Representation of graphs

**Definition:** An *adjacency list* can be used to represent a graph with no multiple edges by specifying the vertices that are adjacent to each vertex of the graph.

**Example:**



**TABLE 2** An Adjacency List for a Directed Graph.

Initial Vertex	Terminal Vertices
a	b, c, d, e
b	b, d
c	a, c, e
d	
e	b, c, d

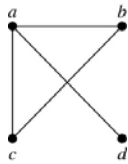
## Adjacency matrices

**Definition:** Suppose that  $G = (V, E)$  is a simple graph where  $|V| = n$ . Arbitrarily list the vertices of  $G$  as  $v_1, v_2, \dots, v_n$ . The *adjacency matrix*  $\mathbf{A}_G$  of  $G$ , with respect to the listing of vertices, is the  $n \times n$  zero-one matrix with 1 as its  $(i, j)$ th entry when  $v_i$  and  $v_j$  are adjacent, and 0 as its  $(i, j)$ th entry when they are not adjacent.

– In other words, if the graph's adjacency matrix is  $\mathbf{A}_G = [a_{ij}]$ ,

then 
$$a_{ij} = \begin{cases} 1 & \text{if } \{v_i, v_j\} \text{ is an edge of } G, \\ 0 & \text{otherwise.} \end{cases}$$

**Example:**



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

*The ordering of vertices is a, b, c, d.*

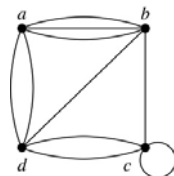
CS 441 Discrete mathematics for CS

M. Hauskrecht

## Adjacency matrices

- Adjacency matrices can also be used to represent graphs with loops and multiple edges.
- A loop at the vertex  $v_i$  is represented by a 1 at the  $(i, i)$ th position of the matrix.
- When multiple edges connect the same pair of vertices  $v_i$  and  $v_j$ , (or if multiple loops are present at the same vertex), the  $(i, j)$ th entry equals the number of edges connecting the pair of vertices.

**Example:** The adjacency matrix of the pseudograph shown here using the ordering of vertices  $a, b, c, d$ .



$$\begin{bmatrix} 0 & 3 & 0 & 2 \\ 3 & 0 & 1 & 1 \\ 0 & 1 & 1 & 2 \\ 2 & 1 & 2 & 0 \end{bmatrix}$$

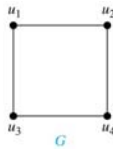
CS 441 Discrete mathematics for CS

M. Hauskrecht

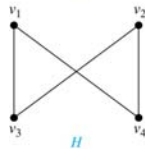
## Graph isomorphism

**Definition:** The simple graphs  $G_1 = (V_1, E_1)$  and  $G_2 = (V_2, E_2)$  are *isomorphic* if there is a **one-to-one and onto function**  $f$  from  $V_1$  to  $V_2$  with the property that  $a$  and  $b$  are adjacent in  $G_1$  if and only if  $f(a)$  and  $f(b)$  are adjacent in  $G_2$ , for all  $a$  and  $b$  in  $V_1$ . Such a function  $f$  is called an *isomorphism*. Two simple graphs that are not isomorphic are called *nonisomorphic*.

**Example:**



Are the two graphs isomorphic?



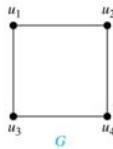
CS 441 Discrete mathematics for CS

M. Hauskrecht

## Graph isomorphism

**Definition:** The simple graphs  $G_1 = (V_1, E_1)$  and  $G_2 = (V_2, E_2)$  are *isomorphic* if there is a **one-to-one and onto function**  $f$  from  $V_1$  to  $V_2$  with the property that  $a$  and  $b$  are adjacent in  $G_1$  if and only if  $f(a)$  and  $f(b)$  are adjacent in  $G_2$ , for all  $a$  and  $b$  in  $V_1$ . Such a function  $f$  is called an *isomorphism*. Two simple graphs that are not isomorphic are called *nonisomorphic*.

**Example:**



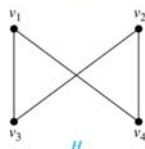
Are the two graphs isomorphic?

$u_1 \rightarrow v_1$

$u_2 \rightarrow v_4$

$u_3 \rightarrow v_2$

$u_4 \rightarrow v_3$



CS 441 Discrete mathematics for CS

M. Hauskrecht



## Connectivity in the graphs, paths

**Informal Definition:** A *path* is a sequence of edges that begins at a vertex of a graph and travels from vertex to vertex along edges of the graph. As the path travels along its edges, it visits the vertices along this path, that is, the endpoints of these.

**Applications:** Numerous problems can be modeled with paths formed by traveling along edges of graphs such as:

- determining whether a message can be sent between two computers.
- efficiently planning routes for mail/message delivery.

## Connectivity in the graphs

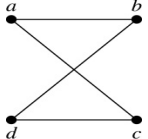
- We can use the adjacency matrix of a graph to find the **number of the different paths between two vertices in the graph.**

**Theorem:** Let  $G$  be a graph with adjacency matrix  $\mathbf{A}$  with respect to the ordering  $v_1, \dots, v_n$  of vertices (with directed or undirected edges, multiple edges and loops allowed). The number of different paths of length  $r$  from  $v_i$  to  $v_j$ , where  $r > 0$  is a positive integer, equals the  $(i,j)$ th entry of  $\mathbf{A}^r$ .

## Connectivity in the graphs

**Theorem:** Let  $G$  be a graph with adjacency matrix  $\mathbf{A}$  with respect to the ordering  $v_1, \dots, v_n$  of vertices (with directed or undirected edges, multiple edges and loops allowed). The number of different paths of length  $r$  from  $v_i$  to  $v_j$ , where  $r > 0$  is a positive integer, equals the  $(i,j)$ th entry of  $\mathbf{A}^r$ .

**Example:**



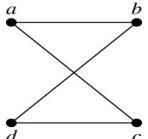
$$A = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix}$$

**Paths of length 4.**

## Connectivity in the graphs

**Theorem:** Let  $G$  be a graph with adjacency matrix  $\mathbf{A}$  with respect to the ordering  $v_1, \dots, v_n$  of vertices (with directed or undirected edges, multiple edges and loops allowed). The number of different paths of length  $r$  from  $v_i$  to  $v_j$ , where  $r > 0$  is a positive integer, equals the  $(i,j)$ th entry of  $\mathbf{A}^r$ .

**Example:**



$$A = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix}$$

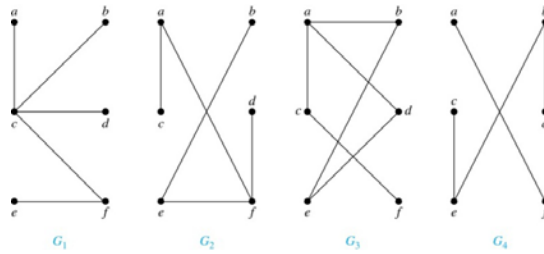
**Paths of length 4:** The adjacency matrix of  $G$  (ordering the vertices as  $a, b, c, d$ ) is given above. Hence the number of paths of length four from  $a$  to  $d$  is the  $(1, 4)$ th entry of  $\mathbf{A}^4$

$$\mathbf{A}^4 = \begin{bmatrix} 8 & 0 & 0 & 8 \\ 0 & 8 & 8 & 0 \\ 0 & 8 & 8 & 0 \\ 8 & 0 & 0 & 8 \end{bmatrix}$$

## Trees

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

**Examples:**

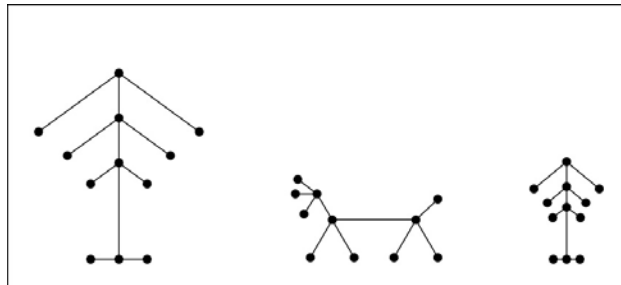


Tree: yes
Tree: yes
Tree: no
Tree: no

## Connectivity in the graphs

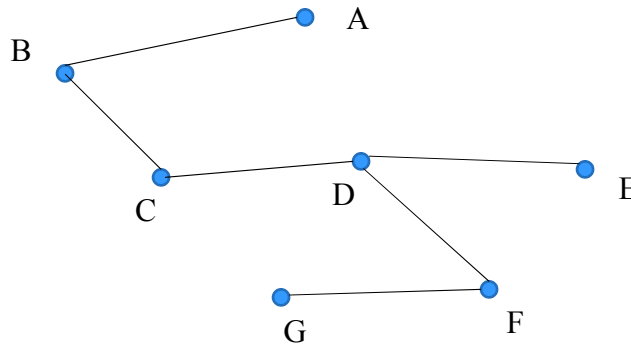
**Definition:** A *forest* is a graph that has no simple circuit, but is not connected. Each of the connected components in a forest is a tree.

**Example:**



## Trees

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



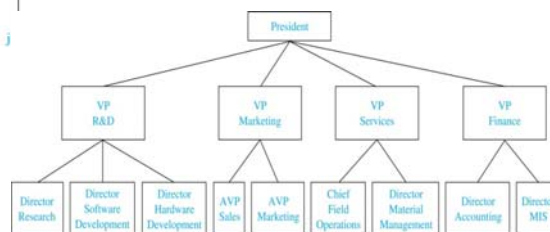
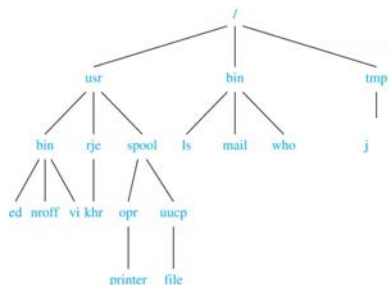
CS 441 Discrete mathematics for CS

M. Hauskrecht

## Application of trees

### Examples:

- The organization of a computer file system into directories, subdirectories, and files is naturally represented as a tree.
- structure of organizations.



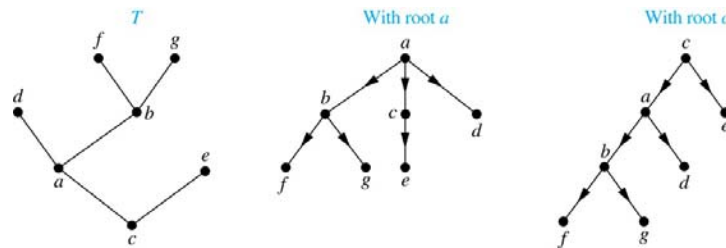
CS 441 Discrete mathematics for CS

M. Hauskrecht

## 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.

**Note:** An *unrooted tree* can be converted into different rooted trees when one of the vertices is chosen as the root.

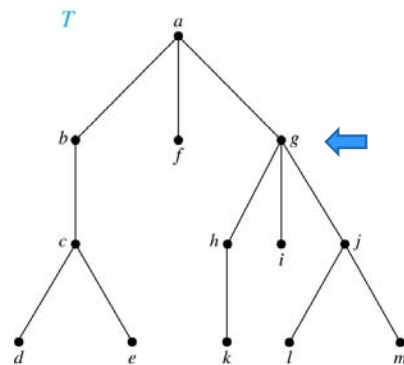


CS 441 Discrete mathematics for CS

M. Hauskrecht

## Rooted trees - terminology

- If  $v$  is a vertex of a rooted tree other than the root, the *parent* of  $v$  is the unique vertex  $u$  such that there is a directed edge from  $u$  to  $v$ . When  $u$  is a parent of  $v$ ,  $v$  is called a *child* of  $u$ . Vertices with the same parent are called *siblings*.



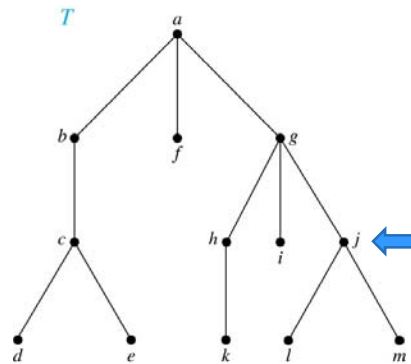
Parent of  $g$ :  $a$   
 Children of  $g$ :  $h, j, k$   
 Siblings of  $g$ :  $b, f$

CS 441 Discrete mathematics for CS

M. Hauskrecht

## Rooted trees - terminology

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



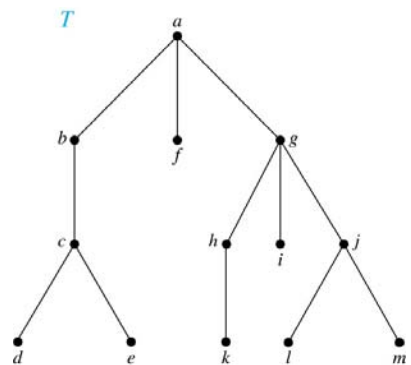
Ancestors of  $j$ :  $g, a$   
Descendants of  $j$ :  $l, m$

CS 441 Discrete mathematics for CS

M. Hauskrecht

## Rooted trees - terminology

- A vertex of a rooted tree with no children is called a *leaf*. Vertices that have children are called *internal vertices*.



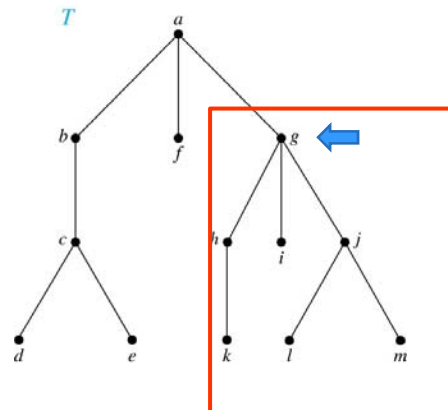
Leaves:  $d, e, k, l, m$   
Examples of internal nodes:  
 $b, g, h$

CS 441 Discrete mathematics for CS

M. Hauskrecht

## Rooted trees - terminology

- If  $a$  is a vertex in a tree, the *subtree* with  $a$  as its root is the subgraph of the tree consisting of  $a$  and its descendants and all edges incident to these descendants.



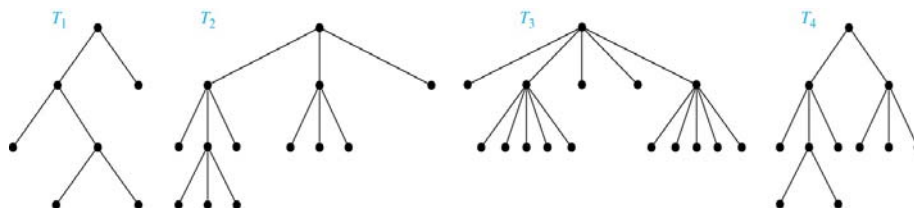
CS 441 Discrete mathematics for CS

M. Hauskrecht

## M-ary tree

**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. An  $m$ -ary tree with  $m = 2$  is called a *binary tree*.

**Example:** Are the following rooted trees full  $m$ -ary trees for some positive integer  $m$ ?



CS 441 Discrete mathematics for CS

M. Hauskrecht

## Binary trees

**Definition:** A *binary tree* is an ordered rooted tree where each internal vertex has at most two children. If an internal vertex of a binary tree has two children, the first is called the *left child* and the second 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.

