

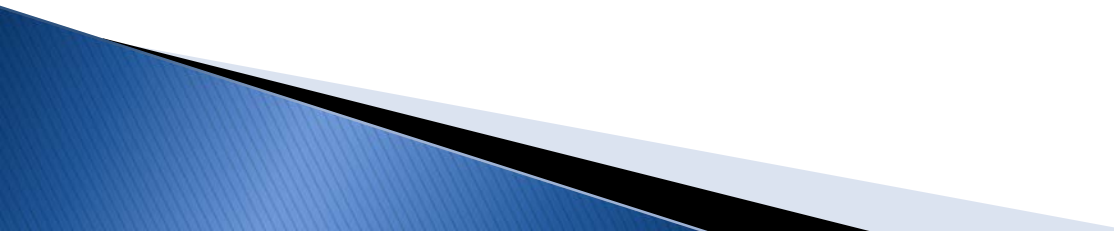
CSE 15

Discrete Mathematics

Lecture 24 – Equivalence Relations
Graphs



Equivalence Relations (Ch. 9.5)

- ▶ Equivalence Relations
 - ▶ Equivalence Classes
 - ▶ Equivalence Classes and Partitions
- 

Equivalence Relations

Definition 1: A relation on a set A is called an *equivalence relation* if it is reflexive, symmetric, and transitive.

Definition 2: Two elements a and b that are related by an equivalence relation are called *equivalent*.

Strings

Example: Suppose that R is the relation on the set of strings of English letters such that aRb if and only if $l(a) = l(b)$, where $l(x)$ is the length of the string x .

Is R an equivalence relation?

Solution: Show that all of the properties of an equivalence relation hold.

- *Reflexivity:* Because $l(a) = l(a)$, it follows that aRa for all strings a .
- *Symmetry:* Suppose that aRb . Since $l(a) = l(b)$, then $l(b) = l(a)$ also holds and bRa .
- *Transitivity:* Suppose that aRb and bRc . Since $l(a) = l(b)$, and $l(b) = l(c)$, then $l(a) = l(c)$ also holds and aRc .

NOTE: An equivalence relation breaks a set into many disjoint sets (classes) such that all elements in a class satisfy that relation.

Equivalence Classes

Definition 3: Let R be an equivalence relation on a set A .

The set of all elements that are related to an element a of A is called the *equivalence class* of a .

- ▶ The equivalence class of a with respect to R is denoted by $[a]_R$.
- ▶ Note that $[a]_R = \{s \mid (a,s) \in R\}$.
- ▶ If $b \in [a]_R$ then b is called a representative of this equivalence class.

Example: The relation R on the set of strings of English letters such that aRb if and only if $l(a) = l(b)$ divides R in equivalence classes such that each class contains letters of the same length.

Graphs and Graph Models (Ch. 10.1)

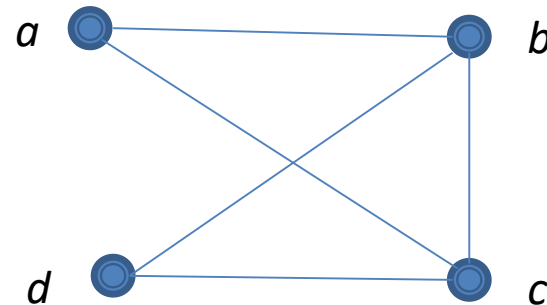
- ▶ Introduction to Graphs
- ▶ Graph Taxonomy

Graphs

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

Example:

This is a graph with four vertices and five edges.



A Remark:

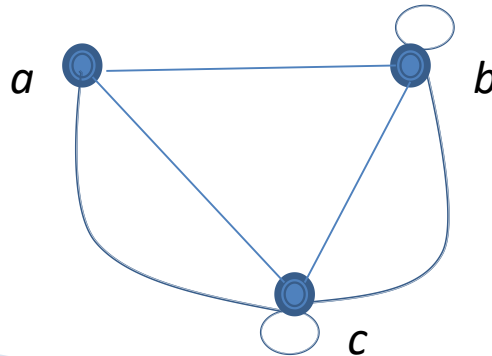
- A graph with an infinite vertex set is called an *infinite graph*. A graph with a finite vertex set is called a *finite graph*. We restrict our attention to finite graphs.

Terminology

- ▶ In a *simple graph* each edge connects two different vertices and no two edges connect the same pair of vertices.
- ▶ *Multigraphs* may have multiple edges connecting the same two vertices. When m different edges connect the vertices u and v , we say that $\{u,v\}$ is an edge of *multiplicity* m .
- ▶ An edge that connects a vertex to itself is called a *loop*.
- ▶ A *pseudograph* may include loops, as well as multiple edges connecting the same pair of vertices.

Example:

This pseudograph has both multiple edges and a loop.



Remark: There is no standard terminology for graph theory. So, it is crucial that you understand the terminology being used whenever you read material about graphs.

Directed Graphs

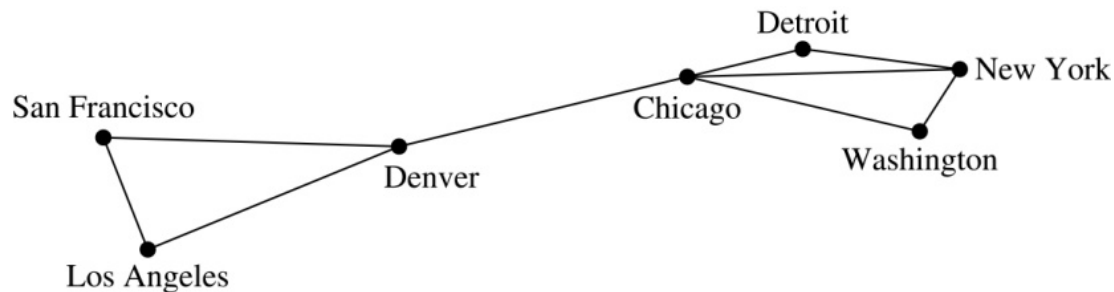
Definition: A *directed graph* (or *digraph*) $G = (V, E)$ consists of a nonempty set V of *vertices* (or *nodes*) and a set E of *directed edges* (or *arcs*). Each edge is associated with an ordered pair of vertices. The directed edge associated with the ordered pair (u, v) is said to *start at* u and *end at* v .

Remark:

- Graphs where the end points of an edge are not ordered are said to be *undirected graphs*.

Graph Models: Computer Networks

- ▶ To model a computer network where we are only concerned whether two data centers are connected by a communications link, we use a simple graph.
- ▶ This is the appropriate type of graph when we only care whether two data centers are directly linked (and not how many links there may be) and all communications links work in both directions.



Other Applications of Graphs

- ▶ Graph theory can be used in models of:
 - Social networks
 - Communications networks
 - Information networks
 - Software design
 - Transportation networks
 - Biological networks