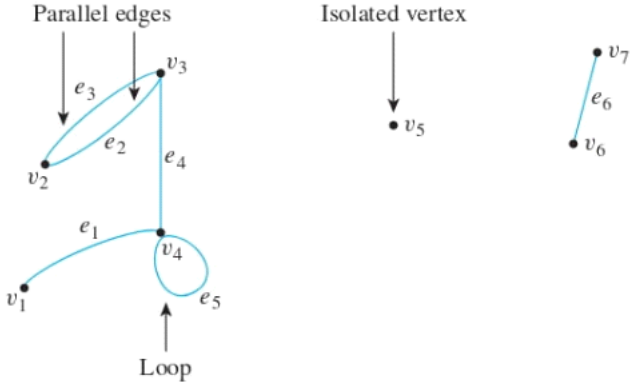
Chapter 10 Review

Saturday, April 24, 2021 7:20 PM

Chapter 10: Graphs and Trees

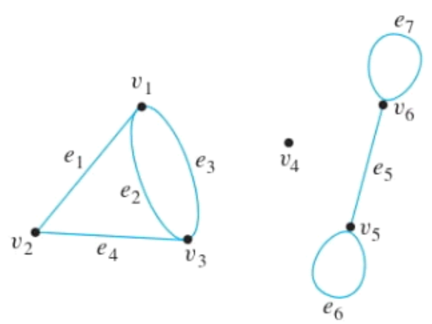
Section 10.1: Graphs: Definitions and Basic Properties



Graph

- A **graph** G consists of two finite sets:
  - A nonempty set V(G) of **vertices**
  - A set E(G) of **edges**, where each edge is associated with a set consisting of either one or two vertices called its **endpoints**
- The correspondence from edges to endpoints is called the **edge-endpoint function**.
- An edge with just one endpoint is called a **loop**.
- Two or more distinct edges with the same endpoints are said to be **parallel**.
- An edge is said to **connect** to its endpoints.
- Two vertices that are connected by an edge are called **adjacent**.
- A vertex that is an endpoint of a loop is said to be **adjacent to itself**.
- An edge is said to be **incident on** each of its endpoints.
- Two edges incident on the same endpoint are called **adjacent**.
- A vertex on which no edges are incident is called **isolated**.

Graph Example:



vertex set $= \{v_1, v_2, v_3, v_4, v_5, v_6\}$
edge set $= \{e_1, e_2, e_3, e_4, e_5, e_6, e_7\}$
edge-endpoint function:

Edge	Endpoints
e_1	$\{v_1, v_2\}$
e_2	$\{v_1, v_3\}$
e_3	$\{v_1, v_3\}$
e_4	$\{v_2, v_3\}$
e_5	$\{v_5, v_6\}$
e_6	$\{v_5\}$
e_7	$\{v_6\}$

Directed Graph (Digraph)

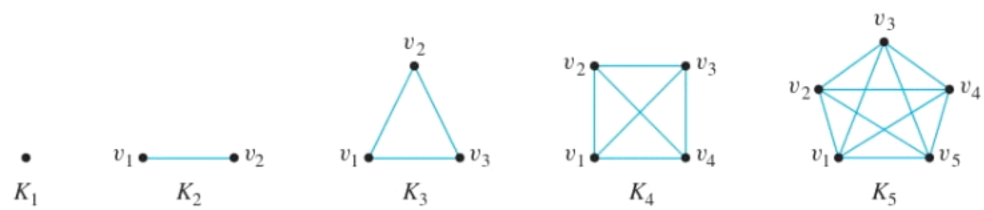
- A **directed graph**, or **digraph**, consists of two finite sets:
  - A nonempty set V(G) of vertices
  - A set D(G) of directed edges, where each is associated with an ordered pair of vertices called its **endpoints**
- If edge e is associated with the pair (a, w) of vertices then e is said to be the (**directed**) **edge** from v to w.
- Note that each directed graph has an associated ordinary (undirected) graph, which is obtained by ignoring the directions of the edges.

Simple Graph

- A **simple graph** is a graph that does not have any loops or parallel edges.
- In a simple graph, an edge with endpoints v and w is denoted {v, w}.

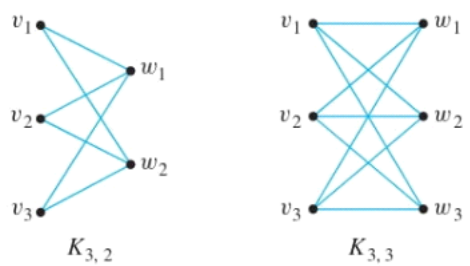
Complete Graph

- Let n be a positive integer. A **complete graph on n vertices**, denoted $\mathbf{K_n}$, is a simple graph with n vertices and exactly one edge connecting each pair of distinct vertices.
- Examples:



Complete Bipartite Graph

- Let m and n be positive integers. A **complete bipartite graph on (m, n) vertices**, denoted $\mathbf{K_{m,n}}$, is a simple graph with distinct vertices $v_1, v_2, \ldots, v_m$ and $w_1, w_2, \ldots, w_n$ that satisfies the following properties: For all i, k = 1, 2,…, m and for all j, l = 1, 2,…, n
  1. There is an edge from each vertex v_i to each vertex w_j
  2. There is no edge from any vertex v_i to any other vertex v_k
  3. There is no edge from any vertex w_j to any other vertex w_l
- Examples:



Subgraph

- A graph H is said to be a **subgraph** of graph G IFF every vertex in H is also a vertex in G, every edge in H is also in G, and every edge in H has the same endpoints as it has in G.

The Concept of Degree

- Let G be a graph and v a vertex of G. The **degree of v**, denoted **deg(v)**, equals the number of edges that are incident on v, with an edge that is a loop counted twice.
- The **total degree of G** is the sum of the degrees of all the vertices of G.

The Handshake Theorem (Theorem 10.1.1)

- If G is any graph, then the sum of the degrees of all the vertices of G equals twice the number of edges of G.

- Specifically, if the vertices of G are $v_1, v_2, \ldots, v_n$, where n is a nonnegative integer, then
  $$\text{the total degree of G} = \deg(v_1) + \deg(v_2) + \cdots + \deg(v_n)$$
  $$= 2 \cdot (\text{the number of edges of G})$$

Corollary 10.1.2: The total degree of a graph is even.

Proposition 10.1.3: In any graph there are an even number of vertices of odd degree.

Section 10.2: Trails, Paths, and Circuits

Definitions

- Let G be a graph, and v and w be vertices in G.
  - A **walk from v to w** is a finite alternating sequence of adjacent vertices and edges of G. Thus a walk has the form $v_0 e_1 v_1 e_2 \cdots v_{n-1} e_n v_n$ where the v's represent vertices, the e's represent edges, $v_0 = v$, $v_n = w$, and for $i = 1, 2, \ldots, n$, and v_i are the endpoints of e_i.
  - The **trivial walk from v to v** consists of the single vertex v.
  - A **trail from v to w** is a walk from v to w that does not contain a repeated edge.
  - A **path form v to w** is a trail that does not contain a repeated vertex.
  - A **closed walk** is a walk that starts and ends at the same vertex.
  - A **circuit** is a closed walk that contains at least one edge and does not contain a repeated edge.
  - A **simple circuit** is a circuit that does not have any other repeated vertex except the first and last.
- Summary:

	Repeated Edge?	Repeated Vertex?	Starts and Ends at Same Point?	Must Contain at Least One Edge?
Walk	allowed	allowed	allowed	no
Trail	no	allowed	allowed	no
Path	no	no	no	no
Closed walk	allowed	allowed	yes	no
Circuit	no	allowed	yes	yes
Simple circuit	no	first and last only	yes	yes

Connectedness

- Let G be a graph. Two **vertices v and w of G are connected** IFF there is a walk from v to w.
- The **graph G is connected** IFF given any two vertices v and w in G, there is a walk form v to w. Symbolically,
  $$\text{G is connected } \Leftrightarrow \forall \text{ vertices } v, w \in V(G), \exists \text{ a walk from } v \text{ } to \text{ } w.$$

Lemma 10.2.1:

- Let G be a graph.
  a. If G is connected, then any two distinct vertices of G can be connected by a path.
  b. If vertices v and w are part of a circuit in G and one edge is removed from the circuit, then there still exists a trail from v to w in G.
  c. If G is connected and G contains a circuit, the an edge of the circuit can be removed without disconnecting G.

Connected Component

- A graph H is a **connected component** of graph G IFF
  1. H is a subgraph of G
  2. H is connected
  3. No connected subgraph of G has H as a subgraph and contains vertices or edges that are not in H

Euler Circuits

- Let G be a graph. An **Euler circuit** for G is a circuit that contains every vertex and every edge of G.
- That is, an Euler circuit for G is a sequence of adjacent vertices and edges in G that…
  - Has at least one edge
  - Starts and ends at the same vertex
  - Uses every vertex of G at least once
  - Uses every edge of G exactly once

<u>Theorem 10.2.2</u>: If a graph has an Euler circuit, then every vertex of the graph has positive even degree.

Contrapositive of Theorem 10.2.2: If some vertex of a graph has odd degree, then the graph does not have an Euler circuit.

<u>Theorem 10.2.3</u>: If a graph G is connected and the degree of every vertex of G is a positive even integer, then G has an Euler circuit.

<u>Theorem 10.2.4</u>: A graph G has an Euler circuit IFF G is connected and every vertex of G has positive even degree.

Euler Trail

- Let G be a graph, and let v and w be two distinct vertices of G. An **Euler trail from v to w** is a sequence of adjacent edges and vertices that…
  - Starts at v and ends at w
  - Passes through every vertex of G at least once
  - Traverses every edge of G exactly once

<u>Corollary 10.2.5</u>: Let G be a graph, and let v and w be two distinct vertices of G. There is an Euler path from v to w IFF G is connected, v and w have odd degree, and all other vertices of G have positive even degree.

Section 10.4: Isomorphisms of Graphs (Partially Covered)

Isomorphic (10.4.1)

- Let G and G' be graphs with vertex sets V(G) and V(G') and edge sets E(G) and E(G'), respectively. **G is isomorphic to G'** IFF there exists one-to-one correspondences $g: V(G) \to V(G')$ and $h: E(G) \to E(G')$ that preserve the edge-endpoint functions of G and G' in the sense that for all $v \in V(G)$ and $e \in E(G)$, $v \text{ is an endpoint of } e \Leftrightarrow g(v) \text{ is an endpoint of } h(e)$.

Section 10.5: Trees

Circuit-Free: A graph is said to be **circuit-free** IFF it has no circuits.

Trees

- A graph is called a **tree** IFF it is circuit-free and connected.
- A **trivial tree** is a graph that consists of a single vertex.
- A graph is called a **forest** IFF it is circuit-free and not connected.

Characterizing Trees

- Any tree with n vertices, where n is a positive integer, has n-1 edges. (Theorem 10.5.2)
- Any connected graph with n vertices and n-1 edges is a tree. (Theorem 10.5.4)
- If one new edge (but no new vertex) is added to a tree, the resulting graph must contain a circuit.
- Every connected graph has a subgraph that is a tree.

<u>Lemma 10.5.1</u>: Any tree that has more than one vertex has at least one vertex of degree 1.

Terminal and Internal Vertices

- Let T be a tree.
  - If T has only one or two vertices, then each is called a **terminal vertex**.
  - If T has at least three vertices, then a vertex of degree 1 in T is called a **terminal vertex** (or a **leaf**), and a vertex of degree greater than 1 in T is called an **internal vertex** (or a **branch vertex**).

<u>Theorem 10.5.2</u>: For any positive integer n, any tree with n vertices has n-1 edges.

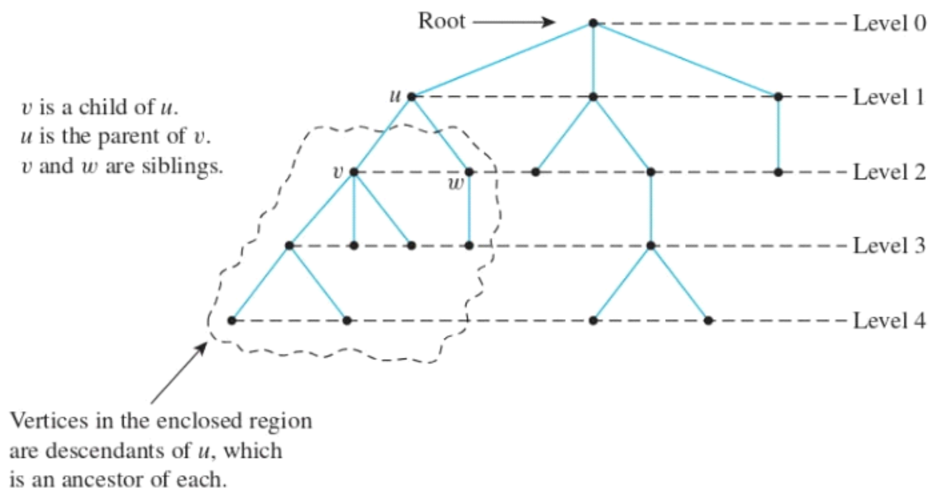
<u>Lemma 10.5.3</u>: If G is any connected graph, C is any circuit in G, and any one of the edges of C is removed from G, then the graph that remains is connected.

<u>Theorem 10.5.4</u>: For any positive integer n, if G is a connected graph with n vertices and n-1 edges then G is a tree.

Section 10.6: Rooted Trees

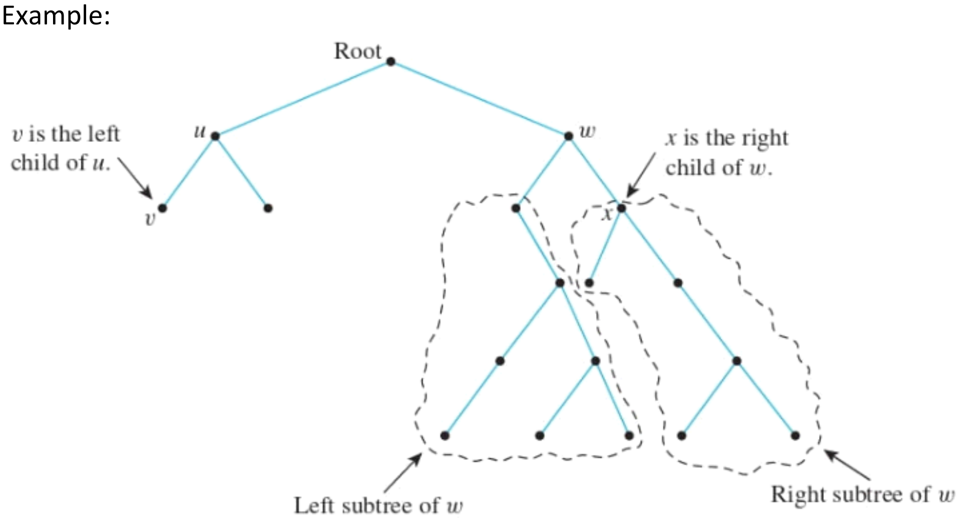
Rooted Trees

- A **rooted tree** is a tree in which there is one vertex that is distinguished form the others and is called the **root**.
- The **level** of a vertex is the number of edges along the unique path between it and the root.
- The **height** of a rooted tree is the maximum level of any vertex of the tree.
- Given the root or any internal vertex v of a rooted tree, the **children** of v are those vertices that are adjacent to c and are one level farther away from the root than v.
- If w is a child of v, the v is called the **parent** of w.
- Two distinct vertices that are both children of the same parent are called **siblings**.
- Given two distinct vertices v and w, if v lies on the unique path between w and the root, the v is an **ancestor** or w and w is a **descendant** of v.
- Example:



Binary Trees

- A **binary tree** is a rooted tree in which every parent has as most two children.
- Each child in a binary tree is designated either a **left child** or a **right child** (but not both), and every parent has at most one left child and one right child.
- A **full binary tree** is a binary tree in which each parent has exactly two children.
- Given any parent v in a binary tree T, if v has a left child, then the **left subtree** of v is the binary tree:
  - whose root is the left child of v
  - whose vertices consist of the left child of v and all its descendants
  - whose edges consist of all those edges of T that connect the vertices of the left subtree
- Given any parent v in a binary tree T, if v has a right child, then the **right subtree** of v is the binary tree:
  - whose root is the right child of v
  - whose vertices consist of the right child of v and all its descendants
  - whose edges consist of all those edges of T that connect the vertices of the right subtree
- Example:



Theorem 10.6.1: If k is a positive integer and T is a full binary tree with k internal vertices, then T has a total of 2k+1 vertices and has k+1 terminal vertices.

Theorem 10.6.2: For all integers $h \geq 0$, if T is any binary tree with a height of h and t terminal vertices, then $t \leq 2^h$. Equivalently, $\log_2 t \leq h$.