

MATH221

Mathematics for Computer Science

Unit 12

Graphs

Learning Objectives

- Graph Terminologies
- Euler Circuits
- Tree and Minimum Spanning Tree: Kruskal's algorithm and Prim's Algorithm

2

Graph Theory

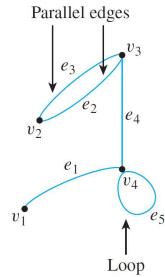
Many real-world problems concern certain classes of objects and the relations between these objects. The objects could be people connected through friendship, cities connected through travel routes, or webpages connected through links.

The mathematical abstraction of these situations is the study of graph theory. It has applications in almost all fields of study.

The idea of a graph is a simple one: it is just a collection of points, some of which may be joined together by lines. We now give the formal definition.

Graphs: Definitions and Basic Properties

- A **graph** G consists of
 - a set of **vertices** $V(G)$, and
 - a set of **edges** $E(G)$.
- Sometimes, we write $G = \{V, E\}$.



An edge connects one vertex to another, or a vertex to itself.

Graphs: Definitions and Basic Properties

Definition: Graph

A **graph** G consists of 2 finite sets: a nonempty set $V(G)$ of **vertices** and a set $E(G)$ of **edges**, where each edge is associated with a set consisting of either one or two vertices called its **endpoints**.

An edge with just one endpoint is called a **loop**. Two distinct edges with the same endpoints are called **parallel**.

An edge is said to **connect** its endpoints; two vertices that are connected by an edge are called **adjacent vertices**; and 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, and two edges incident on the same endpoint are called **adjacent edges**.

We write $e = \{v, w\}$ for an edge e incident on vertices v and w .

6

Graphs: Definitions and Basic Properties

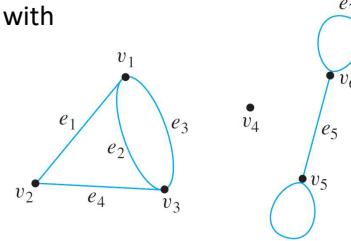
Example: Consider the following graph:

- Write the vertex set V and the edge set E , and give the list of edges with their end-points.

$$V = \{v_1, v_2, v_3, v_4, v_5, v_6\}$$

$$E = \{e_1, e_2, e_3, e_4, e_5, e_6, e_7\}$$

$$\begin{array}{ll} e_1 = \{v_1, v_2\} & e_4 = \{v_2, v_3\} \\ e_2 = \{v_1, v_3\} & e_5 = \{v_5, v_6\} \\ e_3 = \{v_1, v_3\} & e_6 = \{v_5\} \\ e_7 = \{v_6\} & \end{array}$$



7

Graphs: Definitions and Basic Properties

Example: Consider the following graph:

- Find all edges that are incident on v_1 , all vertices that are adjacent to v_1 , all edges that are adjacent to e_1 , all loops, all parallel edges, and all vertices that are adjacent to themselves.

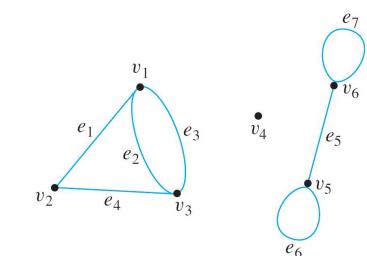
Edges incident on v_1 : e_1, e_2 and e_3 .

Vertices adjacent to v_1 : v_2 and v_3 .

Edges adjacent to e_1 : e_2, e_3 and e_4 .

Loops: e_6 and e_7 .

e_2 and e_3 are parallel.



v_5 and v_6 are adjacent to themselves.

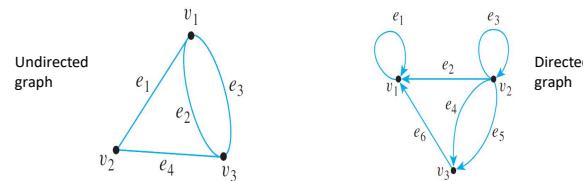
8

Graphs: Definitions and Basic Properties

Definition: Directed Graph

A **directed graph**, or **digraph**, G , consists of 2 finite sets: a nonempty set $V(G)$ of **vertices** and a set $D(G)$ of **directed edges**, where each edge is associated with an ordered pair of vertices called its **endpoints**.

If edge e is associated with the pair (v, w) of vertices, then e is said to be the **(directed) edge** from v to w . We write $e = (v, w)$.



9

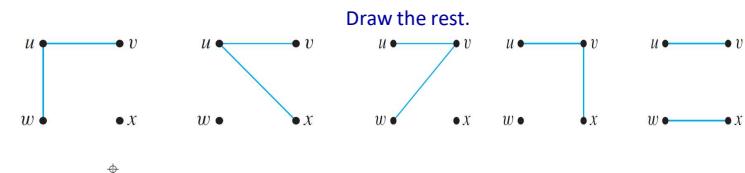
Simple Graphs

Definition: Simple Graph

A **simple graph** is an undirected graph that does not have any loops or parallel edges.

Example: Draw all simple graphs with the 4 vertices $\{u, v, w, x\}$ and two edges, one of which is $\{u, v\}$.

There are at most $\binom{4}{2} = 6$ edges in a simple graph with 4 vertices. One edge is given. Hence we need to pick another from the remaining 5.



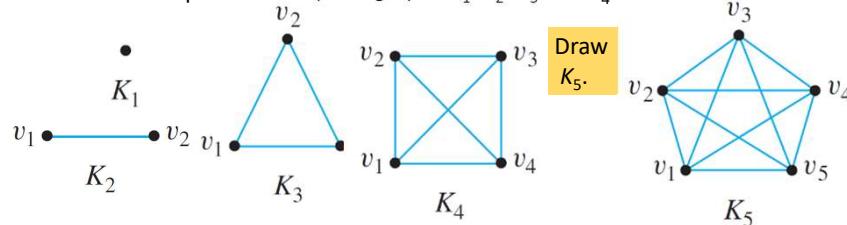
10

Complete Graphs

Definition: Complete Graph

A **complete graph** on n vertices, $n > 0$, denoted K_n , is a simple graph with n vertices and exactly one edge connecting each pair of distinct vertices.

Example: The complete graphs K_1, K_2, K_3 and K_4 .



11

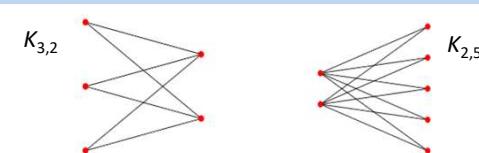
Complete Bipartite Graphs

Definition: Complete Bipartite Graph

A **complete bipartite graph** on (m, n) vertices, where $m, n > 0$, denoted $K_{m,n}$, is a simple graph with distinct vertices v_1, v_2, \dots, v_m and w_1, w_2, \dots, w_n that satisfies the following properties:

For all $i, k = 1, 2, \dots, m$ and for all $j, l = 1, 2, \dots, n$,

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 .



12

General Bipartite Graphs

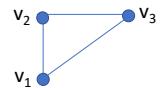
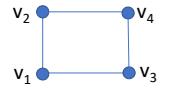
Definition: Bipartite

A simple graph G is bipartite if there are two subsets U and W of the vertex set V of G such that

1. $U \cup W = V$ and $U \cap W = \emptyset$, and
2. every edge of G connects a vertex in U with a vertex in W .

So there are no edges connecting between vertices of U or vertices of W .

Example: Which of these is bipartite?



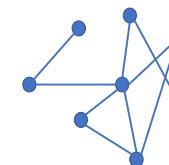
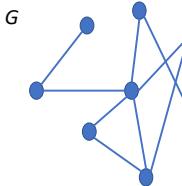
Ans: Only the left-hand side graph: $U=\{v_1, v_4\}$, $V=\{v_2, v_3\}$

13

Subgraph of a Graph

Definition: Subgraph of a Graph

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

A graph G 

A subgraph
of G

14

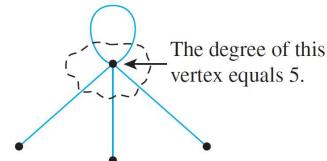
Degree of a Vertex and Total Degree of a Graph

Definition: Degree of a Vertex and Total Degree of a Graph

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 degree of a vertex can be obtained from the drawing of a graph by counting how many end segments of edges are incident on the vertex.

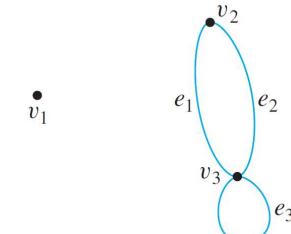


15

Degree of a Vertex and Total Degree of a Graph

Example: Find the degree of each vertex of the graph G shown below. Then find the total degree of G .

$\deg(v_1) = 0$
 $\deg(v_2) = 2$
 $\deg(v_3) = 4$
 Total degree of $G = 6$



16

The Concept of Degree

Theorem 12.1.1 The Handshake Theorem

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, \dots, v_n , where $n \geq 0$, then

$$\begin{aligned}\text{The total degree of } G &= \deg(v_1) + \deg(v_2) + \dots + \deg(v_n) \\ &= 2 \times (\text{the number of edges of } G).\end{aligned}$$

Corollary 12.1.2

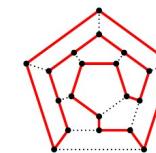
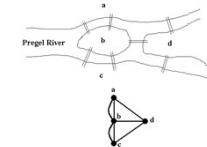
The total degree of a graph is even.

Proposition 12.1.3

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

17

2: Trails, Paths, and Circuits



18

Let's Have Some Fun

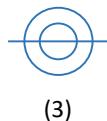
Can you draw the following figures without lifting up your pencil?



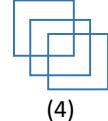
(1)



(2)



(3)



(4)



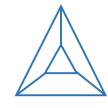
(5)



(6)



(7)



(8)

19

Königsberg bridges

The subject of graph theory began in the year 1736 when the great mathematician [Leonhard Euler](#) published a paper giving the solution to the following puzzle:

The town of Königsberg in Prussia (now Kaliningrad in Russia) was built at a point where two branches of the Pregel River came together. It consisted of an island and some land along the river banks.

These were connected by 7 bridges.

20

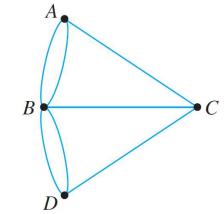
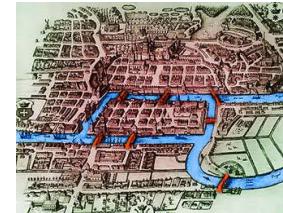
Königsberg bridges



Question: Is it possible to take a walk around town, starting and ending at the same location and crossing each of the 7 bridges **exactly once**?

21

Königsberg bridges



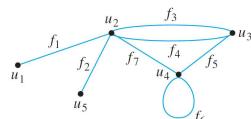
In terms of this graph, the question is:

Is it possible to find a route through the graph that starts and ends at some vertex, one of A, B, C, or D, and traverses each edge exactly once and every vertex at least once?

22

Definitions

Travel in a graph is accomplished by moving from one vertex to another along a sequence of adjacent edges. In the graph below, for instance, you can go from u_1 to u_4 by taking f_1 to u_2 and then f_7 to u_4 . This is represented by writing $u_1f_1u_2f_7u_4$.



Or, you could take a roundabout route:

$u_1f_1u_2f_3u_3f_4u_2f_3u_3f_5u_4f_6u_4f_7u_2f_3u_3f_5u_4$.

23

Walk, Trail, Path, Closed Walk, Circuit, Simple Circuit

Definitions

Let G be a graph, and let v and w be vertices of 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

$$v_0 e_1 v_1 e_2 \dots v_{n-1} e_n v_n,$$

where the v 's represent vertices, the e 's represent edges, $v_0=v$, $v_n=w$, and for all $i \in \{1, 2, \dots, n\}$, v_{i-1} 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 from v to w** is a trail that does not contain a repeated vertex.

24

Walk, Trail, Path, Closed Walk, Circuit, Simple Circuit

Definitions

A **closed walk** is a walk that starts and ends at the same vertex.

A **circuit (or cycle)** is a closed walk that contains at least one edge and does not contain a repeated edge.

A **simple circuit (or simple cycle)** is a circuit that does not have any other repeated vertex except the first and last.

25

Walk, Trail, Path, Closed Walk, Circuit, Simple Definitions

	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

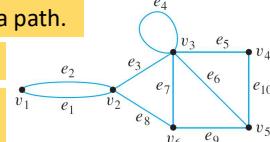
Often a walk can be specified unambiguously by giving either a sequence of edges or a sequence of vertices.

26

Walk, Trail, Path, Closed Walk, Circuit, Simple Circuit

In this graph, determine which of the following walks are trails, paths, circuits, or simple circuits.

- a. $v_1e_1v_2e_3v_3e_4v_3e_5v_4$ Trail; not a path.
- b. $e_1e_3e_5e_6e_7$ Walk; not a trail.
- c. $v_2v_3v_4v_5v_3v_6v_2$ Circuit; not a simple circuit.
- d. $v_2v_3v_4v_5v_6v_2$ Simple circuit.
- e. $v_1e_1v_2e_1v_1$ Closed walk; not a circuit.
- f. v_1 Closed walk; not a circuit.



27

Notes

Because most of the major developments in graph theory have happened relatively recently and in a variety of different contexts, the terms used in the subject have not been standardized.

Susanna Epp's book	Others
Graph	Multigraph
Simple graph	Graph
Vertex	Node
Edge	Arc
Trail	Path
Path	Simple path
Simple circuit	Cycle

The terminology in this book is among the most common, but if you consult other sources, be sure to check their definitions.

For MATH 221, we will follow the terminology in Epp's book.

28

Connectedness

A graph is connected if it is possible to travel from any vertex to any other vertex along a sequence of adjacent edges of the graph.

Definition: Connectedness

Two vertices v and w of a graph G are connected if, and only if, there is a walk from v to w .

The graph G is connected if, and only if, given *any* two vertices v and w in G , there is a walk from v to w . Symbolically,

G is connected iff \forall vertices $v, w \in V(G), \exists$ a walk from v to w .

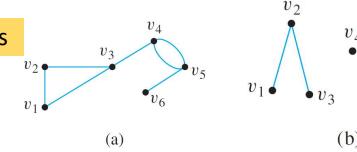
The graph G is disconnected if it is not connected. So, G is disconnected if there are two vertices v and w in G with no walk between v and w .

29

Connectedness

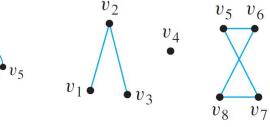
Example: Which of the following graphs are connected?

Yes



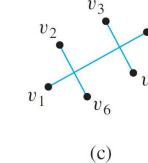
(a)

No



(b)

No



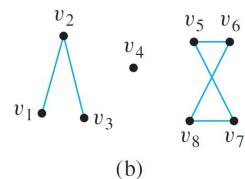
(c)

30

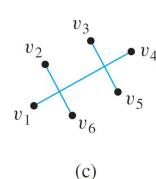
Connected Component

A **connected component** of a graph is a connected subgraph of largest possible size.

The graphs in (b) and (c) are both made up of three pieces, each of which is itself a connected graph.



(b)



(c)

31

Connectedness

Definition: Connected Component

A graph H is a **connected component** of a graph G if, and only if,

1. The graph H is a subgraph of G ;
2. The graph H is connected; and
3. No connected subgraph of G has H as a subgraph and contains vertices or edges that are not in H .

The fact is that any graph is a kind of union of its connected components.

32

Connected Component

Find all connected components of the following graph G .



G has 3 connected components H_1, H_2 and H_3 with vertex sets V_1, V_2 and V_3 and edge sets E_1, E_2 and E_3 , where

$$V_1 = \{v_1, v_2, v_3\}, \quad E_1 = \{e_1, e_2\}$$

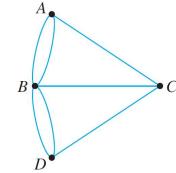
$$V_2 = \{v_4\}, \quad E_2 = \emptyset$$

$$V_3 = \{v_5, v_6, v_7, v_8\}, \quad E_3 = \{e_3, e_4, e_5\}$$

33

Euler Circuits

Now, let's go back to the puzzle of the Königsberg bridges.



Is it possible to find a route through the graph that starts and ends at some vertex, one of A, B, C , or D , and traverses each edge exactly once and each vertex at least once?

34

Euler Circuits

Definition: Euler Circuit

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, and uses every edge of G exactly once.

Definition: Eulerian Graph

An **Eulerian graph** is a graph that contains an Euler circuit.

35

Euler Circuits

Theorem 12.2.1

If a graph has an Euler circuit, then every vertex of the graph has positive even degree.

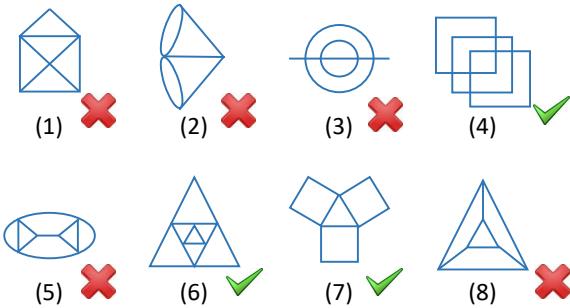
Contrapositive Version of Theorem 12.2.1

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

36

Euler Circuits

Does each of the following graphs have an Euler circuit?



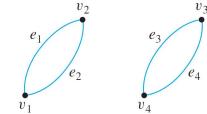
37

Euler Circuits

Is the converse of Theorem 12.2.1 true?

If every vertex of a graph has even degree, then the graph has an Euler circuit.

Not true!



38

Euler Circuits

Theorem 12.2.2

A graph G has an Euler circuit if, and only if, G is connected and every vertex of G has positive even degree.

The proof of this theorem is not within our scope.

A corollary to Theorem 12.2.2 gives a criterion for determining when it is possible to find a walk from one vertex of a graph to another, passing through every vertex of the graph at least once and every edge of the graph exactly once.

39

Euler Path

Definition: Euler Path

Let G be a graph, and let v and w be two distinct vertices of G . An **Euler path from v to w** is a sequence of adjacent edges and vertices that starts at v , ends at w , passes through every vertex of G at least once, and traverses every edge of G exactly once.

Corollary 12.2.3

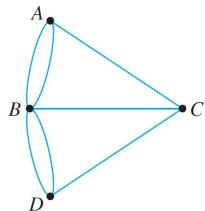
Let G be a graph, and let v and w be two distinct vertices of G . There is an Euler path from v to w if, and only if, G is connected, v and w have odd degree, and all other vertices of G have positive even degree.

The proof of this corollary is not within our scope.

40

Back to Königsberg bridges

Is it possible to find a route through the graph that starts and ends at some vertex, one of A, B, C, or D, and traverses each edge exactly once and every vertex at least once?



Now, we should be able to answer.

Ans: No as there is no Euler circuit exist in this graph.

41

42

3: Isomorphisms of Graphs

Isomorphisms of Graphs

The two drawings shown in Figure 12.3.1 both represent the [same graph](#): Their vertex and edge sets are identical, and their edge-endpoint functions are the same. Call this graph G .

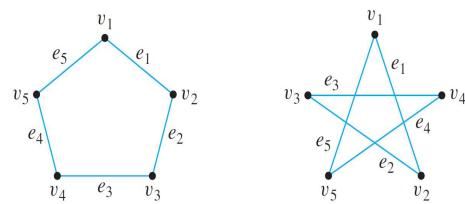


Figure 12.3.1

43

Isomorphisms of Graphs

Now consider the graph G' represented in Figure 12.3.2.

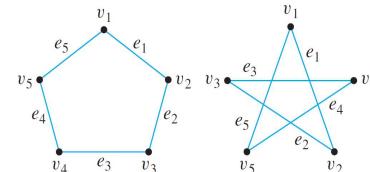


Figure 12.3.1

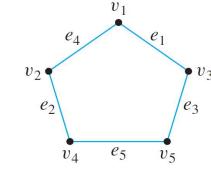


Figure 12.3.2

Observe that G' is a different graph from G (for instance, in G the endpoints of e_1 are v_1 and v_2 , whereas in G' the endpoints of e_1 are v_1 and v_3).

44

Isomorphisms of Graphs

Yet G' is certainly very similar to G . In fact, if the vertices and edges of G' are relabeled by the functions shown in Figure 12.3.3, then G' becomes the same as G .

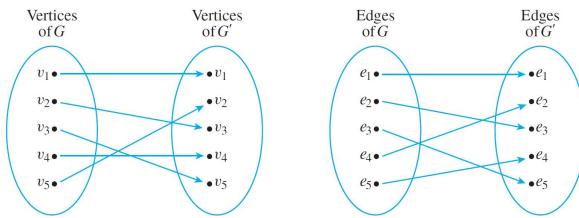


Figure 12.3.3

Note that these relabeling functions are **one-to-one** and **onto**.

45

Isomorphisms of Graphs

Two graphs that are the same except for the labeling of their vertices and edges are called **isomorphic**.

Definition: Isomorphic Graph

Let G and G' be graphs with vertex sets $V(G)$ and $V(G')$ and edge sets $E(G)$ and $E(G')$ respectively.

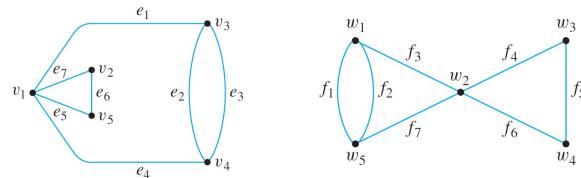
G is isomorphic to G' if, and only if, there exist one-to-one correspondences $g: V(G) \rightarrow V(G')$ and $h: E(G) \rightarrow E(G')$ that preserve the edge-endpoint functions of G and G' in the sense that for all $v \in V(G)$ and $e \in E(G)$,

$$v \text{ is an endpoint of } e \Leftrightarrow g(v) \text{ is an endpoint of } h(e).$$

46

Isomorphisms of Graphs

Example: Show that the following two graphs are isomorphic.

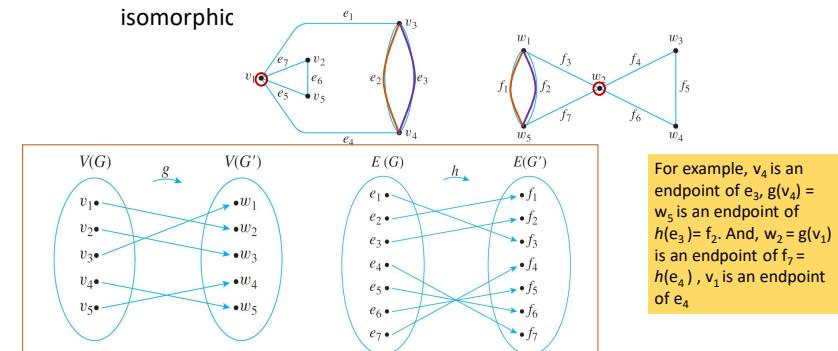


To solve this, you find functions $g: V(G) \rightarrow V(G')$ and $h: E(G) \rightarrow E(G')$ such that for all $v \in V(G)$ and $e \in E(G)$, v is an endpoint of e if, and only if, $g(v)$ is an endpoint of $h(e)$.

47

Isomorphisms of Graphs

Example: Show that the following two graphs are isomorphic



48

4: Tree and Minimum Spanning Tree

Definition

Definition: Tree

A **graph** is said to be **circuit-free** if, and only if, it has no circuits.

A graph is called a **tree** if, and only if, it is circuit-free and connected.

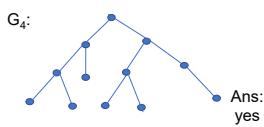
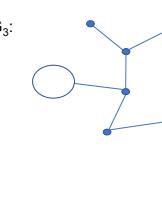
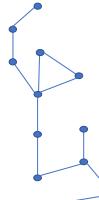
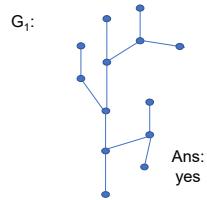
A **trivial tree** is a graph that consists of a single vertex.

49

50

Trees: Examples

Which of the following are trees?

 $G_5:$

Ans: yes

$G_6:$

Ans: no

Characterizing Trees

Theorem 12.4.1

Any tree with n vertices ($n > 0$) has $n - 1$ edges.

Theorem 12.4.2

If G is a connected graph with n vertices and $n - 1$ edges, then G is a tree.

The proof of these theorems are not within our scope

Example: Draw a graph with 5 vertices and 4 edges, but which is not a tree.

52

Definitions

Definition: Spanning Tree

A **spanning tree** for a graph G is a subgraph of G that contains every vertex of G and is a tree.

Proposition 12.4.3

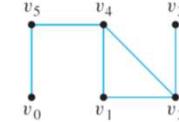
1. Every connected graph has a spanning tree.
2. Any two spanning trees for a graph have the same number of edges.

The proof of this proposition is not within our scope

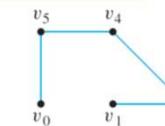
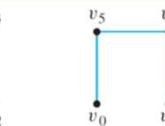
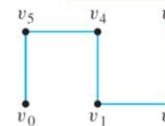
53

Definitions

Example: Find all spanning trees for the graph G below.



The graph G has one circuit $v_2v_1v_4v_2$ and removal of any edge of the circuit gives a tree. Hence there are three spanning trees for G .



54

Minimum Spanning Trees

Definitions: Weighted Graph, Minimum Spanning Tree

A **weighted graph** is a graph for which each edge has an associated positive real number **weight**. The sum of the weights of all the edges is the **total weight** of the graph.

A **minimum spanning tree** for a connected weighted graph is a spanning tree that has the least possible total weight compared to all other spanning trees for the graph.

If G is a weighted graph and e is an edge of G , then $w(e)$ denotes the weight of e and $w(G)$ denotes the total weight of G .

55

Example and Application of Minimum Spanning Tree

An East Coast airline company wants to expand service to the Midwest and has received permission from the Federal Aviation Authority to fly any of the routes shown in Figure 12.4.1.

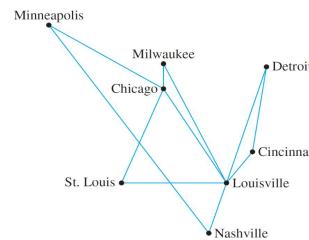


Figure 12.4.1

56

Example and Application of Minimum Spanning Tree

The graph of the routes allowed by the Federal Aviation Authority shown in Figure 12.4.1 can be annotated by adding the distances (in miles) between each pair of cities.

Now suppose the airline company wants to serve all the cities shown, but with a route system that minimizes the total mileage.

This problem can be solved by finding minimum spanning trees in the graph shown in Figure 12.4.2.

Finding minimum spanning tree is certainly an important problem. It is in general a hard problem. We need good algorithm to find minimum spanning trees.

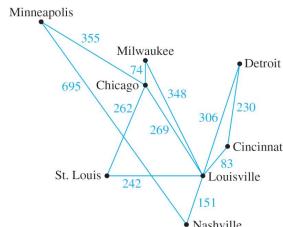


Figure 12.4.2

57

Kruskal's Algorithm (Joseph B. Kruskal, 1956)

In **Kruskal's algorithm**, the edges of a connected weighted graph are examined one by one in order of increasing weight.

At each stage the edge being examined is added to what will become the minimum spanning tree, provided that this addition does not create a circuit.

After $n - 1$ edges have been added (where n is the number of vertices of the graph), these edges, together with the vertices of the graph, form a minimum spanning tree for the graph.

58

Kruskal's Algorithm

Example: Describe the action of Kruskal's algorithm on the graph shown in Figure 12.4.3, where $n = 8$.

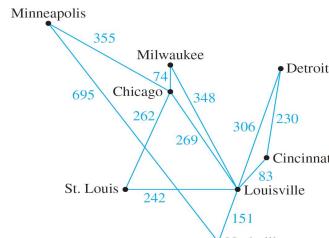


Figure 12.4.3

59

Kruskal's Algorithm

Using Kruskal's algorithm we can formulate the following table.

	Edge considered	Wt	Action taken
1	Chi – Mil	74	added
2	Lou – Cin	83	added
3	Lou – Nas	151	added
4	Cin – Det	230	added
5	StL – Lou	242	added
6	StL – Chi	262	added
7	Chi – Lou	269	not added
8	Lou – Det	306	not added
9	Lou – Mil	348	not added
10	Min – Chi	355	added

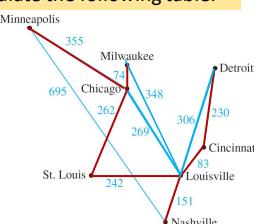


Figure 12.4.3

When Kruskal's algorithm is used on a graph in which some edges have the same weight as others, more than one minimum spanning tree can occur as output.

60

Prim's Algorithm (Robert C. Prim, 1957)

Prim's algorithm works differently from Kruskal's.

In this algorithm we build a minimum spanning tree T for a connected graph of n vertices by starting from any vertex, and at each stage, we add one new edge of least weight that connects one vertex in T and another vertex not in T until T has $(n-1)$ edges.

Prim's Algorithm

Example: Describe the action of Prim's algorithm on the graph shown in Figure 10.4.4, using the Minneapolis vertex as a starting point.

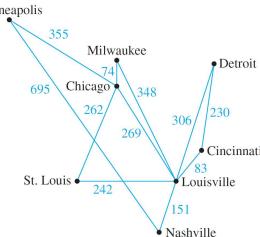


Figure 12.4.4

61

62

Prim's Algorithm

Using Prim's algorithm we can formulate the following table.

	Vertex added	Edge added	Weight
0	Minneapolis		
1	Chicago	Min - Chi	355
2	Milwaukee	Chi - Mil	74
3	St. Louis	Chi - StL	262
4	Louisville	StL - Lou	242
5	Cincinnati	Lou - Cin	83
6	Nashville	Lou - Nas	151
7	Detroit	Cin - Det	230

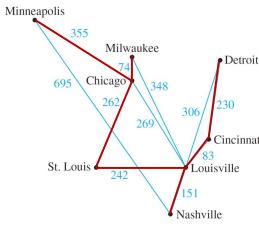


Figure 12.4.4

Same as Kruskal's algorithm, when prim's algorithm is used on a graph in which some edges have the same weight as others, more than one minimum spanning tree can occur as output.

63

Correctness of Kruskal's and Prim's Algorithms

It can be proved that both algorithms are always correct --- always output a minimum spanning tree.

The proof of their correctness is not within our scope.

64

End of Unit 12