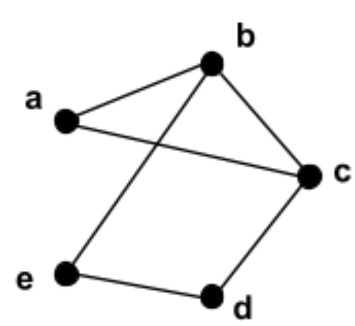
13.1 Introduction to graphs

Graphs are fundamental objects in discrete mathematics that model relationships between pairs of objects. Graphs arise in a wide array of disciplines but play an especially important role in computer science.

Directed graphs were introduced in the context of relations. Here we are concerned with undirected graphs. In an ***undirected graph***, the edges are unordered pairs of vertices, which is useful for modeling relationships that are symmetric. For example, an undirected graph could be used to model sibling relationships within a family. Unlike parent/child relationships in which the two people have different roles, sibling relationships are symmetric. Two people are mutual siblings or neither one is the sibling of the other.

A graph consists of a pair of sets (V, E), where V is a set of vertices and E is a set of edges. A graph is ***finite*** if the vertex set is finite. This material will only be concerned with finite graphs. A single element of V is called a ***vertex*** and is usually represented pictorially by a dot with a label. Each edge in E is a set of two vertices from V and is drawn as a line connecting the two vertices. In the graph below, the vertex set is V = {a, b, c, d, e}. The graph has six edges.

Figure 13.1.1: An undirected graph.



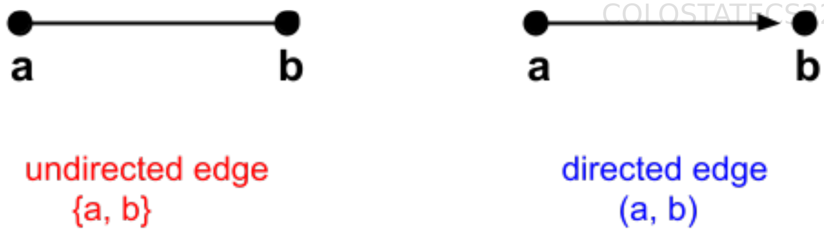
Two of the edges cross each other but there is no vertex at the crossing. The crossing is just a byproduct of how the graph is drawn on a two-dimensional surface. The edge connecting vertex a and vertex b is denoted as {a, b}. The graph drawn above can also be described by listing the vertex set and the edge set:

$$V = \{a, b, c, d, e\}$$

$$E = \{\{a, b\}, \{a, c\}, \{b, c\}, \{b, e\}, \{c, d\}, \{d, e\}\}$$

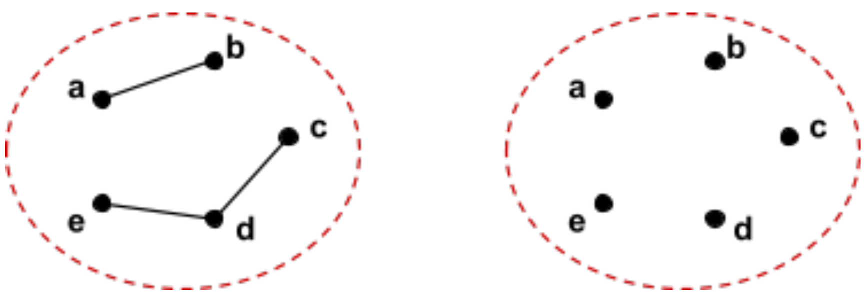
Curly braces are used to denote the vertices participating in an edge. Recall that curly braces denote a set in which the order of the elements does not matter. In an undirected graph there is no particular direction to the edge. The edge {a, b} is the same as the edge {b, a}. By contrast, an edge in a directed graph would be denoted as an ordered pair (a, b). A directed edge is drawn with an arrow from the vertex a to the vertex b. The diagram below shows the undirected edge {a, b} and the directed edge (a, b).

Figure 13.1.2: An undirected and a directed edge.



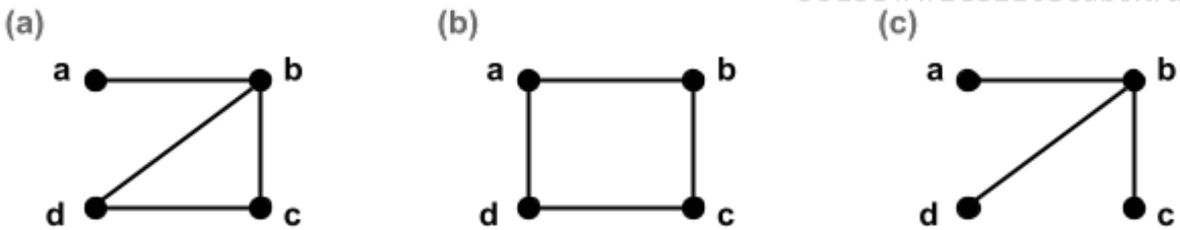
A graph may appear to be disconnected into more than one piece but is still considered to be one graph. It is also possible for a graph to have no edges at all. The diagram below shows two graphs whose vertex set is {a, b, c, d, e}. The edge set for the graph on the left is { {a, b}, {c, d}, {d, e} }. The edge set for the graph on the right is ∅.

Figure 13.1.3: Two graphs with five vertices.



PARTICIPATION ACTIVITY 13.1.1: Match the graph drawing with the corresponding edge set.

Use the three graphs below, all of which have a vertex set V = {a, b, c, d}.



Match the edge set to the corresponding picture of a graph.

If unable to drag and drop, refresh the page.

E = { {a, b}, {b, c}, {c, d}, {b, d} } **E = { {a, b}, {b, c}, {b, d} }**

E = { {a, b}, {b, c}, {c, d}, {a, d} }

(a)

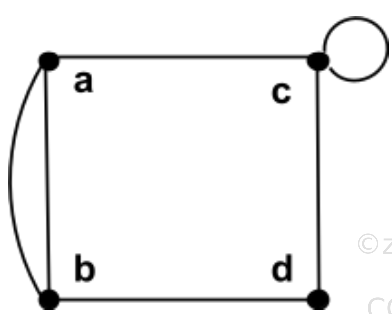
(b)

(c)

Reset

Parallel edges are multiple edges between the same pair of vertices. Imagine a graph whose vertex set is a set of cities and whose edges are roads connecting pairs of cities. It is possible for there to be two different roads between the same two cities. In defining graphs with parallel edges, it would be important to have an additional label besides the two endpoints to specify an edge in order to distinguish between different parallel edges. A graph can also have a ***self-loop*** which is an edge between a vertex and itself. The graph below has two parallel edges between vertices a and b. There is also a self-loop at vertex c.

Figure 13.1.4: A graph with parallel edges and a self-loop.

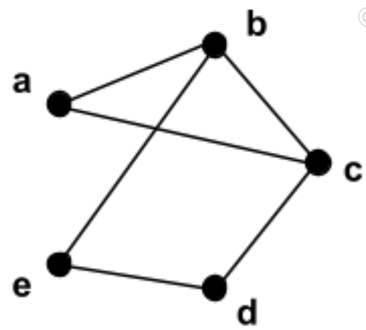


If a graph does not have parallel edges or self-loops, it is said to be a ***simple graph***. Unless otherwise specified, an undirected graph in this material is assumed to be a simple graph.

Graph terminology

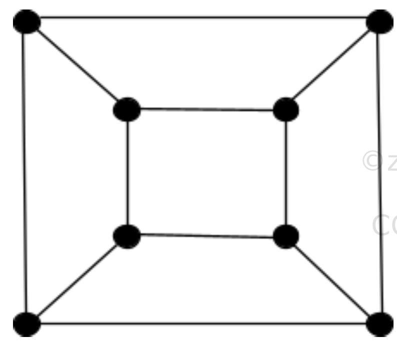
Here are some definitions for common graph terms. They are illustrated with respect to the graph below.

Figure 13.1.5: Undirected graph example.



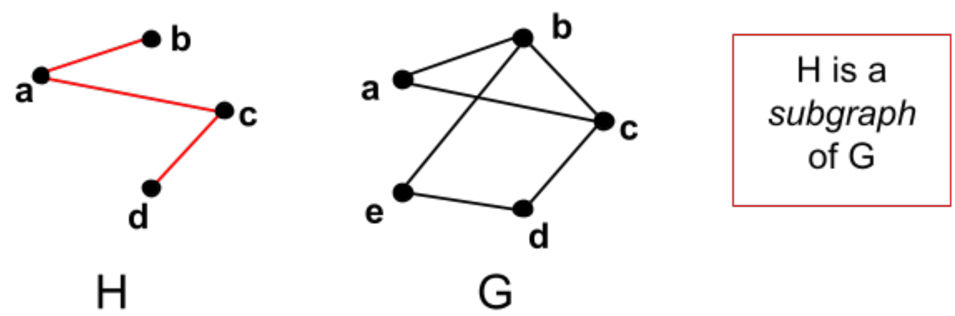
- If there is an edge between two vertices, they are said to be ***adjacent***. In the graph above, d and e are adjacent, but d and b are not adjacent.
- Vertices b and e are the ***endpoints*** of edge {b, e}. The edge {b, e} is ***incident*** to vertices b and e.
- A vertex c is a ***neighbor*** of vertex b if and only if {b, c} is an edge. In the graph above, the neighbors of b are the vertices a, c, and e.
- In a simple graph, the ***degree*** of a vertex is the number of neighbors it has. In the graph above, the degree of b is 3 and the degree of vertex a is 2. The degree of vertex b is denoted by deg(b).
- The ***total degree*** of a graph is the sum of the degrees of all of the vertices. The total degree of the graph above is 2 + 3 + 3 + 2 + 2 = 12.
- In a ***regular graph***, all the vertices have the same degree. In a ***d-regular graph***, all the vertices have degree d. The graph above is not regular because deg(a) ≠ deg(b). However the graph below is 3-regular.

Figure 13.1.6: A 3-regular graph.



- A graph H = (V_H, E_H) is a ***subgraph*** of a graph G = (V_G, E_G) if $V_H \subseteq V_G$ and $E_H \subseteq E_G$. Note that any graph G is a subgraph of itself. The diagram below shows a subgraph H of the graph G:

Figure 13.1.7: An example of a subgraph.



The animation below reviews the graph terminology presented so far:

PARTICIPATION ACTIVITY 13.1.2: Graph terminology review.

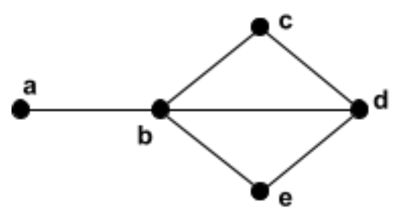
Animation captions:

1. Vertices a and c are adjacent because {a, c} is an edge. In the drawing of the graph, there is a line connecting vertices a and c.
2. Vertices a and b are not adjacent because there is no edge {a, b}. In the drawing of the graph, there is no line connecting vertices a and b.
3. Vertices e and c are the endpoints of the edge {e, c}. Edge {e, c} is incident to vertices c and e.
4. Vertices a, b, d, and e are all neighbors of vertex c. The degree of vertex c is 4. Vertex c has

4 neighbors.
5. The graph is not regular: deg(c) = 4 and deg(d) = 1. Therefore c and d do not have the same degree.
6. A graph is 2-regular if every vertex has degree 2.
7. A graph is 4-regular if every vertex has degree 4.

PARTICIPATION ACTIVITY 13.1.3: Graph terminology.

The questions below refer to the graph depicted below:



1) Is vertex c adjacent to vertex e?

Check Show answer

2) What is the degree of vertex d?

Check Show answer

3) Is the graph a regular graph?

Check Show answer

4) Is it correct to say that vertices d and e are incident?

Check Show answer

5) What is the total degree of the graph?

Check Show answer

6) Is the following graph a subgraph of the graph above: V' = {b, c, e}, E' = {{b, c}, {b, e}}?

Check Show answer

The field of graph theory seeks to understand the properties of graphs. Facts about graphs are useful in computer science because they help in designing efficient algorithms that operate on graphs. The following theorem below is a simple fact about undirected graphs. The proof uses induction as is common in proofs related to graphs. Typically, an inductive proof on a graph uses induction on the number of vertices or the number of edges. The proof below uses induction on the number of edges. The inductive step assumes that the theorem holds for graphs with m edges and proves that the theorem holds for graphs with m + 1 edges. The proof starts with a graph that has m + 1 edges and removes an edge. The inductive hypothesis states that the theorem holds for the graph with the edge removed. Then the proof shows that the theorem holds when the edge is added back into the graph.

Theorem 13.1.1: Number of edges and total degree.

Let G=(V, E) be an undirected graph. Then twice the number of edges is equal to the total degree:

$$\sum_{v \in V} deg(v) = 2 \cdot |E|$$

Proof 13.1.1: Number of edges and total degree.

Proof.
The proof is by induction on the number of edges in the graph. Let m = |E| be the number of edges in G.

Base case: m=0. If there are no edges in the graph then the degree of every vertex is 0.

$$\sum_{v \in V} deg(v) = 0 = 2m$$

Now assume that the theorem holds for all graphs with m edges. We will prove that the theorem holds with all graphs with m+1 edges. Consider a graph with m+1 edges. Consider an edge in the graph {a, b}. Remove the edge {a, b} so that the graph now has only m edges. Let deg(v) denote the degree of vertex v after the edge {a, b} has been removed. By induction, we know that the theorem holds for all graphs with m edges, so it holds after the edge {a, b} is removed from the graph:

$$\sum_{v \in V} deg(v) = 2m$$

When the edge {a, b} is added back into G, the degree of vertex a becomes deg(a) + 1 and the degree of vertex b becomes deg(b) + 1. The degree of all other vertices in G remain unchanged. Therefore, the new total degree is

$$2 + \sum_{v \in V} deg(v)$$

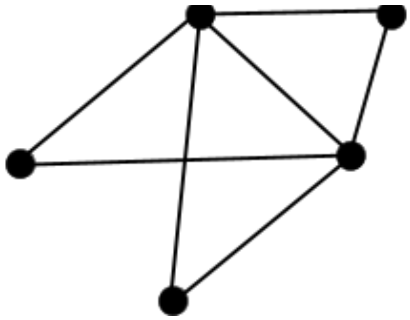
By the inductive hypothesis,

$$2 + \sum_{v \in V} deg(v) = 2 + 2m = 2(m + 1).$$

The first equality uses the inductive hypothesis (shown in red) by substituting 2m for $\sum_{v \in V} deg(v)$. Therefore, after the edge {a, b} is added back into the graph, the total degree is equal to 2(m + 1) which is twice the number of edges. ■

PARTICIPATION ACTIVITY 13.1.4: Induction on graphs.

1) What is the sum of the degrees of the vertices in the graph depicted below?



- 12
- 14
- 13

2) Which of the following statements describes the inductive step in a proof that proves a theorem about a graph by induction on the number of vertices?

- Assume that the theorem holds for a graph and then show that the theorem holds after a vertex is removed from the graph.
- Start with a graph and remove a vertex. Assume that the theorem holds for the graph with the vertex removed and prove that the theorem holds for the graph when the vertex is put back in.

Here are some examples for how undirected graphs model relationships between pairs of entities in a variety of application domains.

Example 13.1.1: Undirected graphs in applications.

- Molecular graphs: the vertices of the graph are the atoms in a molecule. There is an edge between two atoms if they form a bond. For molecular graphs, vertices are labeled with the type of atom they represent and edges are labeled with the type of bond they represent. The structure of a molecular graph reveals important information about the chemical properties of the molecule.
- Scheduling constraints: consider a school that has to schedule a set of courses for a given semester. The vertices of the graph represent the courses to be scheduled. There is an edge between two vertices if the corresponding courses have a conflict. For example, if the meeting times of the courses have already been determined, two courses that have overlapping times can not be scheduled in the same room. There would be an edge between any two courses whose meeting times overlap.
- Communication network: the vertices of a graph denote switches in a communication network. There is an edge between two switches if there is a two-way communication link between the two switches. Network designers would like to design a communication network such that even if a few communication links fail, it is still possible for every switch in the network to send a message to every other switch.
- Social network: consider a graph whose vertices represent individuals. There is an edge between two people if they are acquainted. (Assume that the property of being acquainted is mutual: if person A is acquainted with person B, then person B is acquainted with person A). Sociologists study social networks to understand how information spreads in communities or how societies evolve. The advent of online social networks has allowed social scientists to study social patterns on a much larger scale.

PARTICIPATION ACTIVITY 13.1.5: Constraint graphs.

Consider the set of classes and their corresponding times:

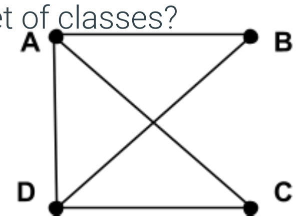
- Class A: MWF, 3:00PM - 5:00PM
- Class B: W, 2:00PM - 4:00PM
- Class C: F, 4:30PM - 5:30PM
- Class D: MWF, 2:30 - 3:30PM

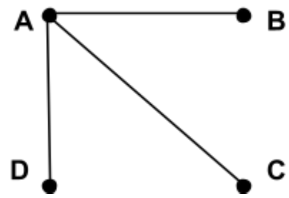
©zyBooks 12/15/22 00:30 1361995
John Farrell
COLOSTATECS220SeaboltFall2022

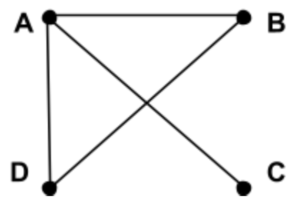
Two classes conflict if their meeting times overlap.

1) Which of the following graphs

correspond to the constraint graph for the set of classes?

○ 

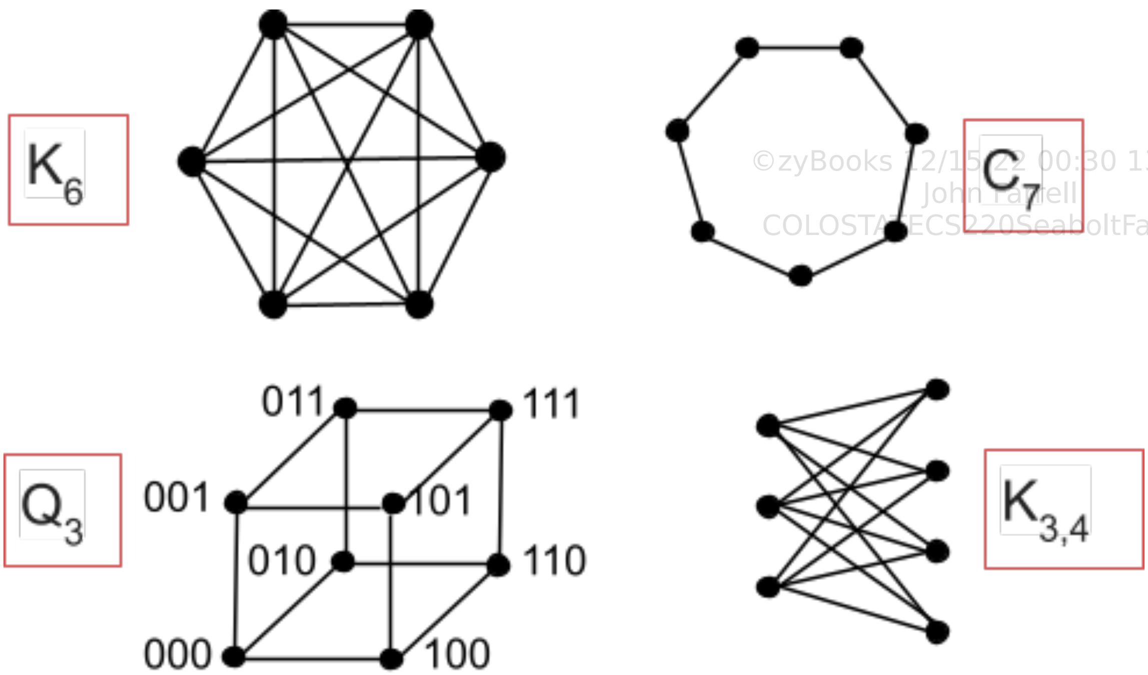
○ 

○ 

Common graphs

Some graphs with special structure are given their own name and notation because they come up frequently in graph theory. The figure below shows some common examples. The general definition for each class is given below:

Figure 13.1.8: Common graphs in graph theory.



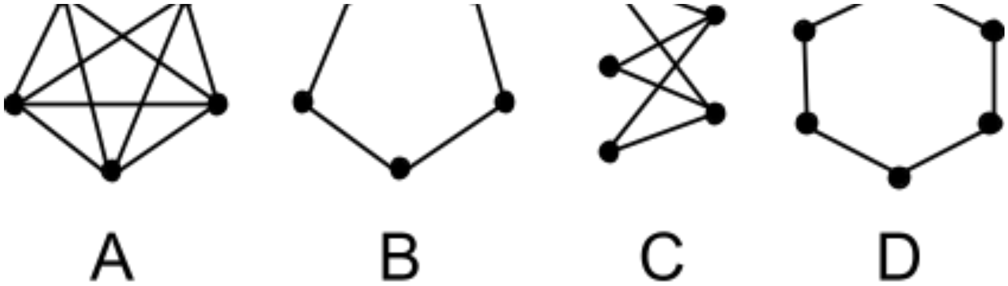
The following graphs are parameterized by a positive integer n. In the case of $K_{n,m}$, n and m must both be positive integers.

- K_n is called the ***complete graph*** on n vertices. K_n has an edge between every pair of vertices. The figure shows K_6. K_n is sometimes called a ***clique*** of size n or an n-clique.
- C_n is called a cycle on n vertices. The edges connect the vertices in a ring. The picture above depicts C_7. Note that C_n is well defined only for $n \geq 3$.
- The n-dimensional hypercube, denoted Q_n, has 2^n vertices. Each vertex is labeled with an n-bit string. Two vertices are connected by an edge if their corresponding labels differ by only one bit. For example in a 5-dimensional hypercube, the vertex labeled 11001 would have an edge to 11011 because the two strings only differ in the 4^{th} location. The figure above shows a diagram of the 3-dimensional hypercube
- $K_{n,m}$ has n+m vertices. The vertices are divided into two sets: one with m vertices and one set with n vertices. There are no edges between vertices in the same set, but there is an edge between every vertex in one set and every vertex in the other set. The figure shows a diagram of $K_{3,4}$.

©zyBooks 12/15/22 00:30 1361995
John Farrell
COLOSTATECS220SeaboltFall2022

PARTICIPATION ACTIVITY 13.1.6: Common graphs and their notation.

Match each graph with its notation.



If unable to drag and drop, refresh the page.

$K_{3,2}$ C_5 C_6 K_5

A

B

C

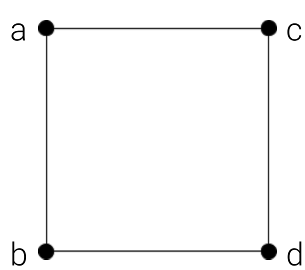
D

Reset

CHALLENGE ACTIVITY 13.1.1: Introduction to graphs.

422102.2723990.qx3zqy7

Start



What is the graph's vertex set V?

V = { Ex: w, x }

What is the graph's edge set E?

E = { Ex: {w, x}, {x, y} } Put commas between {x, y} values.

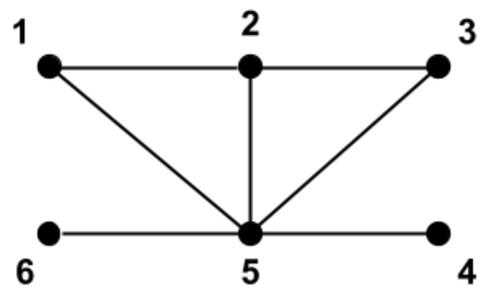
1	2	3

Check Next

Additional exercises

EXERCISE 13.1.1: Graph definitions.

A graph G is depicted in the diagram below.



(a) What is the total degree of G?

(b) List the neighbors of vertex 5.

(c) What is the degree of vertex 6?

(d) Which vertices are adjacent to vertex 3?

(e) Is G a regular graph? Why or why not?

(f) Is K_3 a subgraph of G? If so, name the vertices in the subgraph.

(g) Is K_4 a subgraph of G? If so, name the vertices in the subgraph.

EXERCISE | 13.1.2: Bounds on the number of edges in a graph.

(a) Let G be an undirected graph with n vertices. Let Δ(G) be the maximum degree of any vertex in G, δ(G) be the minimum degree of any vertex in G, and m be the number of edges in G. Prove that

$$\frac{\delta(G)n}{2} \leq m \leq \frac{\Delta(G)n}{2}$$

EXERCISE | 13.1.3: Reasoning about regular graphs.

(a) Is it possible to have a 3-regular graph with five vertices? If such a graph is possible, draw an example. If such a graph is not possible, explain why not.

(b) Is it possible to have a 3-regular graph with six vertices? If such a graph is possible, draw an example. If such a graph is not possible, explain why not.

EXERCISE | 13.1.4: Reasoning about common graphs.

(a) How many edges are in $K_{3,4}$? Is $K_{3,4}$ a regular graph?

(b) How many edges are in K_5? Is K_5 a regular graph?

(c) What is the largest n such that $K_n = C_n$?

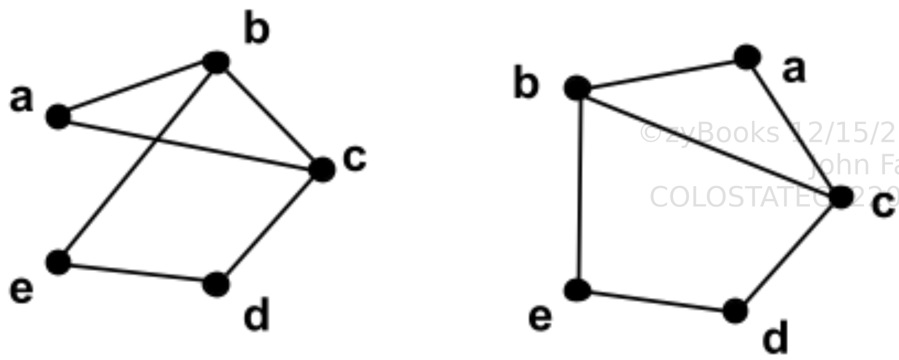
(d) For what value of n is $Q_2 = C_n$?

(e) Is Q_n a regular graph for $n \geq 1$? If so, what is the degree of the vertices in Q_n?

13.2 Graph representations

Below are drawings of two graphs:

Figure 13.2.1: Two drawings of an undirected graph.



The two graphs look different, but that is only because they are drawn differently. The two graphs are actually the same graph because they have the same vertex and edge sets as shown below:

$$V = \{a, b, c, d, e\}$$

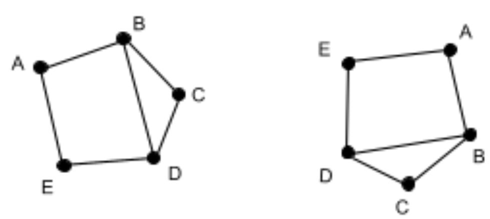
$$E = \{\{a, b\}, \{a, c\}, \{b, c\}, \{b, e\}, \{c, d\}, \{d, e\}\}$$

The way a graph is drawn on a 2-dimensional surface (paper or screen) is not part of the graph itself.

PARTICIPATION ACTIVITY 13.2.1: Identifying identical graphs.

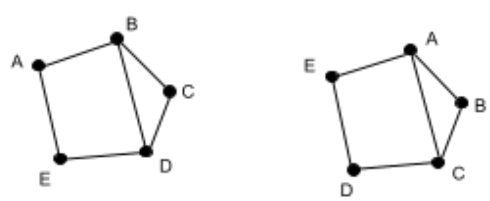
Indicate whether the graphs shown are the same graph.

1)



- Same
- Different

2)



- Same
- Different

A drawing is a convenient way to represent a small graph for a person to visualize. However, drawings can be deceptive in that identical graphs can be made to look very different if they are drawn in different ways. If a graph is part of the input to a computer program, a more formal specification of a graph is required. One way to represent a graph unambiguously is to list the edges. However, a list of the edges in the graph is not the most convenient way for a program to access the information about the graph. A typical step in an algorithm operating on a graph would be to determine whether two vertices are connected by an edge or to list all the neighbors of a given vertex. Each of these steps would require scanning the entire list of edges. There are two common ways to represent graphs for use in algorithms that make the information about the graph more accessible.

Adjacency list representation for graphs

In the ***adjacency list representation*** of a graph, each vertex has a list of all its neighbors. Note that since the graph is undirected if vertex a is in b's list of neighbors, then b must also be in a's list of neighbors. The animation below shows the adjacency list representation for a small example:

PARTICIPATION ACTIVITY 13.2.2: Adjacency list representation.

Animation captions:

1. Adjacency list representation. Each vertex has a list of neighbors. a's neighbors are b and c, so vertices b and c are in a's list.
2. b's neighbors are a, c, and e, so vertices a, c, and e are in b's list.
3. c's neighbors are a, b, and d, so vertices a, b, and d are in c's list. d's list contains neighbors are c and e. e's list contains neighbors are b and d.
4. Is b adjacent to c? Scan b's list to look for c. Vertex c is found in b's list, so yes, b is adjacent to c. Worst case time to scan b's list is proportional to deg(b).

If a graph is represented using adjacency lists, the time required to list the neighbors of a vertex v is proportional to deg(v), the number of vertices to be listed. In order to determine if {a, b} is an edge, it is necessary to scan the list of a's neighbors or the list of b's neighbors. In the worst case, the time required is proportional to the larger of deg(a) or deg(b).

Matrix representation for graphs

The ***matrix representation*** for a graph with n vertices is an n by n matrix whose entries are all either 0 or 1, indicating whether or not each edge is present. If the matrix is labeled M, then $M_{i,j}$ denotes the entry in row i and column j. For a matrix representation, the vertices of the graph are labeled with integers in the range from 1 to n. Entry $M_{i,j}$=1 if and only if {i, j} is an edge in the graph. Since the graph is undirected, $M_{i,j}$ = $M_{j,i}$ because {i, j} and {j, i} refer to the same edge which is either present in the graph or not. Thus the matrix representation of an undirected graph is symmetric about the diagonal, meaning that it is a mirror image of itself along the diagonal extending from the upper left corner to the lower right corner. The animation below illustrates.

PARTICIPATION ACTIVITY | 13.2.3: Matrix representation.

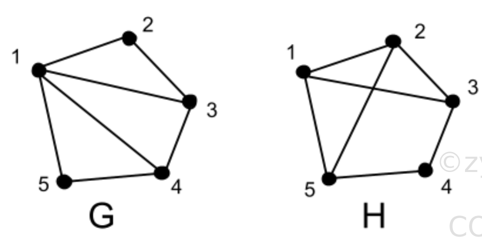
Animation captions:

1. The matrix representation of the given graph uses a 5 by 5 matrix because there are 5 vertices. There is an edge {1, 2}, so $M_{2,1} = M_{1,2} = 1.$
2. There is an edge {1, 3}, so $M_{3,1} = M_{1,3} = 1.$
3. There is an edge {2, 3}, so $M_{3,2} = M_{2,3} = 1,$ an edge {2, 5}, so $M_{5,2} = M_{2,5} = 1,$ an edge {3, 4}, so $M_{4,3} = M_{3,4} = 1,$ and an edge {5, 4}, so $M_{5,4} = M_{4,5} = 1.$
4. All other entries in the matrix are set to 0.
5. The matrix representation of an undirected graph is symmetric about the diagonal.
6. Is 2 adjacent to 5? Look at $M_{2,5}$, the entry in row 2, column 5. $M_{2,5} = 1$, so the answer is yes. O(1) time to answer.
7. To determine all the neighbors of vertex 5, it is necessary to scan all of row 5, which takes time proportional to the number of vertices.

If a graph is represented with a matrix, determining if {i, j} is an edge only requires examining matrix element $M_{i,j}$ which can be done in O(1) time. In order to list the neighbors of a vertex j, it is necessary to scan all of row j. Each 1 encountered corresponds to a neighbor of vertex j. The time required to list the neighbors of vertex j is Ω(n) time.

PARTICIPATION ACTIVITY | 13.2.4: Graph representations.

The diagram below shows two graphs.



1) Which of the two graphs above corresponds to the following matrix representation? (The rows and columns of the matrix are numbered 1 through 5).

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

- ○ G
- ○ H

2) Which of the two graphs above corresponds to the following adjacency list representation?

1	→ 2, 3, 5
2	→ 1, 3, 5
3	→ 1, 2, 4
4	→ 3, 5
5	→ 1, 2, 4

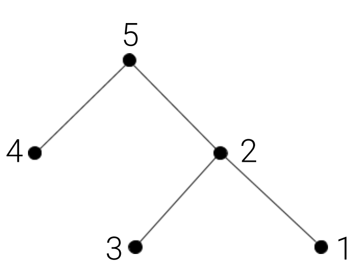
- ○ G
- ○ H

CHALLENGE ACTIVITY 13.2.1: Graph representations.

422102.2723990.qx3zqy7

Start

Fill in the missing entries to complete the matrix representation of the given graph.



$$\begin{array}{c|ccccc} & 1 & 2 & 3 & 4 & 5 \\ \hline 1 & 0 & (a) & 0 & 0 & 0 \\ 2 & (b) & 0 & 1 & 0 & 1 \\ 3 & 0 & 1 & 0 & 0 & 0 \\ 4 & 0 & 0 & 0 & 0 & (c) \\ 5 & 0 & 1 & 0 & 1 & (d) \end{array}$$

(a): Ex: 0/1

(b):

(c):

(d):

1	2	3	4

Check Next

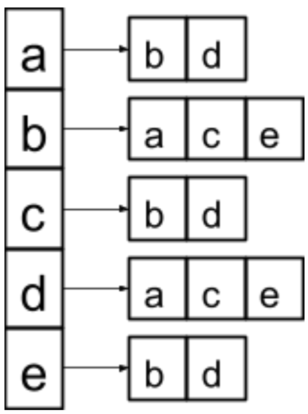
Additional exercises

EXERCISE 13.2.1: From graph representations to graph drawings.

Draw the graph based on its representation.

(a) V = {a, b, c, d}
E = {{a, b}, {a, d}, {b, c}, {b, d}, {c, d}}

(b)

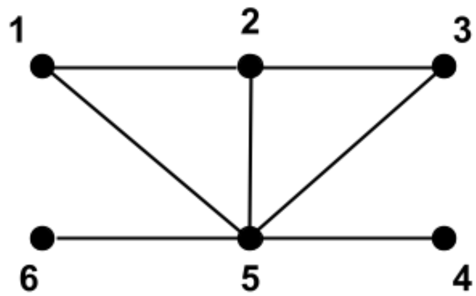


(c)

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

EXERCISE | 13.2.2: From a graph drawing to other graph representations.

A graph G is depicted in the diagram below.



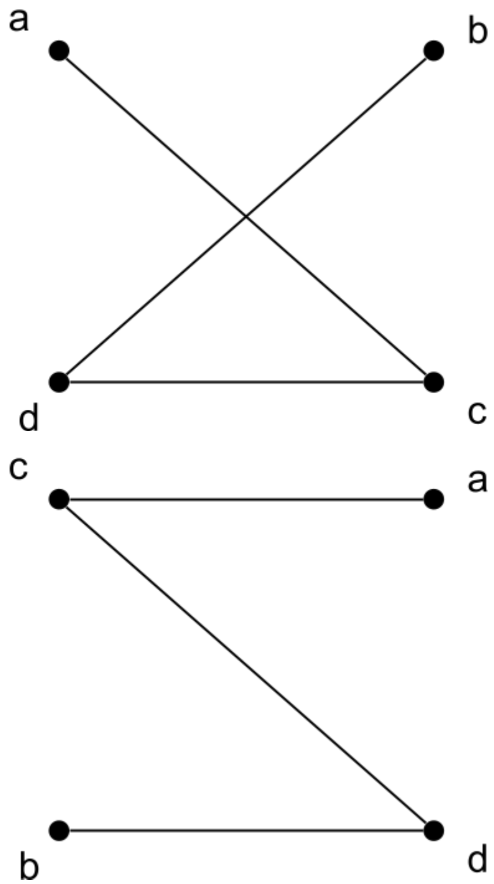
(a) Give the adjacency list representation of G.

(b) Give the matrix representation of G.

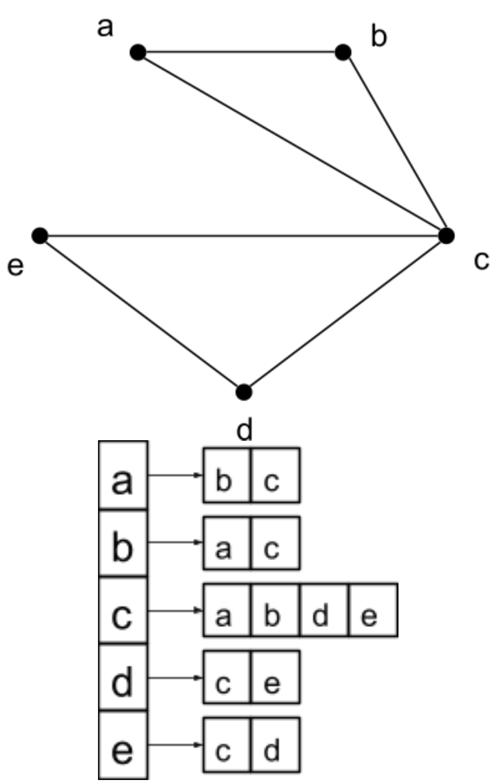
EXERCISE 13.2.3: Determining graph equality.

Indicate if the two graphs are equal.

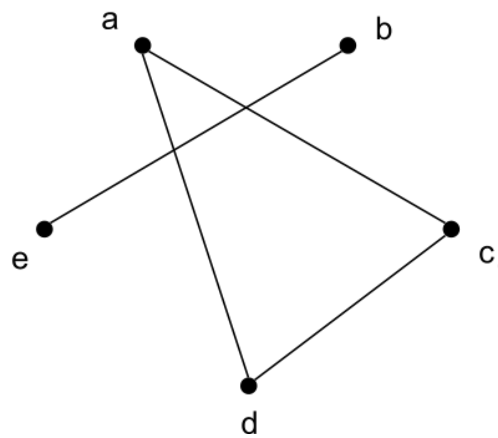
(a)



(b)



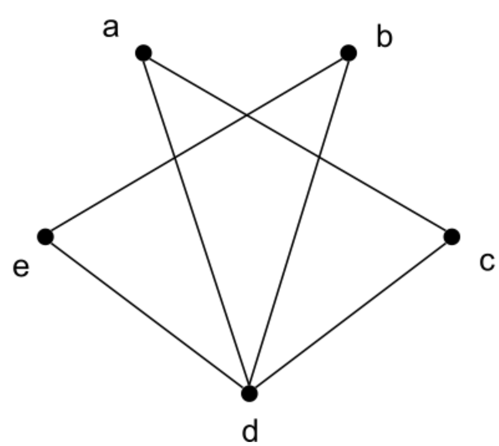
(c)



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

The rows and columns of the matrix represent vertices a, b, c, d, e, in order.

(d)



V = {a, b, c, d, e}
E = {{a, c}, {a, d}, {b, d}, {b, e}, {c, d}, {d, e}}

EXERCISE 13.2.4: Time complexity of graph operations with matrix and adjacency list representations.

Each question below describes an operation on a graph. Give the worst-case time complexity of performing the operation for a d-regular graph with a matrix representation and with an adjacency list representation. The number of vertices in the graph is n. Give your answer first for the case d = 4 and then for d = $\sqrt{n}$. The variable d should not be in your final expression.

With the adjacency list representation, assume you can find the list for a particular vertex

With the adjacency list representation, assume you can find the list for a particular vertex in O(1) time. Also the neighbors of a vertex are stored in an arbitrary order, so searching for a particular neighbor in a list takes time proportional to the length of the list in the worst case.

(a) Given a vertex i, list all the neighbors of vertex i.

(b) Find any neighbor of vertex i.

(c) Given a vertex i, list all the neighbors of the neigh

13.3 Walks, trails, circuits, paths, and cycles

Suppose a graph represents a road network with the vertices corresponding to intersections and the edges to roads that connect intersections. A natural way to use such a graph would be to plan routes from one point to another that pass through a series of intersections. In the language of graphs, a walk is a way to travel from one vertex to another by a series of hops along the undirected edges of the graph.

Definition 13.3.1: A walk in an undirected graph.

A ***walk*** from v_0 to v_l in an undirected graph G is a sequence of alternating vertices and edges that starts and ends with a vertex:

$$\langle v_0, \{v_0, v_1\}, v_1, \{v_1, v_2\}, v_2, \ldots, v_{l-1}, \{v_{l-1}, v_l\}, v_l\rangle$$

The vertices just before and after each edge are the two endpoints of that edge.

Since the edges in a walk are completely determined by the vertices, a walk can also be denoted by the sequence of vertices:

$$\langle v_0, v_1, \ldots, v_l\rangle.$$

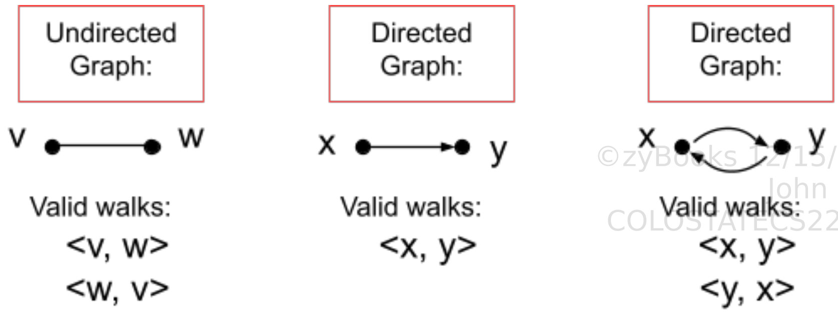
The sequence of vertices is a walk only if $\{v_{i-1}, v_i\} \in E$ for each i = 1, 2,...,l. Two consecutive vertices ..., v_{i-1}, v_i, ... in a walk represent an occurrence of the edge $\{v_{i-1}, v_i\}$ in the walk.

The ***length of a walk*** is l, the number of edges in the walk.

An ***open walk*** is a walk in which the first and last vertices are not the same. A ***closed walk*** is a walk in which the first and last vertices are the same.

The edges in an undirected graph do not have a particular orientation, so an edge can be traversed in either direction. If {v, w} is an edge in an undirected graph, then a walk can have vertex v followed by w or w followed by v. Thus, if the vertices in a walk are reversed, the resulting sequence is also a walk . By contrast, in directed graphs, the endpoints of an edge have a well-defined order. An edge (x, y) in a directed graph can only be traversed from x to y. If the graph happens to have edges (x, y) and (y, x) then a walk can go from x to y or from y to x. The idea is illustrated in the diagram below with walks of length 1:

Figure 13.3.1: Walks in directed and undirected graphs.



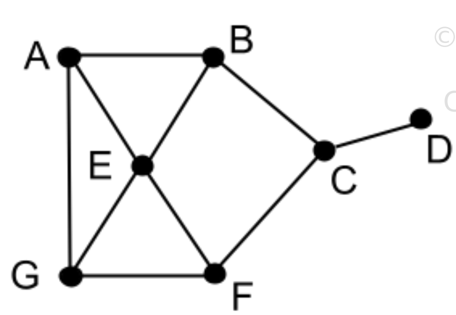
PARTICIPATION ACTIVITY | 13.3.1: Walks in an undirected graph.

Animation captions:

1. A walk in an undirected graph starts at vertex A and follows the edge {A, E}. The walk so far is $\langle A, E, \ldots$
2. Then the walk proceeds along edges {E, B} but can not continue from B to D because {B, D} is not an edge. $\langle A, E, B, D\rangle$ is not a walk.
3. The walk $\langle A, E, B, C\rangle$ is a walk of length 3. There are three edges in the walk.
4. The sequence $\langle A\rangle$ consisting of a single vertex is a walk of length 0. The walk has zero edges.
5. The walk $\langle A, E, C, B, E, A\rangle$ is a walk of length 5. There are five edges, including repetitions. Edge { A, E } occurs twice in the walk.
6. $\langle A, E, C, B, E, A\rangle$ is a closed walk because the first and last vertices are the same.

PARTICIPATION ACTIVITY | 13.3.2: Walks in undirected graphs.

Refer to the undirected graph provided below:



1) Which of the following sequences of vertices is not a walk in the graph?

- ⟨ A, B, E, F, E ⟩
- ⟨ A, B, E, B, C ⟩
- ⟨ A, B, C, E, G ⟩

2) What is the length of the walk: ⟨ A, B, E, G, E, B ⟩

- 3
- 5
- 6

3) What is the length of the walk ⟨ A, E, A ⟩?

- 1
- 2
- 3

4) Which of the following walks is a closed walk?

- ⟨ A, B, E, F, E, B ⟩
- ⟨ B, A, B, E, G, E, B ⟩
- ⟨ A, B, C, F ⟩

In many contexts, walks that do not repeat vertices or edges are preferable. For example in the road network example, a repeated vertex in a walk likely means that the driver is lost. The following list of terms define specific kinds of walks.

- A ***trail*** is an open walk in which no edge occurs more than once.
- A ***circuit*** is a closed walk in which no edge occurs more than once.
- A ***path*** is a trail in which no vertex occurs more than once.
- A ***cycle*** is a circuit of length at least 1 in which no vertex occurs more than once, except the first and last vertices which are the same.

Here are some examples of closed walks:

- ⟨ A, C, D, A ⟩ This closed walk is a circuit because no edge occurs more than once. The circuit is also a cycle because the only repeated vertices are the first and the last.
- ⟨ A, C, B, A, D, E, A ⟩ This closed walk is a circuit because no edge occurs more than once. The circuit is not a cycle because the vertex A appears in the middle of the circuit as well as at the

beginning and the end.

- ⟨ B, C, D, A, C, E, B ⟩ This closed walk is a circuit because no edge occurs more than once. The circuit is not a cycle because the vertex C is repeated in the middle.
- ⟨ A, B, A ⟩ This closed walk is not a cycle because the edge {A, B} occurs twice. A cycle is, by definition, also a circuit and a circuit does not have any repeated edges.

Note that paths and cycles, by definition, do not have any repeated edges. Therefore if a graph is simple (no self-loops or multiple edges between the same pair of vertices), any cycle must have length at least 3. The sequence ⟨ v ⟩ is not a cycle because a cycle, by definition, must have length at least 1. The sequence ⟨ v, v ⟩ is only a walk if there is a self-loop at vertex v. The sequence ⟨ v, w, v ⟩ uses the edges {v, w} twice.

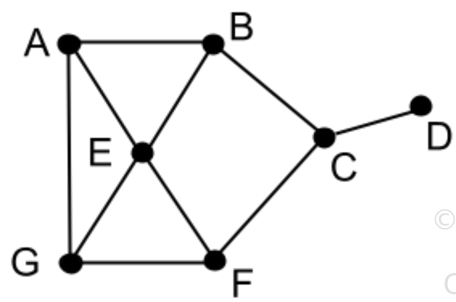
PARTICIPATION ACTIVITY 13.3.3: Paths and cycles in an undirected graph.

Animation captions:

1. The walk $\langle A, E, B, C\rangle$ is a path because the walk is open and there are no repeated vertices.
2. $\langle A, E, C, B, E\rangle$ is a walk because {A, E}, {E, C}, {C, B}, and {B, E} are all edges. The walk is a trail but not a path because vertex E is repeated.
3. $\langle B, C, E, B\rangle$ is a circuit (a walk whose first and last vertices are the same). The circuit is also a cycle because only the first and last vertices are repeated.
4. $\langle E, B, C, E, F, A, E\rangle$ is a circuit but is not a cycle because E is repeated and appears in the middle.

PARTICIPATION ACTIVITY 13.3.4: Walks, paths, circuits, and cycles in undirected graphs.

Refer to the undirected graph provided below:



1) Which of the following sequence of vertices is a circuit in the graph?

- ⟨ A, B, E, B ⟩
- ⟨ A, E, F, G, E, A ⟩
- ⟨ A, E, F, G, A ⟩
- ⟨ B, G, F, C, B ⟩

2) Which of the following sequence of vertices is a trail in the graph?

- ⟨ A, B, E, B ⟩
- ⟨ A, E, F, G, E, B, A ⟩
- ⟨ A, E, F, G, E, B ⟩

3) Is the following walk a path: ⟨ A, E, F, G, E ⟩?

- Yes
- No

4) Is the following closed walk a cycle: ⟨ A, B, C, F, E, A ⟩?

- Yes
- No

Additional exercises

EXERCISE | 13.3.1: Identifying paths, circuits, and cycles in a small graph.

The sequences given below are all walks in a graph. For each sequence give all the attributes from the list below that apply to that sequence.

- Open walk
- Closed walk
- Trail
- Circuit
- Path
- Cycle

(a) ⟨ A, B, C, D, A, C ⟩

(b) ⟨ A, B, C, D, B, A ⟩

(c) ⟨ A, C, D, E, B, A ⟩

(d) ⟨ A, C, D, E, C, B, A ⟩

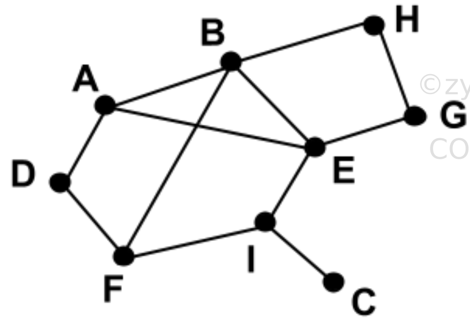
(e) ⟨ A, C, D, E, B ⟩

(f) ⟨ A, B, A ⟩

(g) ⟨ A ⟩

EXERCISE 13.3.2: Examples of walks, paths, circuits, and cycles in a small graph.

Refer to the undirected graph provided below:



(a) What is the maximum length of a path in the graph? Give an example of a path of that length.

(b) What is the maximum length of a cycle in the graph? Give an example of a cycle of that length.

(c) Give an example of an open walk of length five in the graph that is a trail but not a path.

(d) Give an example of a closed walk of length four in the graph that is not a circuit.

(e) Give an example of a circuit of length zero in the graph.

EXERCISE 13.3.3: Making a path from a walk.

(a) Prove that if there is an open walk in a graph that begins at vertex v and ends at vertex w, then there is a path in the graph that begins at vertex v and ends at vertex w.

EXERCISE 13.3.4: Paths that are also circuits or cycles.

(a) Is it possible for a path to also be a circuit? Explain your reasoning.

(b) Is it possible for a path to also be a cycle? Explain your reasoning.

EXERCISE | 13.3.5: Longest walks, paths, circuits, and cycles.

(a) What is the longest possible walk in a graph with n vertices?

(b) What is the longest possible path in a graph with n vertices?

(c) What is the longest possible cycle in a graph with n vertices?

13.4 Graph connectivity

Connected components

In an undirected graph, if there is a path from vertex v to vertex w, then there is also a path from w to v. The two vertices, v and w, are said to be ***connected***. A vertex is always considered to be connected to itself. If the graph represents a road or communication network, then it is very desirable for every pair of vertices to be connected. The property of being connected can be extended to sets of vertices and the entire graph:

- A set of vertices in a graph is said to be connected if every pair of vertices in the set is connected.
- A graph is said to be connected if every pair of vertices in the graph is connected, and is ***disconnected*** otherwise.

A disconnected graph can be divided into more than one connected component. The animation below illustrates the idea of a connected component. Then the formal definition is given afterwards.

A ***connected component*** consists of a maximal set of vertices that are connected as well as all the edges between any two vertices in the set. The word "maximal" means that if any vertex is added to a connected component, then the set of vertices will no longer be connected. A set of vertices can be used to specify a connected component since it is understood that all the edges between any two vertices in the set are included as well.

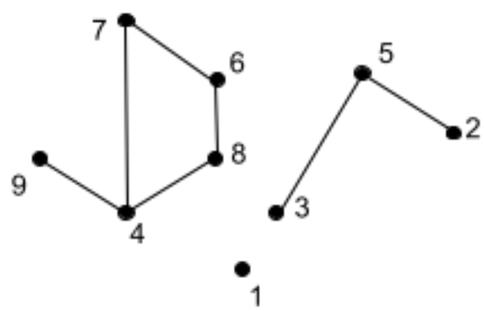
A vertex that is not connected with any other vertex is called an ***isolated vertex*** and is therefore a connected component with only one vertex.

PARTICIPATION ACTIVITY | 13.4.1: Connected components.

Animation captions:

1. D is connected to I in the graph because there is a path from D to I.
2. A is not connected to E because there is no path from A to E.
3. B is an isolated vertex because B is not connected to any other vertex.
4. The connected components are {A, D, I, F, H}, {C, E, G}, {B}. Vertices in the same connected component are connected to each other.
5. {A, D, F, H} is not a connected component because it is not maximal. Vertex I can be added and the vertices would still be mutually connected.

PARTICIPATION ACTIVITY 13.4.2: Connectivity.



1) Is vertex 8 connected to vertex 9?

- Yes
- No

2) Is vertex 8 connected to vertex 3?

- Yes
- No

3) Is {4, 6, 7, 8} a connected component of the graph?

- Yes
- No

4) Is {1} a connected component of the graph?

- Yes
- No

5) Is the graph connected?

- Yes
- No

k-connectivity

In some networks, it is important to be able to guarantee connectivity, even if one or more vertices or edges are removed from a graph. For example, in a communications network, switches (corresponding to vertices in the graph) can fail. In a road network, a road could become unavailable due to an emergency or heavy traffic. The definition of connectivity can be extended to encompass resilience to vertex or edge failures.

Definition 13.4.1: K-vertex-connected graph.

An undirected graph G is ***k-vertex-connected*** if the graph contains at least k + 1 vertices and remains connected after any k - 1 vertices are removed from the graph. The ***vertex connectivity*** of a graph is the largest k such that the graph is k-vertex-connected. The vertex connectivity of a graph G is denoted $\kappa(G)$.

The vertex connectivity of a graph is the minimum number of vertices whose removal disconnects the graph into more than one connected component.

When the graph is a complete graph, there is no set of vertices whose removal disconnects the graph. For the special case of K_n, the vertex connectivity $\kappa(K_n)$ is just defined to be n - 1.

Definition 13.4.2: K-edge-connected graph.

An undirected graph G is ***k-edge-connected*** if removing any k - 1 or fewer edges results in a connected graph. The ***edge connectivity*** of a graph is the largest k such that the graph is k-edge-connected. The edge connectivity of a graph G is denoted $\lambda(G)$.

The edge connectivity of a graph is the minimum number of edges whose removal disconnects the graph into more than one connected component.

The animation below illustrates vertex and edge connectivity:

PARTICIPATION ACTIVITY | 13.4.3: Vertex and edge connectivity.

Animation captions:

1. There are 2 vertices whose removal disconnects the graph. The graph is NOT 3-vertex-connected.
2. Removing any single vertex keeps the graph connected. The graph is 2-vertex-connected. The vertex connectivity of the graph is 2.
3. There are 3 edges whose removal disconnects the graph. The graph is NOT 4-edge-connected.
4. Removing any two edges keeps the graph connected. The graph is 3-edge-connected. The edge connectivity of the graph is 3.

Showing that a graph is k-edge-connected involves showing that removing any subset of k - 1 edges leaves the graph connected. If a graph has m edges, there are $\binom{m}{k-1}$ subsets with k - 1 edges. For large k, exhaustively searching all subsets would be infeasible. Fortunately, there are more efficient ways to determine the edge connectivity and vertex connectivity of a graph. The theorem below shows that the minimum degree of any vertex (which is easy to compute) is at least an upper bound for both the edge and vertex connectivity of a graph.

Theorem 13.4.1: Upper bound for vertex and edge connectivity.

Let G be an undirected graph. Denote the minimum degree of any vertex in G by $\delta(G)$. Then

$$\kappa(G) \le \delta(G) \quad \text{and}$$
$$\lambda(G) \le \delta(G).$$

Proof 13.4.1: Upper bound for vertex and edge connectivity.

Proof.

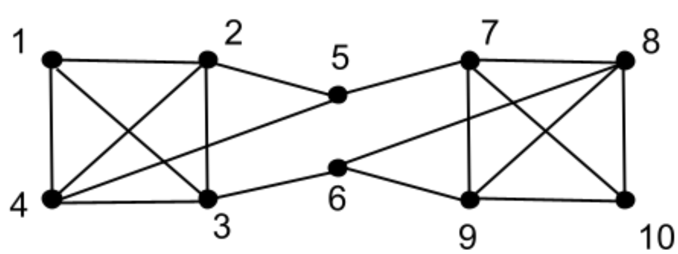
Let k = δ(G). Note that the graph must have at least k+1 vertices if there is a vertex with k neighbors. Since k is the minimum degree in the graph, there is at least one vertex whose degree is exactly k. Select one such vertex and call it v. Removing all k edges incident to v disconnects v from the rest of the graph. Therefore, G is not (k+1)-edge-connected and λ(G) ≤ δ(G).

If the graph has exactly k + 1 vertices and the minimum degree is k, then every vertex has an edge to all the other vertices and the graph is the complete graph on k + 1 vertices. By definition, κ(G) is exactly k = δ(G);. The remainder of the proof addresses the case that G has at least k + 2 vertices. Note that if the number of vertices is at least k+2 and there is a vertex with degree k, then the graph is not a complete graph. To prove that κ(G) ≤ k, we will show that there are k vertices whose removal disconnects the graph. Therefore, G can not be (k+1)-vertex-connected and the vertex connectivity of G is at most k.

Let v be a vertex that has exactly k neighbors. Removing the neighbors of v disconnects v from the rest of the graph. Since there are at least k + 2 vertices, there must be at least one other vertex in the rest of the graph besides v after the k vertices are removed. ■

PARTICIPATION ACTIVITY 13.4.4: K-connectivity.

Define G to be the following graph:



1) Is G 3-vertex-connected? Type: Yes or No

Check Show answer

2) Is G 3-edge-connected? Type: Yes or No

Check Show answer

3) What is the edge connectivity of the graph K_5?

Check Show answer

4) What is the vertex connectivity of the graph K_5?

Check Show answer

5) What is the vertex connectivity of the graph C_{10}?

Check Show answer

Additional exercises

EXERCISE | 13.4.1: Find the connected components of a graph.

Find the connected components of each graph.

(a) G = (V, E). V = {a, b, c, d, e, f, g, h, i, j}. E = { {f,h}, {e,d}, {c,b}, {i,j}, {a,b}, {i,f}, {f,j} }

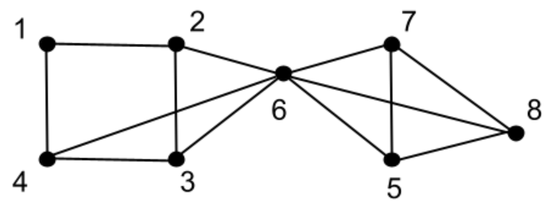
(b) G = (V, E). V = {a, b, c, d, e}. E = ∅

(c) G = (V, E). V = {a, b, c, d, e, f}. E = { {c, f}, {a, b}, {d, a}, {e, c}, {b, f} }

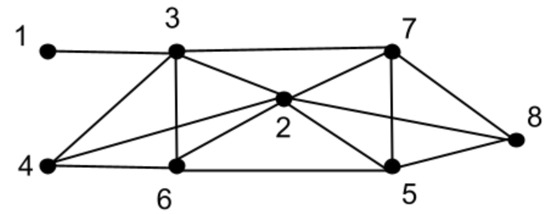
EXERCISE | 13.4.2: Determine the vertex and edge connectivity of a graph.

Determine the edge connectivity and the vertex connectivity of each graph.

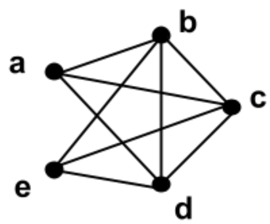
(a)



(b)



(c)



EXERCISE | 13.4.3: Examples in which vertex and edge connectivity is less than the minimum degree.

(a) Give an example of a graph in which the edge connectivity is strictly less than the minimum degree.

(b) Give an example of a graph in which the vertex connectivity is strictly less than the minimum degree.

EXERCISE 13.4.4: Edge connectivity between two vertices.

Two vertices v and w in a graph G are said to be 2-edge-connected if the removal of any edge in the graph leaves v and w in the same connected component.

(a) Prove that G is 2-edge-connected if every pair of vertices in G are 2-edge-connected.

(b) Is being 2-edge-connected a transitive property? That is, if v and w are 2-edge-connected and w and y are 2 edge-connected, can you conclude that v and y are 2-edge-connected? Prove your answer.
You can use the fact that if there is a walk from vertex v to vertex w, then there is a path from vertex v to vertex w.

(c) Two vertices v and w in a graph G are said to be 2-vertex-connected if the removal of any vertex in the graph besides v and w leaves v and w in the same connected component.
Is being 2-vertex-connected a transitive property? That is, if v and w are 2-vertex-connected and w and y are 2-vertex-connected, can you conclude that v and y are 2-vertex-connected? Justify your answer.
You can use the fact that if there is a walk from vertex v to vertex w, then there is a path from vertex v to vertex w.

EXERCISE 13.4.5: Connectivity is an equivalence relation.

(a) Show that the property of being connected is an equivalence relation. That is, for any undirected graph G and any two vertices, v and w, in G, the vertices v and w are related (v~w) if and only if there is a path from v to w. Show that the relation v~w is an equivalence relation.
You can use the fact that if there is a walk from vertex v to vertex w, then there is a path from vertex v to vertex w.

13.5 Euler circuits and trails

An ***Euler circuit*** in an undirected graph is a circuit that contains every edge and every vertex. Note that a circuit, by definition, has no repeated edges, so an Euler circuit contains each edge exactly once.

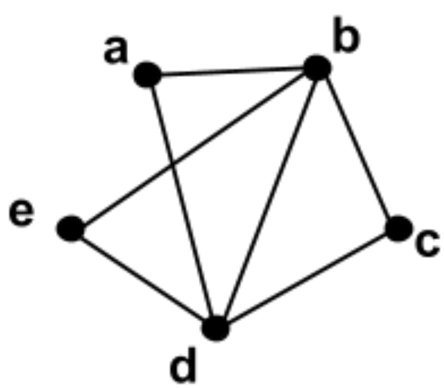
PARTICIPATION ACTIVITY | 13.5.1: Euler circuits in undirected graphs.

Animation captions:

1. The circuit ⟨ a, b, d, c, a ⟩ is not an Euler circuit in the graph because edges {a, d} and {b, c} are not in the circuit.
2. The closed walk ⟨ a, b, d, c, a, d, c, b, a ⟩ is not a circuit in the graph because edges {c, d} and {a, b} occur twice.
3. The closed walk ⟨ a, c, d, b, f, c, b, a, d, e, a ⟩ is an Euler circuit in a new graph because every vertex is reached and every edge occurs exactly once.

PARTICIPATION ACTIVITY | 13.5.2: Identify an Euler circuit.

1) Select the sequence that is an Euler circuit in the graph below



- $\langle a, b, c, d, e, a \rangle$
- $\langle a, b, c, d, a \rangle$
- $\langle a, b, c, d, e, b, d, a \rangle$
- $\langle a, b, c, d, e, b, a, d, b, a \rangle$

Not every graph has an Euler circuit. If a graph is not connected, then the graph can not have an Euler circuit because there would be no edges to get from one connected component to another along the circuit.

Also, if a graph has a vertex whose degree is odd, then the graph can not have an Euler circuit. Think of a person walking along the edges of an Euler circuit. Every time she arrives at a vertex v in the middle of the Euler circuit, there must be an edge incident to v, that she has not yet traversed. The edges incident to v can be paired up according to the order in which she travels: if she arrives at v along edge {u, v} and leaves v along edge {v, w} then the edges {u, v} and {v, w} are paired together. If there are an odd number of edges incident to v, then there will be a time when she arrives at v and has no edge along which she can leave v, implying that the graph does not have an Euler circuit. Thus, if the graph has an Euler circuit, then v must have even degree. (If v happens to be the vertex at the beginning and end of the Euler circuit, reorder the Euler circuit so that it begins and ends at some other vertex. The same argument can be used to show that v also must have even degree.)

Therefore, if an undirected graph G is not connected or has a vertex with odd degree, then G does not have an Euler circuit. An equivalent version of this statement is given as a theorem below.

Theorem 13.5.1: Required conditions for an Euler circuit in a graph.

If an undirected graph G has an Euler circuit, then G is connected and every vertex in G has an even degree.

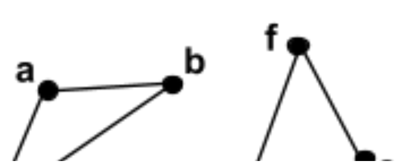
PARTICIPATION ACTIVITY 13.5.3: Odd degree vertex implies no Euler circuit.

Animation captions:

1. Start Euler circuit at x. Arrive at v by edge {x, v}. Leave v by edge {v, w}. Edges {x, v} and {v, w} are paired.
2. Continue Euler circuit ⟨ x, v, w, z, v, y, ... Arrive at v by edge {z, v}. Leave v by edge {v, y}. Edges {z, v} and {v, y} are paired.
3. Finish Euler circuit ⟨ x, v, w, z, v, y, z, x ⟩. Edges incident to v are paired. Vertex v has even degree.
4. Start a trail at vertex x on a new graph. Vertices v and z have odd degree. Try to pair edges incident to v in a walk starting at x.
5. Arrive at v by edge {x, v}. Leave v by edge {v, w}. Edges {x, v} and {v, w} are paired.
6. Continue trail ⟨ x, v, w, z, y, v, ... There are no unused edges to leave v. Vertex v has odd degree. No Euler circuit in the graph.

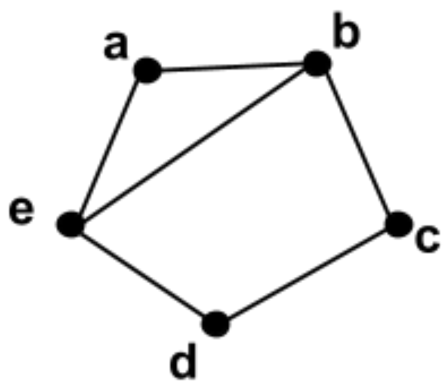
PARTICIPATION ACTIVITY 13.5.4: Graphs that do not have Euler circuits.

1) Identify the property which implies that the graph below does not have an Euler circuit.



- The graph is not connected.
- The graph has a vertex with an odd degree.
- All of the vertices in the graph have even degree.

2) Identify the property which implies that the graph below does not have an Euler circuit.



- The graph is not connected.
- The graph has a vertex with an odd degree.
- All of the vertices in the graph have even degree.

The required conditions for an Euler circuit say that if an undirected graph G has an Euler circuit, then G is connected and every vertex in G has an even degree. In fact, the converse of the theorem is also true:

Theorem 13.5.2: Sufficient conditions for an Euler circuit in a graph.

If an undirected graph G is connected and every vertex in G has an even degree, then G has an Euler circuit.

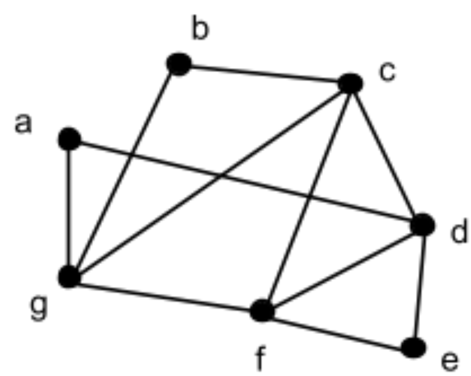
The required and sufficient conditions for an Euler circuit together give a complete characterization of graphs that have a Euler circuit.

Theorem 13.5.3: Characterization of graphs that have an Euler circuit.

An undirected graph G has an Euler circuit if and only if G is connected and every vertex in G has even degree.

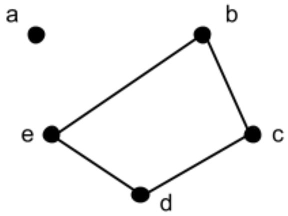
Figure 13.5.1: A graph that has an Euler circuit.

Every vertex in the graph below has even degree. Therefore according to the sufficient conditions for an Euler circuit, the graph must have an Euler circuit.



PARTICIPATION ACTIVITY 13.5.5: Selecting the graph with an Euler circuit.

1) Select the graph that has an Euler circuit.



Finding an Euler circuit and the proof of sufficient conditions for an Euler circuit

The sufficient conditions for an Euler circuit say that if an undirected graph G is connected and every vertex has even degree, then G has an Euler circuit. The proof of the theorem is an algorithm that shows a systematic way to find an Euler circuit in any graph that satisfies the conditions of the theorem.

We start with an easier task which is to find any circuit (not necessarily a Euler circuit) in an undirected graph. The circuit starts with any vertex that is not isolated. The procedure requires that all the vertices have even degree but does not require that the graph be connected.

Figure 13.5.2: Procedure to find a circuit in a graph.

```
Find a vertex w, that is not an isolated vertex.
Select any edge {w, x} incident to w. (Since w is not isolated, there is always at least one
such edge.)
Current trail T := ⟨w, x⟩
last := x
While there is an edge {last, y} that has not been used in T:
    Add y to the end of T
    last := y
```

The proof below establishes that if the degree of every vertex in the graph is even and the trail is open (meaning that the first and last vertices are not equal), then the conditions for the while loop are met and the process will continue. Therefore, the only way for the process to end is when the first and last vertices in the trail are equal. Thus, the procedure always terminates with a closed trail, which is a circuit.

Proof 13.5.1: Proof of correctness for the procedure to find a circuit in a graph.

Let G be an undirected graph in which every vertex has even degree. Let T be an open trail in G and let v be the last vertex in the trail. Then there is an edge in G incident to v that is not in the trail T.

Proof.
Let u be the second to last vertex in the trail. The edge {u, v} is in the trail. Since there are no self-loops, then u and v are not equal. Since v is not the first vertex in the trail, every other occurrence of v in the trail accounts for two more edges in the trail: the edge with the vertex preceding v and the edge with the vertex after v. For example if the sequence ⟨ ... x, v, y, ... ⟩ occurs in the trail, then {x, v} and {v, y} are both edges in the trail. Therefore, the number of edges in the trail incident to v must be odd: 1 for the last occurrence of v in the trail and 2 for every occurrence before the last. Since the degree of v in the graph is even, there must be at least one edge in the graph that is incident to v and is not included in the trail. ■

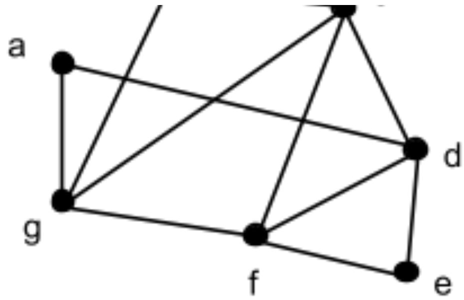
PARTICIPATION ACTIVITY | 13.5.6: Finding a circuit in a graph with no odd degree vertex.

Animation captions:

1. Start a walk at vertex a. Pick any edge incident to a.
2. At each step select any unused edge to leave the current vertex.
3. Eventually, the procedure gets stuck when there are no unused edges incident to the current vertex.
4. The procedure can only get stuck if the current vertex is the starting vertex, in which case the walk is a circuit.

PARTICIPATION ACTIVITY | 13.5.7: The procedure to find a circuit in a graph.

1) The algorithm to find a circuit in a graph is applied to the graph given below. The current trail T is ⟨ a, d, f, e ,d ⟩. Which edge is a candidate to be the next edge in the trail?



- {d, a}
- {d, c}
- {d, f}
- {e, f}

We now present a systematic way to find an Euler circuit in a graph. The requirements for the algorithm to work are that all the vertices in the graph have even degree and that the graph is connected.

The algorithm maintains a circuit C in the graph G. In each iteration, more edges are added to C while maintaining the property that C is a circuit. Since the number of edges in C increases after each iteration, C must eventually contain all the edges in the graph. G does not have any isolated vertices (otherwise G would not be connected). Therefore, a circuit C that visits every edge exactly once must also contain every vertex. Thus, at the end of the algorithm, C is an Euler circuit.

Figure 13.5.3: Procedure to find an Euler circuit in a graph.

Use the procedure described above to find any circuit in G. Call the circuit C. The algorithm continues to iterate the following steps until all the edges in G are included in C:

1. Remove all edges in C from G. Remove any isolated vertices from G. Call the resulting graph G'.
2. Find a vertex w that is in G' and C.
3. Find a circuit in G' that begins and ends with w. Call the circuit C'.
4. Combine circuit C and C'. Suppose C starts and ends at vertex v. Create a new circuit that starts at v and follows the edges in C until w is reached. The new circuit then follows the edges in C' back to w and then follows the rest of the edges in C back to v. The new circuit is renamed C for the next iteration.

In order to complete the proof, it is necessary to show that it is possible to execute steps 2 and 3 in the algorithm. The first proof below shows that there is always a vertex w in G' that is incident to an edge in C. The second proof shows that every vertex in G' has even degree which means that the procedure given above can be used to find a new circuit C' in G'.

Proof 13.5.2: Proof of correctness for the algorithm to find an Euler circuit.

If an undirected graph G is connected and every vertex in G has an even degree, then there is always a vertex w in G' that is also incident to an edge in C.

Proof.

There must be at least one vertex with some incident edges in C and some incident edges not in C. Suppose that no such vertex exists. Let X be the set of vertices whose incident edges are all not in C and let Y be the set of vertices whose incident edges are all in C. Since C is not empty, then Y is not empty. Since C is not yet an Euler circuit, then X is not empty. Suppose there is an edge {x, y} with x ∈ X and y ∈ Y. If {x, y} is an edge in the circuit C, then x does not belong in X because x is incident to an edge in C. If {x, y} is not in C, then y does not belong in Y because y is incident to an edge that is not in C. Therefore, there can be no edge {x, y} connecting a vertex in X to a vertex in Y and the graph G is disconnected, contradicting the assumptions of the theorem. Therefore there must be at least one vertex with some incident edges in C and some incident edges not in C. Vertex w is incident to an edge in C. Furthermore, when the edges are removed, w will not be an isolated vertex, so w is also in G'. ■

Proof 13.5.3: Every vertex in G' has an even degree.

If an undirected graph G is connected and every vertex in G has an even degree, then every vertex in G' is also even.

Proof.

Consider the graph H formed by taking the edges in C and all the vertices that are incident to an edge in C. The circuit C is an Euler circuit for the graph H. Therefore the degree of every vertex in H is even. The graph G' is formed by removing the edges in C from G and throwing out any isolated vertices. Therefore, the degree of a vertex v in G' is equal to the degree of v in G minus the degree of v in H. Since the degree of v in G is even and the degree of v in H is even, then the degree of v in G' is also even. ■

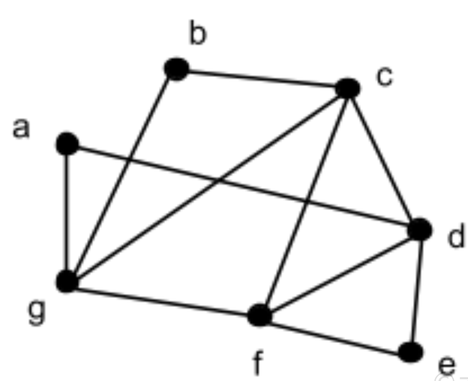
PARTICIPATION ACTIVITY | 13.5.8: The algorithm to find an Euler circuit.

Animation captions:

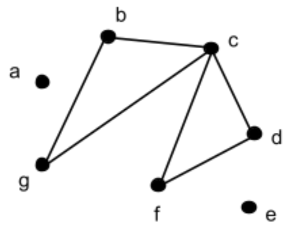
1. Start algorithm with any circuit C in G. C = ⟨ a, b, f, i, c, a ⟩.
2. Start loop because C is not an Euler circuit. Create G' by starting with G, removing edges from C and then removing isolated vertices.
3. Find a vertex w that is in C and G'. Candidates are c, b, f, and i. Select b to assign to w.
4. Find a circuit C' that starts at b. Circuit selected is ⟨ b, d, f, e, b ⟩.
5. Add vertices from C' to C (except the first vertex in C' which is b). The result is a new circuit ⟨a, b, d, f, e, b, f, i, c, a⟩.
6. In the next iteration, C is ⟨a, b, d, f, e, b, f, i, c, a⟩.
7. The new circuit is C in the next iteration. Create a new G' by starting with G, removing the edges from C and removing any isolated vertices.
8. Find a vertex w that is in C and G'. Candidates are c, i. Select c to assign to w.
9. Find a circuit C' in G' that starts at c: ⟨ c, h, i, g, c ⟩.
10. Combine C and C' to get ⟨ a, b, d, f, e, b, f, i, c, h, i, g, c, a ⟩. The new circuit (renamed C) includes all the edges and is therefore an Euler circuit.

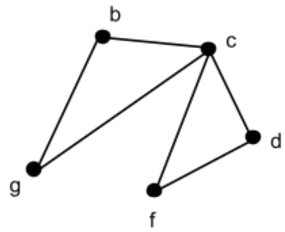
PARTICIPATION ACTIVITY | 13.5.9: The procedure to find an Euler circuit in a graph.

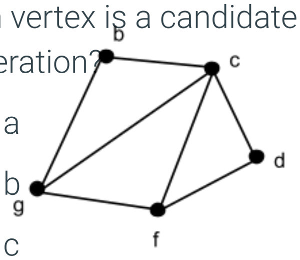
The algorithm to find a circuit in a graph is applied to the graph given below.
At the beginning of an iteration, the circuit C is ⟨ a, d, e, f, g, a ⟩.



1) Which graph represents G'?

- 

- 

- 

2) Which vertex is a candidate to be w in this iteration?

- a
- b
- c
- d

3) Which circuit could possibly be C' in this iteration?

- ⟨ d, e, f, d ⟩
- ⟨ d, f, c, d ⟩
- ⟨ c, d, f, c ⟩

4) Suppose circuit C' is ⟨ d, f, c, d ⟩. Circuit C is ⟨ a, d, e, f, g, a ⟩. Select the sequence that is C in the next iteration.

- ⟨ a, d, d, e, f, g, a, d, f, c, d ⟩
- ⟨ a, d, e, f, g, a ⟩
- ⟨ a, d, f, c, d, e, f, g, a ⟩

Euler trails

An ***Euler trail*** is an open trail that includes each edge. Note that a trail, by definition, has no repeated edges, so an Euler trail contains each edge exactly once. In an open trail, the first and last vertices are not equal. As with Euler circuits, there is a simple set of conditions that characterize when an undirected graph has an Euler trail. The proof of the following theorem is left as an exercise

Theorem 13.5.4: Characterizations of graphs that have an Euler trail.

An undirected graph G has an Euler trail if and only if G is connected and has exactly two vertices with odd degree.

PARTICIPATION ACTIVITY 13.5.10: Euler trails in undirected graphs.

Animation content:

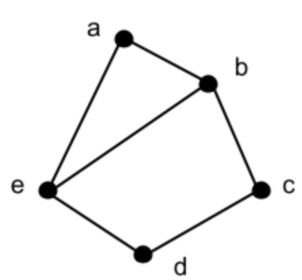
`undefined`

Animation captions:

1. The degrees of vertices a through f in the graph are: 4, 4, 4, 4, 2, 2. Since every vertex has even degree, the graph has no Euler trail.
2. The degrees of vertices a through f of a new graph are: 3, 3, 3, 3, 2, 2. There are four vertices with odd degree, so the graph does not have an Euler trail.
3. The degrees of vertices a through f of a new graph are: 4, 3, 4, 3, 2, 2. There are exactly two vertices with odd degree, so the graph does have an Euler trail.
4. The Euler trail ⟨b, f, c, b, a, c, d, a, e, d⟩ begins and ends with the vertices of odd degree.

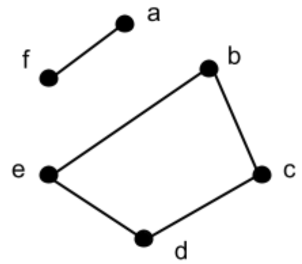
PARTICIPATION ACTIVITY 13.5.11: Euler trails.

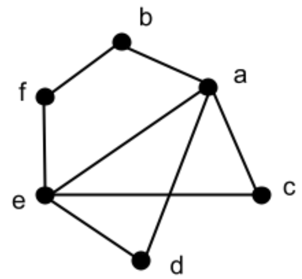
1) Which sequence is an Euler trail in the graph below?

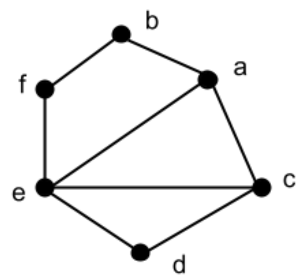


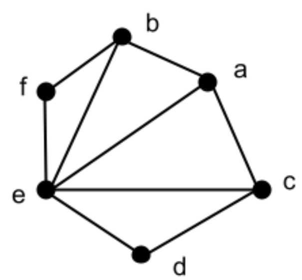
- ⟨ a, b, c, d, e ⟩
- ⟨ a, b, c, d, e, a ⟩
- ⟨ b, c, d, e, b, a, e ⟩
- ⟨ a, b, c, d, e, b, a, e ⟩

2) Which graph has an Euler trail?







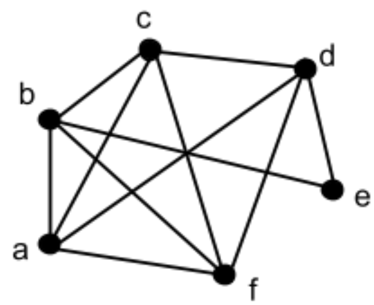


Additional exercises

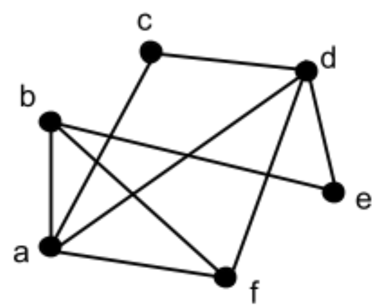
EXERCISE 13.5.1: Finding Euler circuits in graphs.

For each graph, find an Euler circuit in the graph or explain why the graph does not have an Euler circuit.

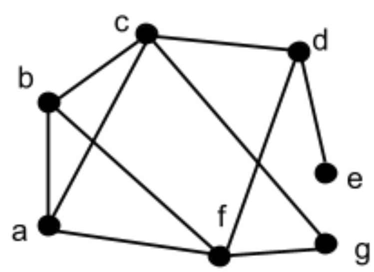
(a)



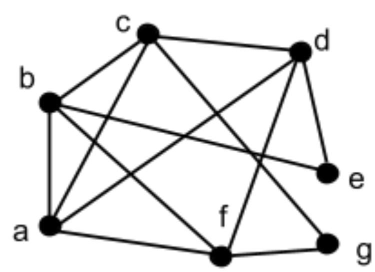
(b)



(c)

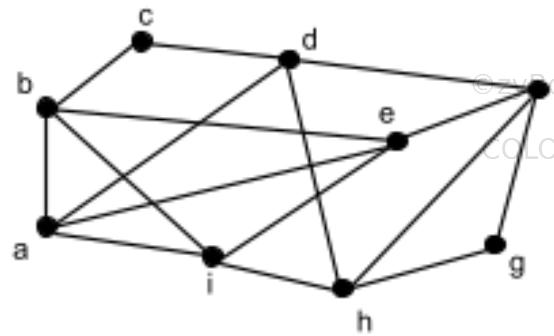


(d)



EXERCISE 13.5.2: Implementing the algorithm to find an Euler circuit.

The algorithm to find an Euler circuit in a graph is applied to the graph below.



(a) At the beginning of an iteration, the cycle C is ⟨ a, b, c, d, h, i, a ⟩. Draw the graph G' for this iteration.

(b) Which vertices are candidates to play the role of w in this iteration?

(c) Select a vertex for w. Then use the algorithm for finding a circuit in a graph to find a circuit C' in G', starting the vertex you selected for w.

(d) Use your choice for C' to find the circuit C for the next iteration.

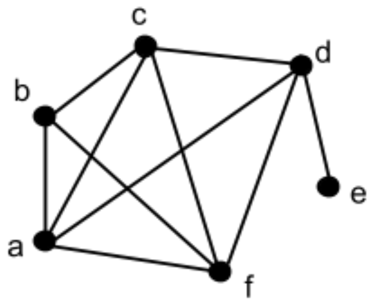
(e) Complete the algorithm to find an Euler circuit. For each iteration, give the graph G', the choice for w, the circuit C' that starts at w, and the circuit C for the next iteration. Give the final Euler circuit produced by the algorithm.

EXERCISE | 13.5.3: Find Euler trails in graphs.

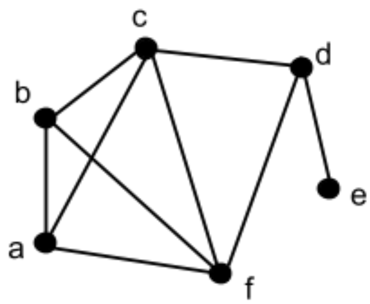
For each graph, find an Euler trail in the graph or explain why the graph does not have an Euler trail .

(Hint: One way to find an Euler trail is to add an edge between two vertices with odd degree, find an Euler circuit in the resulting graph and then delete the added edge from the circuit.)

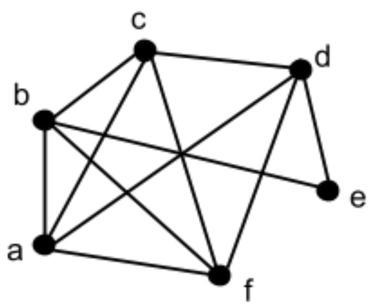
(a)



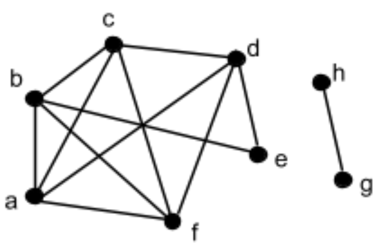
(b)



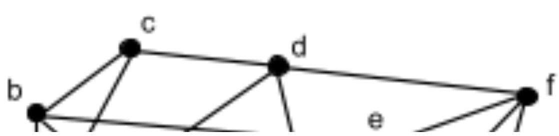
(c)

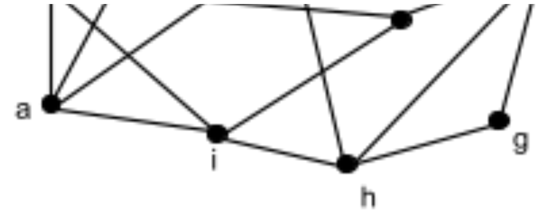


(d)



(e)





EXERCISE 13.5.4: Finding Euler circuits and paths in common graphs.

Answer each question below and justify your answer.

(a) For which values of n does C_n have an Euler circuit?

(b) For which values of n does C_n have an Euler trail?

(c) For which values of n does K_n have an Euler circuit?

(d) For which values of n does K_n have an Euler trail?

(e) For which values of n does Q_n have an Euler circuit?

(f) For which values of n does Q_n have an Euler trail?

(g) For which values of n and m does $K_{n, m}$ have an Euler circuit?

(h) For which values of n and m does $K_{n, m}$ have an Euler trail?

13.6 Hamiltonian cycles and paths

A ***Hamiltonian cycle*** in an undirected graph is a cycle that includes every vertex in the graph. Note that a cycle, by definition, has no repeated vertices or edges, except for the vertex which is at the beginning and end of the cycle. Therefore, every vertex in the graph appears exactly once in a Hamiltonian cycle, except for the vertex which is at the beginning and end of the cycle. A ***Hamiltonian path*** in an undirected graph is a path that includes every vertex in the graph. Note that a path, by definition, has no repeated vertices or edges, so every vertex appears exactly once in a Hamiltonian path.

Note that a Hamiltonian cycle can be transformed into a Hamiltonian path by deleting the last vertex. Therefore if a graph has a Hamiltonian cycle, then the graph also has a Hamiltonian path.

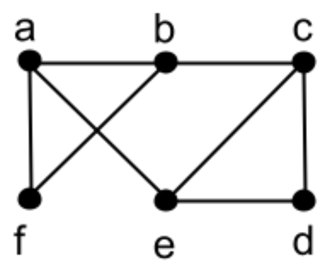
PARTICIPATION ACTIVITY 13.6.1: Hamiltonian cycles and paths.

Animation captions:

1. The closed walk ⟨ x, v, w, y, z, x ⟩ is a cycle. The cycle is a Hamiltonian cycle because every vertex is reached.
2. The open walk ⟨ x, v, w, y, z ⟩ is a path. The path is a Hamiltonian path because every vertex is reached.

PARTICIPATION ACTIVITY 13.6.2: Identifying Hamiltonian cycles and paths.

The questions below refer to the graph G shown below.



1) Select the sequence that corresponds to a Hamiltonian cycle in G.

- ⟨ a, b, c, d, e, a ⟩
- ⟨ a, f, b, c, d, e ⟩
- ⟨ a, f, b, c, d, e, a ⟩
- ⟨ a, b, c, d, e, a, f, a ⟩

2) Select the sequence that corresponds to a Hamiltonian path in G.

- ⟨ a, b, c, d, e⟩
- ⟨ a, f, b, c, d, e ⟩
- ⟨ a, f, b, c, d, e, a ⟩
- ⟨ f, a, b, c, d, e, a ⟩

Unlike Euler circuits and trails, there are no known conditions describing exactly which graphs have a Hamiltonian cycle or path. In fact, there is no efficient algorithm known that can determine whether a graph has a Hamiltonian cycle or path. A brute force algorithm to find a Hamiltonian path would enumerate every possible ordering of the vertices and check whether the sequence is a path.

This algorithm would take a very long time to execute on any computer, even on graphs of modest size.

In some cases, it can be proven that a graph with particular properties does or does not have a Hamiltonian cycle or path. The animation below gives two examples.

PARTICIPATION ACTIVITY 13.6.3: Graphs that do and do not have Hamiltonian cycles.

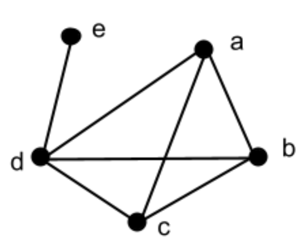
Animation captions:

1. The only neighbor of x is vertex z. Any closed walk that includes x and z must include the edge {x, z} twice.
2. The graph does not have a Hamiltonian cycle. By the same reasoning, any graph with a degree 1 vertex does not have a Hamiltonian cycle.
3. The next graph is K_5, the graph with five vertices and an edge between every pair of vertices.
4. Select any ordering of the vertex and add a copy of the first vertex to the end. The resulting sequence is a Hamiltonian cycle because every pair of vertices has an edge.
5. By the same reasoning K_n has a Hamiltonian cycle for any n $\geq$ 3.

PARTICIPATION ACTIVITY 13.6.4: Graphs with Hamiltonian cycles or paths.

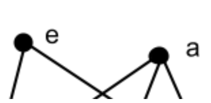
Select the correct description for each graph given below.

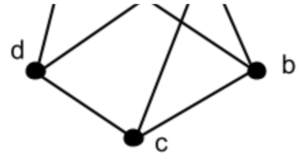
1)



- The graph has a Hamiltonian cycle and a Hamiltonian path.
- The graph has a Hamiltonian path but not a Hamiltonian cycle.
- The graph has neither a Hamiltonian path nor a Hamiltonian cycle.

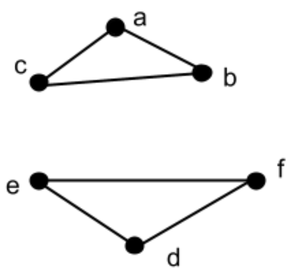
2)





- The graph has a Hamiltonian cycle and a Hamiltonian path.
- The graph has a Hamiltonian path but not a Hamiltonian cycle.
- The graph has neither a Hamiltonian path nor a Hamiltonian cycle.

3)



- The graph has a Hamiltonian cycle and a Hamiltonian path.
- The graph has a Hamiltonian path but not a Hamiltonian cycle.
- The graph has neither a Hamiltonian path nor a Hamiltonian cycle.

Additional exercises

EXERCISE 13.6.1: Hamiltonian paths and cycles in a 2-vertex graph.

The vertex set of graph G is {a, b}. There is one undirected edge {a, b} in G.

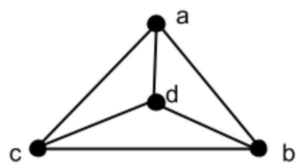
(a) Does G have a Hamiltonian cycle? Justify your answer.

(b) Does G have a Hamiltonian path? Justify your answer.

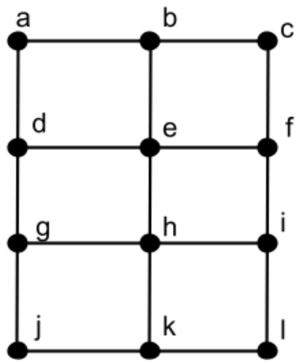
EXERCISE 13.6.2: Finding Hamiltonian cycles in small graphs.

Find a Hamiltonian cycle in each of the graphs.

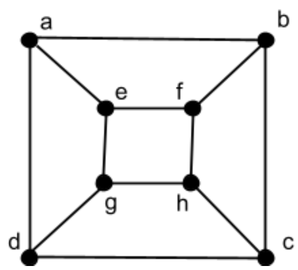
(a)



(b)



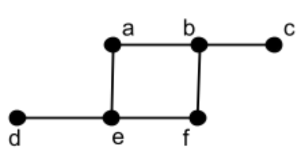
(c)



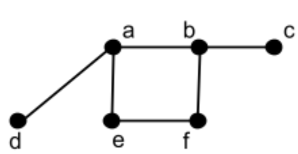
EXERCISE 13.6.3: Hamiltonian paths in small graphs.

Determine whether each graph has a Hamiltonian path. Prove your answer.

(a)



(b)



EXERCISE 13.6.4: Hamiltonian paths and cycles in complete bipartite graphs.

For each class of graphs, determine whether every graph in the class has a Hamiltonian cycle or path. Prove your answer.

(a) $K_{n,m}$, where $n = m$ and $n, m > 1$.

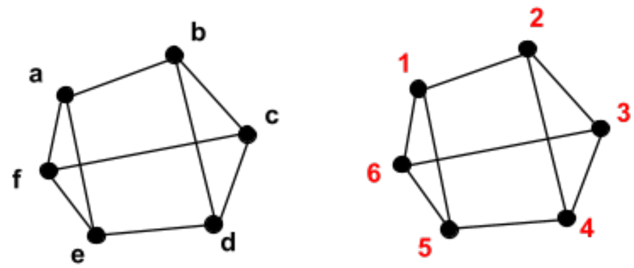
(b) $K_{n,m}$, where $n + 1 = m$.

(c) $K_{n,m}$, where $m \geq n + 2$.

13.7 Graph isomorphism

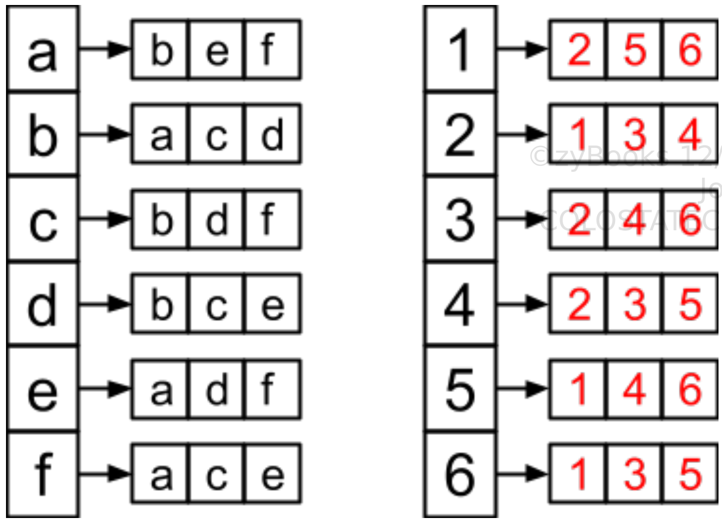
Consider the two graphs pictured below. They look very similar but they are not quite the same. The difference between the two graphs is that the vertices are labeled differently.

Figure 13.7.1: Two similar graphs.



Of course, when the two graphs are drawn as they are, it is easy to verify that the structure of the graph is the same. It is much more difficult to determine that the graphs have the same structure when given the adjacency list representation:

Figure 13.7.2: Adjacency list representation for two similar graphs.



The drawings of the two graphs reveal a correspondence between the two sets of vertices that makes it easy to verify that every edge in the graph on the left corresponds to an edge in the graph on the right, and vice versa. Two graphs are said to be ***isomorphic*** if there is a correspondence between the vertex sets of each graph such that there is an edge between two vertices of one graph if and only if there is an edge between the corresponding vertices of the second graph. The graphs are not identical but the vertices can be relabeled so that they are identical. More formally:

Definition 13.7.1: Isomorphic graphs.

Let G = (V, E) and G'=(V',E'). G and G' are isomorphic if there is a bijection f: V → V' such that for every pair of vertices x, y ∈ V, {x, y} ∈ E if and only if {f(x), f(y)} ∈ E'. The function f is called an ***isomorphism*** from G to G'.

The animation below illustrates:

PARTICIPATION ACTIVITY 13.7.1: Graph isomorphism definition.

Animation content:

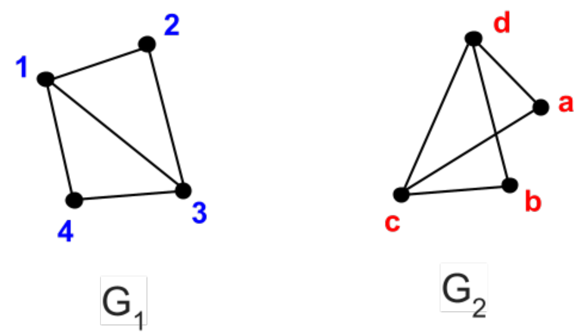
undefined

Animation captions:

1. A function f maps vertices in G_1 to vertices in G_2. {A, E} is not an edge in G_1. f(A) = 3 and f(E) = 4, but {3, 4} is an edge in G_2, so f is not an isomorphism.
2. G_1 can be redrawn so that it looks like G_2, except for the vertex labels. A function g maps the label of each vertex in G_1 to its corresponding label in G_2.
3. {A, B} is an edge in G_1. {f(A), f(B)} = {4, 3} is an edge in G_2. g is an isomorphism because g is a one-to-one correspondence between the edges of G_1 and G_2.

PARTICIPATION ACTIVITY | 13.7.2: Graph isomorphisms.

The diagram below shows two graphs:



1) Is the following function from the vertices of G_1 to the vertices of G_2 an isomorphism?
f(1) = d; f(2) = a; f(3) = b; f(4) = c

- True
- False

2) Is the following function from the vertices of G_1 to the vertices of G_2 an isomorphism?
f(1) = d; f(2) = a; f(3) = c; f(4) = b

- True
- False

In order to verify that graph G is isomorphic to graph G', it is sufficient to find the isomorphism from G to G'. Checking that a function from V to V' is an isomorphism can be done efficiently. However, what if G is not isomorphic to G'? Is there a way to verify that an isomorphism does not exist? In general, there is no efficient algorithm known that can provably determine whether two graphs are isomorphic or not. (An "efficient" algorithm is an algorithm that runs in worst-case polynomial time.) However, in many situations, two non-isomorphic graphs have properties that make it easy to verify that they are not isomorphic.

Consider an isomorphism f from a graph G to another graph G'. The bijection f maps vertex v in G to a vertex f(v) in G'. By the definition of an isomorphism, a vertex w is a neighbor of v in G if and only if f(w) is a neighbor of f(v) in G'. Therefore, the degree of v in G must be the same as the degree of f(v) in G'.

If graph G is isomorphic to graph G', then G has a vertex of degree d if and only if G' has a vertex of degree d. A property is said to be ***preserved under isomorphism*** if whenever two graphs are isomorphic, one graph has the property if and only if the other graph also has the property. The theorem below states that an isomorphism maps each vertex v into a vertex with the same degree. Thus, the property of having a degree d vertex is preserved under isomorphism. If there is an isomorphism from G to G', then G has a degree d vertex if and only if G' has a degree d vertex.

Theorem 13.7.1: Vertex degree preserved under isomorphism.

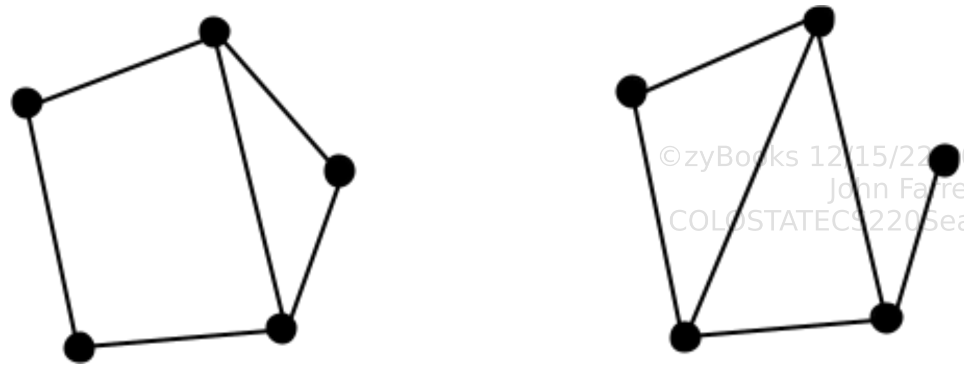
Consider two graphs, G and G'. Let f be an isomorphism from G to G'. For each vertex v in G, the degree of vertex v in G is equal to the degree of vertex f(v) in G'.

Proof 13.7.1: Vertex degree preserved under isomorphism.

Proof.
Let N(v) denote the set of neighbors of vertex v in G. The degree of vertex v is the cardinality of the set N(v). A vertex w ∈ N(v) if and only if {v, w} is an edge in G. By the definition of isomorphism, {v, w} is an edge in G if and only if {f(v), f(w)} is an edge in G'. The neighbors of vertex f(v) in G' is the set {f(w): w ∈ N(v)}. Since f is an isomorphism, f is one-to-one. Therefore, the cardinality of {f(w): w ∈ N(v)} is the same as the cardinality of N(v). Thus, the degree of v in G is the same as the degree of f(v) in G'. ■

Properties preserved under isomorphism can often be used to show that two graphs are not isomorphic. If one graph has the property and another graph does not have the property, then they can not be isomorphic. For example, the two graphs below are not isomorphic because the graph on the right has a vertex of degree 1 and the graph on the left does not.

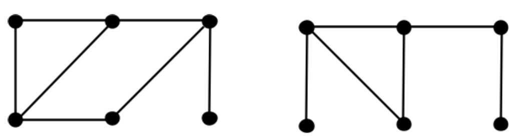
Figure 13.7.3: Two non-isomorphic graphs.



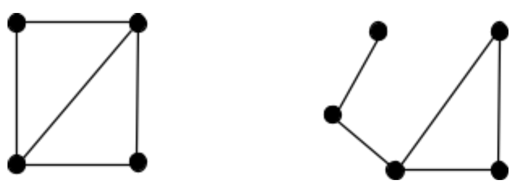
The number of vertices and the number of edges in a graph are also properties that are preserved under isomorphism. If two graphs have different numbers of vertices or edges, then they can not be isomorphic.

PARTICIPATION ACTIVITY 13.7.3: Graph properties preserved under isomorphism.

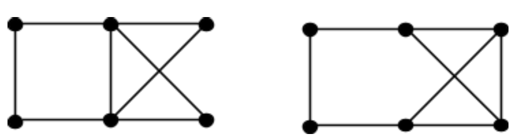
Each question shows a pair of graphs which are not isomorphic. Select the graph property that shows that the pair is not isomorphic.

1) 

- Has a degree 2 vertex.
- Has a degree 4 vertex.
- Number of edges.

2) 

- Number of vertices.
- Number of edges.
- Has a degree 4 vertex.

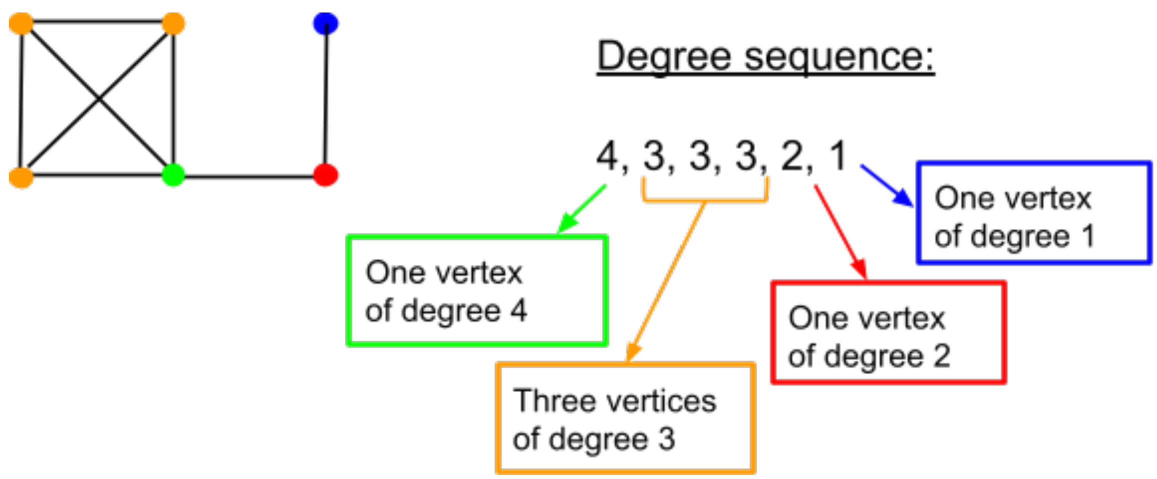
3) 

- Number of edges.

- Number of vertices.
- Has a degree 2 vertex.
- Has a degree 4 vertex.

The fact that having a vertex with a particular degree is preserved under isomorphism can be generalized. If two graphs G and G' are isomorphic and G has r vertices of degree d, then G' also has r vertices of degree d. In fact the entire sequence of degree numbers is preserved under isomorphism. The ***degree sequence*** of a graph is a list of the degrees of all of the vertices in non-increasing order. Non-increasing order means that each number is less than or equal to the preceding number in the sequence. The diagram below shows a graph and its degree sequence.

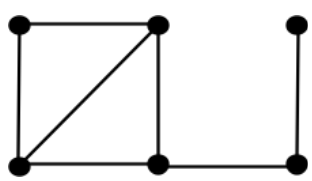
Figure 13.7.4: The degree sequence of a graph.



PARTICIPATION ACTIVITY | 13.7.4: Degree sequence.

What is the degree sequence of the following graph?

1)



- 2, 2, 3, 3, 3, 1
- 3, 3, 3, 2, 2, 1
- 3, 3, 2, 2, 2, 1

The theorem and proof below establish that the degree sequence of a graph is also preserved under isomorphism, meaning that if two graphs are isomorphic they have the same degree sequence. The animation below gives some intuition behind the theorem and shows how degree sequence can be used to established that two graphs are not isomorphic.

PARTICIPATION ACTIVITY 13.7.5: Degree sequence and graph isomorphism.

Animation content:

undefined

Animation captions:

1. G_1 and G_2 are isomorphic. Both graphs have one vertex of degree 4 and two vertices of degree 3, so the degree sequences for both graphs start with 4, 3, 3.
2. The other two vertices in G_1 and G_2 have degree 2, so G_1 and G_2 have the same degree sequence: 4, 3, 3, 2, 2.
3. In another example, G_1 has a degree sequence that starts with 4 because G_1 has one vertex of degree 4.
4. G_1 has three vertices of degree 3, so the degree sequence so far is 4, 3, 3, 3.
5. G_1 has one vertex of degree 2, so the degree sequence so far is 4, 3, 3, 3, 2.
6. Finally, G_1 has one vertex of degree 1, so the degree sequence for G_1 is 4, 3, 3, 3, 2, 1.
7. The degree sequence for G_2 is 4, 4, 3, 2, 2, 1. Since G_1 and G_2 have different degree sequences, G_1 and G_2 are not isomorphic.

Theorem 13.7.2: Degree sequence preserved under isomorphism.

The degree sequence of a graph is preserved under isomorphism.

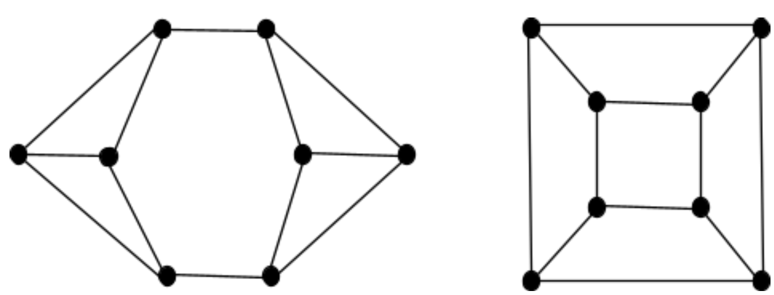
Proof 13.7.2: Degree sequence preserved under isomorphism.

Proof.
Suppose that there is an isomorphism f from graph G to G'. We will show that the degree sequence of G is the same as the degree sequence of G'.

Let the degree sequence of G be $d_1, d_2, \ldots, d_n$. The degree sequence is, by definition, non-increasing, so $d_1 \geq d_2 \geq \ldots \geq d_n$. Order the vertices of G by degree so that vertex v_j has degree d_j. For each vertex v_j in G, the degree of v_j in G is equal to the degree of vertex $f(v_j)$ in G'. Therefore, if the degrees of each of the vertices $f(v_1), \ldots, f(v_n)$ are listed in order, that corresponds to a non-increasing degree sequence which is identical to $d_1, d_2, \ldots, d_n$. ■

The degree sequence is not only useful in determining that two graphs are not isomorphic, it can also help in finding an isomorphism when two graphs are isomorphic. The fact that a vertex in graph G can only be mapped to a vertex of the same degree in G' helps to limit the number of possible mappings from G to G' that need to be checked. For some pairs of graphs, however, the degree sequence turns out not to be helpful in determining if two graphs are isomorphic. The diagram below shows a pair of two non-isomorphic graphs that are both 3-regular. The degree sequence does not help in determining that the two graphs are not isomorphic because the degree sequence for both graphs is just: 3, 3, 3, 3, 3, 3, 3, 3.

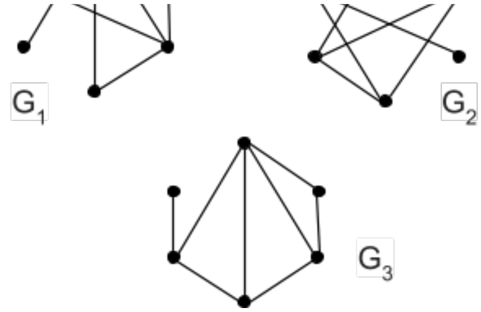
Figure 13.7.5: Two non-isomorphic 3-regular graphs.



PARTICIPATION ACTIVITY 13.7.6: Degree sequence and graph isomorphism.

Two of the graphs pictured below are isomorphic. The third graph is not isomorphic to the other two. Use the degree sequence to determine which of the three graphs is not isomorphic to the other two.

1)  



- G_1
- G_2
- G_3

Graph theory is concerned with properties of graphs that are preserved under isomorphism. Properties preserved under isomorphism relate to the structure of graphs. By contrast, properties that are not preserved under isomorphism depend on the labels of the vertices. We have seen that the number of vertices, the number of edges and the degree sequence are all preserved under isomorphism. Here are some properties that are not preserved under isomorphism:

- The lowest numbered vertex has degree 3
- Every even numbered vertex has odd degree

PARTICIPATION ACTIVITY 13.7.7: Graph properties preserved under isomorphism.

Which graph properties are preserved under isomorphism?

1) Total degree.
   - Preserved
   - Not preserved

2) Number of vertices whose degree is an even number.
   - Preserved
   - Not preserved

3) Sum of the degrees of the even numbered vertices.
   - Preserved
   - Not preserved

4) Number of edges minus the number of vertices.

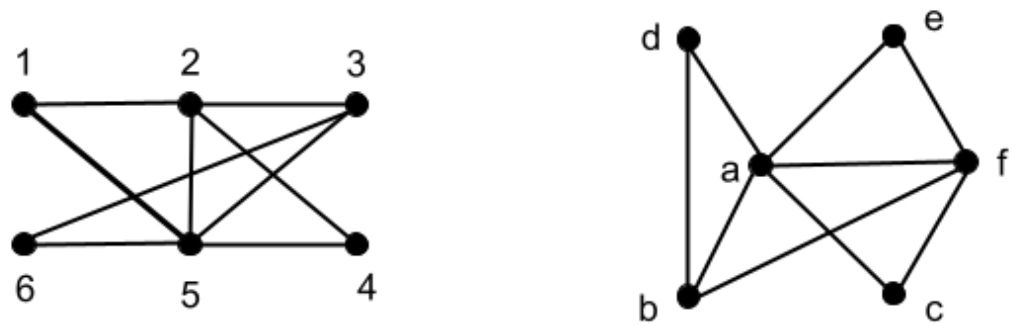
○ Preserved

○ Not preserved

Additional exercises

EXERCISE 13.7.1: Show an isomorphism between two graphs.

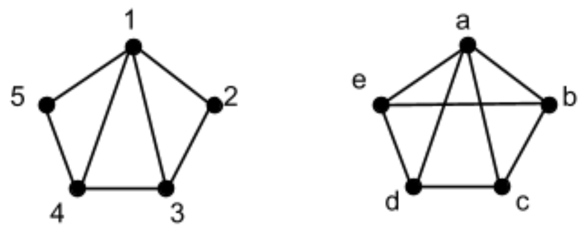
(a) Prove that the two graphs below are isomorphic.



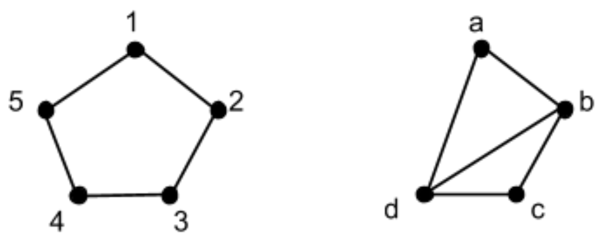
EXERCISE 13.7.2: Showing two graphs are not isomorphic.

For each pair of graphs, show that they are not isomorphic by showing that there is a property that is preserved under isomorphism which one graph has and the other does not.

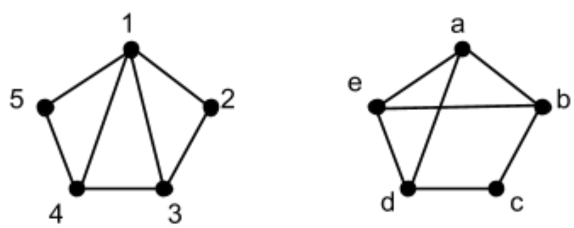
(a)



(b)



(c)



EXERCISE 13.7.3: Draw all non-isomorphic graphs with 3 or 4 vertices.

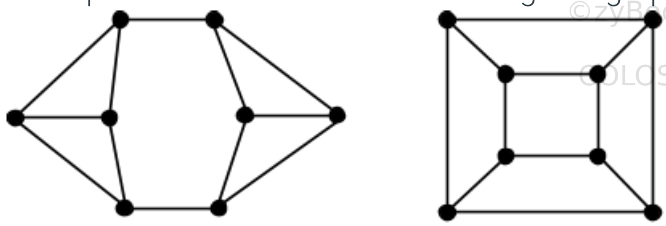
(a) Draw all non-isomorphic simple graphs with three vertices. Do not label the vertices of the graph. You should not include two graphs that are isomorphic. Remember that it is possible for a graph to appear to be disconnected into more than one piece or even have no edges at all.

(b) Draw all non-isomorphic simple graphs with four vertices. Do not label the vertices of the graph. You should not include two graphs that are isomorphic. Remember that it is possible for a graph to appear to be disconnected into more than one piece or even have no edges at all.

EXERCISE 13.7.4: Subgraphs preserved under isomorphism.

(a) Prove that for any k, the property of having C_k as a subgraph is preserved under isomorphism.

(b) Use the fact from part a to show that the following two graphs are not isomorphic:

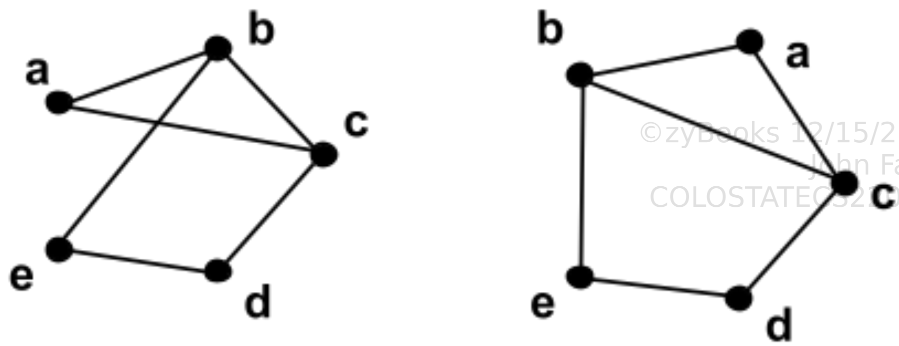


13.8 Planar graphs

A circuit designer has determined the components of a circuit she is designing along with which pairs of components need a direct wire connection. She is now planning the layout of the circuit and would like to place the components on a planar sheet so that the wires connecting different components do not cross. How can she determine if a layout with no crossings is possible? Placing graphs on two-dimensional surfaces to avoid crossings is a classic problem in graph theory. The problem also arises in the field of graph drawing in which the goal is to draw complex graphs in a way that helps people visualize structure and patterns.

An ***embedding*** for G = (V, E) is an assignment of the vertices to points in the plane and an assignment of each edge to a continuous curve. The curve for each edge must start and end at the two points corresponding to the endpoints of the edge. Intuitively, an embedding of a graph corresponds to a drawing of the graph in the plane. Giving a precise definition for "continuous curve" requires ideas from calculus, so a formal definition will not be included here. Instead, we rely on a more intuitive understanding that a continuous curve is one that can be drawn from one end to the other without sharp corners or picking up the stylus. We will also never actually give the specifications for a planar embedding. Instead, we rely on diagrams to represent embeddings of a graph. For example, the diagram below shows two different embeddings of the same graph. Remember that the graph itself is just a set of vertices and a set of edges.

Figure 13.8.1: Two embeddings of the same graph.



An embedding is said to be a ***planar embedding*** if none of the edges cross. There is a ***crossing*** between two edges in an embedding if their curves intersect at a point that is not a common endpoint. For example, in both embeddings shown above, the lines corresponding to edges {a, b} and {b, c} intersect at the point corresponding to vertex b. An intersection at a common vertex is not considered a crossing. However, in the graph on the left, the lines corresponding to edges {b, e} and {a, c} intersect at a point that is not a vertex. The intersection of the two lines corresponding to {b, e} and {a, c} is considered a crossing.

An embedding of a graph can be planar or not planar, depending on whether it has a crossing. Planarity can also be defined as a property of the graph itself:

Definition 13.8.1: Planar graphs.

A graph G is a ***planar graph*** if the graph has a planar embedding.

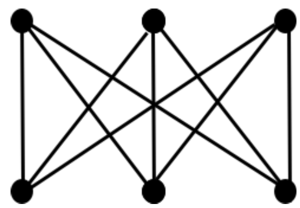
PARTICIPATION ACTIVITY 13.8.1: Definition of a planar embedding and graph.

Animation captions:

1. In an embedding of graph G, the vertices are placed on a 2-dimensional surface. Every pair of vertices that have an edge are connected with a line.
2. An embedding is not planar if there are two edges that cross each other.
3. A different embedding of a graph may have no crossing edges, in which case the embedding is planar. G is a planar graph if G has a planar embedding.

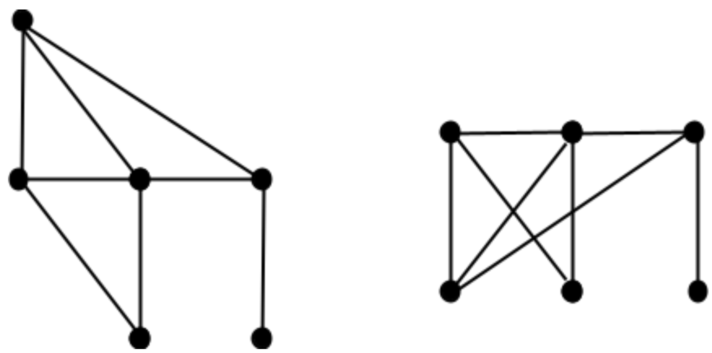
The graph $K_{3,3}$ pictured below is not planar because there is no way to draw the graph on a plane without at least one edge crossing. Try drawing $K_{3,3}$ on paper without crossings. Can you prove that there is no planar embedding of $K_{3,3}$?

Figure 13.8.2: A non-planar graph.



PARTICIPATION ACTIVITY | 13.8.2: Planar graphs.

Two embeddings of the same graph G are depicted below:



1) Is the embedding on the left planar?

Check **Show answer**

2) Is the embedding on the right planar?

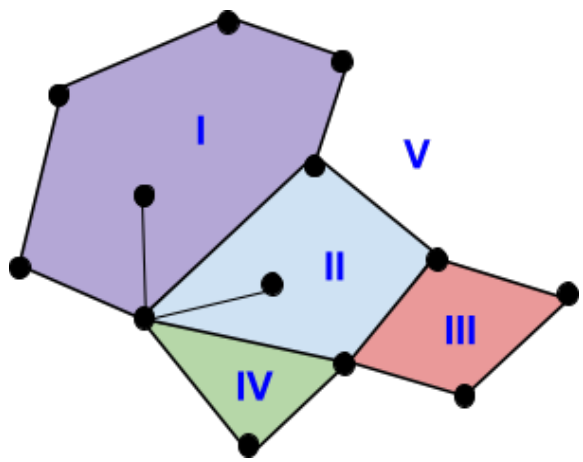
Check **Show answer**

3) Is K_4 planar?

Check Show answer

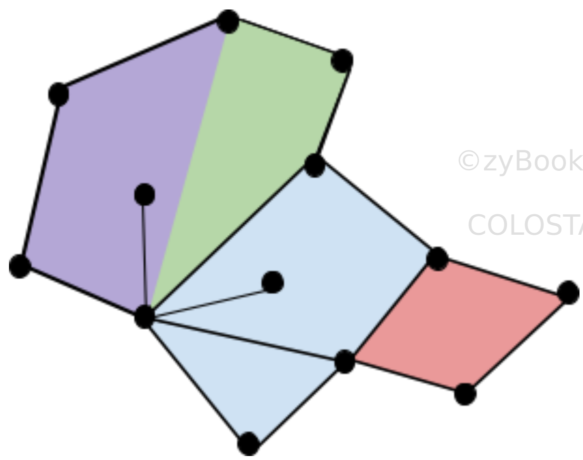
A planar embedding carves up the plane into continuous regions. The diagram below shows an embedding of a graph in the plane. Each region is colored a different color and assigned a number. In a planar embedding, there is always an infinite region called the ***exterior region***. In the diagram below, the exterior region is white.

Figure 13.8.3: The regions of a planar embedding of a graph.



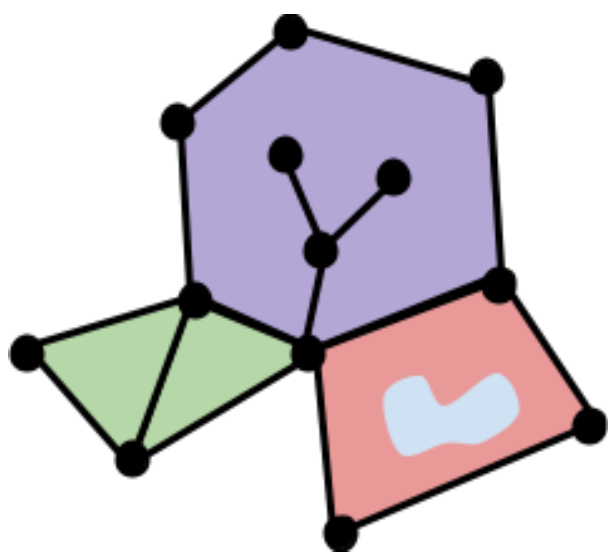
The ***complement of an embedding*** is the set of all points in the plane that are not a vertex or part of a curve corresponding to an edge. In picture above, the complement is any part of the diagram that is not black, since the vertices and edges are black.A ***region*** is a set of points in the complement of an embedding that forms a maximal continuous set. "Continuous" means that it is possible to travel from any point in the region to any other point in the region without leaving the region. "Maximal" means that if any point were added to the region, it would no longer be continuous. In the diagram below, the blue area is not continuous because it is impossible to travel from one part to another without crossing an edge (which is not part of the region). The purple area is not maximal because it can be expanded to include the green area and still remain continuous.

Figure 13.8.4: Areas that are not regions.



PARTICIPATION ACTIVITY | 13.8.3: Regions in a planar embedding.

The diagram below shows an embedding of a planar graph.



1) Is the green area a region of the embedding?

- Yes
- No, because it is not continuous.
- No. The green area is continuous but it is not a maximal continuous area.

2) Is the purple area a region of the embedding?

- Yes
- No, because it is not continuous.
- The purple area is continuous but it is not a maximal continuous area.

3) Is the pink area a region of the embedding?

- Yes
- No, because it is not continuous.
- The pink area is continuous but it is not a maximal continuous area.

4) How many regions are there in the embedding?

- 3
- 4
- 5

Characterizing planar graphs

Proving constraints on planar graphs helps to determine whether a given graph is planar. The following theorem is due to 18th century Swiss mathematician, Leonhard Euler.

Theorem 13.8.1: Euler's Identity.

Consider a planar embedding of a connected graph G. Let n be the number of vertices in G, m the number of edges, and r the number of regions in the embedding. Then,

$$n - m + r = 2$$

The proof of Euler's Identity given below relies on a way to build a graph G by adding edges one at a time. Constructing G incrementally will result in series of graphs $G_0,\ldots,G_m$, where G_0 is a single vertex with no edges, G_m is the final graph G, and G_j has j edges. To get from G_j to G_{j+1} we either add a single edge, or we add an edge and a vertex. The animation below illustrates with an example:

PARTICIPATION ACTIVITY | 13.8.4: Incremental construction of a graph.

Animation content:

undefined

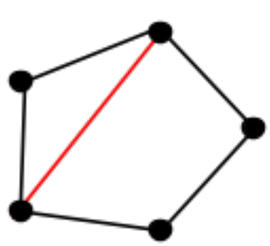
Animation captions:

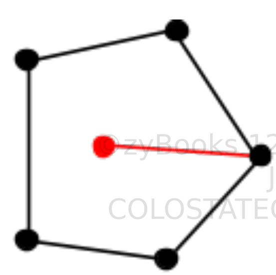
1. Incremental construction of a graph G. G_0 is a graph consisting of a single vertex from G. Add a vertex and an edge connecting the two vertices to get G_1.
2. Add a new vertex and edge to get G_2. Add an edge between two vertices already in G_2 to get G_3.
3. Add a new vertex and edge to get G_4. Add an edge between two vertices already in G_4 to get G_5. G_5 is the final graph.

There are two types of steps in the incremental construction of G.

1. Type 1: only an edge is added. It must be the case that both endpoints of the edge have already been added to the graph.
2. Type 2: an edge and a vertex are added. The new edge must connect a vertex that has already been added to one that has not yet been added.

Type 1 additions do not change the number of vertices and Type 2 additions increase the number of vertices by 1. The incremental construction starts with G_0 which has 1 vertex and ends with G_m which has n vertices. Therefore, if there are n vertices in G, then there are exactly n - 1 type 2 steps in an incremental construction of G. Here is the proof of Euler's Identity:





Type 1:
r: (number of regions) increases
n: (number of vertices) stays the same

Type 2:
r: (number of regions) stays the same
n: (number of vertices) increases

Proof 13.8.1: Euler's Identity.

Theorem: Consider a planar embedding of a connected graph G. Let n be the number of vertices in G, m the number of edges, and r the number of regions in the embedding. Then,

$$n - m + r = 2$$

Proof.

Pick any embedding and any incremental construction of G. As edges and vertices are added to G, they are included in the embedding. After each addition, the partial graph is embedded in the plane.

A type 2 addition does not change the number of regions because it connects the new edge to a new vertex. A type 1 addition splits a region in half and therefore increases the number of regions by 1.

There are m steps in the construction, exactly n - 1 of which are type 2 additions. Therefore, there are m - (n - 1) type 1 additions. The number of regions for G_0 is one since the whole plane is a single region. After the m - (n - 1) type 1 additions, the number of regions r is:

$$r = 1 + (m - (n - 1)) = 2 + m - n$$

It follows that r - m + n = 2. ■

It follows immediately from Euler's identity that any planar embedding of a graph will have the same number of regions. The fact that the number of regions does not depend on the embedding follows from the fact that the number of regions r is determined by r = 2 -n + m. The number of vertices and edges in a graph are fixed properties of the graph, not the embedding.

PARTICIPATION ACTIVITY | 13.8.5: Number of regions in a planar graph.

1) How many regions does the following graph G=(V,E) have?

- V = {a, b, c, d}.
- E = {{a,b},{b,c},{c,d},{d,a},{a,c}}

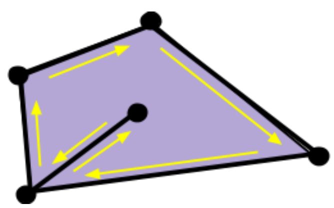
Check Show answer

2) How many regions are in K_4?

Check Show answer

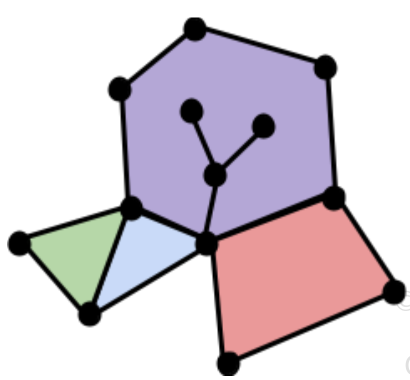
We can use Euler's Identity to give an upper bound on the number of edges in any planar graph as a function of the number of vertices. The proof requires a new definition: the degree of a region. Consider a planar embedding of a graph G. Think of a tiny bug walking all the way around the region along the edges of the graph. The degree of a region is the number of times the bug traverses an edge until it gets back to its starting location. Note that if an edge sticks out into a region, the edge can be traversed twice by the bug and therefore contributes 2 towards the degree of the region as illustrated in the diagram below:

Figure 13.8.5: Degree of a region: the path of a bug around a region.



Degree of the purple region is 6:
6 edges traversed around the region

PARTICIPATION ACTIVITY 13.8.6: Degree of a region.



1) What is the degree of the purple region in the graph embedding pictured above?

Check Show answer

2) What is the degree of the external region?

Check Show answer

Theorem 13.8.2: Number of edges in a planar graph.

Let G be a connected planar graph. Let n be the number of vertices in G and m the number of edges. If $n \geq 3$, then

$$m \leq 3n - 6$$

Proof 13.8.2: Number of edges in a planar graph.

Proof.

Consider a planar embedding of G. For each region R, let deg(R) be the degree of region R. The sum of deg(R) over all the regions in G is equal to 2m because each edge contributes exactly 2 to the total degree. (If the edge is on the boundary of two regions, it contributes 1 to the degree of each region. If the edge is contained within a region, it is counted twice for that region.) Therefore,

$$\sum_R deg(R) = 2m$$

Every interior region has degree at least 3, because there are no multiple edges between the same pair of vertices. The only way for the exterior region to have degree less than 3 is if the graph just consists of two vertices connected by a single edge. Since the number of vertices is at least 3, the exterior region has degree at least 3 as well. Therefore:

$$3r \leq \sum_R deg(R) = 2m.$$

Now substitute the expression r = 2 + m - n (Euler's identity) into the inequality above:

$$3(2 + m - n) \leq 2m.$$

The inequality can be rearranged to give the final result $m \leq 3n - 6$. ■

PARTICIPATION ACTIVITY | 13.8.7: Applying restrictions on planar graphs.

1) What is 3n - 6 for K_5?

Check Show answer

2) How many edges are in K_5?

Check Show answer

3) Is it possible that K_5 is planar?

Check Show answer

Additional exercises

EXERCISE 13.8.1: Draw a planar embedding of a graph.

(a) Give a planar embedding of the following graph:
V = {a, b, c, d, e, f}.
E = { {f, b}, {f, e}, {d, e}, {b, c}, {b, d}, {a, d}, {f, d}, {b, e}, {a, b} }.

(b) How many regions does the graph have?

EXERCISE 13.8.2: Incremental constructions.

(a) Consider a graph G with vertex set V = {a, b, c, d, e, f}. An incremental construction of G starts with vertex e. Then the edges are added in the following order:
{e, f}, {e, c}, {c, f}, {b, c}, {a, c}, {b, a}, {d, b}, {e, b}.
Which edge additions require the addition of a vertex and which additions increase the number of regions?

EXERCISE 13.8.3: A k-regular planar graph.

(a) If a planar graph is k-regular and has n vertices, how many regions does it have? Give your answer as an expression in terms of k and n.

EXERCISE 13.8.4: Is a 6-clique planar?

(a) Is K_6 planar? Justify your answer.

EXERCISE 13.8.5: Subgraphs and planarity.

(a) Prove the following statement: If H is a subgraph of G and G is planar, then H is also planar.

(b) Use the fact proven in the previous question to show that if K_5 is a subgraph of G, then G is not planar.

(c) If a graph G with n vertices and m edges is planar, then m ≤ 3n - 6. Is the converse true? If a graph has m edges and n vertices such that m ≤ 3n - 6, then is it always true that G is planar?

13.9 Graph coloring

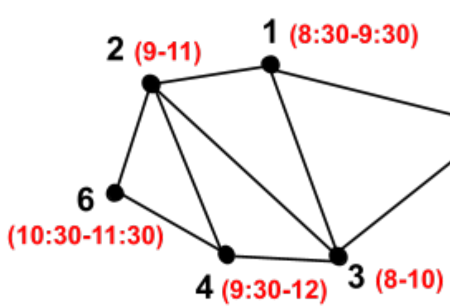
A graph can be used to represent a set of scheduling constraints. For example, suppose that a tennis club is scheduling matches. Groups of players request a court for a particular interval of time but they are not able to specify which court they are assigned. The scheduler for the club would like to make sure that there are enough courts to accommodate all the reservations. Suppose that the list below is the set of reservations for a given day:

Table 13.9.1: Tennis court reservations.

Group	Time
1	8:30AM-9:30AM
2	9:00AM-11:00AM
3	8:00AM-10:00AM
4	9:30AM-11:30AM
5	7:00AM-9:00AM
6	10:30AM-11:30AM

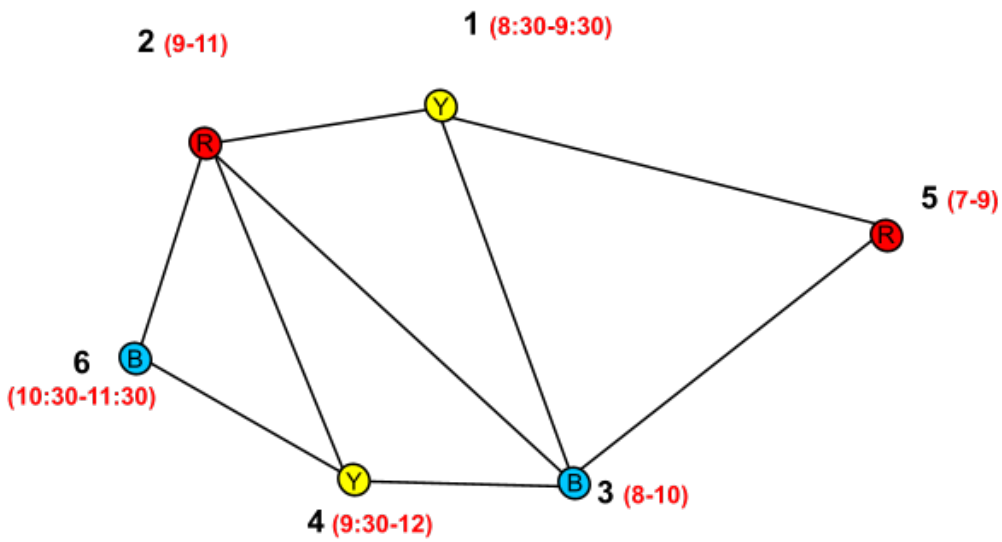
The scheduling problem can be represented by a graph. There is a vertex for each reservation. There is an edge between two reservations if their times overlap, meaning that the two reservations can not be assigned to the same court. The set of scheduling constraints listed in the table give rise to the graph below:

Figure 13.9.1: Graph representing constraints for tennis reservations.



Suppose that the tennis club has three courts. Is there a way to accommodate all the reservations with the three courts? To help her visualize the problem, the club scheduler assigns a color to each court. Court 1 is red, court 2 is blue, and court 3 is yellow. If a group is scheduled in a particular court, that vertex is scheduled the color associated with the court. For example, she decides to schedule group 2 in court 1, so vertex 2 is colored red. The question is, can she color all the vertices so that no two vertices connected by an edge are assigned the same color? The problem is equivalent to assigning each group to a court so that no two groups playing at the same time are assigned to the same court. The coloring below represents a valid assignment:

Figure 13.9.2: A coloring representing a valid assignment.



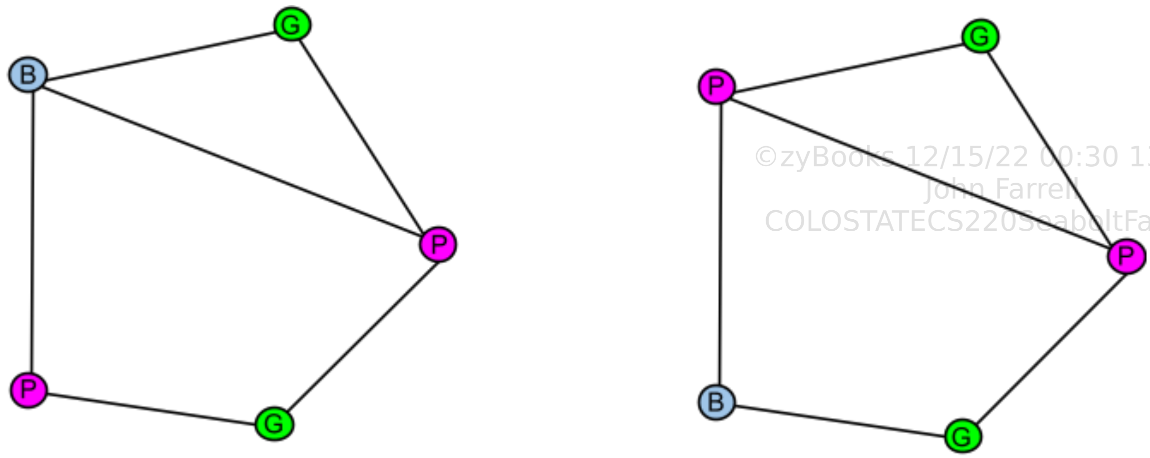
Graph coloring is a classic problem in graph theory because it is useful in modeling resource constraints like tennis court scheduling. Here is the formal definition:

Definition 13.9.1: Graph coloring.

Let G=(V, E) be an undirected graph and C a finite set of colors. A ***valid coloring*** of G is a function f: V → C such that for every edge {x, y} ∈ E, f(x) ≠ f(y). If the size of the range of function f is k, then f is called a ***k-coloring*** of G.

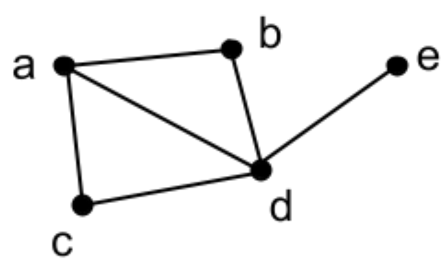
The coloring on the left is a valid coloring because no two adjacent vertices are assigned the same color. The coloring on the left is a 3-coloring because 3 colors are used. The coloring on the right is not a valid coloring because there is an edge whose endpoints are both colored pink.

Figure 13.9.3: Two colorings of a graph: One valid and one invalid.



PARTICIPATION ACTIVITY 13.9.1: Graph colorings.

A graph G is pictured below:



1) Is the following assignment of colors to vertices in G a valid coloring? f(a) = f(d) = RED. f(b) = f(c) = f(e) = BLUE.

- Yes
- No

2) Is the following assignment of colors to vertices in G a valid coloring? g(a) = RED. g(b) = g(c) = g(e) = BLUE. g(d) = GREEN.

- Yes
- No

3) Is the coloring g from the previous

question a 2-coloring?

- Yes
- No

The chromatic number of a graph

Suppose that the organizers of a local tennis tournament have fixed the set of games and game times for a tournament. They would like to find a club that has enough courts to host the tournament. How do they know how many courts they need? If the jobs and constraints are already fixed, the constraint graph is fixed. The question is how many colors are required for a valid coloring of the graph?

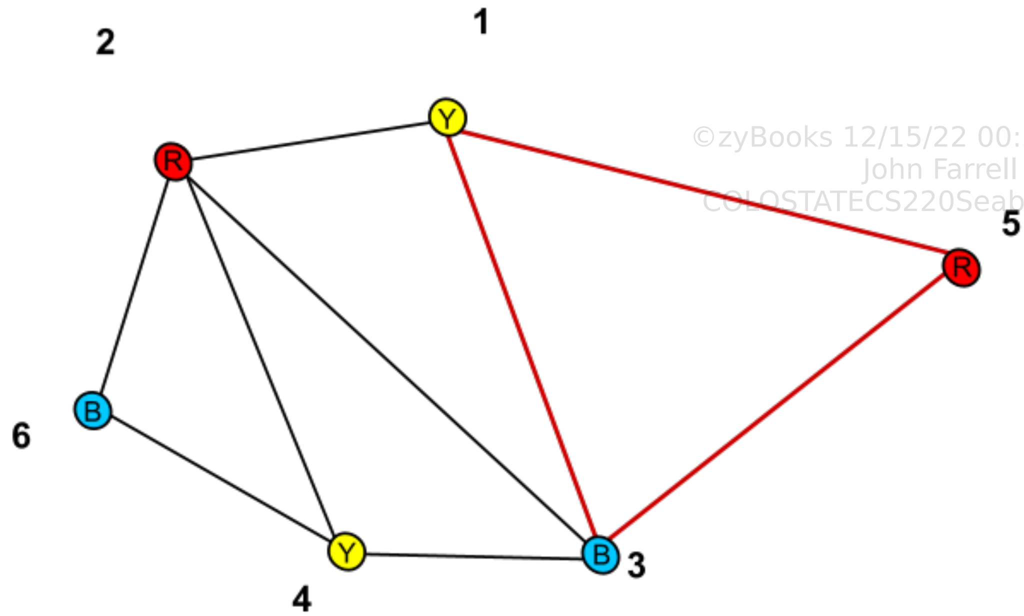
Definition 13.9.2: Chromatic number of a graph.

The ***chromatic number*** of a graph G (denoted X(G)) is the smallest k such that there is a valid k-coloring of G.

For example, the graph G corresponding to the tennis reservations has X(G) ≤ 3 because there is a valid 3-coloring of the graph. Is it possible that X(G) = 2? In other words, is there a valid 2-coloring of the tennis reservation graph? The answer is no because there is no valid coloring of the tennis reservation graph that uses only two colors. Therefore, at least three colors are required in a valid coloring of G which means that X(G) ≥ 3. Since X(G) ≥ 3 and X(G) ≤ 3, then X(G) = 3.

It is easy to verify that the tennis reservation graph requires at least three colors for a valid coloring because there are three group reservations (1, 3, and 5) that have mutually conflicting times. Each of the three groups requires a different court. In the language of graph theory, there are three vertices in the tennis reservation graph such that every pair of vertices in the set of three is connected by an edge. In other words, the tennis reservation graph has K_3 as a subgraph. The diagram below shows the 3-coloring for the tennis reservation graph and highlights one of the subgraphs of K_3 (also called a 3-clique) which establishes that three colors are required for a valid coloring.

Figure 13.9.4: A 3-clique in the tennis reservation graph.



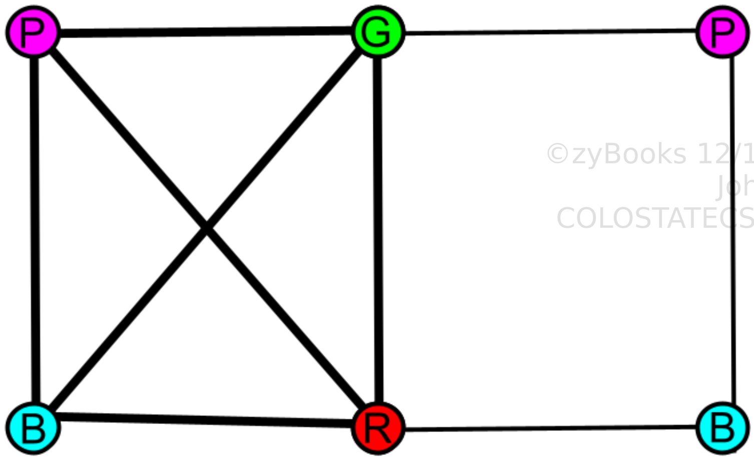
In general, if K_r is a subgraph of G, then there is a subset of r vertices in G such that every pair of vertices in the subset is connected by an edge. In any valid coloring of G, each of the r vertices in the subset must be assigned a distinct color which implies that $X(G) \geq r$. The ***clique number*** of a graph G (denoted $\omega(G)$) is the largest r such that K_r is a subgraph of G. We have just proven the following theorem:

Theorem 13.9.1: Relationship between the clique number and chromatic number.

If G is an undirected graph, then $\omega(G) \leq X(G)$.

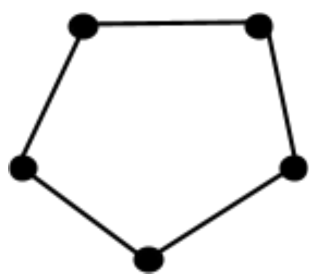
The graph below has K_4 as a subgraph, but the graph does not have K_5 as a subgraph. Therefore, $\omega(G) = 4$. At least four colors are required in a valid coloring of G which means that $X(G) \geq 4$. The graph also has a 4-coloring, so $X(G) = 4$.

Figure 13.9.5: Clique and chromatic number of a graph.



PARTICIPATION ACTIVITY | 13.9.2: Chromatic number and clique number of a graph.

A picture of C_5 is given below:



1) What is $\omega(C_5)$?

Check Show answer

2) Is $X(C_5) \leq 3$?

Check Show answer

3) Is there a 2-coloring for C_5?

Check Show answer

4) What is $X(C_5)$?

Check Show answer

5) The theorem above says that for any graph G, $X(G) \geq \omega(G)$.
Is there a graph G for which $X(G) > \omega(G)$? Type: Yes or No.

Check Show answer

Greedy coloring

It is, in general, difficult to determine the chromatic number of a graph G. However, there is an easy and natural method to color the vertices of a graph called the greedy coloring algorithm. The ***greedy coloring algorithm*** often leads to a coloring that uses a small number of colors, although there is no guarantee that the greedy algorithm uses the smallest number of colors possible for a given graph. Here is a description of the algorithm:

Figure 13.9.6: The greedy coloring algorithm.

- Number the set of possible colors. Assume that there is a very large supply of different colors, even though they might not all be used.
- Order the vertices in any arbitrary order.
- Consider each vertex v in order:
  - Assign v a color that is different from the color of v's neighbors that have already been assigned a color. When selecting a color for v, use the lowest numbered color possible.

The animation below illustrates.

PARTICIPATION ACTIVITY 13.9.3: Greedy coloring of a graph.

Animation content:

`undefined`

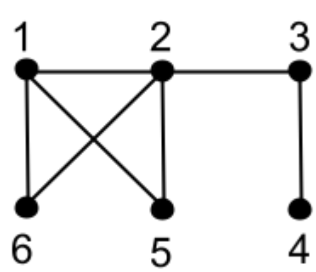
Animation captions:

1. Greedy coloring. Colors are ordered: red, blue, green, pink, ... Pick an ordering of the vertices: 2, 6, 5, 4, 1, 3.
2. The first vertex 2 is colored with the first color, red.
3. The next vertex 6 cannot be colored red because 6 is adjacent to 2 which is already colored red, so 6 is colored with the second color blue.
4. The next vertex 5 cannot be colored red or blue because 5 is adjacent to 2 and 6, so 5 is colored with the third color green.
5. The next vertex 4 can be colored with the first color, red, because 4 is not connected to any red vertices.
6. 1 cannot be colored red because 1 is adjacent to 4 which is already red, but 1 can be colored blue because 1 is not adjacent to any blue vertices.
7. 3 is adjacent to 2 and 4 which are already red, but 3 can be colored blue because 3 is not adjacent to any blue vertices.

The term "greedy" is used to describe a general paradigm for solving many different kinds of problems. Greedy algorithms typically solve a problem one piece at a time. For the coloring problem, the greedy algorithm colors the vertices one at a time and never reconsiders a coloring assignment for a vertex once the vertex has been assigned a color. A greedy algorithm makes the best choice for each piece of the problem based on the partial solution determined so far. For coloring, the goal is to use as few colors as possible, so the greedy algorithm colors each vertex using the smallest numbered color possible. The number of colors used in a greedy coloring can depend on the order that the vertices are colored.

PARTICIPATION ACTIVITY 13.9.4: The greedy coloring algorithm.

Suppose that the greedy algorithm is used to color the vertices of the graph below:

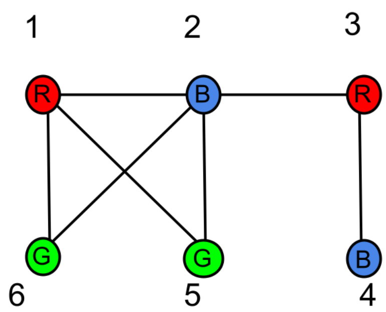


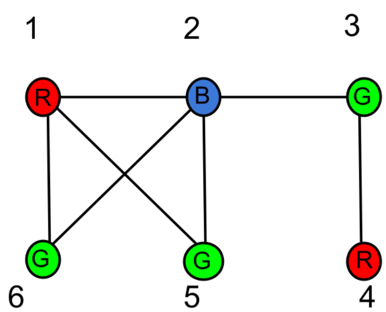
1) Suppose that the colors are:

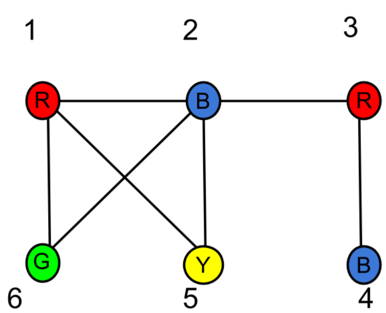
1-Red
2-Blue
3-Green
4-Yellow

The vertices are ordered and colored in the following order: 1, 2, 3, 4, 5, 6.

Which diagram shows the coloring of the graph resulting from the greedy algorithm?







©zyBooks 12/15/22 00:30 1361995
John Farrell
COLOSTATECS220SeaboltFall2022

The greedy coloring algorithm provides a way to upper bound the chromatic number of a graph:

Theorem 13.9.2: Upper bound for X(G).

Let G be an undirected graph. Let Δ(G) be the maximum degree of any vertex in G. Then, X(G) ≤ Δ(G) + 1.

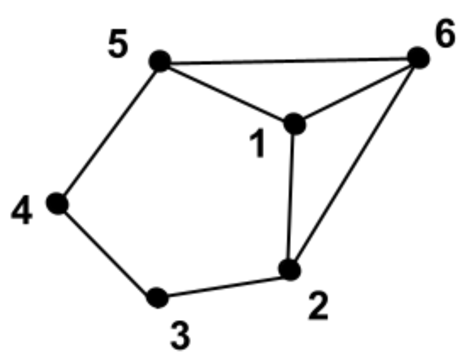
Proof 13.9.1: Upper bound for X(G).

Proof.
We will prove that the greedy algorithm never uses a color numbered more than Δ(G) + 1. Since the greedy algorithm always produces a valid coloring, X(G) is less than or equal to the number of colors used by the greedy algorithm.

Since each vertex has degree at most Δ(G), when a vertex v is assigned a color according to the greedy algorithm, there are at most Δ(G) colors that are ruled out because v is adjacent to a vertex that has already been assigned that color. Therefore, there must be at least one color among the first Δ(G) + 1 colors that can be assigned to vertex v. ■

PARTICIPATION ACTIVITY | 13.9.5: Maximum degree and the greedy coloring algorithm.

Consider the graph G given below.



1) What is Δ(G) + 1?

Check Show answer

2) Suppose that the vertices of G are colored by the greedy

algorithm in order 1, 2, 3, 4, 5, 6.
How many colors are used?

Check Show answer

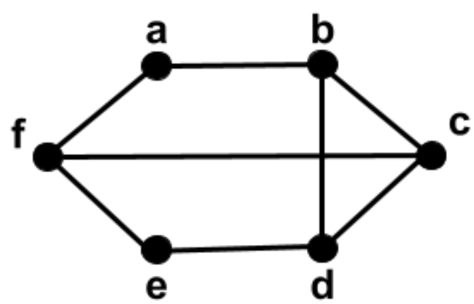
3) Is there a coloring for the graph that uses fewer colors?

Check Show answer

Additional exercises

EXERCISE 13.9.1: Coloring a small graph.

The diagram below shows an undirected graph G:



(a) Show the coloring produced by the greedy algorithm when the vertices are colored in the following order:

b, d, e, a, c, f.

(b) Give a coloring of the graph that uses the fewest number of colors possible.

EXERCISE 13.9.2: Chromatic and clique numbers for common graphs.

What is the chromatic number and the clique number of the following graphs? Your answer may depend on the parameters of the graph (e.g., n or m).

(a) K_n

(b) $K_{n, m}$

(c) C_n

EXERCISE 13.9.3: Clique number and chromatic number.

(a) Give an example of a graph whose clique number is 3 and whose chromatic number is 4.

EXERCISE 13.9.4: Ordering for the greedy algorithm to produce the optimal coloring.

(a) Suppose that the chromatic number of a graph G is k. Is there an ordering of the vertices in G such that the greedy Algorithm will color the vertices in G with k colors if the vertices are colored in that particular order? Justify your answer.