Graphs

R Pruim

2019-02-21

Contents

Preface

These exercises were assembled to accompany a Discrete Mathematics course at Calvin College.

1 Introduction to Graphs

1.1 Graphs

A Graph G consists of two sets

- V, a set of vertices (also called nodes)
- E, a set of edges (each edge is associated with two[1] vertices)

In modeling situations, the edges are used to express some sort of relationship between vertices.

1.1.1 Graph flavors

There are several flavors of graphs depending on how edges are associated with pairs of vertices and what constraints are placed on the edge set.

graph type	edges	multi-edges allowed?	self-loops allowed?	cycles allowed?
simple graph	set	no	no	yes
directed graph	tuple	no	yes	yes
multigraph	set	yes	no	yes
directed multigraph	tuple	yes	yes	yes
pseudo-graph	set	yes	yes	yes
tree	set	no	no	no
directed acyclic graph (DAG)	tuple	no	no	no

Directed graphs are sometimes called **digraphs**.

(Small) graphs are often depicted using dots for vertices and line segments or arcs for edges. For directed graphs, an arrow is used to indicate the direction.

1. For each type of graph, draw a picture of an example graph with 5 vertices.

1.1.2 Labeled Graphs

In many situations it is useful to add additional information to either the vertices or the edges. These are sometimes called "labeled" graphs and the extra information referred to as labels.

1.1.3 Graph Models

Many things can be models with graphs of various types. In each situation below,

- identify the vertices
- identify what edges represent
- determine which flavor of graph would be used
- determine what, if any, information might be useful as labels (on vertices or on edges or on both)
- if the graph has any special properties, note them
- if your group disagrees about some of the answers, identify whether it is because you are making different assumptions about the situation being modeled

[1] In some types of graphs, the two vertices may be the same, so there is really only one vertex. This is called a self-loop.

Model	Vertices	Edges	Type of Graph	Labels/Properties?
2				
3				
4				
5				
6				
7				
8				
9				
10				
11				

2. **Niche Overlap.** Some species of animals compete with other species in an ecosystem.
3. **Acquaintanceship.** Some pairs of people are acquainted with each other, some are not.
4. **Precedence and Concurrent Processing.** Computer programs can be executed more rapidly by executing some statements concurrently But it is important not to execute a statement that depends on the results of statements that have not yet been executed. What is the precedence graph for this simple program?

```
a = 0
b = 1
c = a + 1
d = b + a
e = d + 1
e = c + d
```

5. **Influence.** In studies of group behavior it is observed that certain people can influence the thinking of other people.
6. **Hollywood.** Some actors have been in movies together, others have not.
7. **Round-Robin Tournament.** In a round robin tournament, each team plays each other team once. (The graph should keep track of who wins and loses each game.)
8. **General Sports Schedule.** The schedule for most leagues is not a round-robin tournament. How does that change the graph? If the games haven't been played yet, what information might the graph keep track of? How?
9. **Telephone calls.**
10. **World Wide Web.**
11. **Road maps.**

1.2 Bonus Problem

From 1961 through 1977 the NFL (National Football League) had a 14-game season. Until 1976, when two new teams were added, there were 13 teams in each of 2 conferences. The NFL wanted a schedule were each team would play 11 games against other teams in their conference and 3 games against teams from the other conference.

Devise such a schedule or explain why it is not possible. For simplicity, let's name the teams in the American Football Conference (AFC) $A_1, A_2, \ldots, A_{13}$ and the teams in the National Football Conference (NFC) $N_1, N_2, \ldots, N_{13}$.

2 More Graphs

2.1 Some Example Graphs

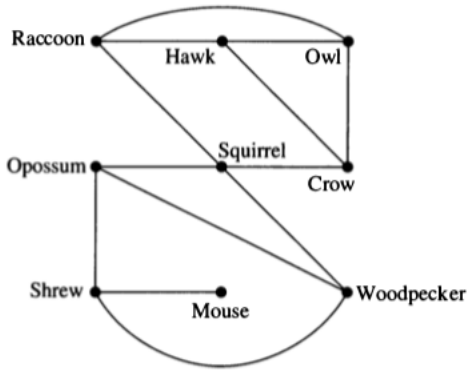


FIGURE 6 A Niche Overlap Graph.

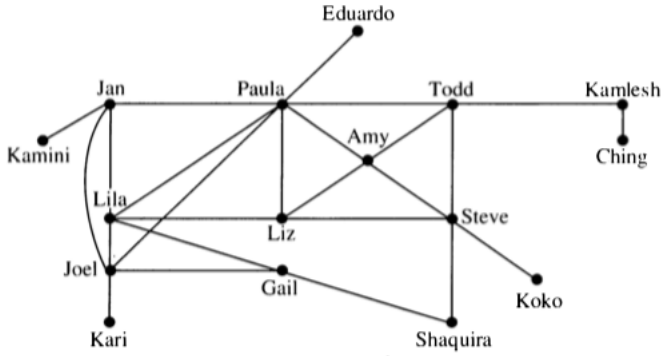


FIGURE 7 An Acquaintanceship Graph.

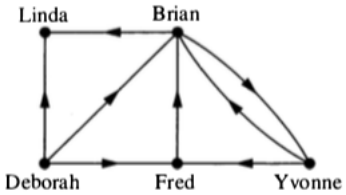


FIGURE 8 An Influence Graph.

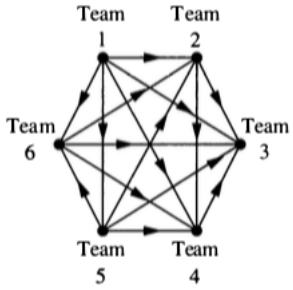


FIGURE 9 A Graph Model of a Round-Robin Tournament.

S_1 $a := 0$
S_2 $b := 1$
S_3 $c := a + 1$
S_4 $d := b + a$
S_5 $e := d + 1$
S_6 $e := c + d$

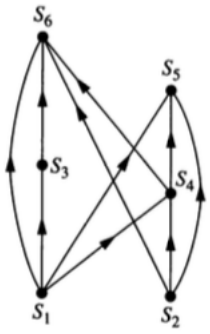


FIGURE 11 A Precedence Graph.

2.2 Some Terminology

- Two vertices are **adjacent** if they are connected by an edge.
- Two edges are **incident** if they share a vertex.
  - For directed graphs, one edge must point into the vertex and one out.
- The **degree** of a vertex in an undirected graph is the number of edges that include the vertex;
  - Self-loops (if they are allowed) contribute 2 to the degree.
  - For a directed graph, we can talk about **in-degree**, **out-degree**, and **total degree**.
- A **path** in a graph is a sequence of edges, each one incident to the next.
  - Can also be described as a sequence of vertices, each one adjacent to the next.
  - For directed graphs, we require that the directions of the edges be compatible.
- A **cycle** is a path that begins and ends at the same vertex.
- A **connected graph** is one where there is a path between any pair of vertices.
  - For a directed graph, we ignore the direction of the edges.
- H is a **subgraph** of G if the vertex set of H is a subset of the vertex set of G and the edge set of H is a subset of the edge set of G, and H is a graph.

2.3 Some Questions

These questions will help make sure you understand the terminology above.

1. In figure 6, which species compete with squirrels?
2. Make a table showing the degree of each vertex in Figure 7. What does a large degree indicate about a person? A small degree?
3. Consider the graph in Figure 6.
   a. Compute the degree of each vertex in Figure 6.
   b. Now add up all the degrees.
   c. How many edges are in the graph?
   d. How is the sum of all the degrees related to the number of edges in the graph? Why? Does this hold for all graphs?
   e. This result is called the **Handshaking Theorem**. Why does it have that name?
4. In Figure 9, assume that $a \to b$ means team a defeated team b.
   a. Compute the in-degree and out-degree of each team in Figure 9.
   b. What do these numbers tell us about the teams?
   c. Who is the winner of the Round-Robin tournament in Figure 9?
5. Consider Figure 8.
   a. Compute the in-degree and the out-degree of each vertex in Figure 8.

b. Add up all the in-degrees.

c. Now add up all the out degrees.

d. What do you notice? Will this hold for all directed graphs, or is this graph special?

6. What is the longest path you can find in Figure 8? What does it represent in terms of the model?

2.4 NFL

From 1961 through 1977 the NFL (National Football League) had a 14-game season. Until 1976, when two new teams were added, there were 13 teams in each of 2 conferences. The NFL wanted a schedule were each team would play 11 games against other teams in their conference and 3 games against teams from the other conference.

Suppose we create such a schedule for the NFL. Consider the part of the schedule that includes only the 13 NFC teams. We can represent this as a directed graph (road team $\rightarrow$ home team). (This graph would be a subgraph of the graph for the entire schedule.)

7. How many vertices does this graph have?
8. What is the total degree of each vertex? Why?
9. What is the sum of all the total degrees?
10. How many edges does this graph have? Why is this impossible?

So a little bit of graph theory shows that the NFL's desired schedule is not possible.

2.5 Some special graphs

- K_n, the complete graph with n vertices.
  - simple graph with every possible edge.
- C_n, the cycle with n vertices.
  - simple graph that consists of a single cycle connecting all the vertices and no other edges.
- W_n, the wheel with $n + 1$ vertices.
  - Formed by adding 1 vertex to C_n and an edge between the new vertex and all the vertices in C_n.
- Q_n, the n-dimensional cube.
  - vertices: bit strings with n bits
  - edges: two vertices are adjacent if and only if their bit strings differ in exactly one position.

11. Draw K_5. How many edges does K_5 have? How many edges does K_n have?
12. Draw C_5. How many edges does C_5 have? How many edges does C_n have?
13. Draw W_5. How many edges does W_5 have? How many edges does W_n have?
14. How many vertices does Q_3 have? How many edges does Q_3 have? Draw Q_3.
15. How many vertices does Q_4 have? How many edges does Q_4 have?
16. In Figure 7 there is a subgraph that is a K_4. List its vertices. Is there a subgraph that is a K_5? How do you know?

3 Graph Representations

3.1 4 Representations of Graphs

There are several ways to represent a (simple) graph. The efficiency of some graph algorithms depends on which representation is used to store the graph.

3.1.1 Edge Lists

An edge list simply lists off the edges (i.e., pairs of vertices) in the graph.

3.1.2 Adjacency Lists

For each vertex i, a list of vertices j such that there is an edge from i to j.

3.1.3 Adjacency Matrix

An adjacency matrix A is a square matrix with a row (and column) for each vertex.

- $A_{ij} = 1$ if there is an edge from vertex i to vertex j
- $A_{ij} = 0$ if there is **not** an edge from vertex i to vertex j

3.1.4 Incidence Matrix

An incidence matrix M has a row for each vertex and a column for each edge.

- $M_{ij} = 1$ if edge j is incident with vertex i.
- $M_{ij} = 0$ if edge j is **not** incident with vertex i.

3.2 Representation to Graphs

Draw graphs with the following representations.

1. edge list: $\{\{1,2\},\{2,3\},\{3,4\},\{4,5\},\{1,3\},\{2,4\},\{3,5\}\}$.

2. adjacency list

vertex	adjacent vertices
1	2, 3, 5
2	1
3	1, 4, 5
4	3, 5
5	1, 3, 4

3. adjacency matrix:

```
0 1 1 1
1 0 1 1
1 1 0 0
1 1 0 0
```

4. adjacency matrix:

```
0 1 1 0
1 0 0 1
1 0 0 1
0 1 1 0
```

5. adjacency matrix

```
0 1 0 0 1 1
1 0 1 1 0 1
0 1 0 0 1 1
0 1 0 0 1 0
1 0 1 1 0 1
1 1 1 0 1 0
```

6. incedence matrix

```
1 1 0 0 0 0
0 0 1 1 0 1
0 0 0 0 1 1
1 0 1 0 0 0
0 1 0 1 1 0
```

7. incedence matrix

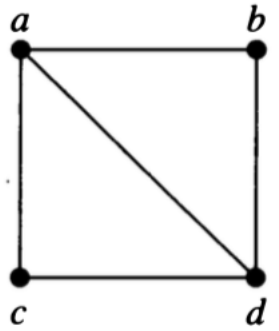
```
1 1 1 0 0 0 0 0
0 1 1 1 0 1 1 0
0 0 0 1 1 0 0 1
0 0 0 0 0 0 1 1
1 0 0 0 1 1 0 0
```

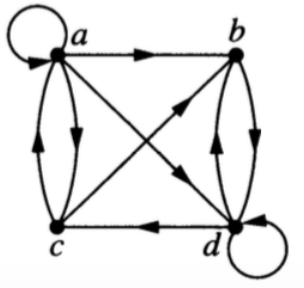
Variations on the theme

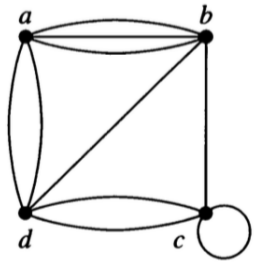
7. Which of these representations can be used (perhaps with slight modification) with **directed graphs**? What adjustments do you need to make?
8. Which of these representations can be used (perhaps with slight modification) for graphs with **multi-edges**? What adjustments do you need to make?
9. Which of these representations can be used (perhaps with slight modification) for graphs with **self-loops**? What adjustments do you need to make?

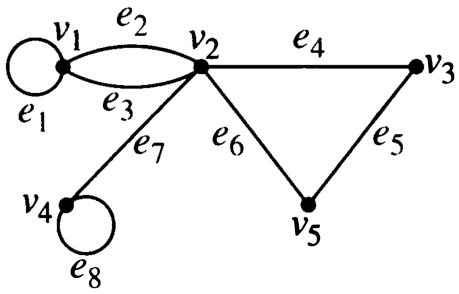
3.3 Graph to representation

7. Use each representation to represent the following graphs or say why it isn't possible.









4 Graph Isomorphism

Two (mathematical) objects are called **isomorphic** if they are "essentially the same" (iso-morph means same-form). What "essentially the same" means depends on the kind of object. For graphs, we mean that the vertex and edge structure is the same. So

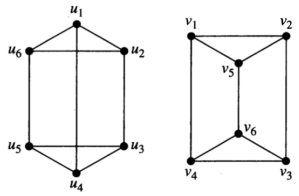
> Graphs G and H are **isomorphic** if there is a bijection (1-1 and onto function)
>
> $$f : G \to H$$
>
> such that there is an edge from a to b in G if and only if there is an edge from $f(a)$ to $f(b)$ in H. The function f is called an **isomorphism**.

(For graphs that allow multi-edges, we require that the number of edges be the same.)

4.1 Showing graphs are isomorphic

When graphs are isomorphic, we can demonstrate this by providing the isomorphism.

1. Show that the following pair of graphs is isomorphic by providing the isomorphism.
   a. What must you demonstrate to convince someone that your function is indeed an isomorphism?
   b. Is there more than one isomorphism? If so, give another one.



4.2 Showing graphs are not isomorphic

Showing that two graphs are *not* isomorphic can be trickier. Somehow you need to demonstrate that no isomorphism exists. Unfortunately, there are a lot of bijections ($n!$ if the graphs each have n vertices), so checking all possible bijections to see if they are isomorphisms would not be a very efficient algorithm.

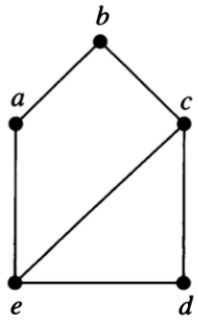
News Flash: Until recently, the best known algorithm for Graph Isomorphism had exponential worst case running time (but was polynomial time on average). In 2015, Laszlo Babai demonstrated an algorithm that runs in sub-exponential time (almost, but not quite polynomial time).

Back to showing two graphs are not isomorphic. We need to demonstrate that no isomorphism exists, but we don't want to check all of them. One good approach is to consider **graph invariants**. A graph invariant is a property that is the same whenever two graphs are isomorphic.

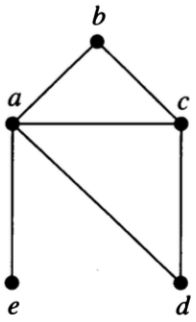
2. Which of the following are graph invariants?
   a. number of vertices
   b. number of edges
   c. degree sequence
   d. adjacency matrix
   e. incidence matrix
   f. having a subgraph isomorphic to K_4
   g. having a subgraph isomorphic to _____ (can you fill in the blank with any graph?)

Unfortunately, matching invariants does not guarantee that graphs are isomorphic, but you can use graph invariants to help narrow the search.

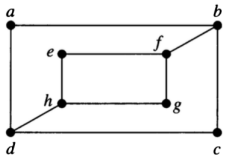
3. Explain why each of the following pairs of graphs is not isomorphic.



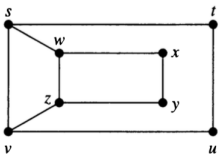
G



H



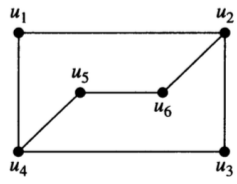
G



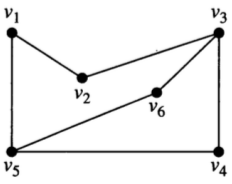
H

4.3 Isomorphic or not?

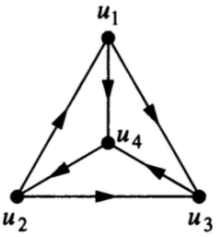
4. For each pair, determine whether the graphs are isomorphic. Be sure to explain how you know.

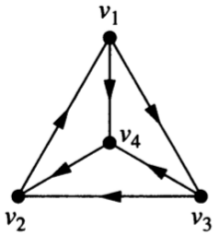


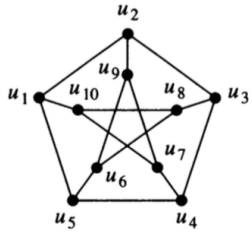
G

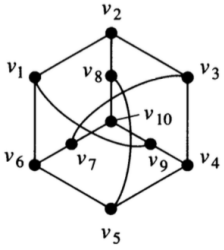


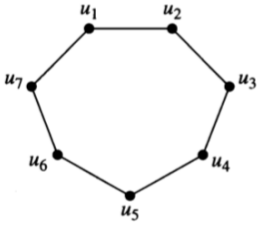
H

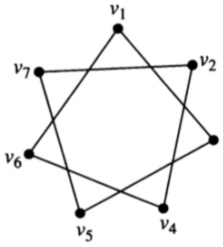












5 Paths in Graphs

5.1 Definition of a path

- Informally, a **path** in a graph is a sequence of edges, each one incident to the next.
  - Can also be described as a sequence of vertices, each one adjacent to the next.
  - For directed graphs, we require that the directions of the edges be compatible.
- More formally, let n be a nonnegative integer and G an undirected [directed] graph. A **path** of **length** n from vertex v_0 to vertex v_n in G is a sequence of n edges $e_1, \ldots, e_n$ of G such that when $1 \leq i \leq n$, then $e_i = \{v_{i-1}, v_i\}$ $[e_i = \langle v_{i-1}, v_i \rangle]$
- A path is a **circuit** (also called a **cycle**) if it begins and ends at the same vertex and has length ≥ 1.
- A path or circuit is **simple** if it does not include the same edge more than once.

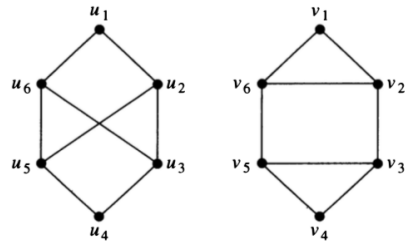
Questions

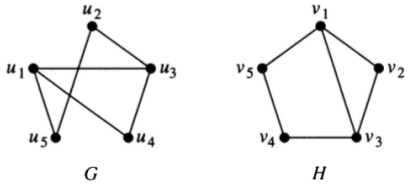
1. What is a path of length 0?
2. Can a path of length 1 be a circuit? If so, draw an example. If not, explain why not.
3. Why are paths defined in terms of edges rather than vertices? In what situations does it matter? When does it not matter?

5.2 Paths and Isomorphism

If $f : G \to H$ is an isomorphism, then f maps paths in G to paths in H. (Why?) This can be useful for finding and isomorphism between G and H or for showing that there is no isomorphism.

4. Answer some questions about the pairs of graphs below.
   a. Compute the degreee sequences for each graph.
   b. Find the shortest and longest simple paths from v_1 to v_4.
   c. Are the graphs isomorphic? Explain.





5.3 Connectedness

- An **undirected graph** is **connected** if there is a path between every pair of vertices.
- A **directed graph** is **strongly connected** if there is a path between every pair of vertices.
- A **directed graph** is **weakly connected** if the underlying undirected graph is connected.
- A connected component of a subgraph H of a graph G is maximal connected subgraph. That is, H must be a subgraph of G, H be connected, but H cannot be a subgraph of a larger connected subgraph of G.

5. Draw a directed graph with 6 vertices that is weakly connected but not strongly connected.
6. Draw an undirected graph with two connected components.

5.4 Euler and Hamilton Paths

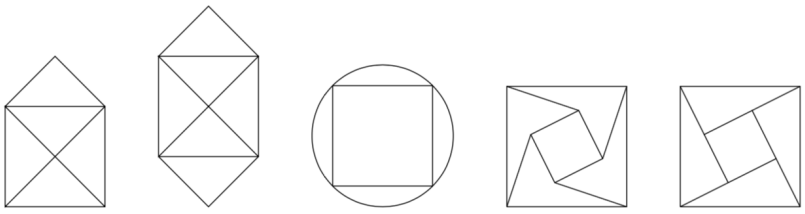
- An **Euler path** is a path that visits every edge of a graph exactly once.
- A **Hamilton path** is a path that visits every vertex exactly once.

Euler paths are named after Leonid Euler who posed the following famous problem about the bridges in Königsberg.

5.4.1 Can you trace it?

Perhaps you have seen the following type of "puzzle". The goal is to trace the figures without lifting your pencil from the paper and without retracing any of the lines.

7. Which kind of path is this puzzle asking for?
8. Which of these can you trace in this manner?

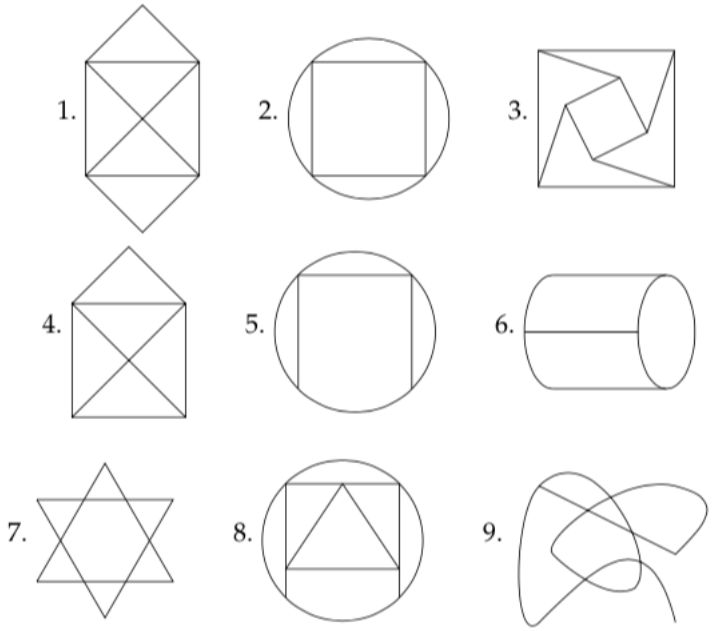


5.5 Relationships in Graphs

A **planar graph** is a graph that *can be* drawn in the plane without any edges crossing.

9. Show that K_4 and Q_3 are both planar graphs.

Here are some graphs that are missing their vertices.



10. Add dots showing vertices so that each graph is a planar graph.

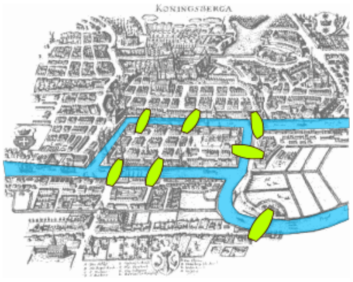
11. Fill in the table below for the graphs above. Do you notice any patterns? Can you make any conjectures? Can you prove any of the conjectures? (Note: an even vertex is a vertex with even degree. An odd vertex is a vertex with odd degree.)

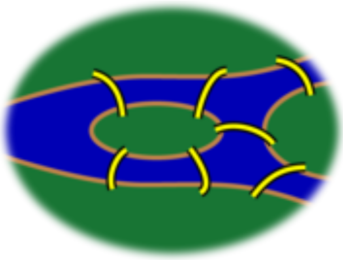
Graph	number of even verticies	number of odd vertices	total degree	number of vertices	number of edges	number of regions	Euler path?	Euler circuit?
1								
2								
3								
4								
5								
6								
7								
8								
9								

5.6 Königsberg Bridge Problem

5.6.1 Original Version

One of the founders of graph theory was Leonid Euler (one of the greatest mathematicians who ever lived). At the time he was alive, the city of Königsberg had seven bridges connecting the two banks of the river flowing through downtown and two islands in that river. Here are a couple maps of downtown Königsberg at the time. One is a bit stylized.





As the story goes, the residents of Königsberg enjoyed walking the bridges. The question is this: How can someone take a Sunday afternoon walk that starts and ends in the same place and crosses each bridge exactly once? (Leaving the map, renting hot air balloons, swimming, etc. are not allowed.)

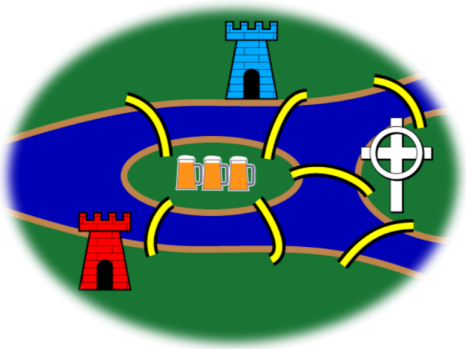
We'll return to this problem in just a moment. But first, a different sort of puzzle.

12. Create a graph that represents the Königsberg Bridge Puzzle (so it can be traced in this manner, if and only if the Königsberg Bridge Puzzle has a solution). What are the vertices? What are the edges? What type of graph is it?

13. Can you trace the Königsberg Bridge graph?

5.6.2 Extentions to the Königsberg bridge problem

Here are some extensions to the Königsberg Bridge problem. First, we embelish the map a bit: The northern bank of the river is occupied by the Schloss, or castle, of the Blue Prince; the southern by that of the Red Prince. The east bank is home to the Bishop's Kirche, or church; and on the small island in the center is a Gasthaus, or inn.



It being customary among the townsmen, after some hours in the Gasthaus, to attempt to walk the bridges, many have returned for more refreshment claiming to have walked each bridge exactly once. However, none have been able to repeat the feat by the light of day.

14. **The Blue Prince's eighth bridge.** The Blue Prince, having analyzed the town's bridge system by means of graph theory, concludes that the bridges cannot be walked. He contrives a stealthy plan to build an eighth bridge so that he can begin in the evening at his castle, walk the bridges, and end at the Gasthaus to brag of his victory. Of course, he wants the Red Prince to be unable to duplicate the feat. Where does the Blue Prince build the eighth bridge?

15. **The Red Prince's ninth bridge.** The Red Prince, infuriated by his brother's solution to the problem, wants to build a ninth bridge, enabling him to begin at his castle, walk the bridges, and end at the Gasthaus to rub dirt in his brother's face. Furthermore, his brother should then no longer walk the bridges himself. Where does the Red Prince build the ninth bridge?

16. **The Bishop's tength bridge.** The Bishop has watched this furious bridge-building with dismay. It upsets the townspeople and, worse, contributes to excessive drunkenness. He wants to build a tenth bridge that allows all the inhabitants to walk the bridges and return to their own beds. Where does the Bishop build the tenth bridge?

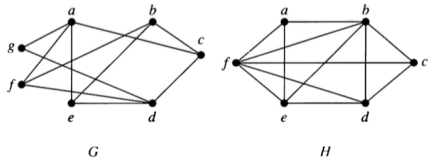
6 Bipartite Graphs

6.1 Definition

A simple graph G is **biparatite** if its vertices can be partitioned into two disjoint subsets V_1 and V_2 and every edge connects a vertex in V_1 with a vertex in V_2. The pair $\langle V_1, V_2 \rangle$ is called a **bipartition** of G.

6.2 Examples

1. Which of the following are bipartite?

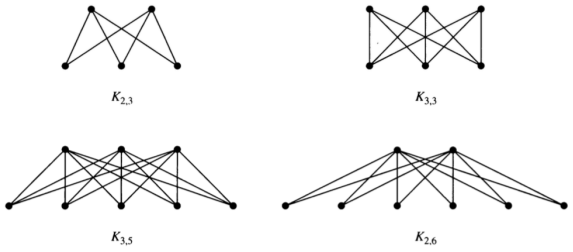


2. For what values of n are the following bipartite?
   a. K_n
   b. C_n
   c. Q_n
3. Can a bipartite graph have more than one bipartition? If so, give an example. If not, explain why not.
4. Describe an algorithm for checking whether a graph is bipartite. How efficient is your algorithm?

6.3 Complete Bipartite Graphs

A complete biparite graph $K_{m,n}$ has two disjoint sets of vertices V_1 and V_2 with m and n elements. There is an edge between x and y if and only if $x \in V_1$ and $y \in V_2$ or $x \in V_2$ and $y \in V_1$.

Here are some examples.



5. Draw the following graphs: $K_{2,2}$, $K_{4,2}$, $K_{4,3}$

7 Planar Graphs

A graph is a **planar graph** if it can be drawn in the plane with no edge crossings. [Note: You may need to rearrange the vertices to avoid crossings.]

7.1 Euler and Planarity

We have already seen that for planar graphs,

$$v - e + r = 1 + \text{number of connected components},$$

where v is the number of vertices, e the number of edges and r the number of regions.

In particular, for a connected graph we have $v - e + r = 2$.

1. Show that the following are planar.
   a. $K_{2,2}$
   b. $K_{2,3}$
   c. $K_{2,4}$
   d. Is $K_{2,n}$ planar for every n?
2. Use Euler's formula to show that K_5 is not planar. (Hint: how many regions would K_5 have it if were planar? Why is that problematic?)
3. Use Euler's formula to show that $K_{3,3}$ is not planar.

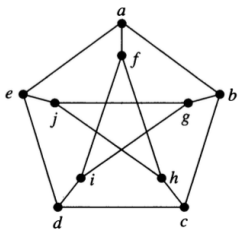
7.2 Kuratowski's Theorem

The proof of the following theorem is too involved for this course, but it turns out that all nonplanar graphs contain a "homeomorphic copy" of either K_5 or $K_{3,3}$.

Kuratowski's Theoerm: A graph G is nonplanar if and only if it contains a subgraph H that is homeomorphic to either K_5 or $K_{3,3}$.

- two graphs are **homeomorphic** if they can each be obtained from the same graph by a sequence of elementary subdivisions.
- an **elementary subdivision** of a graph G removes one edge (u, v) and adds a new vertex n and two new edges (u, n) and (v, n).

4. Create a graph with 5 vertices and 7 edges. Now perform three elementary subdivisions on your graph.
5. Describe what an elementary subdivision is in simple words a kindergarterner could understand.
6. Explain why two homeomorphic graphs are either both planar or both nonplanar.
7. Use Kuratowski's Theorem to show that the Petersen graph is nonplanar.



8 Euler Paths

8.1 An Euler Path Algorithm

Here is python code for an Euler path algorithm.

```
# find an Euler path/circuit or report there is none.
# this version assumes (without checking) that the graph is connected.
def euler_path(graph, verbose = False):
  degrees = graph.degrees()
  odd_vertices = [v for v in degrees.keys() if degrees[v] % 2 == 1]

  if len(odd_vertices) == 2:
    v_init = odd_vertices[0]
  elif len(odd_vertices) == 0:
    v_init = graph.some_vertex()
  else:
    return(None)

  stack = [v_init]
  path = []

  while stack:
    if verbose:
      print "stack =", stack, " path =", path
    u = stack[-1]
    v = graph.adjacent_vertex(u)
    if v:
      stack.append(v)
      graph.delete_edge((u,v))
    else:
      # move vertex from stack to path
      path.append(stack.pop())
  return(path)
```

8.1.1 Examples

```
verbose = False
G1 = Graph([(1,2), (1,3), (1,4), (1,5), (2,3), (2,5),
            (2,6), (3,4), (3,5), (4,5), (4,6)])
print euler_path(G1, verbose = verbose)
```

```
[1, 5, 4, 6, 2, 5, 3, 4, 1, 3, 2, 1]
```

```
G2 = Graph([(1,2), (1,3), (1,4), (2,3), (2,5),
            (2,6), (3,4), (3,5), (4,5), (4,6)])
print euler_path(G2, verbose = verbose)
```

```
[5, 4, 6, 2, 5, 3, 4, 1, 3, 2, 1]
```

```
G3 = Graph([(1,2), (1,3), (1,4), (2,3), (2,5), (2,6),
            (3,4), (3,5), (4,5), (4,6), (5,6), (3,7)])
print euler_path(G3, verbose = verbose)
```

```
None
```

8.1.2 Questions

1. What would the output of `euler_path(G1, verbose = True)` be?

   (For this question, you may assume that `adjacent_vertex()` will return the smallest numbered adjacent vertex and `some_vertex()` the smallest numbered vertex in the graph.)

2. Pick a graph representation (edge list, adjacency list, adjacency matrix, incidence matrix) and determine the effiicency of each method of the `Graph` class used in this algorithm.

   (For this question, you do not need to return any particular vertex from `adjacent_vertex()` or `some_vertex()`. Which vertex does you algorithm choose?)

3. For your graph representation, determine the efficiency of the algorithm.
   Be sure to consider the efficiency of the methods in the `Graph` class.

4. Repeat questions 2 and 3 for other representations.

5. What happens in this algorithm if the graph provided is not connected?

6. Can this algorithm handle multi-edges and self loops?

7. Will the algorithm work for directed graphs?

8. This algorithm assumes the graph is connected. What is the efficiency of checking that a graph is connected?