9.4 Traversal: Eulerian and Hamiltonian Graphs

The subject of graph traversals has a long history. In fact, the solution by Leonhard Euler (Switzerland, 1707-83) of the Königsberg Bridge Problem is considered by many to represent the birth of graph theory.

The Königsberg Bridge Problem and Eulerian Graphs

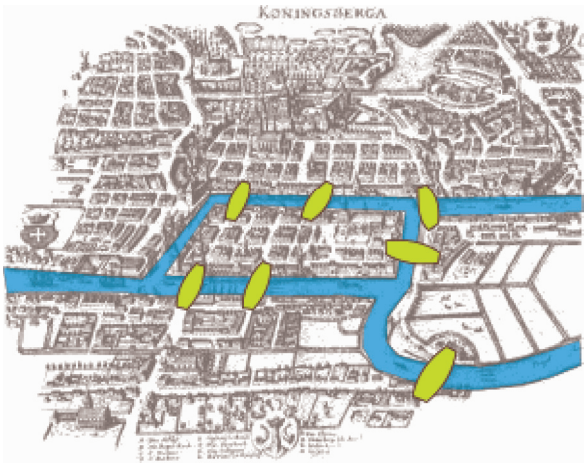


Figure 9.4.1
Map of Königsberg

A map of the Prussian city of Königsberg (circa 1735) in Figure 9.4.1 shows that there were seven bridges connecting the four land masses that made up the city. The legend of this problem states that the citizens of Königsberg searched in vain for a walking tour that passed over each bridge exactly once. No one could design such a tour and the search was abruptly abandoned with the publication of Euler's Theorem.

Theorem 9.4.1: Euler' s Theorem—Königsberg Case. *No walking tour of Königsberg can be designed so that each bridge is used exactly once.*

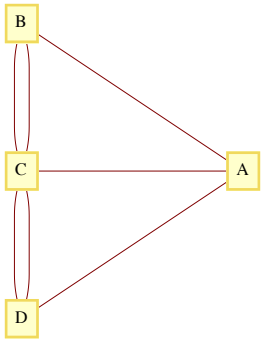


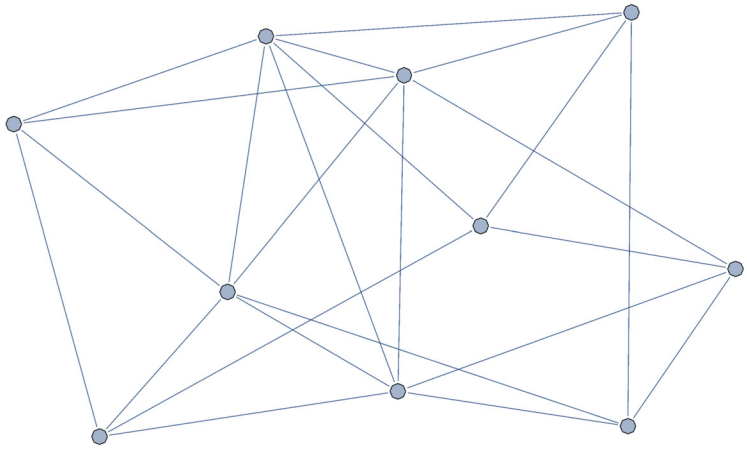
Figure 9.4.2
Multigraph representation of Königsberg

Proof: The map of Königsberg can be represented as an undirected multigraph, as in Figure 9.4.2. The four land masses are the vertices and each edge represents a bridge. The desired tour is then a path that uses each edge once and only once. Since the path can start and end at two different vertices, there are two remaining vertices that must be intermediate vertices in the path. If x is an intermediate vertex, then every time that you visit x, you must use two of its incident edges, one to enter and one to exit. Therefore, there must be an even number of edges connecting x to the other vertices. Since every vertex in the Königsberg graph has an odd number of edges, no tour of the type that is desired is possible. ■

As is typical of most mathematicians, Euler wasn't satisfied with solving only the Königsberg problem. His original theorem, which is paraphrased below, concerned the existence of paths and circuits like those sought in Königsberg. These paths and circuits have become associated with Euler's name.

Definitions: Eulerian Paths, Circuits, Graphs. *A Eulerian path through a graph is a path whose edge list contains each edge of the graph exactly once. If the path is a circuit, then it is called a Eulerian circuit. A Eulerian graph is a graph that possesses a Eulerian path.*

Example 9.4.1. Without tracing any paths, we can be sure that the graph below has an Eulerian circuit because all vertices have an even degree. This follows from the following theorem.



Theorem 9.4.2: Euler's Theorem—General Case. *An undirected graph is Eulerian if and only if it is connected and has either zero or two vertices with an odd degree. If no vertex has an odd degree, then the graph has a Eulerian circuit.*

Proof: It can be proven by induction that the number of vertices in an undirected graph that have an odd degree must be even. We will leave the proof of this fact to the reader as an exercise. The necessity of having either zero or two vertices of odd degree is clear from the proof of the Königsberg case of this theorem. Therefore, we will concentrate on proving that this condition is sufficient to ensure that a graph is Eulerian. Let k be the number of vertices with odd degree.

Phase 1. If $k = 0$, start at any vertex, v_0, and travel along any path, not using any edge twice. Since each vertex has an even degree, this path can always be continued past each vertex that you reach except v_0. The result is a circuit that includes v_0. If $k = 2$, let v_0 be either one of the vertices of odd degree. Trace any path starting at v_0 using up edges until you can go no further, as in the $k = 0$ case. This time, the path that you obtain must end at the other vertex of odd degree that we will call v_1. At the end of Phase 1, we have an initial path that may or may not be Eulerian. If it is not Eulerian, Phase 2 can be repeated until all of the edges have been used. Since the number of unused edges is decreased in any use of Phase 2, a Eulerian path must be obtained in a finite number of steps.

Phase 2. As we enter this phase, we have constructed a path that uses a proper subset of the edges in our graph. We will refer to this path as the current path. Let V be the vertices of our graph, E the edges, and E_u the edges that have been used in the current path. Consider the graph $G' = (V, E - E_u)$. Note that every vertex in G' has an even degree. Select any edge, e, from G'. Let v_a and v_b be the vertices that e connects. Trace a new path starting at v_a whose first edge is e. We can be sure that at least one vertex of the new path is also in the current path since (V, E) is connected. Starting at v_a, there exists a path in (V, E) to any vertex in the current path. At some point along this path, which we can consider the start of the new path, we will have intersected the current path. Since the degree of each vertex in G' is even, any path that we start at v_a can be continued until it is a circuit. Now, we simply augment the current path with this circuit. As we travel along the current path, the first time that we intersect the new path, we travel along it (see Figure 9.4.3). Once we complete the circuit that is the new path, we resume the traversal of the current path.

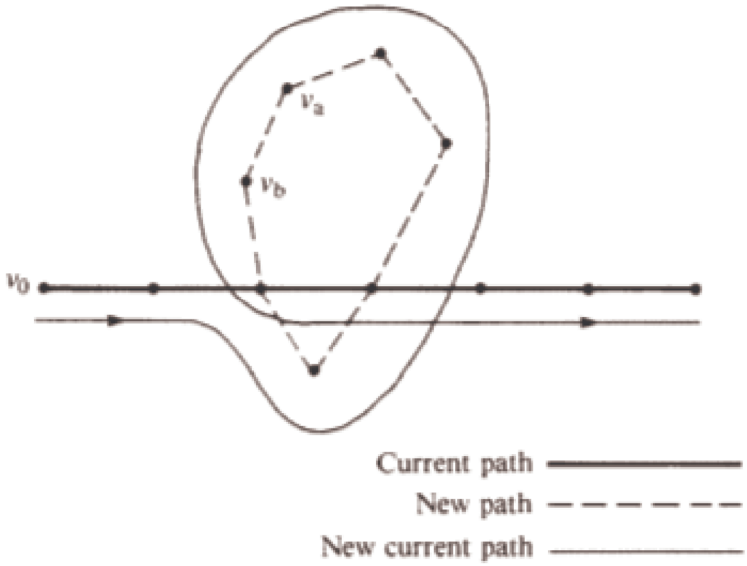


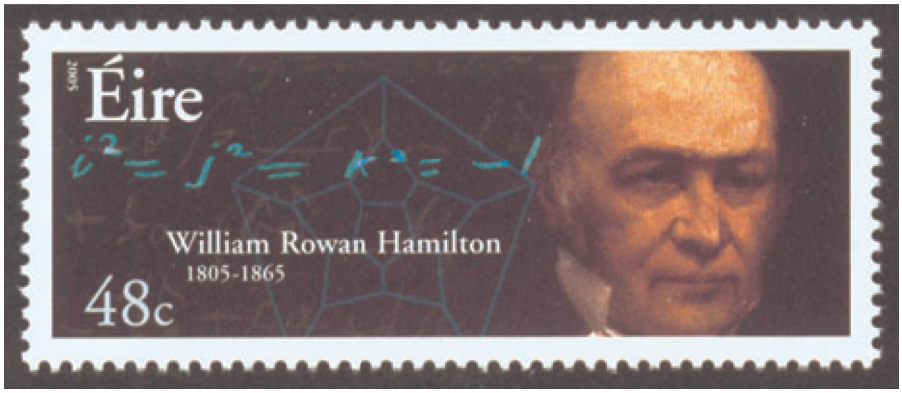
Figure 9.4.3
Augmenting the current path in the proof of Theorem 9.4.2

If the result of this phase is a Eulerian path, then we are finished; otherwise, repeat this phase. #

Example 9.4.2. The complete undirected graphs K_2 and $K_{2n+1}, n = 1, 2, 3, \ldots..$, are Eulerian. If $n > 1$, then K_{2n} is not Eulerian.

HAMILTONIAN GRAPHS

To search for a path that uses every vertex of a graph exactly once seems to be a natural next problem after you have considered Eulerian graphs. The Irish mathematician Sir William Hamilton (1805-65) is given credit for first defining such paths. He is also credited with discovering the quaternions, for which he was honored by the Irish government with a postage stamp in 2004.



Definition: Hamiltonian Paths, Circuits, and Graphs. *A Hamiltonian path through a graph is a path whose vertex list contains each vertex of the graph exactly once, except if the path is a circuit, in which case the initial vertex appears a second time as the terminal vertex. If the path is a circuit, then it is called a Hamiltonian circuit. A Hamiltonian graph is a graph that possesses a Hamiltonian path.*

Example 9.4.3. Figure 9.4.4 shows a graph that is Hamiltonian. In fact, it is the graph that Hamilton used as an example to pose the question of existence of Hamiltonian paths in 1859. In its original form, the puzzle that was posed to readers was called "Around the World." The vertices were labeled with names of major cities of the world and the object was to complete a tour of these cities. The graph is also referred to as the dodecahedron graph, where vertices correspond with the corners of a dodecahedron and the edges are the edges of the solid that connect the corners.

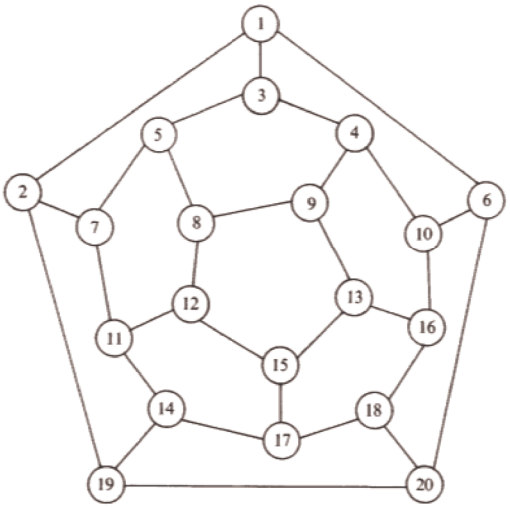


Figure 9.4.4
The dodecahedron graph, a Hamiltonian graph.

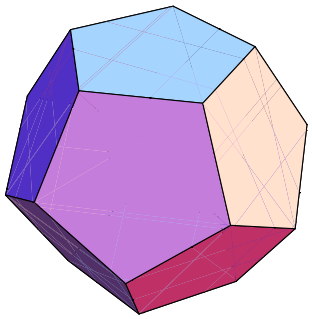


Figure 9.4.5
The regular dodecahedron

Unfortunately, a simple condition doesn't exist that characterizes a Hamiltonian graph. An obvious necessary condition is that the graph be connected; however, there is a connected undirected graph with four vertices that is not Hamiltonian. Can you draw such a graph?

A Note on What Is Possible and What Is Impossible. The search for a Hamiltonian path in a graph is typical of many simple-sounding problems in graph theory that have proven to be very difficult to solve. Although there are simple algorithms for conducting the search, they are impractical for large problems because they take such a long time to complete as graph size increases. Currently, every algorithm to search for a Hamiltonian path in a graph takes exponential time to complete. That is, if $T(n)$ is the time it takes to search a graph of n vertices, then there is a positive real number a, $a > 1$, such that $T(n) > a^n$ for all but possibly a finite number of positive values for n. No matter how close to 1 we can make a, a^n will grow at such a fast rate that the algorithm will not be feasible for large values of n. For a given algorithm, the value of a depends on the relative times that are assigned to the steps, but in the search for Hamiltonian paths, the actual execution time for known algorithms is large with 20 vertices. For 1,000 vertices, no algorithm is likely to be practical, and for 10,000 vertices, no currently known algorithm could be executed.

It is an unproven but widely held belief that no faster algorithm exists to search for Hamiltonian paths. A faster algorithm would have to be one that takes only, for example, polynomial time; that is, $T(n) < p(n)$, for some polynomial sequence p. To sum up, the problem of determining whether a graph is Hamiltonian is theoretically possible; however, for large graphs we consider it a practical impossibility. Many of the problems we will discuss in the next section, particularly the Traveling Salesman Problem, are thought to be impossible in the same sense.

> Note: There are so-called *subexponential algorithms* with time complexities that lie between exponential and polynomial. These speeds are also thought to be generally unattainable in finding Hamiltonian paths.

Definition: The n-cube. *Let $n \geq 1$, and let B^n be the set of strings of 0's and 1's with length n. The n-cube is the undirected graph with a vertex for each string in B^n and an edge connecting each pair of strings that differ in exactly one position.*

The 1-cube, 2-cube, 3-cube, and 4-cube are shown in Figure 9.4.6.

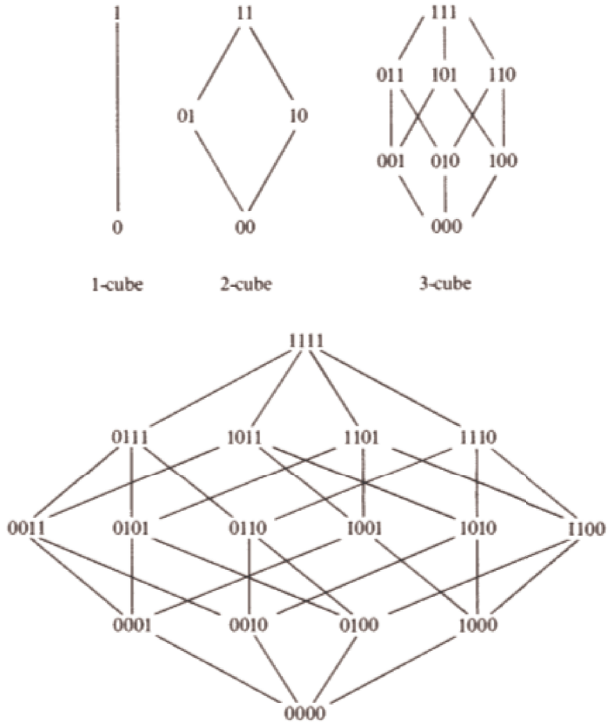


Figure 9.4.6
n-cubes, $n = 1, 2, 3, 4$

The Gray Code. A Hamiltonian circuit of the n-cube can be described recursively. The circuit itself, called the Gray Code, is not the only Hamiltonian circuit of the n-cube, but it is the easiest to describe. The standard way to write the Gray Code is as a column of strings, where the last string is followed by the first string to complete the circuit.

Basis ($n = 1$): The Gray Code for the 1-cube is $G_1 = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$. Note that the edge between 0 and 1 is used twice in this circuit. That doesn't violate any rules for Hamiltonian circuits, but can only happen if a graph as two vertices.

Recursion: Given the Gray Code for the n-cube, $n > 1$, then G_{n+1} is obtained by (1) listing G_n with each string prefixed with 0, and then (2) reversing the list of strings in G_n with each string prefixed with 1. Symbolically, the recursion can be expressed as

$$G_{n+1} = \begin{pmatrix} 0\,G_n \\ 1\,G_n{}^r \end{pmatrix}$$

where $G_n{}^r$ is the reverse of list G_n. The Gray Codes for the 2-cube and 3-cube are

$$G_2 = \begin{pmatrix} 00 \\ 01 \\ 11 \\ 10 \end{pmatrix} \quad \text{and} \quad G_3 = \begin{pmatrix} 000 \\ 001 \\ 011 \\ 010 \\ 110 \\ 111 \\ 101 \\ 100 \end{pmatrix}$$

Applications of the Gray Code. One application of the Gray code was discussed in the Introduction to this book. An other application is in statistics. In a statistical analysis, there is often a variable that depends on several factors, but exactly which factors are significant may not be obvious. For each subset of factors, there would be certain quantities to be calculated. One such quantity is the multiple correlation coefficient for a subset. If the correlation coefficient for a given subset, A, is known, then the value for any subset that is obtained by either deleting or adding an element to A can be obtained quickly. To calculate the correlation coefficient for each set, we simply travel along G_n, where n is the number of factors being studied. The first vertex will always be the string of 0's, which represents the empty set. For each vertex that you visit, the set that it corresponds to contains the k^{th} factor if the k^{th} character is a 1.

EXERCISES FOR SECTION 9.4

A Exercises

1. Locate a map of New York City and draw a graph that represents its land masses, bridges and tunnels. Is there a Eulerian path through New York City? You can do the same with any other city that has at least two land masses.

2. Which of the drawings in Figure 9.4.7 can be drawn without removing your pencil from the paper and without drawing any line twice?

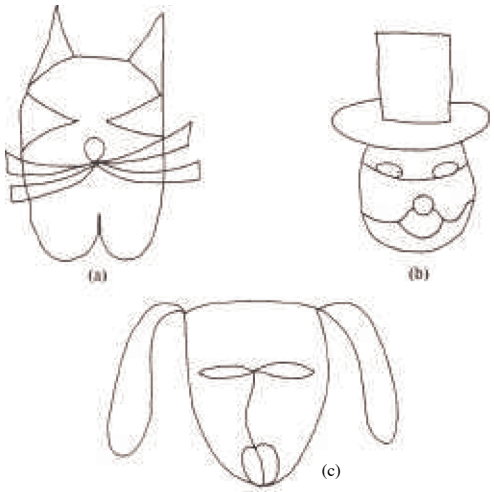


Figure 9.4.7
Exercise 2

3. Write out the Gray Code for the 4-cube.

4. Find a Hamiltonian circuit for the dodecahedron graph in Figure 9.4.4.

5. The Euler Construction Company has been contracted to construct an extra bridge in Königsberg so that a Eulerian path through the town exists. Can this be done, and if so, where should the bridge be built?

6. (a) Determine which of the graphs in Figure 9.4.8 has a Eulerian path?

   (b) Find a Eulerian path for the graphs that have one.

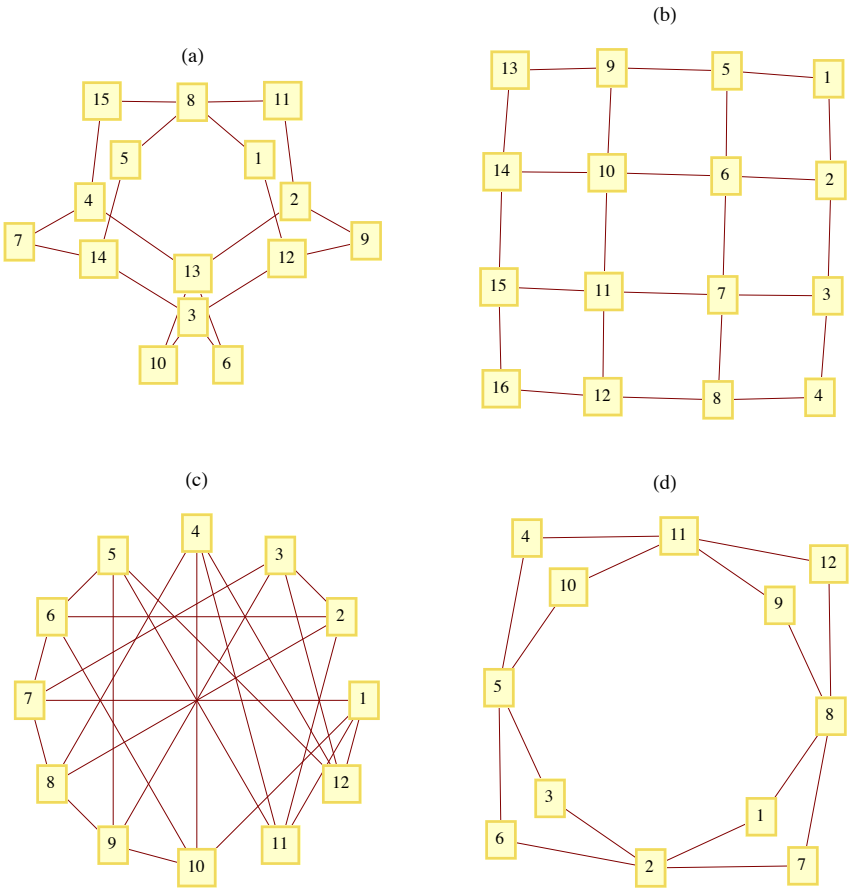


Figure 9.4.8
Exercise 6

B Exercises

7. Formulate Euler's theorem for directed graphs.

8. Prove that the number of vertices in an undirected graph with odd degree must be even. (Hint: Prove by induction on the number of edges.)

9. (a) Under what conditions will a round-robin tournament graph be Eulerian?

   (b) Prove that every round-robin tournament graph is Hamiltonian.

10. For what values of n is n-cube Eulerian.

9.5 Graph Optimization

The common thread that connects all of the problems in this section is the desire to optimize (maximize or minimize) a quantity that is associated with a graph. We will concentrate most of our attention on two of these problems, the Traveling Salesman Problem and the Maximum Flow Problem. At the close of this section, we will discuss some other common optimization problems.

Definition: Weighted Graph. *A weighted graph,* (V, E, w)*, is a graph* (V, E) *together with a weight function* $w : E \to \mathbb{R}$*. If* $e \in E$*,* $w(e)$ *is the weight on edge* e*.*

As you will see in our examples, $w(e)$ is usually a cost associated with the edge e; therefore, most weights will be positive.

Example 9.5.1. Let V be the set of six capital cities in New England: Boston, Augusta, Hartford, Providence, Concord, and Montpelier. Let E be $\{\{a, b\} \in V \times V \mid a \neq b\}$; that is, (V, E) is a complete unordered graph. An example of a weight function on this graph is

$$w(c_1, c_2) = \text{the distance from } c_1 \text{ to } c_2 .$$

Many road maps define distance functions as in Figure 9.5.1.

	ME	MA	NH	CT	VT	RI
Augusta, ME	–	165	148	266	190	208
Boston, MA	165	–	75	103	192	43
Concord, NH	148	75	–	142	117	109
Hartford, CT	266	103	142	–	204	70
Montpelier, VT	190	192	117	204	–	223
Providence, RI	208	43	109	70	223	–

FIGURE 9.5.1
Distances between capital cities of New England

The Traveling Salesman Problem

The Traveling Salesman Problem is, given a weighted graph, to find a circuit $(e_1, e_2, \ldots, e_n)$ that visits every vertex at least once and minimizes the sum of the weights,

$$\sum_{i=1}^{n} w(e_i)$$

Any such circuit is called an *optimal path*.

Notes. (a) Some statements of the Traveling Salesman Problem require that the circuit be Hamiltonian. In many applications, the graph in question will be complete and this restriction presents no problem.

(b) If the weight on each edge is constant, for example, $w(e) = 1$, then the solution to the Traveling Salesman Problem will be any Hamiltonian circuit, if one exists.

Example 9.5.2. The Traveling Salesman Problem gets its name from the situation of a salesman who wants to minimize the number of miles that he travels in visiting his customers. For example, if a salesman from Boston must visit the other capital cities of New England, then the problem is to find a circuit in the weighted graph of Example 9.5.1. Note that distance and cost are clearly related in this case. In addition, tolls and traffic congestion might also be taken into account in this case.

The search for an efficient algorithm that solves the Traveling Salesman has occupied researchers for years. If the graph in question is complete, there are $(n - 1)!$ different circuits. As n gets large, it is impossible to check every possible circuit. The most efficient algorithms for solving the Traveling Salesman Problem take an amount of time that is proportional to $n\,2^n$. Since this quantity grows so quickly, we can't expect to have the time to solve the Traveling Salesman Problem for large values of n. Most of the useful algorithms that have been developed have to be heuristic; that is, they find a circuit that should be close to the optimal one. One such algorithm is the "closest neighbor" algorithm, one of the earliest attempts at solving the Traveling Salesman Problem. The general idea behind this algorithm is, starting at any vertex, to visit the closest neighbor to the starting point. At each vertex, the next vertex that is visited is the closest one that has not been reached. This shortsighted approach typifies heuristic algorithms called *greedy algorithms*, which attempt to solve a minimization (maximization) problem by minimizing (maximizing) the quantity associated with only the first step.

Algorithm 9.5.1. The Closest Neighbor Algorithm. *Let* $G = (V, E, w)$ *be a complete weighted graph with* $|V| = n$*. The closest neighbor circuit through G starting at* v_1 *is* $(v_1, v_2, \ldots, v_n)$*, defined by the steps:*

1. $V_1 = V - \{v_1\}$.
2. For $k = 2$ *to* $n - 1$
  2.1 $v_k =$ *the closest vertex in* V_{k-1} *to* v_{k-1}
    $(*\ w(v_{k-1}, v_k) = \min(w(v_{k-1}, v) \mid v \in V_{k-1})\ *)$
    In case of a tie for closest, v_k *may be chosen arbitrarily.*
  2.2 $V_k = V_{k-1} - \{v_k\}$
3. $v_n =$ *the only element of* V_n.

The cost of the closest neighbor circuit is

$$\sum_{k=1}^{n-1} w(v_k, v_{k+1}) + w(v_n, v_1)$$

Example 9.5.3. The closest neighbor circuit starting at A in Figure 9.5.2 is (1, 3, 2, 4, 1), with a cost of 29. The optimal path is (1, 2, 3, 4, 1), with a cost of 27.

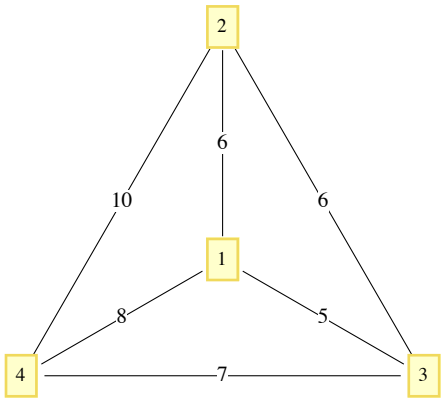


FIGURE 9.5.2
Example 9.5.3

Although the closest neighbor circuit is often not optimal, we may be satisfied if it is close to optimal. If C_{opt} and C_{cn} are the costs of optimal and closest neighbor circuits in a graph, then it is always the case that $C_{\text{opt}} \le C_{\text{cn}}$ or $\frac{C_{\text{cn}}}{C_{\text{opt}}} \ge 1$. We can assess how good the closest neighbor algorithm is by determining how small the quantity $\frac{C_{\text{cn}}}{C_{\text{opt}}}$ gets. If it is always near 1, then the algorithm is good. However, if there are graphs for which it is large, then the algorithm may be discarded. Note that in Example 9.5.3, $\frac{C_{\text{cn}}}{C_{\text{opt}}} = \frac{29}{27} \approx 1.074$. A 7% increase in cost may or may not be considered significant, depending on the situation.

Example 9.5.4. A salesman must make stops at vertices A, B, and C, which are all on the same one-way street. The graph in Figure 9.5.3 is weighted by the function

$w(i, j)$ = the time it takes to drive from vertex i to vertex j.

Note that if j is down the one-way street from i, then $w(i, j) < w(j, i)$. The values of C_{opt} , and C_{cn} are 20 and 32, respectively. Verify that C_{cn} is 32 by using the closest neighbor algorithm. The value of $\frac{C_{\text{cn}}}{C_{\text{opt}}} = 1.6$ is significant in this case since our salesman would spend 60% more time on the road if he used the closest neighbor algorithm.

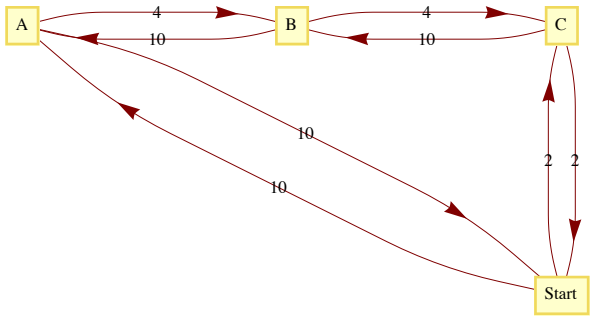


FIGURE 9.5.3
Example 9.5.4

A more general result relating to the closest neighbor algorithm presumes that the graph in question is complete and that the weight function satisfies the conditions

(1) $w(x, v) = w(y, x)$ for all x, y in the vertex set, and

(2) $w(x, y) + w(y, z) \ge w(x, z)$ for all x, y, z in the vertex set.

The first condition is called the *symmetry condition* and the second is the *triangle inequality*.

The following theorem's reference needs to be updated:

Theorem 9.5.1. If (V, E, w) is a complete weighted graph that satisfies the symmetry and triangle inequality conditions, then

$$\frac{C_{cn}}{C_{opt}} \le \frac{\lceil log_2(2n) \rceil}{2} \qquad (9.5a)$$

Proof: See Liu, pages 105-109.

Notes: (a) If $|V| = 8$, then this theorem says that C_{cn} can be no larger than twice the size of C_{opt}; however, it doesn't say that the closest neighbor circuit will necessarily be that far from an optimal circuit. The quantity $\frac{\lceil \log_2(2n) \rceil}{2}$ is called an upper bound for the ratio $\frac{C_{cn}}{C_{opt}}$. It tells us only that things can't be any worse than the upper bound. Certainly, there are many graphs with eight vertices such that the optimal and closest neighbor circuits are the same. What is left unstated in this theorem is whether there are graphs for which the quantities in 9.5a are equal. If there are such graphs, we say that the upper bound is *sharp*.

(b) The value of $\frac{C_{cn}}{C_{opt}}$ in Example 9.5.4 is 1.6, which is greater than $\frac{\lceil \log_2(2 \times 4) \rceil}{2} = 1.5$; however, the weight function in this example does not satisfy the conditions of the theorem.

The Traveling Salesman Problem—Unit Square Version

Example 9.5.5. A robot is programmed to weld joints on square metal plates. Each plate must be welded at prescribed points on the square. To minimize the time it takes to complete the job, the total distance that a robot's arm moves should be minimized. Let $d(P, Q)$ be the distance between P and Q. Assume that before each plate can be welded, the arm must be positioned at a certain point P_0 . Given a list of n points, we want to put them in order so that

$$d(P_0, P_1) + d(P_1, P_2) + \cdots + d(P_{n-1}, P_n) + d(P_n, P_0)$$

is as small as possible.

The type of problem that is outlined in Example 9.5.5 is of such importance that it is probably the most studied version of the Traveling Salesman Problem. What follows is the usual statement of the problem. Let $[0, 1] = \{x \in \mathbb{R} \mid 0 \le x \le 1\}$, and let $S = [0, 1]^2$, the unit square. Given n pairs of real numbers $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$ in S that represent the n vertices of a K_n , find a circuit of the graph that minimizes the sum of the distances traveled in traversing the circuit.

Since the problem calls for a circuit, it doesn't matter which vertex we start at; assume that we will start at (x_1, y_1). Once the problem is solved, we can always change our starting position. A function can most efficiently describe a circuit in this problem. Every bijection $f : \{1, \ldots, n\} \to \{1, \ldots, n\}$ with $f(1) = 1$ describes a circuit

$$(x_1, y_1), \left(x_{f(2)}, y_{f(2)}\right), \ldots, \left(x_{f(n)}, y_{f(n)}\right)$$

Since there are $(n - 1)!$ such bijections, an examination of all possible circuits is not feasible for large values of n.

One popular heuristic algorithm is the strip algorithm:

Algorithm 9.5.2: The Strip Algorithm. *Given n points in the unit square:*

Phase 1:

(1.1) Divide the square into $\left\lceil \sqrt{n/2} \right\rceil$ vertical strips, as in Figure 9.5.4. Let d be the width of each strip. If a point lies on a boundary between two strips, consider it part of the left-hand strip.

(1.2) Starting from the left, find the first strip that contains one of the points. Locate the starting point by selecting the first point that is encountered in that strip as you travel from bottom to top. We will assume that the first point is (x_1, y_1)

(1.3) Alternate traveling up and down the strips that contain vertices until all of the vertices have been reached.

(1.4) Return to the starting point.

Phase 2:

(2.1) Shift all strips $d/2$ units to the right (creating a small strip on the left).

(2.2) Repeat Steps 1.2 through 1.4 of Phase 1 with the new strips.

When the two phases are complete, choose the shorter of the two circuits obtained.

Step 1.3 needs a bit more explanation. How do you travel up or down a strip? In most cases, the vertices in a strip will be vertically distributed so that the order in which they are visited is obvious. In some cases, however, the order might not be clear, as in the third strip in Phase I of Figure 9.5.4. Within a strip, the order in which you visit the points (if you are going up the strip) is determined thusly: (x_i, y_i) precedes $\left(x_j, y_j\right)$

if $y_i < y_j$ or if $y_i = y_j$ and $x_i < x_j$. In traveling down a strip, replace $y_i < y_j$ with $y_i > y_j$.

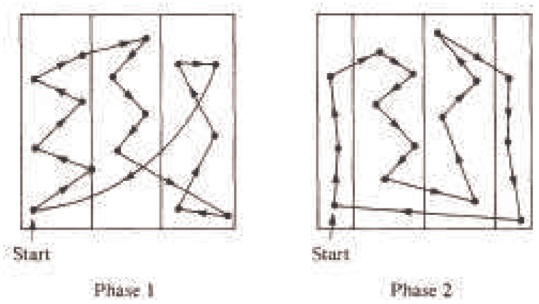


FIGURE 9.5.4
The Strip Algorithm

The selection of $\lceil \sqrt{n/2} \rceil$ strips was made in a 1959 paper by Beardwood, Halton, and Hammersley. It balances the problems that arise if the number of strips is too small or too large. If the square is divided into too few strips, some strips may be packed with vertices so that visiting them would require excessive horizontal motion. If too many strips are used, excessive vertical motion tends to be the result. An update on what is known about this algorithm is contained in the paper by K. J. Supowit, E. M. Reingold, and D. A. Plaisted.

Since the construction of a circuit in the square consists of sorting the given points, it should come as no surprise that the strip algorithm requires a time that is roughly a multiple of $n \log n$ time units when n points are to be visited.

The worst case that has been encountered with this algorithm is one in which the circuit obtained has a total distance of approximately $\sqrt{2n}$ (see Sopowit et al.).

NETWORKS AND THE MAXIMUM FLOW PROBLEM

Definition: Network. *A network is a simple weighted directed graph that contains two distinguished vertices called the source and the sink with the property that the indegree of the source and outdegree of the sink are both zero. The weight function on a network is the capacity function.*

An example of a real situation that can be represented by a network is a city's water system. A reservoir would be the source, while a distribution point in the city to all of the users would be the sink. The system of pumps and pipes that carries the water from source to sink makes up the remaining network. We can assume that the water that passes through a pipe in one minute is controlled by a pump and the maximum rate is determined by the size of the pipe and the strength of the pump. This maximum rate of flow through a pipe is called its capacity and is the information that the weight function of a network contains.

Example 9.5.6. Consider the system that is illustrated in Figure 9.5.5. The numbers that appear next to each pipe indicate the capacity of that pipe in thousands of gallons per minute. This map can be drawn in the form of a network, as in Figure 9.5.6.

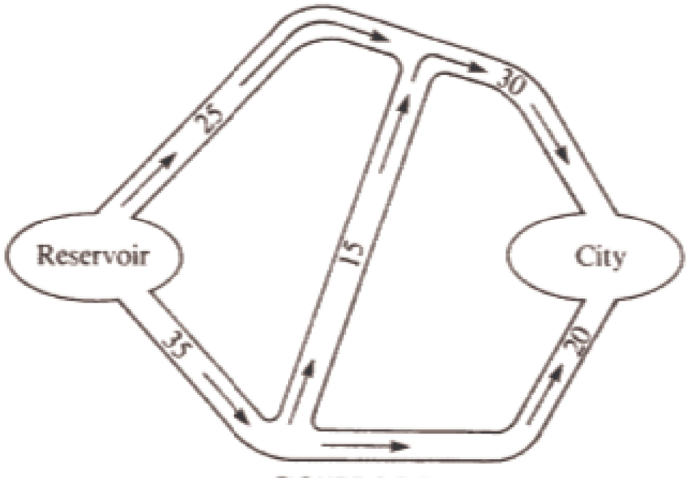


FIGURE 9.5.5
Diagram of a city's water system

Although the material passing through this network is water, networks can also represent the flow of other materials, such as automobiles, electricity, telephone calls or patients in a health system.

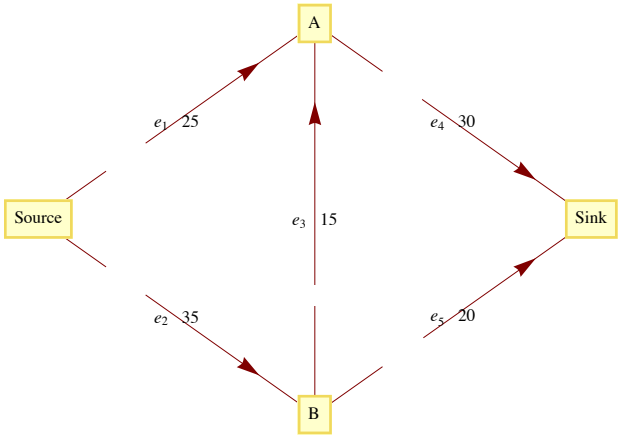


FIGURE 9.5.6
Flow diagram for a city's water system

The Maximum Flow Problem is derived from the objective of moving the maximum amount of water or other material from the source to the sink. To measure this amount, we define a flow as a function $f : E \to \mathbb{R}$ such that (1) the flow of material through any edge is nonnegative and no larger than its capacity: $0 \le f(e) \le w(e)$, for all $e \in E$; and (2) for each vertex other than the source and sink, the total amount of material that is directed into a vertex is equal to the total amount that is directed out:

$$\begin{aligned} \sum_{(x,v)\in E} f(x, v) &= \sum_{(v,y)\in E} f(v, y) \\ \text{Flow into } v &= \text{Flow out of } v \end{aligned} \qquad (9.5b)$$

The summation notation on the left of 9.5b represents the sum of the flows through each edge in E that has v as a terminal vertex. The right-hand side indicates that you should add all of the flows through edges that initiate at v.

Theorem 9.5.2. *If f is a flow, then*

$$\sum_{(source,v)\in E} f(source, v) = \sum_{(v,sink)\in E} f(v, sink)$$

This common value is called the value of the flow. We will denote the value of a flow by $V(f)$. The value of a flow represents the amount of material that passes through the network with that flow.

Proof. Subtract the right-hand side of 9.5b from the left-hand side. The result is:

$$\text{Flow into } v - \text{Flow out of } v = 0$$

Now sum up these differences for each vertex in $V' = V - \{\text{source}, \text{sink}\}$. The result is

$$\sum_{v\in V'} \left(\sum_{(x,v)\in E} f(x, v) - \sum_{(v,y)\in E} f(v, y) \right) = 0 \qquad (9.5c)$$

Now observe that if an edge connects two vertices in V, its flow appears as both a positive and a negative term in 9.5c. This means that the only positive terms that are not cancelled out are the flows into the sink. In addition, the only negative terms that remain are the flows out of the source. Therefore,

$$\sum_{(v,\text{sink})\in E} f(v, \text{sink}) - \sum_{(\text{source},v)\in E} f(\text{source}, v) = 0 \qquad \blacksquare$$

MAXIMAL FLOWS

Since the Maximum Flow Problem consists of maximizing the amount of material that passes through a given network, it is equivalent to finding a flow with the largest possible value. Any such flow is called a maximal flow.

For the network in Figure 9.5.6, one flow is f_1, defined by $f_1(e_1) = 25$, $f_1(e_2) = 20$, $f_1(e_3) = 0$, $f_1(e_4) = 25$, and $f_1(e_5) = 20$. The value of f_1, $V(f_1)$, is 45. Since the total flow into the sink can be no larger than 50 ($w(e_4) + w(e_5) = 30 + 20$), we can tell that f_1 is not very far from the solution. Can you improve on f_1 at all? The sum of the capacities into the sink can't always be obtained by a flow. The same is true for the sum of the capacities out of the source. In this case, the sum of the capacities out of the source is 60, which obviously can't be reached in this network.

A solution of the Maximum Flow Problem for this network is the maximal flow f_2, where $f_2(e_1) = 25$, $f_2(e_2) = 25$, $f_2(e_3) = 5$, $f_2(e_4) = 30$, and $f_2(e_5) = 20$, with $V(f_2) = 50$. This solution is not unique. In fact, there is an infinite number of maximal flows for this problem.

There have been several algorithms developed to solve the Maximal Flow Problem. One of these is the Ford and Fulkerson Algorithm (FFA).

The FFA consists of repeatedly finding paths in a network called flow augmenting paths until no improvement can be made in the flow that has been obtained.

Definition: Flow Augmenting Path. *Given a flow f in a network* (V, E)*, a flow augmenting path with respect to f is a simple path from the source to the sink using edges both in their forward and their reverse directions such that for each edge e in the path,* $w(e) - f(e) > 0$ *if e is used in its forward direction and* $f(e) > 0$ *if e is used in the reverse direction.*

Example 9.5.7. For f_1 in Example 9.5.6, a flow augmenting path would be (e_2, e_3, e_4) since

$$w(e_2) - f_1(e_2) = 15,\ w(e_3) - f_1(e_3) = 5, \text{ and } w(e_4) - f_1(e_4) = 5.$$

These positive differences represent unused capacities, and the smallest value represents the amount of flow that can be added to each edge in the path. Note that by adding 5 to each edge in our path, we obtain f_2, which is maximal. If an edge with a positive flow is used in its reverse direction, it is contributing a movement of material that is counterproductive to the objective of maximizing flow. This is why the algorithm directs us to decrease the flow through that edge.

Algorithm 9.5.3: The Ford and Fulkerson Algorithm.

(1) Define the flow function f_0 *by* $f_0(e) = 0$ *for each edge* $e \in E$.

(2) $i = 0$.

(3) Repeat:

(3.1) If possible, find a flow augmenting path with respect to f_i.

(3.2) If a flow augmenting path exists, then:

(3.2.1) Determine

$$d = min\{\{w(e) - f_i(e) \mid e \text{ is used in the forward direction}\},\ \{f_i(e) \mid e \text{ is used in the reverse direction}\}\}$$

(3.2.2) Define f_{i+1} *by*

$$f_{i+1}(e) = f_i(e) \quad \text{if } e \text{ is not part of the flow augmenting path}$$
$$f_{i+1}(e) = f_i(e) + d \quad \text{if } e \text{ is used in the forward direction}$$
$$f_{i+1}(e) = f_i(e) - d \quad \text{if } e \text{ is used in the reverse direction}$$

(3.2.3) $i = i + 1$.

until no flow augmenting path exists.

(4) Terminate with a maximal flow f_i

Notes:

(a) It should be clear that every flow augmenting path leads to a flow of increased value and that none of the capacities of the network can be violated.

(b) The depth-first search should be used to find flow augmenting paths since it is far more efficient than the breadth-first search in this situation. The depth-first search differs from the broadcasting algorithm (a variation of the breadth-first search) in that you sequentially visit vertices until you reach a "dead end" and then backtrack.

(c) There have been networks discovered for which the FFA does not terminate in a finite number of steps. These examples all have irrational capacities. It has been proven that if all capacities are positive integers, the FFA terminates in a finite number of steps. See Ford and Fulkerson, Even, or Berge for details.

(d) When you use the FFA to solve the Maximum Flow Problem by hand it is convenient to label each edge of the network with the fraction $f_i(e)/w(e)$.

A Depth-First Search for the Sink Initiating at the Source. *Let E' be the set of directed edges that can be used in producing a flow augmenting path. Add to the network a vertex called start and the edge (start, source).*

(1) S = vertex set of the network.
(2) p = start.
*(3) p = source (*Move p along the edge (start, source) *)*
(4) ***While*** *p is not equal to start or sink* ***do****.*
  If *an edge in E' exists that takes you from p to another vertex in S*
    then *set p to be that next vertex and delete the edge from E'.*
    else *reassign p to be the vertex that p was reached from (i.e., backtrack).*
(5) ***If*** *p = start,*
  then *no flow augmenting path exists.*
  else *p = sink, you have found a flow augmenting path.*

Example 9.5.8. Consider the network in Figure 9.5.7, where the current flow, f, is indicated by a labeling of the edges. The path $(Source, v_2, v_1, v_3, Sink)$ is a flow augmenting path that allows us to increase the flow by one unit. Note that (v_1, v_3) is used in the reverse direction, which is allowed because $f(v_1, v_3) > 0$. The value of the new flow that we obtain is 8. This flow must be maximal since the capacities

out of the source add up to 8. This maximal flow is defined by the labeling of Figure 9.5.8.

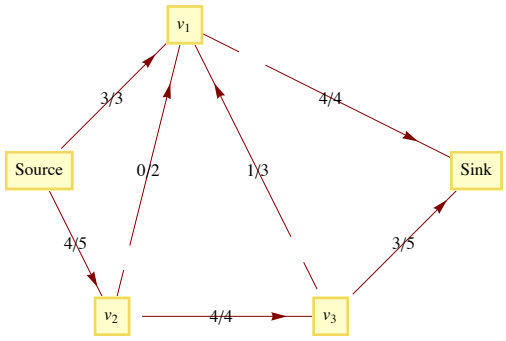


FIGURE 9.5.7
Current flow in Example 9.5.8

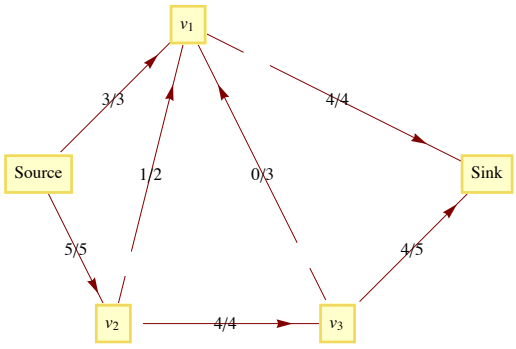


FIGURE 9.5.8
Augmented, optimal flow in Example 9.5.8

OTHER GRAPH-OPTIMIZATION PROBLEMS

(a) *The Minimum Spanning Tree Problem*: Given a weighted graph, (V, E, w), find a subset E' of E with the properties that (V, E') is connected and the sum of the weights of edges in E' are as small as possible. We will discuss this problem in Chapter 10.

(b) *The Minimum Matching Problem*: Given an undirected weighted graph, $(K E, w)$, with an even number of vertices, pair up the vertices so that each pair is connected by an edge and the sum of these edges is as small as possible. A unit square version of this problem has been studied extensively. See References: [Sopowit001] for details on what is known about this version of the problem.

(c) *The Graph Center Problem*: Given a connected, undirected, weighted graph, find a vertex (the center) in the graph with the property that the distance from the center to every other vertex is as small as possible. "As small as possible" could be interpreted either as minimizing the sum of the distances to each vertex or as minimizing the maximum distance from the center to a vertex.

EXERCISES FOR SECTION 9.5

A Exercises

1. Find the closest neighbor circuit through the six capitals of New England starting at Boston. If you start at a different city, will you get a different circuit?

2. Is Theorem 9.5.1 sharp for $n = 3$? For $n = 4$?

3. Given the following sets of points in the unit square, find the shortest circuit that visits all the points and find the circuit that is obtained with the strip algorithm.

(a) $\{(0.1\,k,\ 0.1\,k) : k = 0,\ 1,\ 2,\ \ldots,\ 10\}$

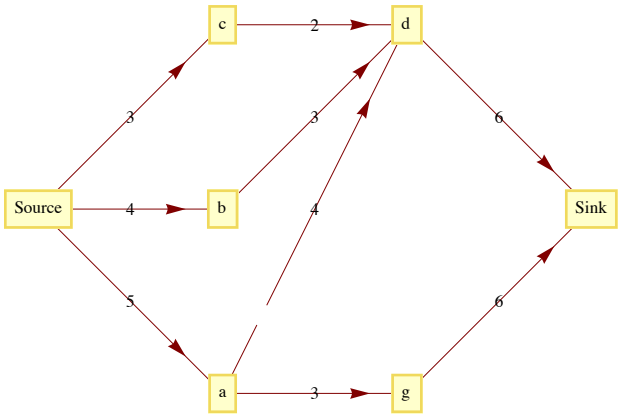
(b) $\{(0.1,\ 0.3),\ (0.3,\ 0.8),\ (0.5,\ 0.3),\ (0.7,\ 0.9),\ (0.9,\ 0.1)\}$

(c) $\{(0.0,\ 0.5),\ (0.5,\ 0.0),\ (0.5,\ 1.0),\ (1.0,\ 0.5)\}$

(d) $\{(0,\ 0),\ (0.2,\ 0.6),\ (0.4,\ 0.1),\ (0.6,\ 0.8),\ (0.7,\ 0.5)\}$

4. For $n = 4,\ 5,$ and 6, locate n points in the unit square for which the strip algorithm works poorly.

5. Consider the network whose maximum capacities are shown on the following graph.



(a) A function f is partially defined on the edges of this network by:

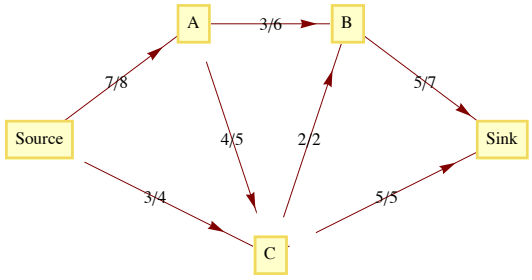
$f(\text{Source}, c) = f(\text{Source}, b) = f(\text{Source}, a) = 2$, and $f(a, d) = 1$.

Define f on the rest of the other edges so that f is a flow. What is the value of f?

(b) Find a flow augmenting path with respect to f for this network. What is the value of the augmented flow?

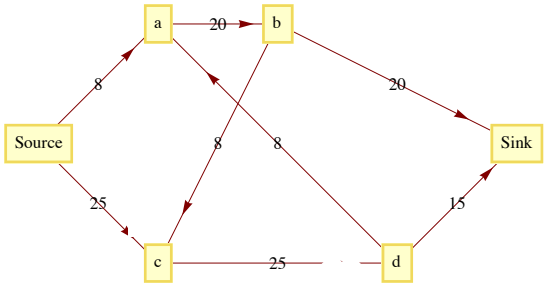
(c) Is the augmented flow a maximum flow? Explain.

6. Given the following network with capacity function c and flow function f, find a maximal flow function. The labels on the edges of the network are of the form $f(e)/c(e)$, where $c(e)$ is the capacity of edge e and $f(e)$ is the used capacity for flow f.

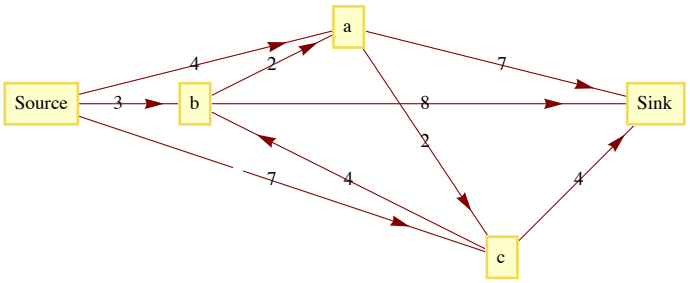


7. Find maximal flows for the following networks.

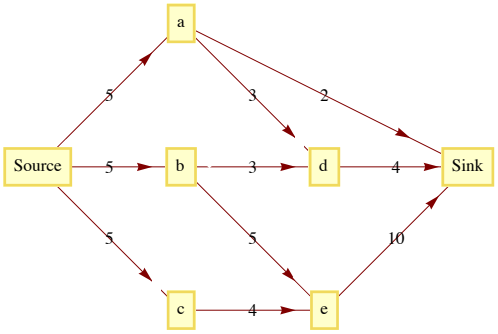
(a)



(b)



(c)



B Exercises

8. (a) [Easy] Find two maximal flows for the network in Figure 9.5.6 other than the one found in the text.

(b) [Harder] Describe the set of all maximal flows for the same network.

(c) [Hardest] Prove that if a network has two maximal flows, then it has an infinite number of maximal flows.

9. Discuss reasons that the closest neighbor algorithm is not used in the unit square version of the Traveling Salesman Problem. (Hint: Count the number of comparisons of distances that must be done.)

C Exercises

10. Explore the possibility of solving the Traveling Salesman Problem in the "unit box": $[0,\ 1]^3$.

11. Devise a "closest neighbor" algorithm for matching points in the unit square.

9.6 Planarity and Colorings

The topics in this section are related to how graphs are drawn.

Planarity: Can a given graph be drawn in a plane so that no edges intersect? Certainly, it is natural to avoid intersections, but up to now we haven't gone out of our way to do so.

Colorings: Suppose that each vertex in an undirected graph is to be colored so that no two vertices that are connected by an edge have the same color. How many colors are needed? This question is motivated by the problem of drawing a map so that no two bordering countries are colored the same. A similar question can be asked for coloring edges.

Definition: Planar Graph/ Plane Graph/Planar Embedding. *A graph is planar if it can be drawn in a plane so that no edges cross. If a graph is drawn so that no edges intersect, it is a plane graph, and such a drawing is a planar embedding of the graph.*

Example 9.6.1. The graph in Figure 9.6.1(a) is planar but not a plane graph. The same graph is drawn as a plane graph in Figure 9.6.1(b)

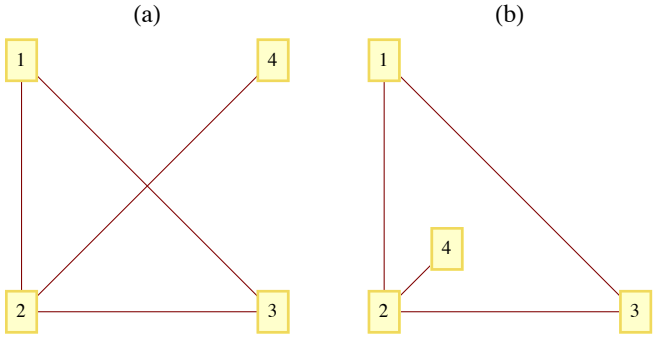


Figure 9.6.1
A planar graph and a planar embedding

Notes:

(a) In discussing planarity, we need only consider simple undirected graphs with no self-loops. All other graphs can be treated as such since all of the edges that relate any two vertices can be considered as one "package" that clearly can be drawn in a plane.

(b) Can you think of a graph that is not planar? How would you prove that it isn't planar? Proving the nonexistence of something is usually more difficult than proving its existence. This case is no exception. Intuitively, we would expect that sparse graphs would be planar and dense graphs would be nonplanar. Theorem 9.6.2 will verify that dense graphs are indeed nonplanar.

(c) The topic of planarity is a result of trying to restrict a graph to two dimensions. Is there an analogous topic for three dimensions? What graphs can be drawn in one dimension?

Answer to Note c: If a graph has only a finite number of vertices, it can always be drawn in three dimensions. This is not true for all graphs with an infinite number of vertices. The only "one-dimensional" graphs are the ones that consist of a finite number of chains, as in Figure 9.6.2, with one or more vertices in each chain.

Figure 9.6.2
Chains of length one, two and three

Example 9.6.2. A discussion of planarity is not complete without mentioning the famous Three Utilities Puzzle. The object of the puzzle is to supply three houses, A, B, and C, with the three utilities, gas, electric, and water. The constraint that makes this puzzle impossible to solve is that no utility lines may intersect i. e., a planar embedding of the graph in Figure 9.6.3, which is commonly denoted $K_{3,3}$. This graph is one of two fundamental nonplanar graphs. The Kuratowski Reduction Theorem states that if a graph is nonplanar then "contains" either a $K_{3,3}$ or a K_5. Containment is in the sense that if you start with a nonplanar graph you can always perform a sequence of edge deletions and contractions (shrinking an edge so that the two vertices connecting it coincide) to produce one of the two graphs.

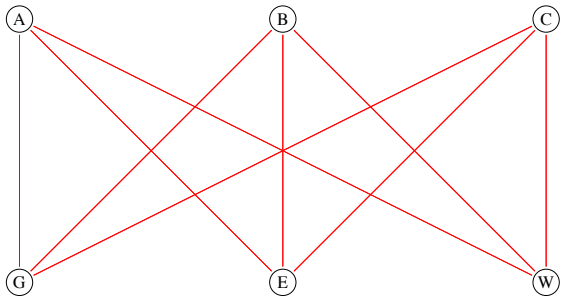


Figure 9.6.3
The Three Utilities Puzzle.

A planar graph divides the plane into one or more regions. Two points on the plane lie in the same region if you can draw a curve connecting the two points that does not pass through an edge. One of these regions will be of infinite area. Each point on the plane is either a vertex, a point on an edge, or a point in a region. A remarkable fact about the geography of planar graphs is the following theorem that is attributed to Euler.

Theorem 9.6.1: Euler's Formula. *If $G = (V, E)$ is a connected planar graph with r regions, v vertices and e edges, then*

$$v + r - e = 2 \qquad (9.6a)$$

Experiment: Jot down a graph right now and count the number of vertices, regions, and edges that you have. If $v + r - e$ is not 2, then your graph is either nonplanar or not connected.

Proof: We prove Euler's Formula by Induction on e, for $e \geq 0$.

Basis: If $e = 0$, then G must be a graph with one vertex, $v = 1$; and there is one infinite region, $r = 1$.

Therefore, $v + r - e = 1 + 1 - 0 = 2$, and the basis is true.

Induction: Suppose that G has k edges, $k \geq 1$, and that all connected planar graphs with less than k edges satisfy 9.6a. Select any edge that is part of the boundary of the infinite region and call it e_1. Let G' be the graph obtained from G by deleting e_1. Figure 9.6.4 illustrates the two different possibilities we need to consider: either G' is connected or it has two connected components, G_1 and G_2.

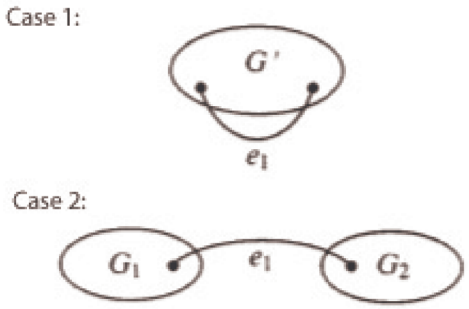


Figure 9.6.4
Two case in the proof of Euler's Formula

If G' is connected, the induction hypothesis can be applied to it. If G' has v' vertices, r' edges and e' edges, then $v' + r' - e' = 2$ and in terms of the corresponding numbers for G,

$v' = v$ No vertices were removed to form G'
$r' = r - 1$ One region of G merged with the infinite region when e_1 is removed
$e' = k - 1$ We assumed that G had k edges.

For the case where G' is connected,

$$\begin{aligned} v + r - e &= v + r - k \\ &= v' + (r' + 1) - (e' + 1) \\ &= v' + r' - e' \\ &= 2 \end{aligned}$$

If G' is not connected, it must consist of two connected components, G_1 and G_2 since we started with a connected graph, G. We can apply the induction hypothesis to each of the two components to complete the proof. We leave it to the students to do this, with the reminder that in counting regions, G_1 and G_2 will share the same infinite region. ■

Theorem 9.6.2. *If $G = (V, E)$ is a connected planar graph with v vertices, $v \geq 3$, and e edges, then*

$$e \leq 3v - 6. \qquad (9.6b)$$

Remark: One implication of 9.6b is that the number of edges in a connected planar graph will never be larger than three times its number.of vertices (as long as it has at least three vertices). Since the maximum number of edges in a graph with v vertices is a quadratic function of v, as v increases, planar graphs are more and more sparse.

Outline of a Proof of Theorem 9.6.2.

(a) Let r be the number of regions in G. For each region, count the number of edges that comprise its border. The sum of these counts must be at least $3r$. Recall that we are working with simple graphs here, so a region made by two edges connecting the same two vertices is not possible.

(b) Based on (a), infer that the number of edges in G must be at least $\frac{3r}{2}$.

(c) $e \geq \frac{3r}{2} \quad \Rightarrow \quad r \leq \frac{2e}{3}$

(d) Substitute $\frac{2e}{3}$ for r in Euler's Formula to obtain an inequality that is equivalent to 9.6.b. ■

The following theorem will be useful as we turn to graph coloring.

Theorem 9.6.3. *If G is a connected planar graph, then it has a vertex with degree 5 or less.*

Proof (by contradiction): We can assume that G has at least seven vertices, for otherwise the degree of any vertex is at most 5. Suppose that G is a connected planar graph and each vertex has a degree of 6 or more. Then, since each edge contributes to the degree of two vertices, $e \geq \frac{6v}{2} = 3v$. However, Theorem 9.6.2 states that the $e \leq 3v - 6 < 3v$, which is a contradiction. ■

GRAPH COLORING

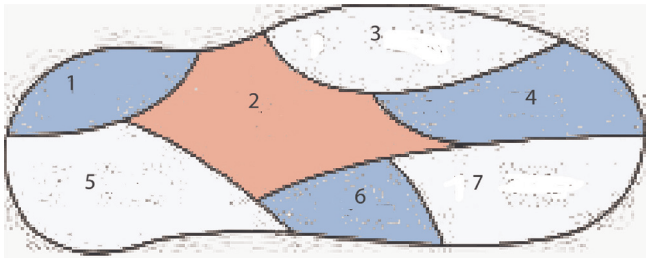


Figure 9.6.5
A 3-coloring of Euler Island

The map of Euler Island in Figure 9.6.5 shows that there are seven towns on the island. Suppose that a cartographer must produce a colored map in which no two towns that share a boundary have the same color. To keep costs down, she wants to minimize the number of different colors that appear on the map. How many colors are sufficient? For Euler Island, the answer is three. This problem motivates a more general problem.

The Graph Coloring Problem. Given an undirected graph $G = (V, E)$, find a "coloring function" f from V into a set of colors H such that $(v_i, v_j) \in E \Rightarrow f(v_i) \neq f(v_j)$ and H has the smallest possible cardinality. The cardinality of H is called the *chromatic number of G*, $\chi(G)$.

Notes:

(a) A coloring function onto an n element set is called an n-coloring.

(b) In terms of this general problem, the chromatic number of the graph of Euler Island is three. To see that no more than three colors are needed, we need only display a 3-coloring: $f(1) = f(4) = f(6) =$ blue, $f(2) =$ red, and $f(3) = f(5) = f(7) =$ white. This coloring is not unique. The next smallest set of colors would be of two colors, and you should be able to convince yourself that no 2-coloring exists for this graph.

In the mid-nineteenth century, it became clear that the typical planar graph had a chromatic number of no more than 4. At that point, mathematicians attacked the Four-Color Conjecture, which is that if G is any planar graph, then its chromatic number is no more than 4. Although the conjecture is quite easy to state, it took over 100 years, until 1976, to prove the conjecture in the affirmative.

Theorem 9.6.4: The Four-Color Theorem. *If G is a planar graph, then* $\chi(G) \leq 4$.

A proof of the Four-Color Theorem is beyond the scope of this text, but we can prove a theorem that is only 25 percent inferior.

Theorem 9.6.5: The Five-Color Theorem. *If G is a planar graph, then* $\chi(G) \leq 5$.

The number 5 is not a sharp upper bound for $\chi(G)$ because of the Four-Color Theorem.

Proof, by Induction on the Number of Vertices in the Graph:

Basis: Clearly, a graph with one vertex has a chromatic number of 1.

Induction: Assume that all planar graphs with $n - 1$ vertices have a chromatic number of 5 or less. Let G be a planar graph. By Theorem 9.6.2,

there exists a vertex v with $\deg v \le 5$. Let $G - v$ be the planar graph obtained by deleting v and all edges that connect v to other vertices in G. By the induction hypothesis, $G - v$ has a 5-coloring. Assume that the colors used are red, white, blue, green, and yellow.

If $\deg v < 5$, then we can produce a 5-coloring of G by selecting a color that is not used in coloring the vertices that are connected to v with an edge in G.

If $\deg v = 5$, then we can use the same approach if the five vertices that are adjacent to v are not all colored differently. We are now left with the possibility that v_1, v_2, v_3, v_4, and v_5 are all connected to v by an edge and they are all colored differently. Assume that they are colored red, white blue, yellow, and green, respectively, as in Figure 9.6.6.

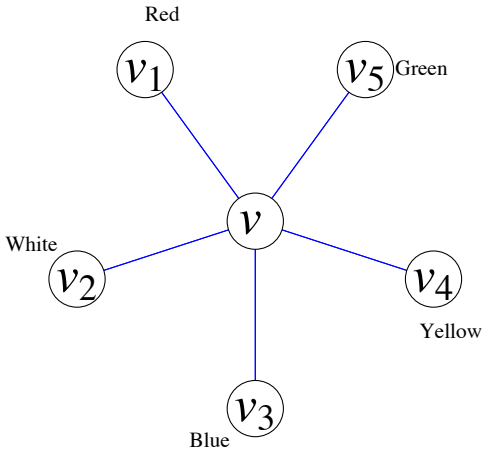


Figure 9.6.6

Starting at v_1 in $G - v$, suppose we try to construct a path v_3 that passes through only red and blue vertices. This can either be accomplished or it can't be accomplished. If it can't be done, consider all paths that start at v_1, and go through only red and blue vertices. If we exchange the colors of the vertices in these paths, including v_1 we still have a 5-coloring of $G - v$. Since v_1 is now blue, we can color the central vertex, v, red.

Finally, suppose that v_1 is connected to v_3 using only red and blue vertices. Then a path from v_1 to v_3 by using red and blue vertices followed by the edges (v_3, v) and (v, v_1) completes a circuit that either encloses v_2 or encloses v_4 and v_5 . Therefore, no path from v_2 to v_4 exists using only white and yellow vertices. We can then repeat the same process as in the previous paragraph with v_2 and v_4 , which will allow us to color v white. ■

Definition: Bipartite Graph. *A bipartite graph is a graph that has a 2-coloring. Equivalently, a graph is bipartite if its vertices can be partitioned into two nonempty subsets so that no edge connects a vertices from the same from each subset.*

Example 9.6.3.

(a) The graph of the Three Utilities Puzzle is bipartite. The vertices are partitioned into the utilities and the homes. Of course a 2-coloring of the graph is to color the utilities red and the homes blue.

(b) For $n \ge 1$, the n-cube is bipartite. A coloring would be to color all strings with an even number of 1's red and the strings with an odd number of 1's blue. By the definition of the n-cube, two strings that have the same color couldn't be connected since they would need to differ in at least two positions.

(c) Let V be a set of 64 vertices, one for each square on a chess board. We can index the elements of V by

v_{ij} = the square on the row i, column j.

Connect vertices in V according to whether or not you can move a knight from one square to another. Using our indexing of V,

$$\left(v_{ij},\ v_{kl}\right) \in E \text{ if and only if } \begin{array}{c} |i-k|+|j-l| = 3 \\ \text{and } |i-k| \cdot |j-l| = 2 \end{array}$$

$(V,\ E)$ is a bipartite graph. The usual coloring of a chessboard is valid 2-coloring.

How can you recognize whether a graph is bipartite? Unlike planarity, there is a nice equivalent condition for a graph to be bipartite.

Theorem 9.6.6. An undirected graph is bipartite if and only if it has no circuit of odd length.

Proof. ($\Rightarrow$) Let $G = (V,\ E)$ be a bipartite graph that is partitioned into two sets, *R(ed)* and *B(lue)* that define a 2-coloring. Consider any circuit in V. If we specify a direction in the circuit and define f on the vertices of the circuit by

$f(u)$ = the next vertex in the circuit after v.

Note that f is a bijection. Hence the number of red vertices in the circuit equals the number of blue vertices, and so the length of the circuit must be even.

($\Longleftarrow$) Assume that G has no circuit of odd length. For each component of G, select any vertex w and color it red. Then for every other vertex v in the component, find the path of shortest distance from w to v. If the length of the path is odd, color v blue, and if it is even, color v red. We claim that this method defines a 2-coloring of G. Suppose that it does not define a 2-coloring. Then let v_a and v_b be two vertices with identical colors that are connected with an edge. By the way that we colored G, neither v_a nor v_b could equal w. We can now construct a circuit with an odd length in G. First, we start at w and follow the shortest path to v_a . Then follow the edge (v_a, v_b), and finally, follow the reverse of a shortest path from w to v_b. Since v_a and v_b have the same color, the first and third segments of this circuit have lengths that are both odd or even, and the sum of their lengths must be even. The addition of the single edge (v_a, v_b) shows us that this circuit has an odd length. This contradicts our premise. ■

Note: An efficient algorithm for finding a 2-coloring of a graph can be designed using the method that is used in the second part of the proof above.

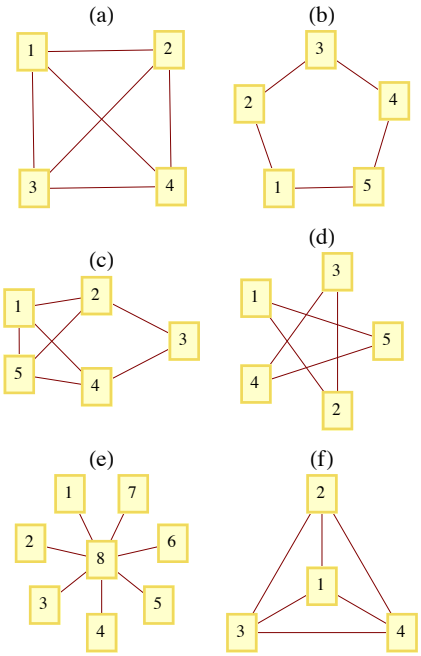
EXERCISES FOR SECTION 9.6

A Exercises

1. Apply Theorem 9.6.2 to prove that once n gets to a certain size, a K_n is nonplanar. What is the largest complete planar graph?

2. Can you apply Theorem 9.6.2 to prove that the Three Utilities Puzzle can't be solved?

3. What are the chromatic numbers of the following graphs?



4. Prove that if an undirected graph has a subgraph that is a K_3 it then its chromatic number is at least 3.

5. What is $\chi(K_n)$, $n \geq 1$?

6. What is the chromatic number of the United States?

B Exercises

7. Complete the proof of Theorem 9.6.1.

8. Use the outline of a proof of Theorem 9.6.2 to write a complete proof. Be sure to point out where the premise $v \geq 3$ is essential.

9. Let $G = (V, E)$ with $|V| \geq 11$, and let U be the set of all undirected edges between distinct vertices in V. Prove that either G or $G' = (V, E^c)$ is nonplanar.

10. Design an algorithm to determine whether a graph is bipartite.

11. Prove that a bipartite graph with an odd number of vertices greater than or equal to 3 has no Hamiltonian circuit.

C Exercises

12. Prove that any graph with a finite number of vertices can be drawn in three dimensions so that no edges intersect.

13. Suppose you had to color the edges of an undirected graph so that for each vertex, the edges that it is connected to have different colors. How can this problem be transformed into a vertex coloring problem?

14. (a) Suppose the edges of a K_6 are colored either red or blue. Prove that there will be either a "red K_3" (a subset of the vertex set with three vertices connected by red edges) or a "blue K_3."

(b) Suppose six people are selected at random. Prove that either there exists a subset of three of them with the property that any two people in the subset can communicate in a common language, or there exist three people, no two of whom can communicate in a common language.