# 6 Traveling Salesman Problems

The Mars Science Laboratory *Curiosity* is a six-wheeled rover that looks like a dune buggy on steroids and cost NASA $2.5 billion to build and launch. *Curiosity*, loaded with fancy cameras and all kinds of scientific instruments, left Cape Canaveral, Florida, on November 26, 2011 and landed on the Gale crater region of Mars on August 6, 2012. *Curiosity*'s mission is to explore an area around the Gale crater where planetary scientists believe there is a good chance of finding chemical and biological markers that might be evidence of past or present life. To put it simply, *Curiosity* is on the hunt for tiny Martians, dead or alive.

## 6.1 What Is a Traveling Salesman Problem?

The term *traveling salesman problem* (TSP) is catchy but a bit misleading, since most of the time the problems that fall under this heading have nothing to do with salespeople living out of a suitcase. The expression "traveling salesman" has traditionally been used as a convenient metaphor for many different real-life problems that share a common mathematical structure.

The three elements common to all TSPs (from now on we simply refer to any traveling salesman problem as a TSP) are the following:

- **A traveler.** The traveler could be a person (or a group), an object (a bus, a truck, an unmanned rover, etc.); it could even be a bee.
- **A set of sites.** These are the places or locations the traveler must visit. We will use $N$ to denote the number of sites.
- **A set of costs.** These are positive numbers associated with the expense of traveling from a site to another site. Here the "cost" variable is not restricted to just *monetary* cost—it can also represent *distance* traveled or *time* spent on travel.

A *solution* to a TSP is a "trip" that starts and ends at a site and visits all the other sites once (but only once). We call such a trip a **tour**. An *optimal solution* (**optimal tour**) is a tour of minimal *total cost*. (Notice that we used *a tour* rather than *the tour*—in general a TSP has more than one optimal solution.)

Let's now look at some examples that should help clarify the types of problems we call TSPs. We will not solve any of these TSPs in this section, but we will discuss solutions in later sections.

### EXAMPLE 6.1 A REAL TRAVELING SALESMAN'S TSP

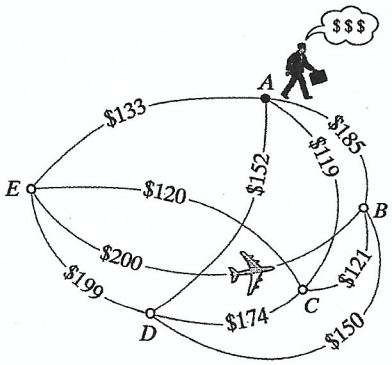



FIGURE 6-1 Cost of travel between cities.

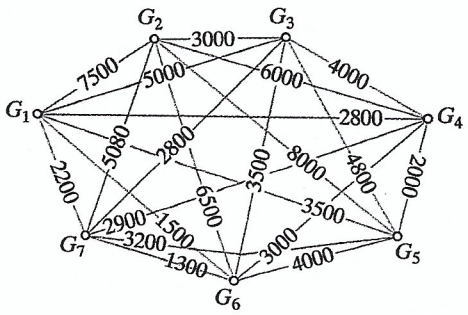
Willy "the Traveler" is a traveling salesman who spends a lot of time on the road calling on customers. He is planning his next business trip, where he will visit customers in five cities we will call $A$, $B$, $C$, $D$, and $E$ for short ($N = 5$). Since $A$ is Willy's hometown, he needs to start and end his trip at $A$.

The graph in Fig. 6-1 shows the *cost* (in dollars) of a one-way ticket between any pair of cities (for simplicity we are assuming the cost of a one-way ticket is the same regardless of the direction of travel—something that is not always true in the crazy world of modern airline ticket prices). Like most people, Willy hates to waste money, so among the many possible ways he can organize the sales tour Willy wants to find the *cheapest* (i.e., an *optimal tour*). We will see the solution to Willy's TSP in Section 6.3, but if you want to give it a try on your own now, please feel free to do so.

## EXAMPLE 6.3 THE *CURIOSITY* TSP

The story of the Mars Science Laboratory nicknamed *Curiosity* was introduced in the chapter opener. *Curiosity* is a six-wheeled $2.5-billion rover loaded with cameras and instruments that roams around a region of Mars known as the Gale crater, collecting rocks and doing chemical and biological assays of the soil. The primary mission objective is to look for evidence of the existence of life (past or present) on Martian soil. The mission is expected to last approximately one Martian year (23 months), and *Curiosity* is a slow traveler (about 660 feet a day), so efficient planning of its travels is a critical part of the mission. The specific details of where *Curiosity* is to roam were not available at the time of the writing of this example, so the details that follow are a fictional but realistic version of how the mission might evolve.

Figure 6-3(a) is a graph showing seven locations around the Gale crater (called $G_1$ through $G_7$) that *Curiosity* is going to visit, with $G_1$ being the landing site. The numbers on each edge represent travel distances in meters. [To make some of these distances meaningful, consider the fact that *Curiosity* can cover only about 200 m a day, so the trek from say $G_2$ to $G_5$ (8000 m) would take roughly 40 days.] Note that in the graph the edges are not drawn to scale—as in any graph, the positioning of the vertices is irrelevant and the only data that are relevant in this case are the numbers (distances) associated with each edge. In spite of a conscious effort to render the graph as clear and readable as possible, it's hard to get around the fact that there are a lot of numbers in the picture and things get pretty crowded. The distance chart in Fig. 6-3(b) provides exactly the same information as the graph in Fig. 6-3(a) but is a little easier to work with. [For TSPs with more than six sites ($N > 6$), a chart is generally preferable to a graph.]




(a)

| | $G_1$ | $G_2$ | $G_3$ | $G_4$ | $G_5$ | $G_6$ | $G_7$ |
|---|---|---|---|---|---|---|---|
| $G_1$ | * | 7500 | 5000 | 2800 | 3500 | 1500 | 2200 |
| $G_2$ | 7500 | * | 3000 | 6000 | 8000 | 6500 | 5000 |
| $G_3$ | 5000 | 3000 | * | 4000 | 4800 | 3500 | 2800 |
| $G_4$ | 2800 | 6000 | 4000 | * | 2000 | 3000 | 2900 |
| $G_5$ | 3500 | 8000 | 4800 | 2000 | * | 4000 | 3200 |
| $G_6$ | 1500 | 6500 | 3500 | 3000 | 4000 | * | 1300 |
| $G_7$ | 2200 | 5000 | 2800 | 2900 | 3200 | 1300 | * |

(b)

FIGURE 6-3 Distance between locations (in meters) for the *Curiosity* TSP.

- **School buses.** A school bus (the *traveler*) picks up children in the morning and drops them off at the end of the day at designated stops (the *sites*). On a typical rural school bus route there may be 20 to 30 such stops. With school buses, total time on the bus is always the most important variable (students have to get to school on time), and there is a known time of travel (the *cost*) between any two bus stops. Since children must be picked up at every bus stop, a *tour* of all the sites (starting and ending at the school) is required. Since the bus repeats its route every day during the school year, finding an *optimal tour* is crucial.
- **Circuit boards.** In the process of fabricating integrated-circuit boards, tens of thousands of tiny holes (the *sites*) must be drilled in each board. This is done by using a stationary laser beam and moving the board (the *traveler*). To do this efficiently, the order in which the holes are drilled should be such that the entire drilling sequence (the *tour*) is completed in the least amount of time (*optimal cost*). This makes for a very high-tech TSP.
- **Errands around town.** On a typical Saturday morning, an average Joe or Jane (the *traveler*) sets out to run a bunch of errands around town, visiting various sites (grocery store, hair salon, bakery, post office). When gas was cheap, *time* used to be the key *cost* variable, but with the cost of gas these days, people are more likely to be looking for the tour that minimizes the total *distance* traveled (see Exercise 59 for an illustration of this TSP).

# 6.2 Hamilton Paths and Circuits

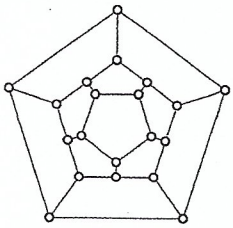


FIGURE 6-5 Hamilton's Icosian game.

In 1857, the Irish mathematician Sir William Rowan Hamilton invented a board game the purpose of which was to find a trip along the edges of the graph shown in Fig. 6-5 that visited each of the vertices once and only once, returning at the end to the starting vertex. (You may want to try your hand at it—the solution is shown on page 205.) Hamilton tried to market the game to make a little money, but he ended up just selling the rights to a London dealer for 25 pounds.

The only reason the story of Hamilton's Icosian game is of any interest to us is that it illustrates an important concept in this chapter—that of traveling along the edges of a connected graph with the purpose of visiting each and every one of the *vertices* once (but only once). This leads to the following two definitions:

- **Hamilton path.** A *Hamilton path* in a connected graph is a path that visits all the vertices of the graph once and only once.
- **Hamilton circuit.** A *Hamilton circuit* in a connected graph is a circuit that visits all the vertices of the graph once and only once.

In spite of the similarities in the definitions, Hamilton paths and circuits are very different from Euler paths and circuits and the two should not be confused. With Hamilton the name of the game is to visit all the *vertices* of the graph once; with Euler the name of the game is to pass through all the *edges* of the graph once.

## EXAMPLE 6.5 GRAPHS WITHOUT HAMILTON CIRCUITS

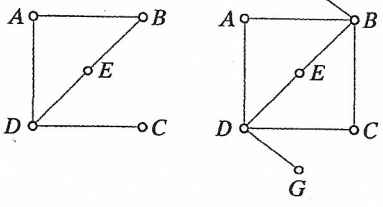


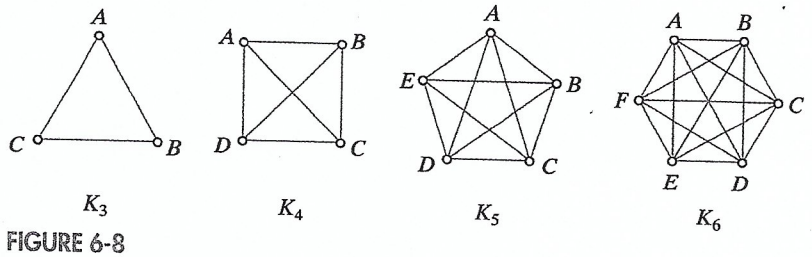

FIGURE 6-6

Figure 6.6 shows two graphs. The graph in Fig. 6.6(a) has no Hamilton circuits because once you visit $C$ you are stuck there. On the other hand, the graph has several Hamilton paths. One of them is $C, D, E, B, A$; another one is $C, D, A, B, E$. We can *reverse* those two paths and get two more: $A, B, E, D, C$ and $E, B, A, D, C$. Clearly, the Hamilton path has to start or end at $C$, so the four Hamilton paths listed above are the only ones possible. Note that the graph has two odd vertices, so from Euler's theorem for paths (page 153) we know that it has Euler paths as well.

The graph in Fig. 6.6(b) has no Hamilton circuits because once you visit $F$ (or $G$) you are stuck there. Nor does the graph have Hamilton paths: The path has to start at either $F$ or $G$ and end at the other one, and there is no way to visit the other five vertices without going through some vertices more than once. Note that this graph also has two odd vertices, so we know that it has an Euler path. ■

## Complete Graphs

A simple graph (i.e., no loops or multiple edges) in which the vertices are completely interconnected (every vertex is connected to every other vertex) is called a **complete graph**. Complete graphs are denoted by the symbol $K_N$, where $N$ is the number of vertices. Figure 6-8 shows the complete graphs for $N = 3, 4, 5$, and 6.




$K_3$ $K_4$ $K_5$ $K_6$

FIGURE 6-8

Listed below are four key properties of $K_N$:

1. The *degree* of every vertex in $K_N$ is $N - 1$.
2. The number of *edges* in $K_N$ is $\frac{N(N-1)}{2}$.
3. The number of *Hamilton paths* in $K_N$ is $N! = 1 \times 2 \times 3 \times \cdots \times N$.
4. The number of *Hamilton circuits* in $K_N$ is $(N-1)! = 1 \times 2 \times 3 \times \cdots \times (N-1)$.

Property (1) follows from the definition of $K_N$: since every vertex is adjacent to each of the other $N - 1$ vertices, the degree of each vertex is $N - 1$. Property (2) follows from property (1) and Euler's sum of degrees theorem (page 153): Since each vertex has degree $N - 1$, the sum of all the degrees is $N(N-1)$ and the number of edges is $\frac{N(N-1)}{2}$. Properties (3) and (4) require a little more detailed explanation, so we will start by exploring in some detail the Hamilton circuits and paths in $K_4$ and $K_5$.

## EXAMPLE 6.7 HAMILTON CIRCUITS AND PATHS IN $K_4$

The six Hamilton circuits in $K_4$ are shown in Fig. 6-9. Listed using $A$ as the starting and ending vertex they are: (a) $A, B, C, D, A$; (b) $A, D, B, C, A$; (c) $A, B, D, C, A$; and their respective reversals (d) $A, D, C, B, A$; (e) $A, C, B, D, A$; (f) $A, C, D, B, A$.

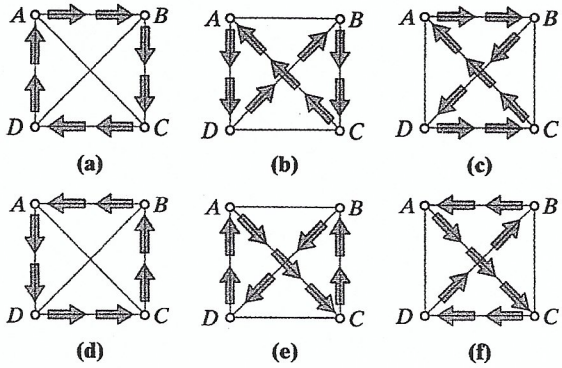



FIGURE 6-9 The six Hamilton circuits in $K_4$.

## EXAMPLE 6.8 HAMILTON CIRCUITS AND PATHS IN $K_5$

| | |
|---|---|
| (1) $A, B, C, D, E, A$ | (13) $A, E, D, C, B, A$ |
| (2) $A, B, C, E, D, A$ | (14) $A, D, E, C, B, A$ |
| (3) $A, B, D, C, E, A$ | (15) $A, E, C, D, B, A$ |
| (4) $A, B, D, E, C, A$ | (16) $A, C, E, D, B, A$ |
| (5) $A, B, E, C, D, A$ | (17) $A, D, C, E, B, A$ |
| (6) $A, B, E, D, C, A$ | (18) $A, C, D, E, B, A$ |
| (7) $A, C, B, D, E, A$ | (19) $A, E, D, B, C, A$ |
| (8) $A, C, B, E, D, A$ | (20) $A, D, E, B, C, A$ |
| (9) $A, C, D, B, E, A$ | (21) $A, E, B, D, C, A$ |
| (10) $A, C, E, B, D, A$ | (22) $A, D, B, E, C, A$ |
| (11) $A, D, B, C, E, A$ | (23) $A, E, C, B, D, A$ |
| (12) $A, D, C, B, E, A$ | (24) $A, E, B, C, D, A$ |

TABLE 6-1 The 24 Hamilton circuits in $K_5$

The $4! = 1 \times 2 \times 3 \times 4 = 24$ Hamilton circuits in $K_5$ are shown in Table 6-1, written using $A$ as the starting and ending vertex. Circuits (13) through (24) are the reversals of circuits (1) through (12), respectively. Any Hamilton circuit that starts and ends with $A$ will have the other four "inside" vertices ($B$, $C$, $D$, and $E$) listed in between in some order. There are $4 \times 3 \times 2 \times 1 = 24$ ways to list the letters $B$, $C$, $D$, and $E$ in order: 4 choices for the first letter, 3 choices for the second letter, 2 choices for the third letter, and a single choice for the last letter. This explains why Table 6-1 is a complete list of all the possible Hamilton circuits in $K_5$.

There are $24 \times 5 = 120$ Hamilton paths in $K_5$. For obvious reasons, we are not going to list them, but by now we should know how to generate them: Take any of the 24 Hamilton circuits and break it up by deleting one of its five edges. Figure 6-10 shows three of the 120 possible Hamilton paths: Fig. 6-10(a) is obtained by deleting edge $DA$ from circuit (2) on Table 6-1, and Fig. 6-10(b) is obtained by deleting edge $BC$ from circuit (11). We leave it as an exercise for the reader to figure out how the Hamilton path in Fig. 6-10(c) comes about.

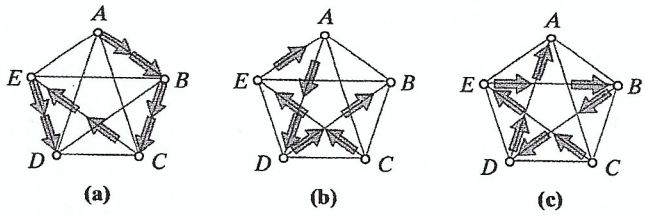



FIGURE 6-10 Three of the 120 possible Hamilton paths in $K_5$.

Another way to count the number of Hamilton paths in $K_5$ is to think in terms of *permutations*. (We first introduced permutations and their connection to factorials in Section 2.4. For a quick review of these concepts the reader is encouraged to revisit that section.) Each Hamilton path in $K_5$ is a permutation of the letters $A$, $B$, $C$, $D$, and $E$. It follows that the number of Hamilton paths equals the number of permutations of the five letters. This number is $5! = 120$.

When we generalize the preceding observations to a complete graph with $N$ vertices we get the following improved version of properties (3) and (4):

3. A *Hamilton path* in $K_N$ is equivalent to a *permutation of the N vertices*, and, therefore, the number of Hamilton paths is $N!$
4. A *Hamilton circuit* in $K_N$ is equivalent to a permutation of the $N-1$ "inside" vertices (i.e., all vertices except the starting/ending vertex), and, therefore, the number of Hamilton circuits is $(N-1)!$

# 6.3 The Brute-Force Algorithm

We start this section with the TSP introduced in Example 6.1.

**EXAMPLE 6.9** THE REAL TRAVELING SALESMAN'S TSP SOLVED

In Example 6.1 we left Willy the traveling salesman hanging. What Willy would like from us, most of all, is to help him find the optimal (in this case *cheapest*) tour of the five cities in his sales territory. The cost of travel between any two cities is shown in Fig. 6.11 (this is exactly the same figure as Fig. 6-1). Notice that the underlying graph in Fig. 6-11 is just a fancy version of $K_5$ (five vertices each adjacent to the other four) with numbers associated with each of the 10 edges. In this TSP the numbers represent the cost of travel (in either direction) along that edge.

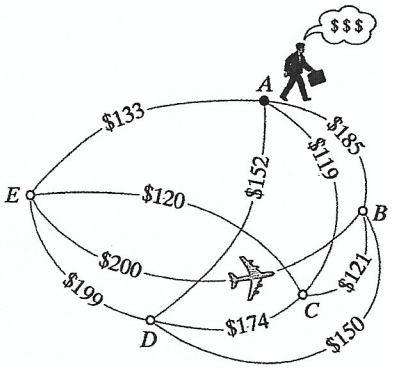



FIGURE 6-11

Table 6-4 lists the 24 possible tours that Willy could potentially take. The first and third columns of Table 6-4 gives the 24 different Hamilton circuits in $K_5$. This part is an exact copy of Table 6-1. Notice once again that the list is organized so that each tour in the third column is the reversed version of the corresponding tour in the first column. Listing the tours this way saves a lot of work because the cost of a tour is the same regardless of the direction of travel: Once we know the cost of a tour we know the cost of its reversal. (Please note that this is true only because we made the assumption that the cost of travel between two cities is the same regardless of the direction of travel. When the costs vary with the direction of travel then the shortcut won't work.) The middle column shows the total cost of each tour and the calculation that leads to it (for each tour just add the costs of the edges that make the tour).

| Tour | Total cost | Tour |
|---|---|---|
| (1) $A, B, C, D, E, A$ | $185 + 121 + 174 + 199 + 133 = 812$ | (13) $A, E, D, C, B, A$ |
| (2) $A, B, C, E, D, A$ | $185 + 121 + 120 + 199 + 152 = 777$ | (14) $A, D, E, C, B, A$ |
| (3) $A, B, D, C, E, A$ | $185 + 150 + 174 + 120 + 133 = 762$ | (15) $A, E, C, D, B, A$ |
| (4) $A, B, D, E, C, A$ | $185 + 150 + 199 + 120 + 119 = 773$ | (16) $A, C, E, D, B, A$ |
| (5) $A, B, E, C, D, A$ | $185 + 200 + 120 + 174 + 152 = 831$ | (17) $A, D, C, E, B, A$ |
| (6) $A, B, E, D, C, A$ | $185 + 200 + 199 + 174 + 119 = 877$ | (18) $A, C, D, E, B, A$ |
| (7) $A, C, B, D, E, A$ | $119 + 121 + 150 + 199 + 133 = 722$ | (19) $A, E, D, B, C, A$ |
| (8) $A, C, B, E, D, A$ | $119 + 121 + 200 + 199 + 152 = 791$ | (20) $A, D, E, B, C, A$ |
| (9) $A, C, D, B, E, A$ | $119 + 174 + 150 + 20[illegible] + 133 = 776$ | (21) $A, E, B, D, C, A$ |
| (10) $A, C, E, B, D, A$ | $119 + 120 + 200 + 150 + 152 = 741$ | (22) $A, D, B, E, C, A$ |
| (11) $A, D, B, C, E, A$ | $152 + 150 + 121 + 120 + 133 = 676$ | (23) $A, E, C, B, D, A$ |
| (12) $A, D, C, B, E, A$ | $152 + 174 + 121 + 200 + 133 = 780$ | (24) $A, E, B, C, D, A$ |

TABLE 6-4 The 24 possible tours in Example 6-9 and their costs, with the optimal tour(s), highlighted

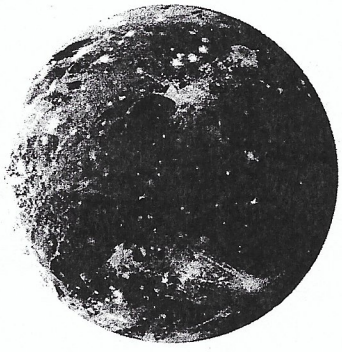
Before moving on to other TSP examples, let's connect a few dots. Our general assumption for a TSP is that there is a way to get from any location to any other location. If we think of the locations as the vertices of a graph, this means that every TSP has an underlying graph that is a complete graph $K_N$. In addition to the underlying graph, there are costs associated with each edge. A graph that has numbers associated with its edges is called a **weighted graph**, and the numbers are called **weights**. A tour in a TSP translates to a Hamilton circuit in the underlying graph, and an optimal tour translates into a Hamilton circuit of least total weight.

In short, the TSP is the concrete, real-life problem and its reformulation in terms of Hamilton circuits in a complete weighted graph is the mathematical model that represents the problem. In this model, *locations are vertices, costs are weights, tours are Hamilton circuits,* and *optimal tours are Hamilton circuits of least total weight.*

We can describe now the approach we used in Example 6.9 to solve Willy's TSP in a somewhat more formal language: We made a list of all possible Hamilton circuits (first and third columns of Table 6-4), calculated the total weight of each (middle column of Table 6-4), and picked the circuits with least total weight. This strategy is formally known as the **brute-force algorithm**.

**THE BRUTE-FORCE ALGORITHM**

- **Step 1.** Make a list of all the Hamilton circuits of the underlying graph $K_N$.
- **Step 2.** Calculate the total weight of each Hamilton circuit.
- **Step 3.** Choose a Hamilton circuit with least total weight.



**EXAMPLE 6.10** THE INTERPLANETARY MISSION TSP SOLVED

We are revisiting the TSP introduced in Example 6-2. In this TSP the goal is to find the *fastest* tour for an interplanetary mission to five of the outer moons in our solar system. Moreover, the tour has to start and end at our home planet, Earth. This makes it a TSP with $N = 6$ vertices. Figure 6-13(a) shows the mission time (in years) for travel between any two moons and between Earth and any moon.

To use the brute-force algorithm we would start with a list of all the possible Hamilton circuits using Earth as the starting and ending vertex. The problem is that this list has $5! = 120$ different Hamilton circuits. That's more work than we care to do, so a full list is out of the question. Imagine now that we get a hint: The first stop in an optimal tour is Callisto. How much help is that? A lot. It means that the optimal tour must be a Hamilton circuit of the form $E$, $C$, *, *, *, *, $E$. The *'s

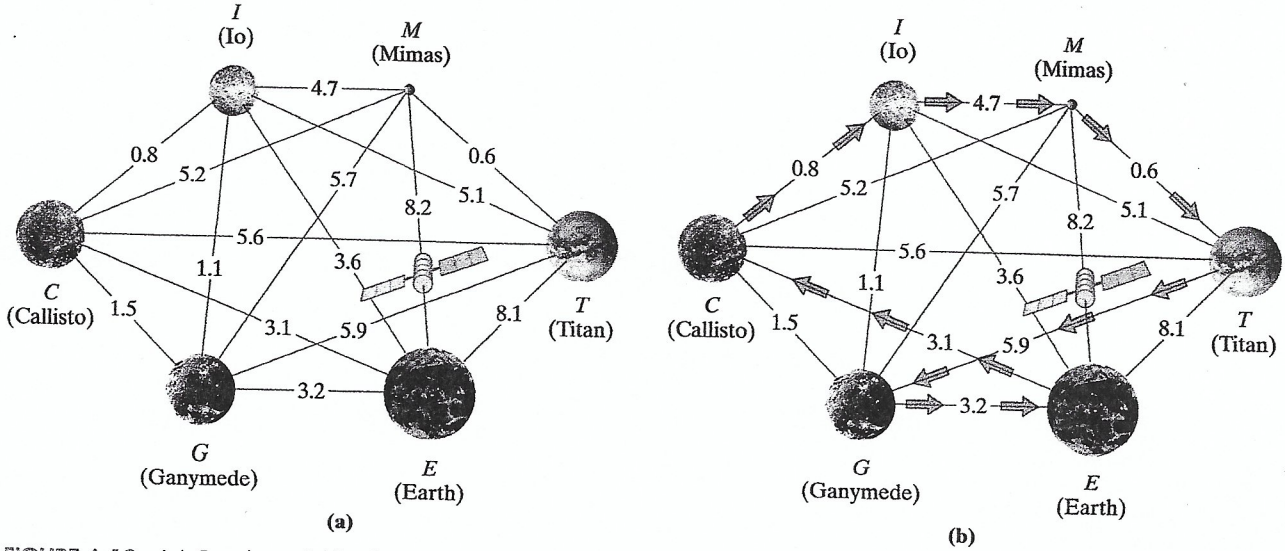
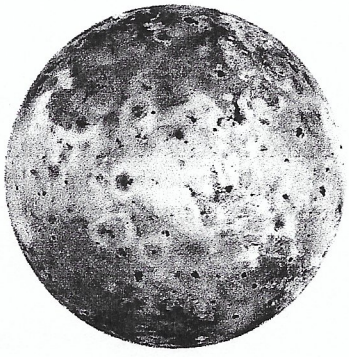



FIGURE 6-13 (a) Graph model for the interplanetary mission TSP. (b) An optimal tour.

# 6.4 The Nearest-Neighbor and Repetitive Nearest-Neighbor Algorithms

In this section we introduce a new method for solving TSPs called the *nearest-neighbor algorithm* (NNA). We will illustrate the basic idea of the nearest-neighbor algorithm using the interplanetary mission TSP once again. We found an optimal tour for this TSP in Example 6.10, so the point now is to see how the NNA does it.

**EXAMPLE 6.11** THE INTERPLANETARY MISSION TSP AND THE NEAREST-NEIGHBOR ALGORITHM



Look at the graph in Fig. 6-13(a) once again and imagine planning the mission. Starting from Earth we could choose any of the moons for our first stop. Of all the choices, Callisto makes the most sense because in terms of travel time it is Earth's "nearest neighbor": Among the edges connecting $E$ to the other vertices, $EC$ is the one with the smallest weight. (Technically speaking we should call $C$ the "smallest weight" neighbor, but that sounds a bit strange, so we use *nearest neighbor* as a generic term for the vertex connected by the edge of least weight.)

So we made it to Callisto. Where to next? Following the same logic, we choose to go to Callisto's nearest neighbor Io. From Io we go to Ganymede (the nearest neighbor we have not yet been to), from Ganymede to Mimas, from Mimas to Titan (the last moon left), and finally from Titan the mission comes back to Earth. This tour, called the *nearest-neighbor tour*, is shown in Fig. 6-14. The total length of this tour is 19.4 years, and that's a bit of bad news. In Example 6.10 we found an optimal tour with total length 18.3 years—this tour is more than a year longer.

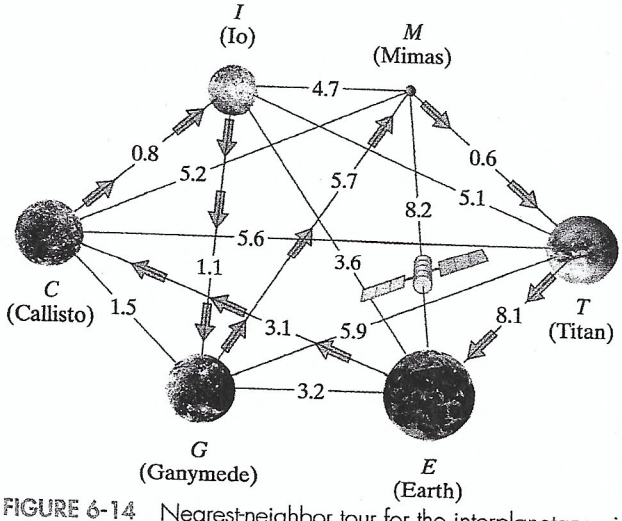



FIGURE 6-14 Nearest-neighbor tour for the interplanetary mission TSP.

Example 6.11 illustrates a common situation when trying to find solutions to TSPs. We can come up with very simple and fast methods for finding "good" tours, but these methods cut corners and don't necessarily produce the "best" (optimal) tour.

Formally, we will use the term **approximate algorithm** to describe any algorithm for solving TSPs that produces a tour, but not necessarily an optimal tour. The *nearest-neighbor* algorithm is an example of an *approximate* algorithm. We will discuss a different approximate algorithm called the *cheapest-link* algorithm in Section 6.5, and in Chapter 8 we will see approximate algorithms for solving a different type of problem (not a TSP).

## EXAMPLE 6.12 THE CONCERT TOUR TSP AND THE NNA

In Example 6.4 we were introduced to the indie rock band Luna Park, just about to embark on a 10-city concert tour. Figure 6-15(a) shows the distances (in miles) between any two cities. Because the band members all live at $A$, the tour needs to start and end at $A$. The goal is to find the optimal (shortest distance) tour.

This is a TSP with $N = 10$ locations. Notice that we don't see a complete weighted graph, but the mileage chart provides exactly the same information as the graph would (and it does it in a much cleaner way). The only way we know for finding an optimal tour is the brute-force algorithm, and to use brute force would mean making a list of $9! = 362{,}880$ possible Hamilton circuits. Obviously, we are not about to do that, at least not without help. But, we now know of a quick-and-dirty shortcut—the nearest-neighbor algorithm. Let's try it and see what we get.

To implement the nearest-neighbor algorithm from a chart we use the following strategy: Start with the row labeled $A$, and look for the smallest number in that row. That number (119) identifies $A$'s nearest neighbor, in this case $C$. We now go to row $C$ and look for $C$'s nearest neighbor. [Before we do that, it helps to cross out

| | A | B | C | D | E | F | G | H | J | K |
|---|---|---|---|---|---|---|---|---|---|---|
| A | * | 185 | 119 | 152 | 133 | 321 | 297 | 277 | 412 | 381 |
| B | 185 | * | 121 | 150 | 200 | 404 | 458 | 492 | 379 | 427 |
| C | 119 | 121 | * | 174 | 120 | 332 | 439 | 348 | 245 | 443 |
| D | 152 | 150 | 174 | * | 199 | 495 | 480 | 500 | 454 | 489 |
| E | 133 | 200 | 120 | 199 | * | 315 | 463 | 204 | 396 | 487 |
| F | 321 | 404 | 332 | 495 | 315 | * | 356 | 211 | 369 | 222 |
| G | 297 | 458 | 439 | 480 | 463 | 356 | * | 471 | 241 | 235 |
| H | 277 | 492 | 348 | 500 | 204 | 211 | 471 | * | 283 | 478 |
| J | 412 | 379 | 245 | 454 | 396 | 369 | 241 | 283 | * | 304 |
| K | 381 | 427 | 443 | 489 | 487 | 222 | 235 | 478 | 304 | * |

(a)

| | (A) | B | (C) | (D) | (E) | F | G | H | J | K |
|---|---|---|---|---|---|---|---|---|---|---|
| A | * | 185 | 119 | 152 | 133 | 321 | 297 | 277 | 412 | 381 |
| B | 185 | * | 121 | 150 | 200 | 404 | 458 | 492 | 379 | 427 |
| C | 119 | 121 | * | 174 | 120 | 332 | 439 | 348 | 245 | 443 |
| D | 152 | 150 | 174 | * | 199 | 495 | 480 | 500 | 454 | 489 |
| E | 133 | 200 | 120 | 199 | * | 315 | 463 | 204 | 396 | 487 |
| F | 321 | 404 | 332 | 495 | 315 | * | 356 | 211 | 369 | 222 |
| G | 297 | 458 | 439 | 480 | 463 | 356 | * | 471 | 241 | 235 |
| H | 277 | 492 | 348 | 500 | 204 | 211 | 471 | * | 283 | 478 |
| J | 412 | 379 | 245 | 454 | 396 | 369 | 241 | 283 | * | 304 |
| K | 381 | 427 | 443 | 489 | 487 | 222 | 235 | 478 | 304 | * |

(b)

**FIGURE 6-15** (a) Mileage chart for the concert tour TSP. (b) The chart after the first four steps of the NNA (*A, C, E, D, . . .*).

the column for $C$ (this helps make sure we don't go back to $C$ in the middle of the tour). For similar reasons we also crossed out the $A$-column.] The smallest number available in row $C$ is 120, and it identifies $C$'s nearest-neighbor $E$. We now cross out the $E$-column and go to row $E$, looking for a nearest-neighbor. The smallest number available in row $E$ is 199, and it identifies $E$'s nearest-neighbor $D$. We cross out the $D$-column and continue [Fig. 6-15(b) shows the mileage chart at this point]: From $D$ to its nearest-available neighbor $B$ (150), from $B$ to $J$ (379), from $J$ to $G$ (241), from $G$ to $K$ (235), from $K$ to $F$ (222), from $F$ to the only city left $H$ (211), and finally end the tour by returning to $A$ (277). This is the nearest-neighbor tour: $A, C, E, D, B, J, G, K, F, H, A$, and it has a total length of 2153 miles. ■

## The Repetitive Nearest-Neighbor Algorithm

One of the interesting features of the nearest-neighbor algorithm is that the tour it produces depends on the choice of starting vertex. For the same TSP, a change in the choice of starting vertex can produce a different nearest-neighbor tour. (Note that this does not imply that the tours are always different—we might change the starting vertex and still get the same tour.) This observation can help us squeeze better tours from the nearest-neighbor algorithm: Each time we try a different starting vertex there is the chance we might get an improved tour; the more vertices we try, the better our chances. This is the key idea behind a refinement of the nearest-neighbor algorithm known as the *repetitive nearest-neighbor* algorithm. Our next example illustrates how this strategy works.

### EXAMPLE 6.13 THE CONCERT TOUR TSP REVISITED

In Example 6.12 we used the nearest-neighbor algorithm to find a concert tour for the Luna Park rock band (this is the TSP introduced in Example 6.4). We used $A$ as the starting and ending vertex because the band lives at $A$, so in some sense we had no choice. The tour we found had a total length of 2153 miles.

We are going to try the same thing again but now will use $B$ as the starting and ending vertex. (Let's disregard for now the fact that the tour really needs to start and end at $A$. We'll deal with that issue later.) From $B$ we go to its nearest-neighbor $C$, from C to its nearest-neighbor $A$, from $A$ to $E$, and so on. The work is a little tedious, but we can finish the tour in a matter of minutes. We leave the details to the enterprising reader, but the bottom line is that we end up with the tour $B, C, A, E, D, J, G, K, F, H, B$ with a total length of 2427 miles.

We will now repeat this process, using each of the other vertices as the starting/ending vertex and finding the corresponding nearest-neighbor tour. Figuring a couple of minutes per vertex, the process might take about 20 minutes. Table 6-5 summarizes the results. The first column shows the 10 nearest-neighbor tours obtained by running over all possible starting/ending vertices; the second column shows the total length of each tour. We are looking for the best among these. Tours (5) through (10) are all tied for best, each with a total length of 2033 miles. We can take any one of these tours, and get a nice improvement over the original tour we found in Example 6.12.

| Nearest-Neighbor tour | Total length |
|---|---|
| (1) $A, C, E, D, B, J, G, K, F, H, A$ | 2153 |
| (2) $B, C, A, E, D, J, G, K, F, H, B$ | 2427 |
| (3) $C, A, E, D, B, J, G, K, F, H, C$ | 2237 |
| (4) $D, B, C, A, E, H, F, K, G, J, D$ | 2090 |
| (5) $E, C, A, D, B, J, G, K, F, H, E$ | 2033 |
| (6) $F, H, E, C, A, D, B, J, G, K, F$ | 2033 |
| (7) $G, K, F, H, E, C, A, D, B, J, G$ | 2033 |
| (8) $H, E, C, A, D, B, J, G, K, F, H$ | 2033 |
| (9) $J, G, K, F, H, E, C, A, D, B, J$ | 2033 |
| (10) $K, F, H, E, C, A, D, B, J, G, K$ | 2033 |

■ TABLE 6-5 Nearest-neighbor tours for every possible starting vertex

### THE NEAREST-NEIGHBOR ALGORITHM

- **Start:** Start at the designated starting vertex. If there is no designated starting vertex pick any vertex.
- **First step:** From the starting vertex go to its *nearest neighbor* (i.e., the vertex for which the corresponding edge has the smallest weight).
- **Middle steps:** From each vertex go to its *nearest neighbor*, choosing only among *the vertices that haven't been yet visited.* (If there is more than one nearest neighbor choose among them at random.) Keep doing this until all the vertices have been visited.
- **Last step:** From the last vertex return to the starting vertex. The tour that we get is called the **nearest-neighbor** tour.

### THE REPETITIVE NEAREST-NEIGHBOR ALGORITHM

- Let $X$ be any vertex. Find the nearest-neighbor tour with $X$ as the starting vertex, and calculate the cost of this tour.
- Repeat the process with each of the other vertices of the graph as the starting vertex.
- Of the nearest-neighbor tours thus obtained, choose one with least cost. If necessary, rewrite the tour so that it starts at the designated starting vertex. The tour that we get is called the **repetitive nearest-neighbor** tour.

# 6.5 The Cheapest-Link Algorithm

**EXAMPLE 6.14** THE INTERPLANETARY MISSION TSP AND THE CLA

The graph in Fig. 6-16(a) is a repeat of Fig. 6-2. The weights of the edges represent the cost (in this case time) of travel between any two locations. We will now show how the cheapest-link algorithm handles this TSP.

- Step 1. We start by scanning the graph in Fig. 6-16(a) looking for the *cheapest link* (*edge*). Here the cheapest link is the one connecting Mimas and Titan (0.6 years). [For convenience we'll denote it by *MT* (0.6).] We select that link and indicate that we are doing so by highlighting it in red.
- Step 2. We go back to scanning the graph looking for the *next cheapest link*. We find *CI* (0.8) and proceed to highlight it in red. (Notice that the two red links are not connected, but it doesn't matter at this point.)

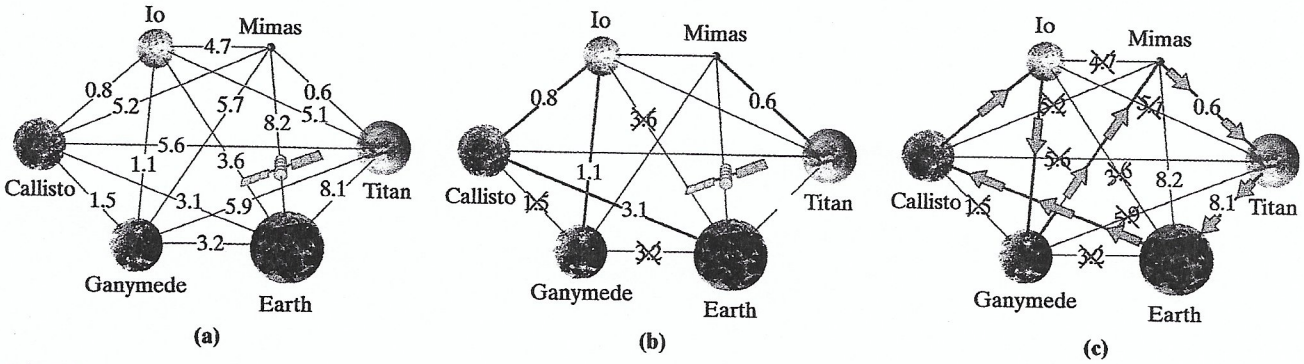



FIGURE 6-16 (a) The interplanetary mission TSP. (b) Partially constructed cheapest-link tour. (c) Cheapest-link tour after completion.

- Step 3. Once again, we scan the graph looking for the next cheapest link and find *IG* (1.1). We highlight it in red.
- Step 4. The next cheapest link in the graph is *CG* (1.5). We pick up our red pen and are about to highlight the link when we suddenly realize that this is not a good move. A red *CG* means that we would be forming a red circuit *C*, *I*, *G*, *C*. But tours (Hamilton circuits) can't contain partial circuits, so any edge that forms a partial circuit must be ruled out. (For convenience, we call this the *partial-circuit rule*.) So, even though *CG* is the next available cheapest link, we can't choose it because of the partial-circuit rule. We indicate this fact by X-ing out *CG* (the X is like a little marker saying "do not travel along this link"). So, we try again. After *CG*, the next cheapest link is *CE* (3.1). No problem here, so we select it and highlight it in red.
- Step 5. The next cheapest link in the graph is *EG* (3.2). If we were to highlight *GE* in red we would be forming the partial red circuit *E*, *G*, *I*, *C*, *E*. Because of the partial-circuit rule, we must X-out *EG*. We scan the graph again and find that the next cheapest link is *IE*. If we were to highlight *IE* in red we would have the following problem: *three* red edges (*IE*, *IG*, and *IC*) meeting at one vertex. This is not possible in a tour, since it would require visiting that vertex more than once. (For convenience, we call this the *three-edge rule*.) So we X-out *IE* as well. [If you are doodling along with this narration, your picture at this point should look something like Fig. 6-16(b).] The next four cheapest links in order are *IM* (4.7), *IT* (5.1), *CM* (5.2), and *CT* (5.6). They all have to be ruled out because of the three-edge rule. This leads us to *GM* (5.7). This edge works, so we select it and highlight it in red.
- Step 6. Since $N = 6$ this should be the last step. The last step is a little easier than the others—there should be only one way to close the circuit. Looking at the graph we see that *E* and *T* are the two loose ends that need to be connected. We do that by adding the link *ET*. Now we are done.

### THE CHEAPEST-LINK ALGORITHM

- **Step 1.** Pick the *cheapest link* available. (If there is more than one, randomly pick one among the cheapest links.) Highlight the link in red (or any other color).
- **Step 2.** Pick the next cheapest link available and highlight it.
- **Steps 3, 4, . . ., $N-1$.** Continue picking and highlighting the cheapest available link that (a) does not violate the *partial-circuit* rule (i.e., does not close a partial circuit) or (b) does not violate the *three-edge* rule (i.e., does not create three edges meeting at the same vertex).
- **Step $N$.** Connect the two vertices that close the red circuit. Once we have the Hamilton circuit, we can add a direction of travel (clockwise or counterclockwise). Either one gives us a **cheapest-link tour**.

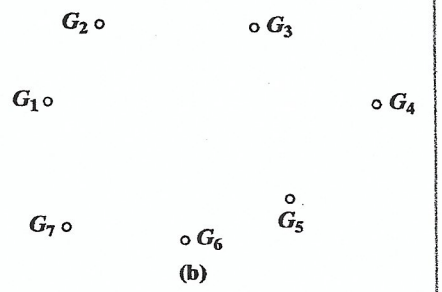
Our next example illustrates how to implement the cheapest-link algorithm when we have to work from a chart rather than a graph. The main difficulty in this situation is that it is not easy to spot violations of the *partial-circuit* and *three-edge* rules when looking at a chart. A simple way around this difficulty is to create an auxiliary picture of the tour as it is being built, one edge at a time.

### EXAMPLE 6.15 THE *CURIOSITY* TSP AND THE CLA

We are finally going to take a look at the *Curiosity* TSP introduced in the chapter opener and described in Example 6.3. The seven locations ($G_1$ through $G_7$) that the Mars rover *Curiosity* must visit and the distances (in meters) between pairs of locations are given in Fig. 6-17(a). We will use the cheapest-link algorithm working directly out of the distance chart. All we will need is an additional auxiliary graph that will help us visualize the links as we move through the steps of the algorithm. Figure 6-17(b) shows the auxiliary graph when we start—a blank slate of seven vertices $G_1$ through $G_7$ and no edges.

| | $G_1$ | $G_2$ | $G_3$ | $G_4$ | $G_5$ | $G_6$ | $G_7$ |
|---|---|---|---|---|---|---|---|
| $G_1$ | * | 7500 | 5000 | 2800 | 3500 | 1500 | 2200 |
| $G_2$ | 7500 | * | 3000 | 6000 | 8000 | 6500 | 5000 |
| $G_3$ | 5000 | 3000 | * | 4000 | 4800 | 3500 | 2800 |
| $G_4$ | 2800 | 6000 | 4000 | * | 2000 | 3000 | 2900 |
| $G_5$ | 3500 | 8000 | 4800 | 2000 | * | 4000 | 3200 |
| $G_6$ | 1500 | 6500 | 3500 | 3000 | 4000 | * | 1300 |
| $G_7$ | 2200 | 5000 | 2800 | 2900 | 3200 | 1300 | * |

(a)



(b)

## EXAMPLE 6.16 THE *CURIOSITY* TSP AND THE NNA

We know only one way to find the optimal tour for the *Curiosity* TSP—use the brute-force algorithm, but this would require us to create a list with 6! = 720 Hamilton circuits. That's a lot of circuits to check out by hand.

| | $G_1$ | $G_2$ | $G_3$ | $G_4$ | $G_5$ | $G_6$ | $G_7$ |
|---|---|---|---|---|---|---|---|
| $G_1$ | * | 7500 | 5000 | 2800 | 3500 | 1500 | 2200 |
| $G_2$ | 7500 | * | 3000 | 6000 | 8000 | 6500 | 5000 |
| $G_3$ | 5000 | 3000 | * | 4000 | 4800 | 3500 | 2800 |
| $G_4$ | 2800 | 6000 | 4000 | * | 2000 | 3000 | 2900 |
| $G_5$ | 3500 | 8000 | 4800 | 2000 | * | 4000 | 3200 |
| $G_6$ | 1500 | 6500 | 3500 | 3000 | 4000 | * | 1300 |
| $G_7$ | 2200 | 5000 | 2800 | 2900 | 3200 | 1300 | * |

**(a)**

FIGURE 6-18 (a) Distance chart for the *Curiosity* TSP;

# EXERCISES

## WALKING

### 6.1 What Is a Traveling Salesman Problem?

No exercises for this section.

### 6.2 Hamilton Paths and Circuits

**1.** For the graph shown in Fig. 6-19,

**(a)** find three different Hamilton circuits.

**(b)** find a Hamilton path that starts at $A$ and ends at $B$.

**(c)** find a Hamilton path that starts at $D$ and ends at $F$.

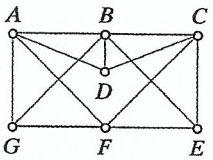



FIGURE 6-19

**2.** For the graph shown in Fig. 6-20,

**(a)** find three different Hamilton circuits.

**(b)** find a Hamilton path that starts at $A$ and ends at $B$.

**(c)** find a Hamilton path that starts at $F$ and ends at $I$.

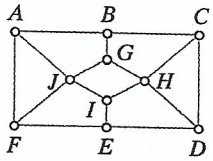



FIGURE 6-20

**3.** Find all possible Hamilton circuits in the graph in Fig. 6-21. Write your answers using $A$ as the starting/ending vertex.

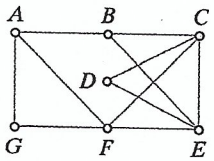



FIGURE 6-21

**4.** Find all possible Hamilton circuits in the graph in Fig. 6-22. Write your answers using $A$ as the starting/ending vertex.

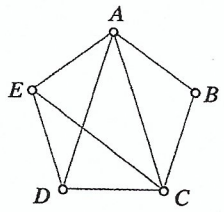



FIGURE 6-22

**5.** For the graph shown in Fig. 6-23,

**(a)** find a Hamilton path that starts at $A$ and ends at $E$.

**(b)** find a Hamilton circuit that starts at $A$ and ends with the edge $EA$.

**(c)** find a Hamilton path that starts at $A$ and ends at $C$.

**(d)** find a Hamilton path that starts at $F$ and ends at $G$.

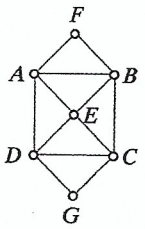



FIGURE 6-23

**6.** For the graph shown in Fig. 6-24,

**(a)** find a Hamilton path that starts at $A$ and ends at $E$.

**(b)** find a Hamilton circuit that starts at $A$ and ends with the edge $EA$.

**(c)** find a Hamilton path that starts at $A$ and ends at $G$.

**(d)** find a Hamilton path that starts at $F$ and ends at $G$.

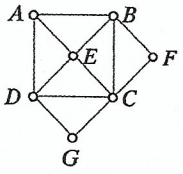



FIGURE 6-24

**7.** Suppose $D, G, E, A, H, C, B, F, D$ is a Hamilton circuit in a graph.

**(a)** Find the number of vertices in the graph.

**(b)** Write the Hamilton circuit using $A$ as the starting/ending vertex.

**(c)** Find two different Hamilton paths in the graph that start at $A$.

**8.** Suppose $G, B, D, C, A, F, E, G$ is a Hamilton circuit in a graph.

**(a)** Find the number of vertices in the graph.

**(b)** Write the Hamilton circuit using $F$ as the starting/ending vertex.

**(c)** Find two different Hamilton paths in the graph that start at $F$.

**9.** Consider the graph in Fig. 6-25.

**(a)** Find the five Hamilton paths that can be obtained by "breaking" the Hamilton circuit $B,A,D,E,C,B$ (i.e., by deleting just one edge from the circuit).

**(b)** Find the eight Hamilton paths that do not come from "broken" Hamilton circuits (i.e., cannot be closed into a Hamilton circuit). (*Hint*: See Example 6.6).

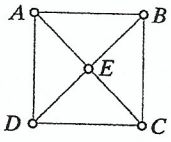



FIGURE 6-25

**10.** Consider the graph in Fig. 6-26.

**(a)** Find all the Hamilton circuits in the graph, using $B$ as the starting/ending vertex. (*Hint*: There are five Hamilton circuits and another five that are reversals of the first five.)

**(b)** Find the four Hamilton paths that start at $B$ and do not come from "broken" Hamilton circuits (i.e., cannot be closed into a Hamilton circuit).

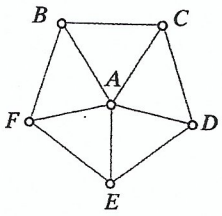



FIGURE 6-26

**11.** Consider the graph in Fig. 6-27.

**(a)** Find all the Hamilton circuits in the graph, using $A$ as the starting/ending vertex. You don't have to list both a circuit and its reversal—you can just list one from each pair.

**(b)** Find all the Hamilton paths that do not come from "broken" Hamilton circuits (i.e., cannot be closed into a Hamilton circuit). You don't have to list both a path and its reversal—you can just list one from each pair.

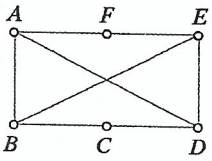



FIGURE 6-27

**12.** Consider the graph in Fig. 6-28.

**(a)** Find all the Hamilton circuits in the graph, using $A$ as the starting/ending vertex. You don't have to list both a circuit and its reversal—you can just list one from each pair.

**(b)** Find all the Hamilton paths that do not come from "broken" Hamilton circuits (i.e., cannot be closed into a Hamilton circuit). You don't have to list both a path and its reversal—you can just list one from each pair. (*Hint*: Such paths must either start or end at $C$. You can just list all the paths that start at $C$— the ones that end at $C$ are their reversals.)

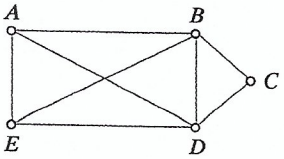



FIGURE 6-28

**13.** For the graph shown in Fig. 6-29,

**(a)** find a Hamilton path that starts at $A$ and ends at $F$.

**(b)** find a Hamilton path that starts at $K$ and ends at $E$.

**(c)** explain why the graph has no Hamilton path that starts at $C$.

**(d)** explain why the graph has no Hamilton circuits.

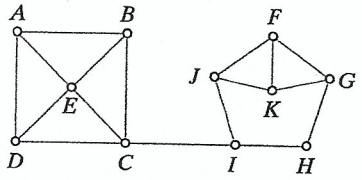



FIGURE 6-29

**14.** For the graph shown in Fig. 6-30,

**(a)** find a Hamilton path that starts at $B$.

**(b)** find a Hamilton path that starts at $E$.

**(c)** explain why the graph has no Hamilton path that starts at $A$ or at $C$.

**(d)** explain why the graph has no Hamilton circuit.

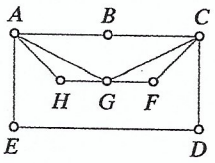



FIGURE 6-30

**15.** Explain why the graph shown in Fig. 6-31 has neither Hamilton circuits nor Hamilton paths.

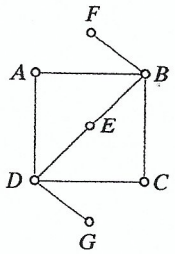



FIGURE 6-31

**16.** Explain why the graph shown in Fig. 6-32 has no Hamilton circuit but does have a Hamilton path.

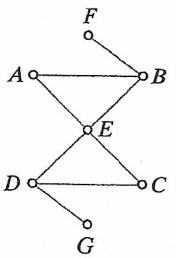



FIGURE 6-32

**17.** For the weighted graph shown in Fig. 6-33,

**(a)** find the weight of edge $BD$.

**(b)** find a Hamilton circuit that starts with edge $BD$, and give its weight.

**(c)** find a Hamilton circuit that ends with edge $DB$, and give its weight.

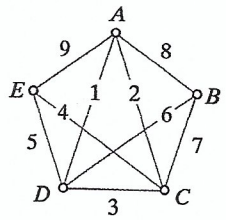



FIGURE 6-33

**18.** For the weighted graph shown in Fig. 6-34,

**(a)** find the weight of edge $AD$.

**(b)** find a Hamilton circuit that starts with edge $AD$, and give its weight.

**(c)** find a Hamilton circuit that ends with edge $DA$, and give its weight.

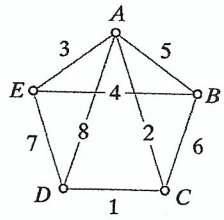



FIGURE 6-34

**19.** For the weighted graph shown in Fig. 6-35,

**(a)** find a Hamilton path that starts at $A$ and ends at $C$, and give its weight.

**(b)** find a second Hamilton path that starts at $A$ and ends at $C$, and give its weight.

**(c)** find the optimal (least weight) Hamilton path that starts at $A$ and ends at $C$, and give its weight.

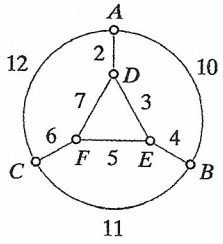



FIGURE 6-35

**20.** For the weighted graph shown in Fig. 6-36,

(a) find a Hamilton path that starts at $B$ and ends at $D$, and give its weight.

(b) find a second Hamilton path that starts at $B$ and ends at $D$, and give its weight.

(c) find the optimal (least weight) Hamilton path that starts at $B$ and ends at $D$, and give its weight.

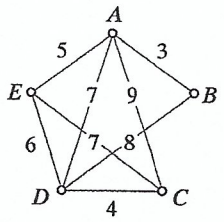



FIGURE 6-36

**21.** Suppose you have a supercomputer that can generate one *billion* Hamilton circuits per second.

(a) Estimate (in years) how long it would take the supercomputer to generate all the Hamilton circuits in $K_{21}$.

(b) Estimate (in years) how long it would take the supercomputer to generate all the Hamilton circuits in $K_{22}$.

**22.** Suppose you have a supercomputer that can generate one *trillion* Hamilton circuits per second.

(a) Estimate (in years) how long it would take the supercomputer to generate all the Hamilton circuits in $K_{26}$.

(b) Estimate (in years) how long it would take the supercomputer to generate all the Hamilton circuits in $K_{27}$.

**23.** (a) How many edges are there in $K_{20}$?

(b) How many edges are there in $K_{21}$?

(c) If the number of edges in $K_{50}$ is $x$ and the number of edges in $K_{51}$ is $y$, what is the value of $y - x$?

**24.** (a) How many edges are there in $K_{200}$?

(b) How many edges are there in $K_{201}$?

(c) If the number of edges in $K_{500}$ is $x$ and the number of edges in $K_{501}$ is $y$, what is the value of $y - x$?

**25.** In each case, find the value of $N$.

(a) $K_N$ has 120 distinct Hamilton circuits.

(b) $K_N$ has 45 edges.

(c) $K_N$ has 20,100 edges.

**26.** In each case, find the value of $N$.

(a) $K_N$ has 720 distinct Hamilton circuits.

(b) $K_N$ has 66 edges.

(c) $K_N$ has 80,200 edges.

## 6.3 The Brute-Force Algorithm

**27.** Find an optimal tour for the TSP given in Fig. 6-37, and give its cost.

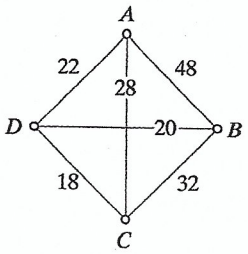



FIGURE 6-37

**28.** Find an optimal tour for the TSP given in Fig. 6-38, and give its cost.

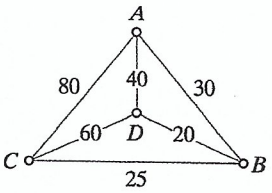



FIGURE 6-38

**29.** A truck must deliver furniture to stores located in five different cities $A$, $B$, $C$, $D$, and $E$. The truck must start and end its route at $A$. The time (in hours) for travel between the cities is given in Fig. 6-39. Find an optimal tour for this TSP and give its cost in hours. (*Hint*: The edge $AD$ is part of an optimal tour.)

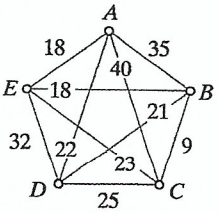



FIGURE 6-39

**30.** A social worker starts from her home $A$, must visit clients at $B$, $C$, $D$, and $E$ (in any order), and return home to $A$ at the end of the day. The graph in Fig. 6-40 shows the distance (in miles) between the five locations. Find an optimal tour for this TSP, and give its cost in miles. (*Hint*: The edge $AC$ is part of an optimal tour.)

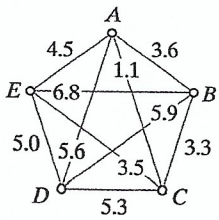



FIGURE 6-40

**31.** You are planning to visit four cities $A$, $B$, $C$, and $D$. Table 6-6 shows the time (in hours) that it takes to travel by car between any two cities. Find an optimal tour for this TSP that starts and ends at $B$.

| | A | B | C | D |
|---|---|---|---|---|
| A | * | 12 | 6 | 14 |
| B | 12 | * | 17 | 15 |
| C | 6 | 17 | * | 11 |
| D | 14 | 15 | 11 | * |

TABLE 6-6

**32.** An unmanned rover must be routed to visit four sites labeled $A$, $B$, $C$, and $D$ on the surface of the moon. Table 6-7 shows the distance (in kilometers) between any two sites. Assuming the rover landed at $C$, find an optimal tour.

| | A | B | C | D |
|---|---|---|---|---|
| A | 0 | 4 | 18 | 16 |
| B | 4 | 0 | 17 | 13 |
| C | 18 | 17 | 0 | 7 |
| D | 16 | 13 | 7 | 0 |

TABLE 6-7

**33.** Consider a TSP with nine vertices labeled $A$ through $I$.

**(a)** How many tours are of the form $A, G, \ldots, A$? (*Hint*: The remaining seven letters can be rearranged in any sequence.)

**(b)** How many tours are of the form $B, \ldots, E, B$?

**(c)** How many tours are of the form $A, D, \ldots, F, A$?

**34.** Consider a TSP with 11 vertices labeled $A$ through $K$.

**(a)** How many tours are of the form $A, B, \ldots, A$? (*Hint*: The remaining nine letters can be rearranged in any sequence.)

**(b)** How many tours are of the form $C, \ldots, K, C$?

**(c)** How many tours are of the form $D, B, \ldots, K, D$?

## 6.4 The Nearest-Neighbor and Repetitive Nearest-Neighbor Algorithms

**35.** For the weighted graph shown in Fig. 6-41, (i) find the indicated tour, and (ii) give its cost. (*Note*: This is the TSP introduced in Example 6.1.)

**(a)** The nearest-neighbor tour with starting vertex $B$

**(b)** The nearest-neighbor tour with starting vertex $C$

**(c)** The nearest-neighbor tour with starting vertex $D$

**(d)** The nearest-neighbor tour with starting vertex $E$

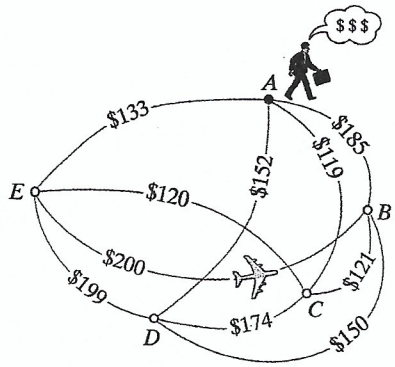



FIGURE 6-41

**36.** A delivery service must deliver packages at Buckman ($B$), Chatfield ($C$), Dayton ($D$), and Evansville ($E$) and then return to Arlington ($A$), the home base. Figure 6-42 shows a graph of the estimated travel times (in minutes) between the cities.

**(a)** Find the nearest-neighbor tour with starting vertex $A$. Give the total travel time of this tour.

**(b)** Find the nearest-neighbor tour with starting vertex $D$. Write the tour as it would be traveled if starting and ending at $A$. Give the total travel time of this tour.

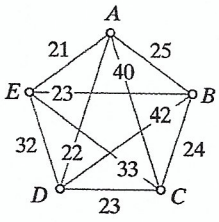



FIGURE 6-42

**37.** The Brute-Force Bandits is a rock band planning a five-city concert tour. The cities and the distances (in miles) between them are given in the weighted graph shown in Fig. 6-43. The tour must start and end at $A$. The cost of the chartered bus in which the band is traveling is \$8 per mile.

**(a)** Find the nearest-neighbor tour with starting vertex $A$. Give the cost (in \$) of this tour.

**(b)** Find the nearest-neighbor tour with starting vertex $B$. Write the tour as it would be traveled by the band, starting and ending at $A$. Give the cost (in \$) of this tour.

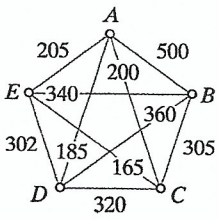



FIGURE 6-43

**38.** A space mission is scheduled to visit the moons Callisto ($C$), Ganymede ($G$), Io ($I$), Mimas ($M$), and Titan ($T$) to collect rock samples at each and then return to Earth ($E$). The travel times (in years) are given in the weighted graph shown in Fig. 6-44. (*Note*: This is the interplanetary TSP discussed in Example 6.11.)

(a) Find the nearest-neighbor tour with starting vertex $E$. Give the total travel time of this tour.

(b) Find the nearest-neighbor tour with starting vertex $T$. Write the tour as it would be traveled by an expedition starting and ending at $E$. Give the total travel time of this tour.

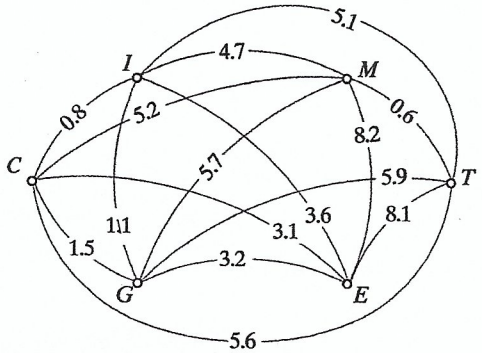



FIGURE 6-44

**39.** This exercise refers to the furniture truck TSP introduced in Exercise 29 (see Fig. 6-39).

(a) Find the nearest-neighbor tour starting at $A$.

(b) Find the nearest-neighbor tour starting at $B$, and give the answer using $A$ as the starting/ending city.

**40.** This exercise refers to the social worker TSP introduced in Exercise 30 (see Fig. 6-40).

(a) Find the nearest-neighbor tour starting at $A$.

(b) Find the nearest-neighbor tour starting at $C$, and give the answer using $A$ as the starting/ending city.

**41.** Darren is a sales rep whose territory consists of the six cities in the mileage chart shown in Fig. 6-45. Darren wants to visit customers at each of the cities, starting and ending his trip in his home city of Atlanta. His travel costs (gas, insurance, etc.) average \$0.75 per mile.

(a) Find the nearest-neighbor tour with Atlanta as the starting city. What is the total cost of this tour?

(b) Find the nearest-neighbor tour using Kansas City as the starting city. Write the tour as it would be traveled by Darren, who must start and end the trip in Atlanta. What is the total cost of this tour?

**Mileage Chart**

| | Atlanta | Columbus | Kansas City | Minneapolis | Pierre | Tulsa |
|---|---|---|---|---|---|---|
| **Atlanta** | * | 533 | 798 | 1068 | 1361 | 772 |
| **Columbus** | 533 | * | 656 | 713 | 1071 | 802 |
| **Kansas City** | 798 | 656 | * | 447 | 592 | 248 |
| **Minneapolis** | 1068 | 713 | 447 | * | 394 | 695 |
| **Pierre** | 1361 | 1071 | 592 | 394 | * | 760 |
| **Tulsa** | 772 | 802 | 248 | 695 | 760 | * |

FIGURE 6-45

**42.** The Platonic Cowboys are a country and western band based in Nashville. The Cowboys are planning a concert tour to the seven cities in the mileage chart shown in Fig. 6-46.

(a) Find the nearest-neighbor tour with Nashville as the starting city. What is the total length of this tour?

(b) Find the nearest-neighbor tour using St. Louis as the starting city. Write the tour as it would be traveled by the band, which must start and end the tour in Nashville. What is the total length of this tour?

**Mileage Chart**

| | Boston | Dallas | Houston | Louisville | Nashville | Pittsburgh | St. Louis |
|---|---|---|---|---|---|---|---|
| **Boston** | * | 1748 | 1804 | 941 | 1088 | 561 | 1141 |
| **Dallas** | 1748 | * | 243 | 819 | 660 | 1204 | 630 |
| **Houston** | 1804 | 243 | * | 928 | 769 | 1313 | 779 |
| **Louisville** | 941 | 819 | 928 | * | 168 | 388 | 263 |
| **Nashville** | 1088 | 660 | 769 | 168 | * | 553 | 299 |
| **Pittsburgh** | 561 | 1204 | 1313 | 388 | 553 | * | 588 |
| **St. Louis** | 1141 | 630 | 779 | 263 | 299 | 588 | * |

FIGURE 6-46

**43.** Find the repetitive nearest-neighbor tour (and give its cost) for the furniture truck TSP discussed in Exercises 29 and 39 (see Fig. 6-39).

**44.** Find the repetitive nearest-neighbor tour for the social worker TSP discussed in Exercises 30 and 40 (see Fig. 6-40).

**45.** This exercise is a continuation of Darren's sales trip problem (Exercise 41). Find the repetitive nearest-neighbor tour, and give the total cost for this tour. Write the answer using Atlanta as the starting city.

**46.** This exercise is a continuation of the Platonic Cowboys concert tour (Exercise 42). Find the repetitive nearest-neighbor tour, and give the total mileage for this tour. Write the answer using Nashville as the starting city.

**47.** Suppose that in solving a TSP you use the nearest-neighbor algorithm and find a nearest-neighbor tour with a total cost of \$13,500. Suppose that you later find out that the cost of an optimal tour is \$12,000. What was the relative error of your nearest-neighbor tour? Express your answer as a percentage, rounded to the nearest tenth of a percent.

**48.** Suppose that in solving a TSP you use the nearest-neighbor algorithm and find a nearest-neighbor tour with a total length of 21,400 miles. Suppose that you later find out that the length of an optimal tour is 20,100 miles. What was the relative error of your nearest-neighbor tour? Express your answer as a percentage, rounded to the nearest tenth of a percent.

## 6.5 Cheapest-Link Algorithm

**49.** Find the cheapest-link tour (and give its cost) for the furniture truck TSP discussed in Exercise 29 (see Fig. 6-39).

**50.** Find the cheapest-link tour for the social worker TSP discussed in Exercise 30 (see Fig. 6-40).

**51.** For the Brute-Force Bandits concert tour discussed in Exercise 37, find the cheapest-link tour, and give the bus cost for this tour (see Fig. 6-43).

**52.** For the weighted graph shown in Fig. 6-47, find the cheapest-link tour. Write the tour using $B$ as the starting vertex.

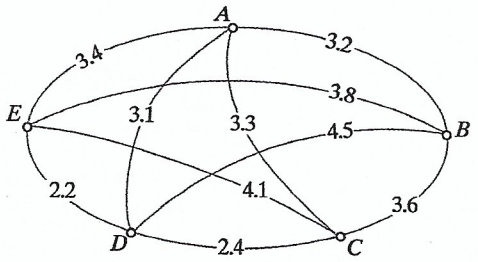



FIGURE 6-47

**53.** For Darren's sales trip problem discussed in Exercise 41, find the cheapest-link tour, and give the total cost for this tour (see Fig. 6-45).

**54.** For the Platonic Cowboys concert tour discussed in Exercise 42, find the cheapest-link tour, and give the total mileage for this tour (see Fig. 6-46).

**55.** A rover on the planet Mercuria has to visit six sites labeled $A$ through $F$. Figure 6-48 shows the time (in days) for the rover to travel between any two sites.

(a) Find the cheapest-link tour for these sites and give its length.

(b) Given that the tour $A, B, D, F, C, E, A$ is an optimal tour, find the relative error of the cheapest-link tour found in (a).

| | A | B | C | D | E | F |
|---|---|---|---|---|---|---|
| A | * | 11 | 19 | 16 | 9 | 10 |
| B | 11 | * | 20 | 13 | 17 | 15 |
| C | 19 | 20 | * | 21 | 13 | 11 |
| D | 16 | 13 | 21 | * | 12 | 16 |
| E | 9 | 17 | 13 | 12 | * | 14 |
| F | 10 | 15 | 11 | 16 | 14 | * |

FIGURE 6-48

**56.** A robotic laser must drill holes on five sites ($A$, $B$, $C$, $D$, and $E$) in a microprocessor chip. At the end, the laser must return to its starting position $A$ and start all over. Figure 6-49 shows the time (in seconds) it takes the laser arm to move from one site to another. In this TSP, a tour is a sequence of drilling locations starting and ending at $A$.

(a) Find the cheapest-link tour and its length.

(b) Given that the tour $A, D, B, E, C, A$ is an optimal tour, find the relative error of the cheapest-link tour found in (a).

| | A | B | C | D | E |
|---|---|---|---|---|---|
| A | * | 1.2 | 0.7 | 1.0 | 1.3 |
| B | 1.2 | * | 0.9 | 0.8 | 1.1 |
| C | 0.7 | 0.9 | * | 1.2 | 0.8 |
| D | 1.0 | 0.8 | 1.2 | * | 0.9 |
| E | 1.3 | 1.1 | 0.8 | 0.9 | * |

FIGURE 6-49

## JOGGING

**57.** Suppose that in solving a TSP you find an approximate solution with a cost of \$1614, and suppose that you later find out that the relative error of your solution was 7.6%. What was the cost of the optimal solution?

**58.** Suppose that in solving a TSP you find an approximate solution with a cost of \$2508, and suppose that you later find out that the relative error of your solution was 4.5%. What was the cost of the optimal solution?

**59.** You have a busy day ahead of you. You must run the following errands (in no particular order): Go to the post office, deposit a check at the bank, pick up some French bread at the deli, visit a friend at the hospital, and get a haircut at Karl's Beauty Salon. You must start and end at home. Each block on the map shown in Fig. 6-50 is exactly 1 mile.

(a) Draw a weighted graph modeling to this problem.

(b) Find an optimal tour for running all the errands. (Use any algorithm you think is appropriate.)

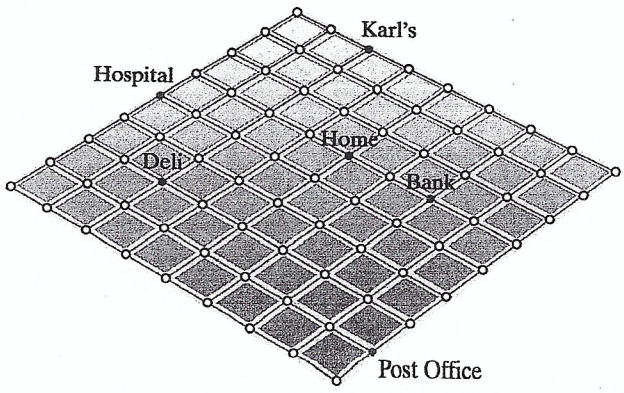



FIGURE 6-50

*In Exercises 60 and 61, you are scheduling a dinner party for six people (A, B, C, D, E, and F). The guests are to be seated around a circular table, and you want to arrange the seating so that each guest is seated between two friends (i.e., the guests to the left and to the right are friends of the guest in between). You can assume that all friendships are mutual (when X is a friend of Y, Y is also a friend of X).*

**60.** Suppose that you are told that all possible friendships can be deduced from the following information:

$A$ is friends with $B$ and $F$; $B$ is friends with $A$, $C$, and $E$; $C$ is friends with $B$, $D$, $E$, and $F$; $E$ is friends with $B$, $C$, $D$, and $F$.

**(a)** Draw a "friendship graph" for the dinner guests.

**(b)** Find a possible seating arrangement for the party.

**(c)** Is there a possible seating arrangement in which $B$ and $E$ are seated next to each other? If there is, find it. If there isn't, explain why not.

**61.** Suppose that you are told that all possible friendships can be deduced from the following information:

$A$ is friends with $C$, $D$, $E$, and $F$; $B$ is friends with $C$, $D$, and $E$; $C$ is friends with $A$, $B$, and $E$; $D$ is friends with $A$, $B$, and $E$.

Explain why it is impossible to have a seating arrangement in which each guest is seated between friends.

**62.** If the number of edges in $K_{500}$ is $x$ and the number of edges in $K_{502}$ is $y$, what is the value of $y - x$?

**63. A 2 by 2 grid graph.** The graph shown in Fig. 6-51 represents a street grid that is 2 blocks by 2 blocks. (Such a graph is called a *2 by 2 grid graph.*) For convenience, the vertices are labeled by type: corner vertices $C_1$, $C_2$, $C_3$, and $C_4$, boundary vertices $B_1$, $B_2$, $B_3$, and $B_4$, and the interior vertex $I$.

**(a)** Find a Hamilton path in the graph that starts at $I$.

**(b)** Find a Hamilton path in the graph that starts at one of the corner vertices and ends at a different corner vertex.

**(c)** Find a Hamilton path that starts at one of the corner vertices and ends at $I$.

**(d)** Find (if you can) a Hamilton path that starts at one of the corner vertices and ends at one of the boundary vertices. If this is impossible, explain why.

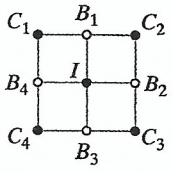



FIGURE 6-51

**64.** Find (if you can) a Hamilton circuit in the 2 by 2 grid graph discussed in Exercise 63. If this is impossible, explain why.

**65. A 3 by 3 grid graph.** The graph shown in Fig. 6-52 represents a street grid that is 3 blocks by 3 blocks. The graph has four corner vertices ($C_1$, $C_2$, $C_3$, and $C_4$), eight boundary vertices ($B_1$ through $B_8$), and four interior vertices ($I_1$, $I_2$, $I_3$, and $I_4$).

**(a)** Find a Hamilton circuit in the graph.

**(b)** Find a Hamilton path in the graph that starts at one of the corner vertices and ends at a different corner vertex.

**(c)** Find (if you can) a Hamilton path that starts at one of the corner vertices and ends at one of the interior vertices. If this is impossible, explain why.

**(d)** Given any two adjacent vertices of the graph, explain why there always is a Hamilton path that starts at one and ends at the other one.

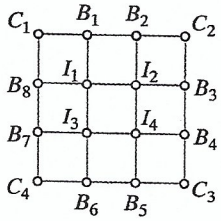



FIGURE 6-52

**66. A 3 by 4 grid graph.** The graph shown in Fig. 6-53 represents a street grid that is 3 blocks by 4 blocks.

**(a)** Draw a Hamilton circuit in the graph.

**(b)** Draw a Hamilton path in the graph that starts at $C_1$ and ends at $C_3$.

**(c)** Draw (if you can) a Hamilton path in the graph that starts at $C_1$ and ends at $C_2$. If this is impossible, explain why.

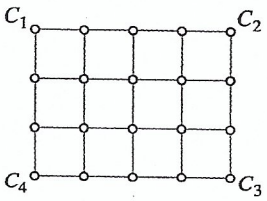



FIGURE 6-53

**67.** Explain why the cheapest edge in any graph is always part of the Hamilton circuit obtained using the nearest-neighbor algorithm.

**68. (a)** Explain why a graph that has a bridge cannot have a Hamilton circuit.

**(b)** Give an example of a graph with bridges that has a Hamilton path.

**69.** Nick is a traveling salesman. His territory consists of the 11 cities shown on the mileage chart in Fig. 6-54. Nick must find a tour that starts and ends in Dallas (that's his home) and visits each of the other 10 cities.

**(a)** Find a nearest-neighbor tour that starts at Dallas.

**(b)** Find the cheapest-link tour.

**Mileage Chart**

| | Atlanta | Boston | Buffalo | Chicago | Columbus | Dallas | Denver | Houston | Kansas City | Louisville | Memphis |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **Atlanta** | * | 1037 | 859 | 674 | 533 | 795 | 1398 | 789 | 798 | 382 | 371 |
| **Boston** | 1037 | * | 446 | 963 | 735 | 1748 | 1949 | 1804 | 1391 | 941 | 1293 |
| **Buffalo** | 859 | 446 | * | 522 | 326 | 1346 | 1508 | 1460 | 966 | 532 | 899 |
| **Chicago** | 674 | 963 | 522 | * | 308 | 917 | 996 | 1067 | 499 | 292 | 530 |
| **Columbus** | 533 | 735 | 326 | 308 | * | 1028 | 1229 | 1137 | 656 | 209 | 576 |
| **Dallas** | 795 | 1748 | 1346 | 917 | 1028 | * | 781 | 243 | 489 | 819 | 452 |
| **Denver** | 1398 | 1949 | 1508 | 996 | 1229 | 781 | * | 1019 | 600 | 1120 | 1040 |
| **Houston** | 789 | 1804 | 1460 | 1067 | 1137 | 243 | 1019 | * | 710 | 928 | 561 |
| **Kansas City** | 798 | 1391 | 966 | 499 | 656 | 489 | 600 | 710 | * | 520 | 451 |
| **Louisville** | 382 | 941 | 532 | 292 | 209 | 819 | 1120 | 928 | 520 | * | 367 |
| **Memphis** | 371 | 1293 | 899 | 530 | 576 | 452 | 1040 | 561 | 451 | 367 | * |

FIGURE 6-54

**70.** Julie is the marketing manager for a small software company based in Boston. She is planning a sales trip to Michigan to visit customers in each of the nine cities shown on the mileage chart in Fig. 6-55. She can fly from Boston to any one of the cities and fly out of any one of the cities back to Boston for the same price (call the arrival city $A$ and the departure city $D$). Her plan is to pick up a rental car at $A$, drive to each of the other cities, and drop off the rental car at the last city $D$. Slightly complicating the situation is that Michigan has two separate peninsulas—an upper peninsula and a lower peninsula—and the only way to get from one to the other is through the Mackinaw Bridge connecting Cheboygan to Sault Ste. Marie. (There is a $3 toll to cross the bridge in either direction.)

**Mileage Chart**

| | Detroit | Lansing | Grand Rapids | Flint | Cheboygan | Sault Ste. Marie | Marquette | Escanaba | Menominee |
|---|---|---|---|---|---|---|---|---|---|
| **Detroit** | * | 90 | 158 | 68 | 280 | | | | |
| **Lansing** | 90 | * | 68 | 56 | 221 | | | | |
| **Grand Rapids** | 158 | 68 | * | 114 | 233 | | | | |
| **Flint** | 68 | 56 | 114 | * | 215 | | | | |
| **Cheboygan** | 280 | 221 | 233 | 215 | * | 78 | | | |
| **Sault Ste. Marie** | | | | | 78 | * | 164 | 174 | 227 |
| **Marquette** | | | | | | 164 | * | 67 | 120 |
| **Escanaba** | | | | | | 174 | 67 | * | 55 |
| **Menominee** | | | | | | 227 | 120 | 55 | * |

FIGURE 6-55

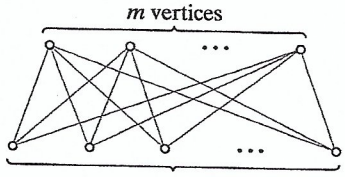
(a) Suppose that the rental car company charges 39 cents per mile plus a drop off fee of $250 if $A$ and $D$ are different cities (there is no charge if $A = D$). Find the optimal (cheapest) route and give the total cost.

(b) Suppose that the rental car company charges 49 cents per mile but the car can be returned to any city without a drop off fee. Find the optimal route and give the total cost.

## RUNNING

**71. Complete bipartite graphs.** A complete bipartite graph is a graph with the property that the vertices can be divided into two sets $A$ and $B$ and each vertex in set $A$ is adjacent to each of the vertices in set $B$. There are no other edges! If there are $m$ vertices in set $A$ and $n$ vertices in set $B$, the complete bipartite graph is written as $K_{n,n}$. Figure 6-56 shows a generic bipartite graph.

(a) For $n > 1$, the complete bipartite graphs of the form $K_{m,n}$ all have Hamilton circuits. Explain why.




$K_{m,n}$

FIGURE 6-56

(b) If the difference between $m$ and $n$ is exactly 1 (i.e., $|m-n|=1$), the complete bipartite graph $K_{m,n}$ has a Hamilton path. Explain why.

(c) When the difference between $m$ and $n$ is more than 1, then the complete bipartite graph $K_{m,n}$ has neither a Hamilton circuit nor a Hamilton path. Explain why.

**72. *m* by *n* grid graphs.** An $m$ by $n$ grid graph represents a rectangular street grid that is $m$ blocks by $n$ blocks, as indicated in Fig. 6-57. (You should try Exercises 63 through 66 before you try this one.)

(a) If $m$ and $n$ are both odd, then the $m$ by $n$ grid graph has a Hamilton circuit. Describe the circuit by drawing it on a generic graph.

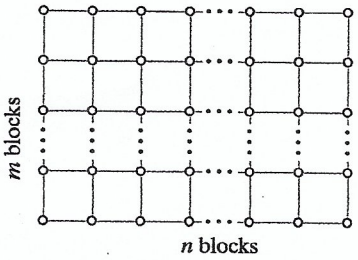



FIGURE 6-57

(b) If either $m$ or $n$ is even and the other one is odd, then the $m$ by $n$ grid graph has a Hamilton circuit. Describe the circuit by drawing it on a generic graph.

(c) If $m$ and $n$ are both even, then the $m$ by $n$ grid graph does not have a Hamilton circuit. Explain why a Hamilton circuit is impossible.

**73. Ore's theorem.** A connected graph with $N$ vertices is said to satisfy *Ore's condition* if $\deg(X)+\deg(Y)\geq N$ for every pair of vertices $X$ and $Y$ of the graph. Ore's theorem states that *if a graph satisfies Ore's condition, then it has a Hamilton circuit.*

(a) Explain why the complete bipartite graph $K_{n,n}$ (see Exercise 71) satisfies Ore's condition.

(b) Explain why for $m\neq n$, the complete bipartite graph $K_{m,n}$ (see Exercise 71) does *not* satisfy Ore's condition.

(c) Ore's condition is sufficient to guarantee that a connected graph has a Hamilton circuit but is not a necessary condition. Give an example of a graph that has a Hamilton circuit but does not satisfy Ore's condition.

**74. Dirac's theorem.** If $G$ is a connected graph with $N$ vertices and $\deg(X)\geq\frac{N}{2}$ for every vertex $X$, then *$G$ has a Hamilton circuit.* Explain why Dirac's theorem is a direct consequence of Ore's theorem.