Euler Circuits

Topics in Contemporary Mathematics

MA 103

Summer II, 2013

Content

- Euler Circuit Problems
- What Is a Graph?
- Graph Concepts and Terminology
- Graph Models
- Euler's Theorems
- Fleury's Algorithm
- Eulerizing Graphs

Euler Paths and Circuits

How do we create *efficient routes* for the delivery of goods and services (such as mail delivery, garbage collection, police patrols, newspaper deliveries, and late-night pizza deliveries) along the streets of a city, town, or neighborhood.

These types of *management science* problems are known as *Euler circuit problems*.

Routing Problems

What is a routing problem? To put it in the most general way, routing problems are concerned with finding ways to route the delivery of goods and/or services to an assortment of destinations.

The goods or services in question could be packages, mail, newspapers, pizzas, garbage collection, bus service, and so on.

The delivery destinations could be homes, warehouses, distribution centers, terminals, and the like.

Two Basic Questions

The existence question is simple: Is an actual route possible? For most routing problems, the existence question is easy to answer, and the answer takes the form of a simple yes or no.

When the answer to the existence question is yes, then a second question–the optimization question–comes into play.

Of all the possible routes, which one is the optimal route? Optimal here means “the best” when measured against some predetermined variable such as cost, distance, or time.

Euler Circuit Problems

The common thread in all Euler circuit problems is what we might call, the exhaustion requirement– the requirement that the route must wind its way through . . . everywhere.

Thus, in an Euler circuit problem, by definition every single one of the streets (or bridges, or lanes, or highways) within a defined area (be it a town, an area of town, or a subdivision) must be covered by the route. We will refer to these types of routes as exhaustive routes.

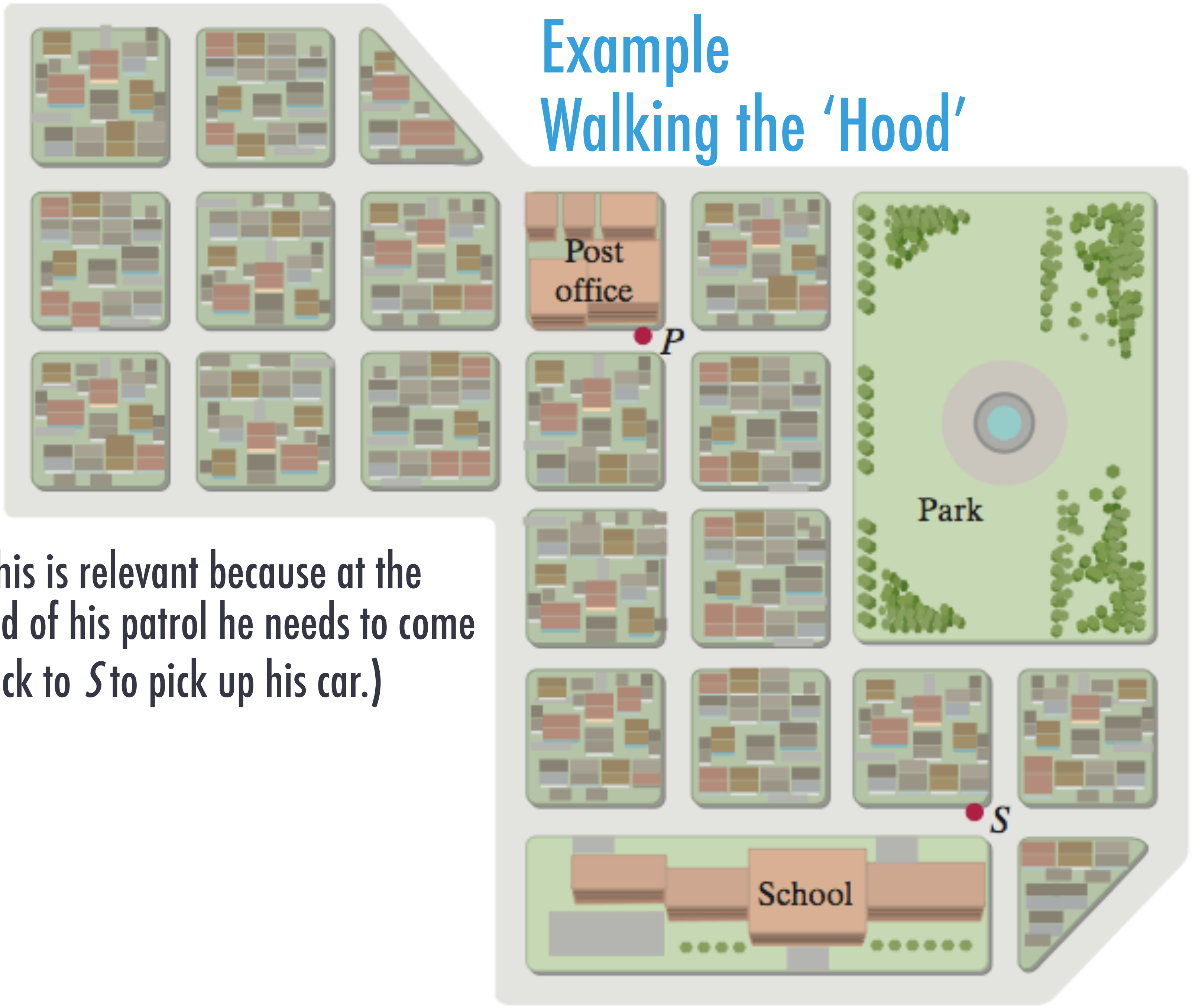
Example Walking the 'Hood'

A private security guard is hired to patrol the streets of the Sunnyside neighborhood shown next.

The security guard's assignment is to make an exhaustive patrol, on foot, through the entire neighborhood.

Obviously, he doesn't want to walk any more than what is necessary. His starting point is the southeast corner across from the school (*S*)–that's where he parks his car.

Example
Walking the 'Hood'



(This is relevant because at the end of his patrol he needs to come back to S to pick up his car.)

Example Walking the 'Hood'

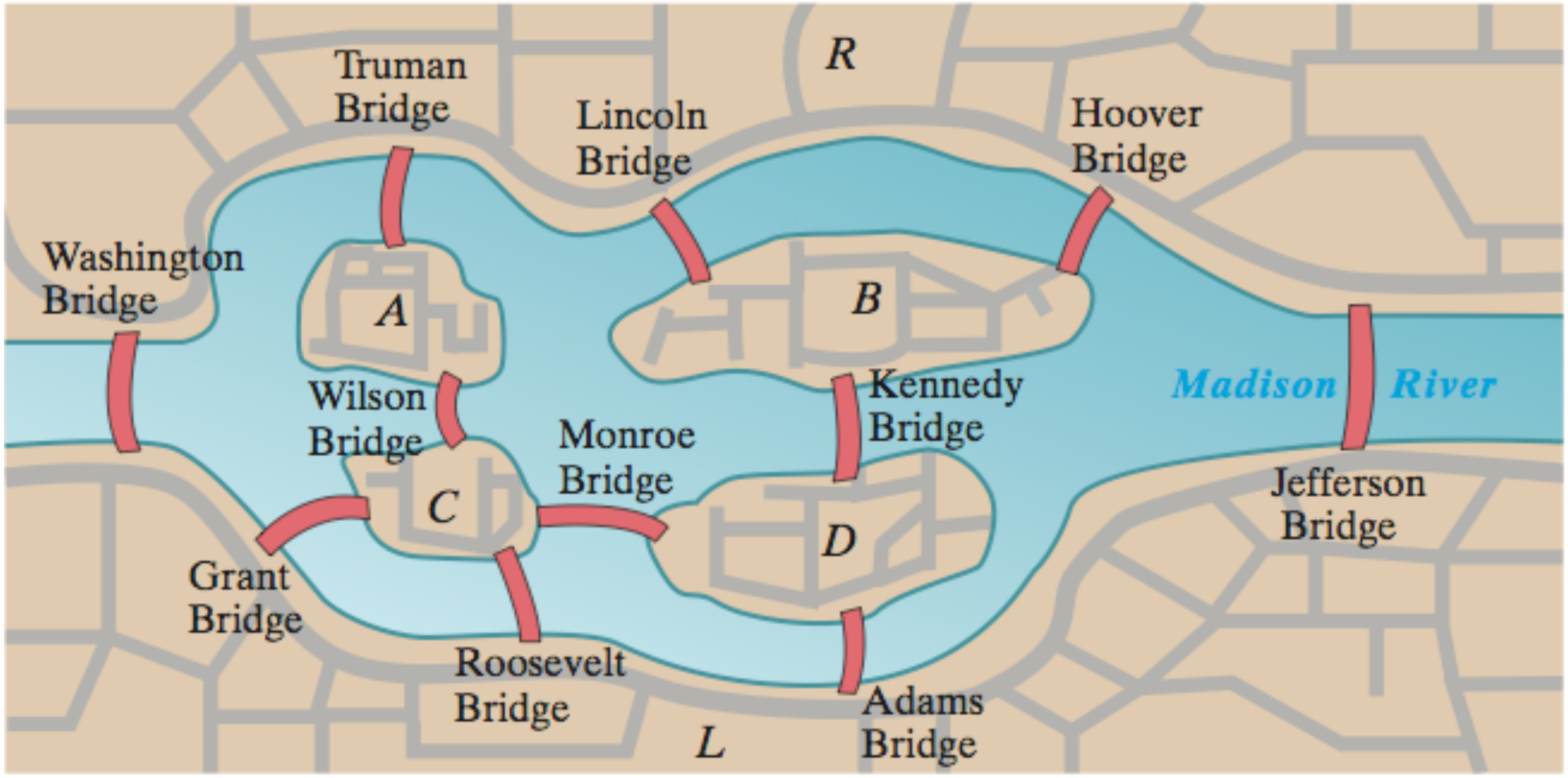
Being a practical person, the security guard would like the answers to two questions:

(1)Is it possible to start and end at *S*, cover every block of the neighborhood, and pass through each block just once?

(2) If some of the blocks will have to be covered more than once, what is an optimal route that covers the entire neighborhood?

Optimal here means "with the minimal amount of walking."

Example The Bridges of Madison County



Example The Bridges of Madison County

A photographer needs to drive across each bridge once for the photo shoot. Moreover, since there is a $25 toll (the locals call it a "maintenance tax") every time an out-of-town visitor drives across a bridge, the photographer wants to minimize the total cost of his trip and to recross bridges only if it is absolutely necessary.

What is the optimal (cheapest) route for him to follow?

Example Child’s Play

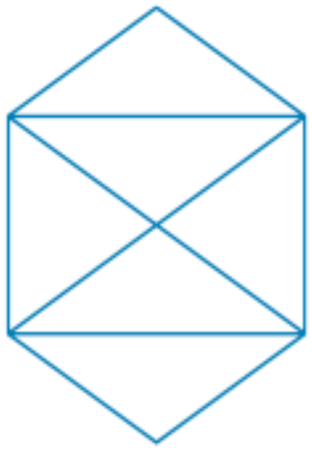
The Figure on the next slide shows a few simple line drawings. The name of the game is to trace each drawing without lifting the pencil or retracing any of the lines.

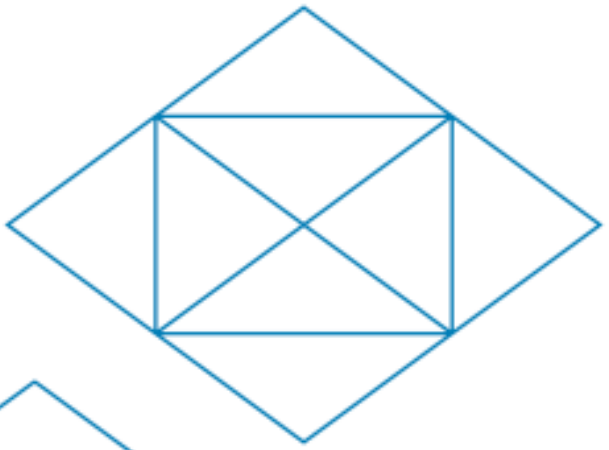
These kinds of tracings are called unicursal tracings. (When we end in the same place we started, we call it a closed unicursal tracing; when we start and end in different places, we call it an open unicursal tracing.)

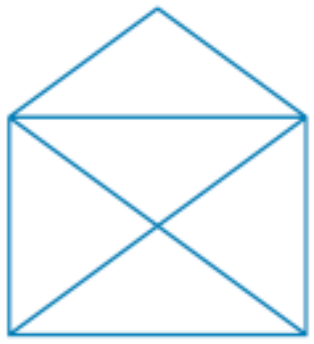
Example Child's Play

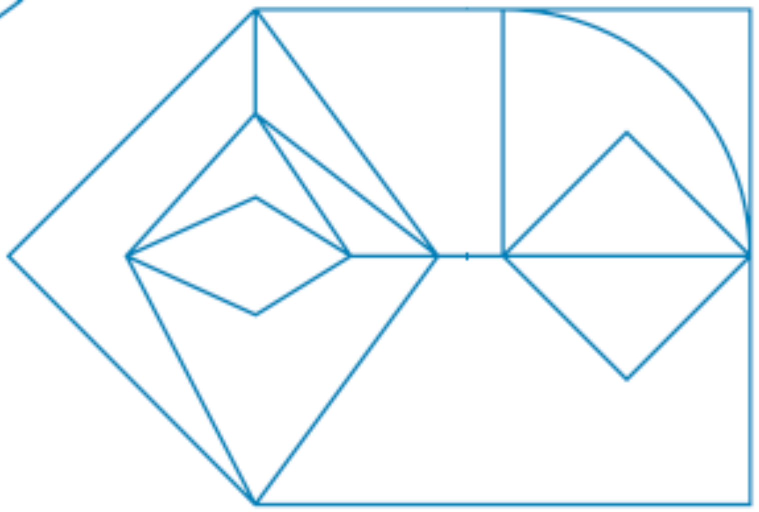
Which of the drawings can be traced with closed unicursal tracings? Which with only open ones?

How can we tell if a unicursal tracing (open or closed) is possible?









What Is a Graph?

Euler circuit problems can all be tackled by means of a single unifying mathematical concept–*the concept of a graph.*

The most common way to describe a graph is by means of a picture. The basic elements of such a picture are:

a set of "dots" called the **vertices** of the graph and

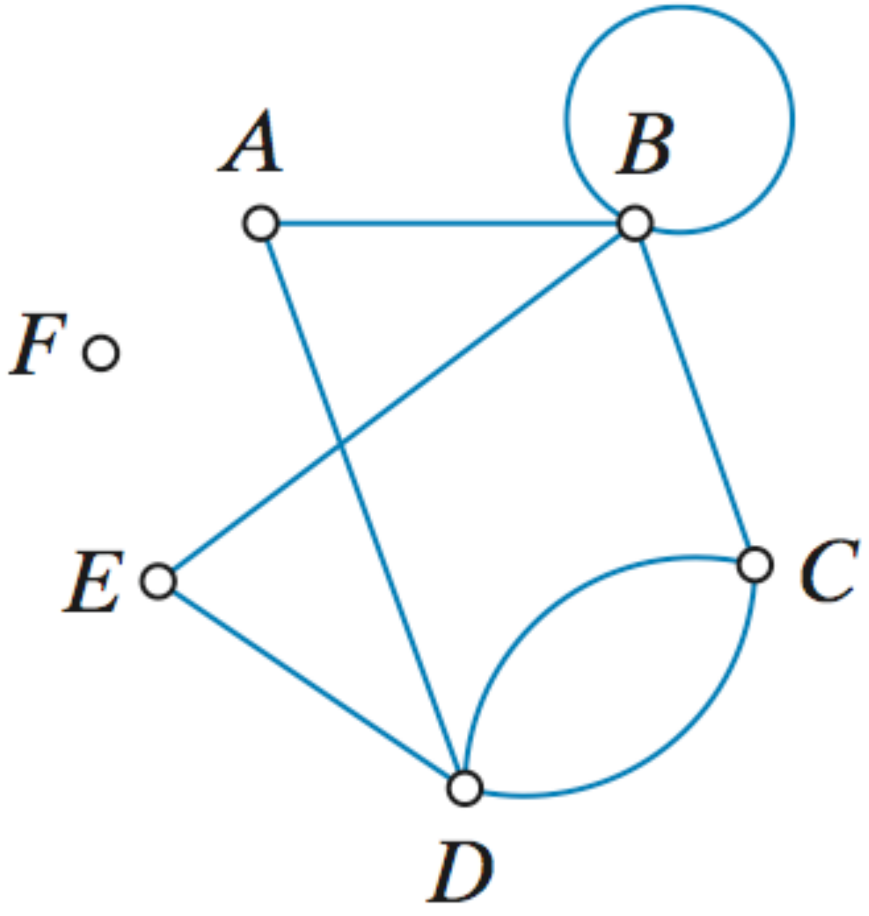
a collection of "lines" called the edges of the graph.

On the surface, that's all there is to it–lines connecting dots! Below the surface there is a surprisingly rich theory. Let's explore a few basic concepts first.

Example Connect the Dots

This graph has six vertices called *A*, *B*, *C*, *D*, *E* and *F*.

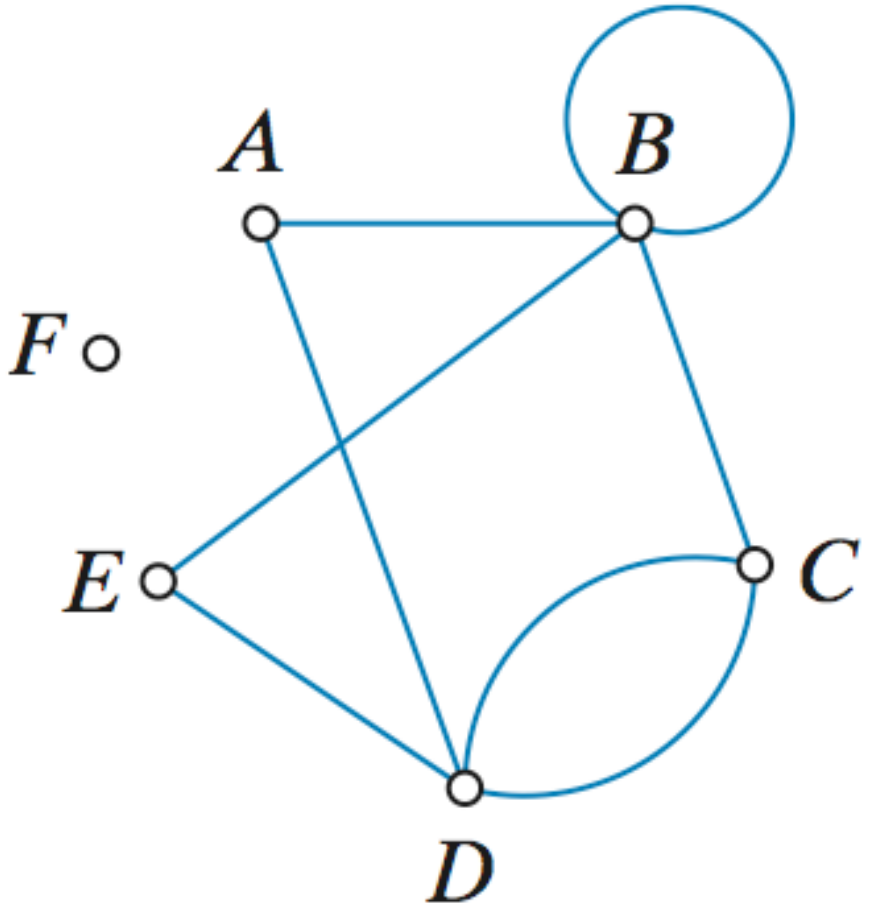
Each edge can be described by listing (in any order) the pair of vertices that are connected by the edge. Thus, the edges of this graph, listed in random order, are *AB*, *BC*, *CD*, *AD*, *DE*, *EB*, *CD*, and *BB*



Example Connect the Dots

Notice several important things about the edges of the graph:

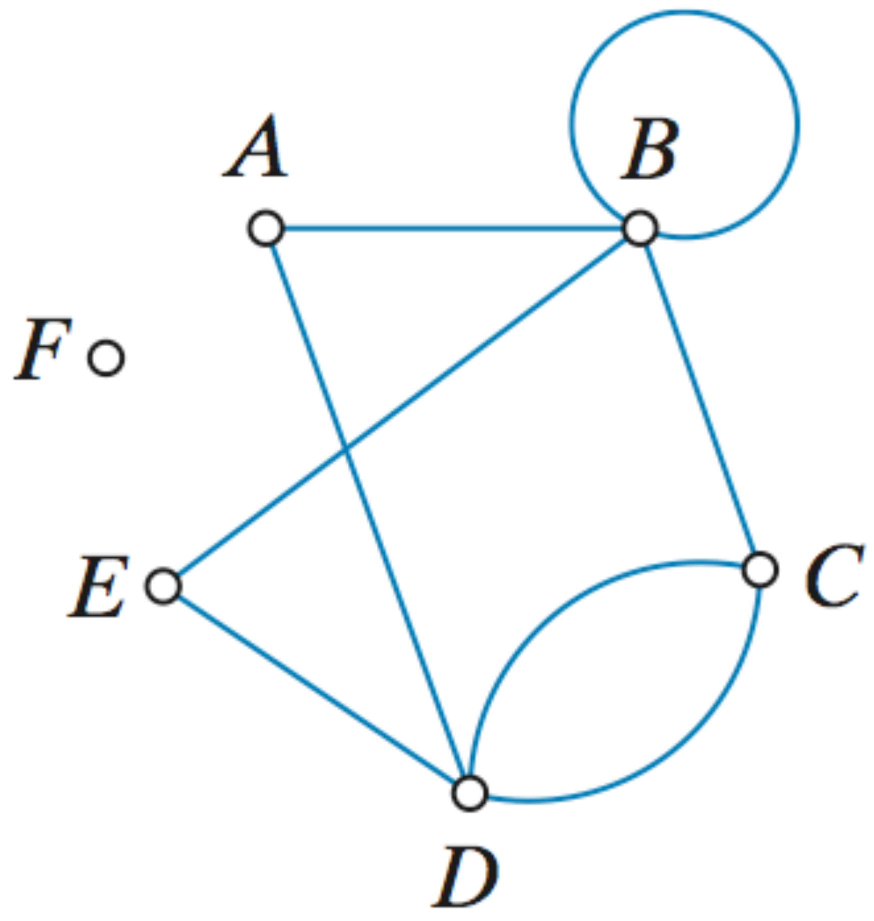
It is possible for an edge to connect a vertex back to itself, as is the case with *BB*. These type of edges are called **loops**.



Example Connect the Dots

- It is possible for two edges to connect the same pair of vertices, as is the case with *CD*, which is a "double edge."

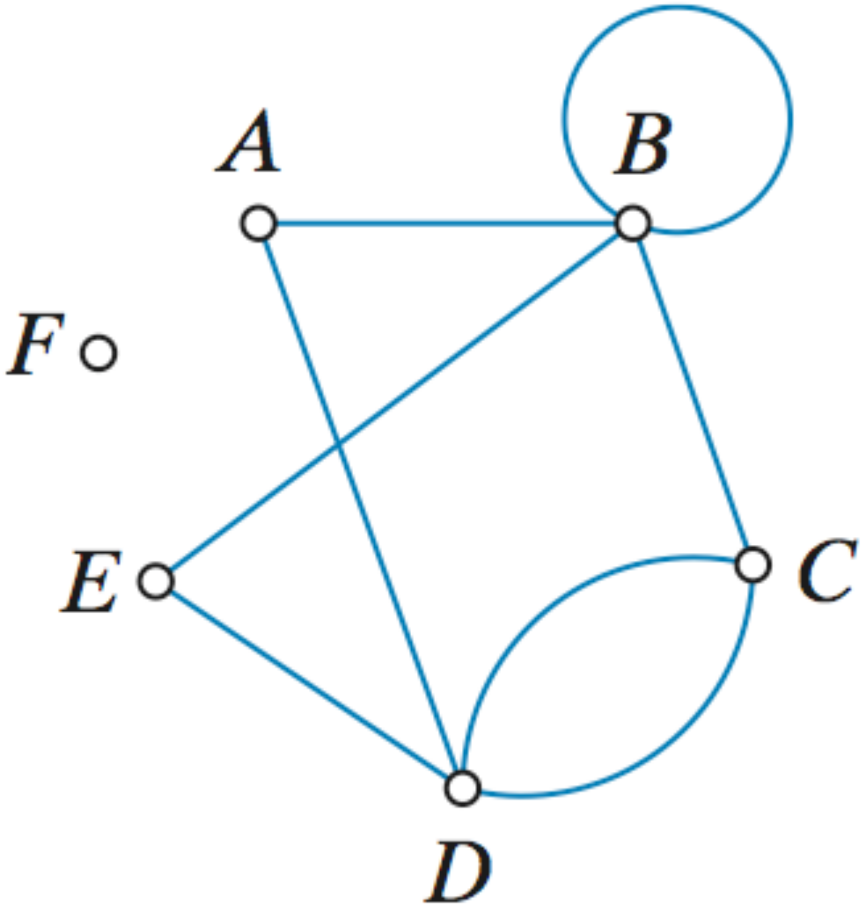
  In general, we refer to such edges as **multiple edges**.



Example Connect the Dots

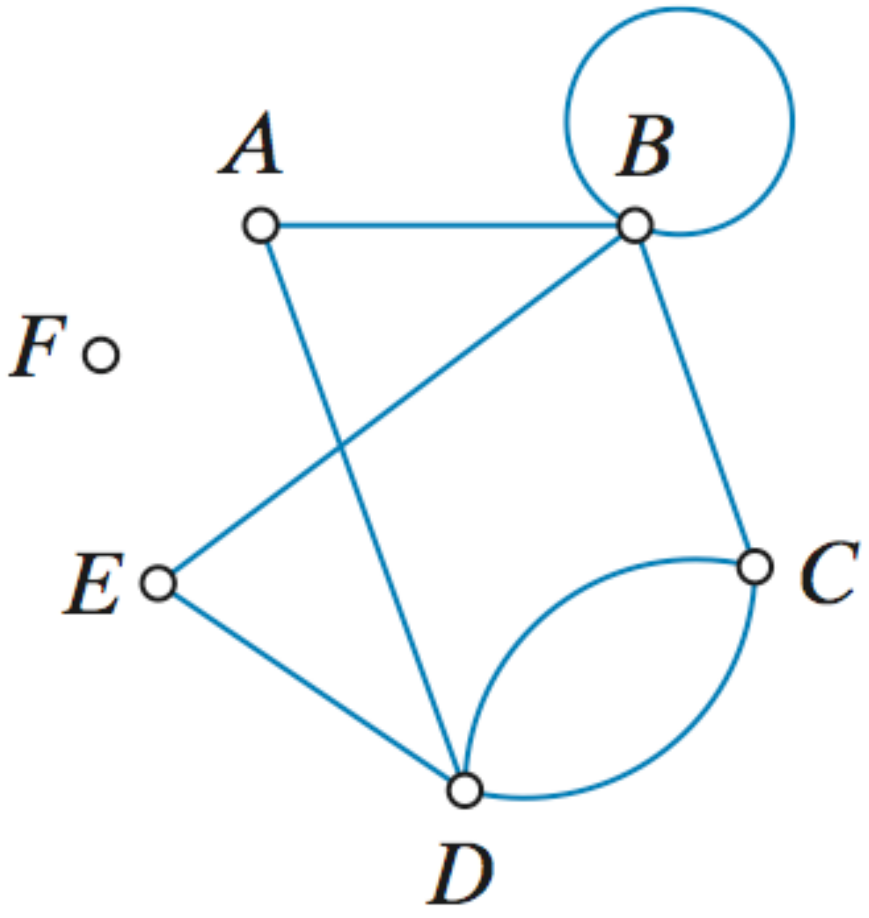
- Sometimes edges "cross" each other at incidental crossing points that are not themselves vertices of the graph.

Such is the case with the crossing point created by edges *AD* and *BE* .



Example Connect the Dots

- Edges do not have a direction; thus, there is no right or wrong order to write an edge–*AB* or *BA* are both acceptable.

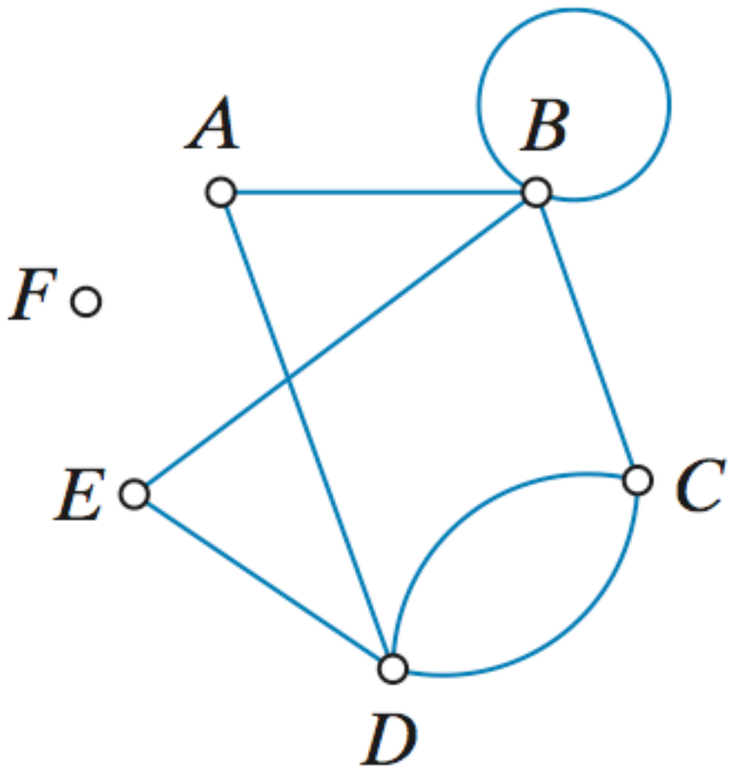


Example Connect the Dots

A convenient way to describe the vertices and edges of a graph is by using the notation of sets.

For the graph shown here the **vertex set** is $\mathcal{V} = \{A, B, C, D, E, F\}$, and the

edge set is $E = \{AB, AD, BB, BC, BE, CD, CD$, and $DE\}$. Notice that CD appears twice in the edge set, indicating that there are two edges connecting C and D.



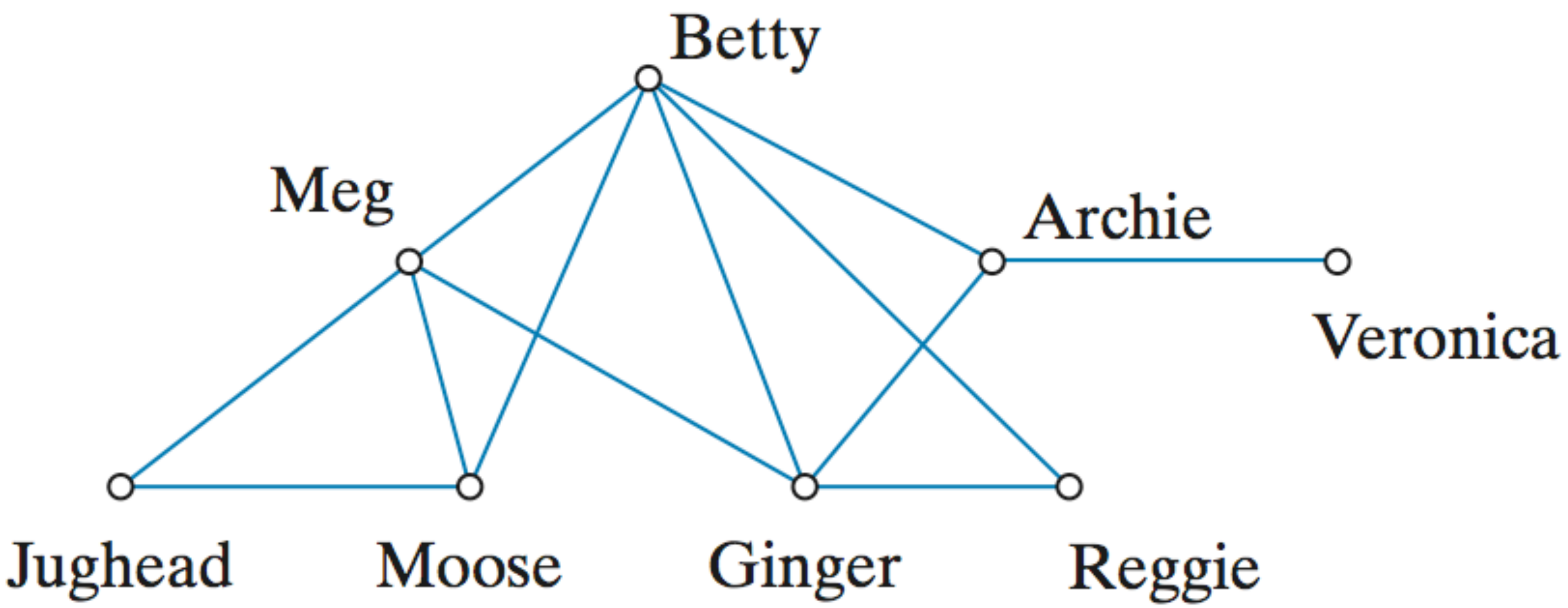
Example Relationship Graphs

Imagine that as part of a sociology study we want to describe the network of “friendships” that develops among a group of students through their Facebook sites.

We can illustrate this very nicely with a graph such as the one on the next slide.

Example Relationship Graphs

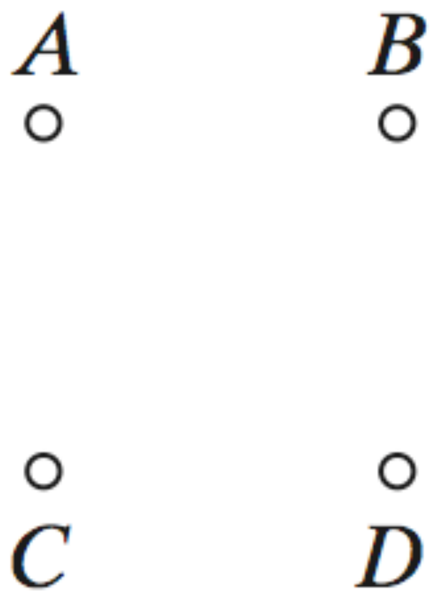
In this graph the vertices represent people (the students), and an edge connecting vertex *X* to vertex *Y* implies that *X* and *Y* are "Facebook friends."



Example Pure Isolation

Here is a graph with four **isolated vertices** and having no edges.

We won't be seeing graphs like this too often, but it's important to know that graphs with no edges are allowed.



The edge set of a graph with no edges is the *empty* set (we can write it as $E = \{\ \}$ or $E = \phi$).

Example Pictures Optional

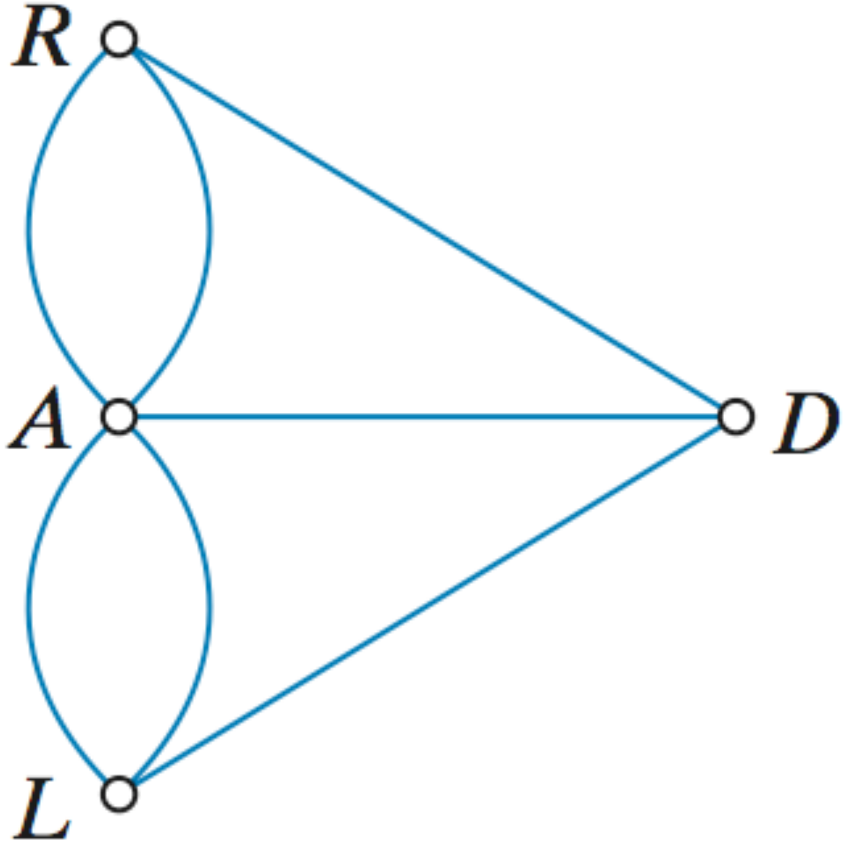
Suppose you are given the following information about a graph: The vertex set is $\mathcal{V} = \{A, D, L, \text{and } R\}$, and the edge set is $\mathcal{E} = \{AD, AL, AL, AR, AR, DL, DR\}$.

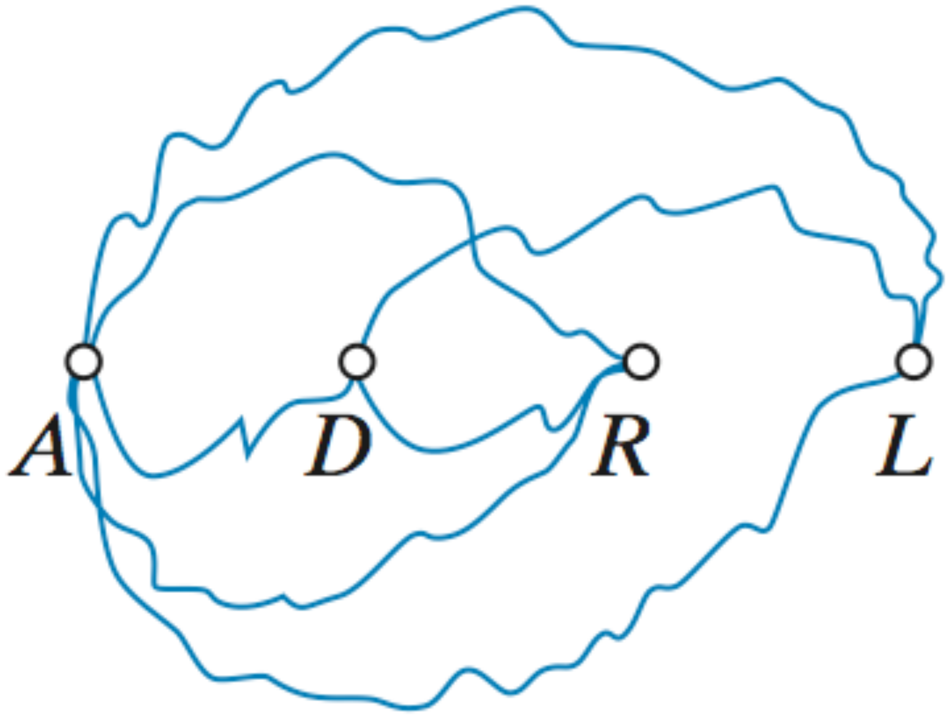
But where is the picture? You are told that if you really want a picture, you can make up your own. That's fine, but where should you place the vertices? What should the edges look like?

Good news: These issues are irrelevant! You have total freedom to place the vertices anywhere you please, and as long as you connect the right pairs of vertices, you can connect them any way you like!

Example Pictures Optional

Here are two pictures of the same graph.





Example Pictures Optional

While they may look like two totally different pictures to the casual observer, *they both describe the same vertex set* $\mathcal{V} = \{A, D, L$, and $R\}$ and the same edge set $\mathcal{E} = \{AD, AL, AL, AR, AR, DL, DR\}$.

Important Point

We think of a graph as a picture consisting of dots and lines, but the picture is just a visual representation of a more abstract idea: a set of objects (represented by the vertices of the graph) and a relationship among pairs of objects (represented by the edges of the graph.)

All one needs to describe these two things are a vertex set $\mathcal{V}$ and an edge set $\mathcal{E}$. A picture is nice but not essential.

Thus, if we simply give the vertex set is $\mathcal{V} = \{A, D, L, \text{and } R\}$, and the edge set is $\mathcal{E} = \{AD, AL, AL, AR, AR, DL, DR\}$. we have defined a graph.

GRAPH

A **graph** is a structure consisting of a set of objects (the vertex set) and a list describing how pairs of objects are related (the edge set). Relations among objects include the possibility of an object being related to itself (a loop) as well as multiple relations between the same pair of objects (multiple edges).

Terminology

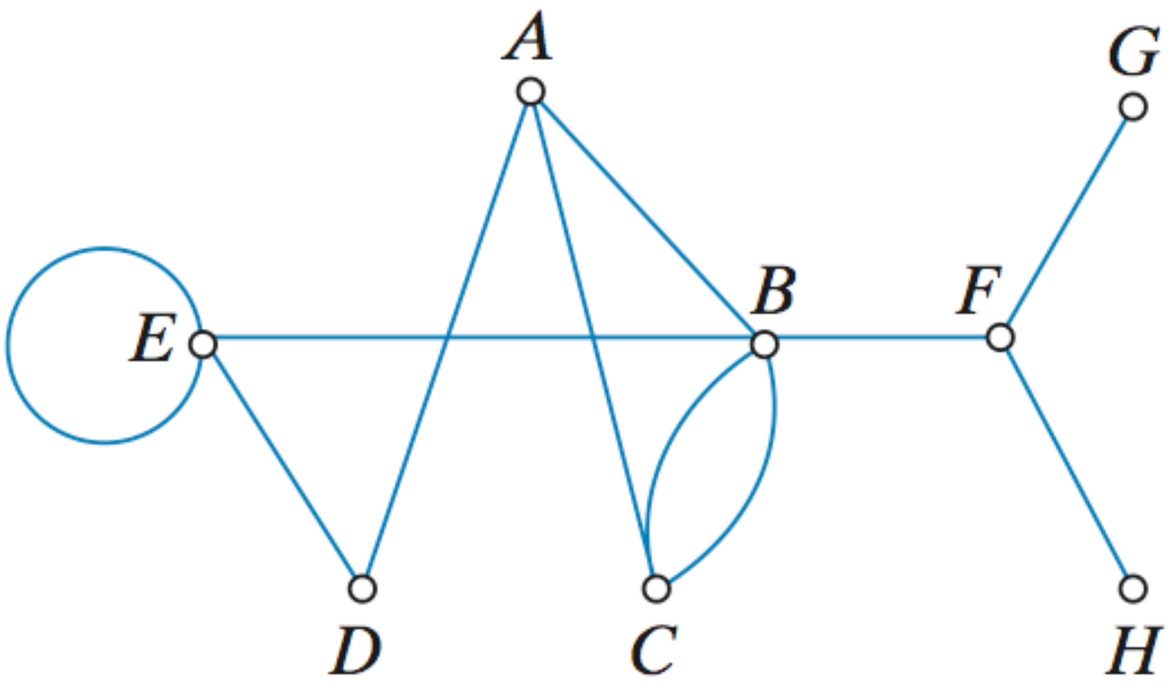
In this section we will introduce some important concepts and terminology that are associated with graph theory.

Example Adjacency

We say that two vertices in a graph are **adjacent** if they are joined by an edge.

Vertices *A* and *B* are adjacent and so are vertices *B* and *C*. Vertices *C* and *D* are not adjacent, and neither are vertices A and E.

Because of the loop EE, we say that vertex E is adjacent to itself.

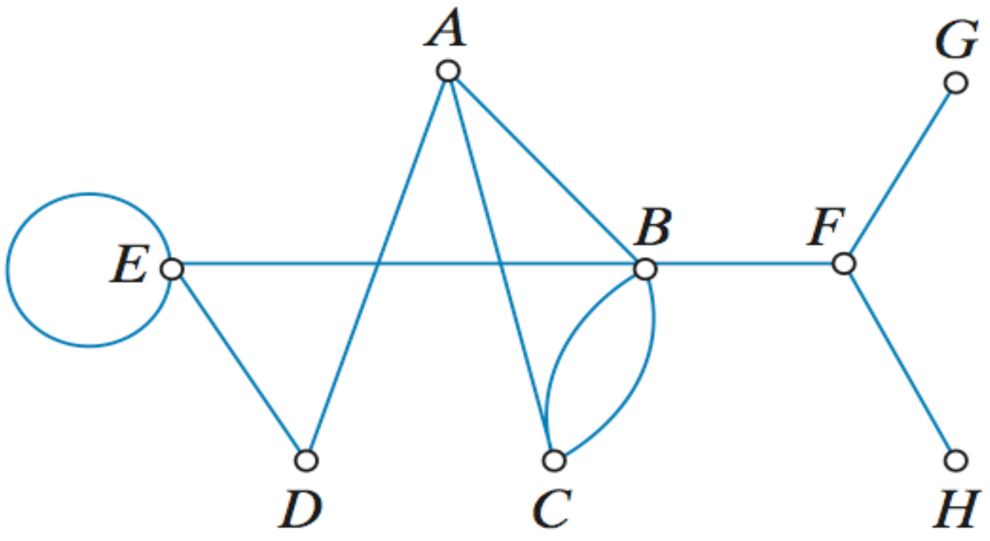


Example Degree of a Vertex

The **degree** *of a vertex is the number of edges meeting at that vertex.*

A loop counts twice toward the degree. We will use the notation deg(*V*) to denote the degree of vertex *V*.

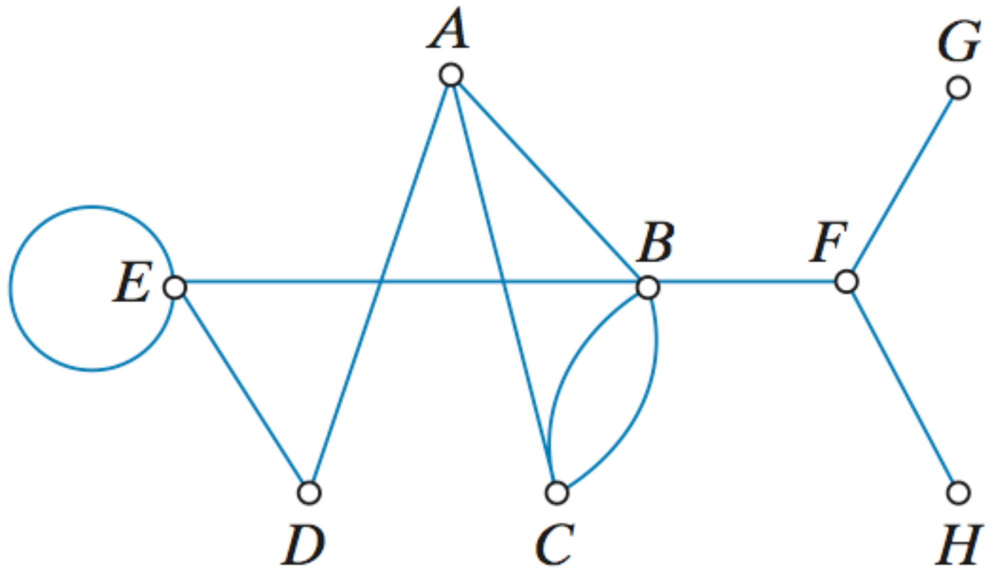
In the figure shown here, the degrees of the vertices are as follows:
deg(*A*) = 3, deg(*B*) = 5,
deg(*C*) = 3, deg(*D*) = 2,
deg(*E*) = 4, deg(*F*) = 3,
deg(*G*) = 1, and deg(*H*) = 1.



Example Degree of a Vertex

Because distinguishing vertices with an even degree from vertices with an odd degree is going to be critical later on, we will often refer to vertices as **even vertices** or **odd vertices**, depending on their degree.

This graph has two even vertices (*D* and *E*) and six odd vertices (all the others).



Example Paths and Circuits

Paths and circuits both describe "trips" along the edges of a graph.

The only real difference between a path and a circuit is that:

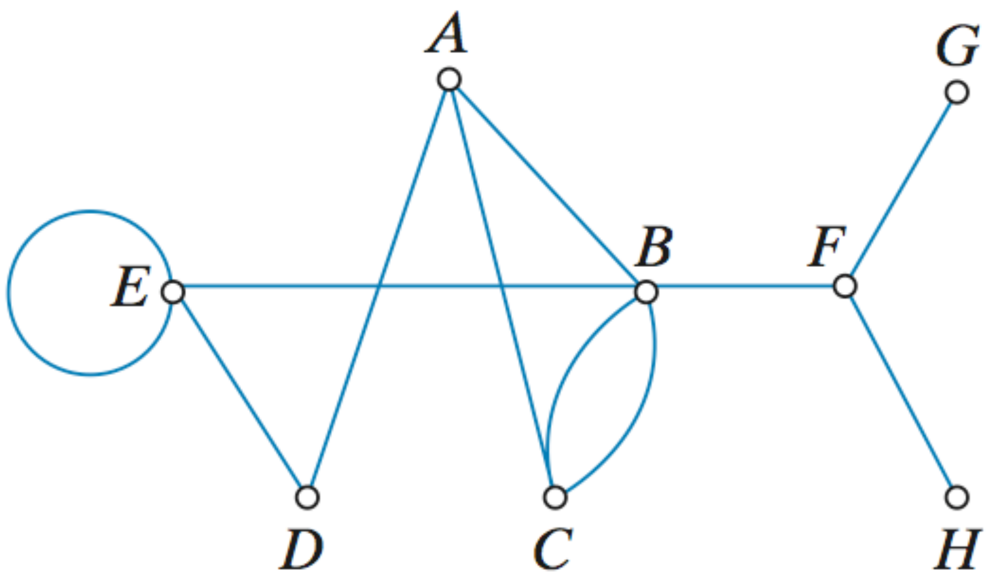
a **circuit** is a "closed" trip (the trip ends back at the starting point), whereas a **path** is an "open" trip (the starting and ending points are different).

In this context, by a "trip" (be it a path or a circuit), we mean a sequence of adjacent edges with the property that *an edge can be traveled just once.*

Example Paths and Circuits

The standard way to describe a path or a circuit is by listing the vertices in order of travel. Here are a few examples of paths and circuits using the graph shown here:

- *A*, *B*, *E*, *D* is a *path* from vertex *A* to vertex *D*. The edges of this path in order of travel
  are *AB*, *BE*, and *ED*. The **length** of the path (i.e., the number of edges in the path)

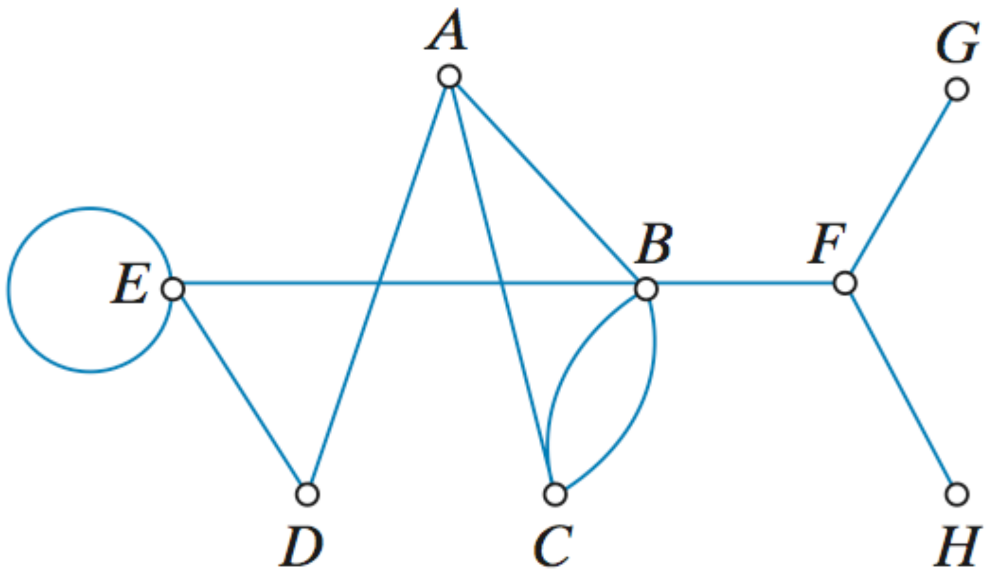


Example Paths and Circuits

- *A*, *B*, *C*, *A*, *D*, *E* is a *path* of length 5 from *A* to *E*. This path visits vertex A twice (that's fine), *but no edge is repeated.*

- *A*, *B*, *C*, *B*, *E* is another *path* from *A* to *E*. This path is only possible because there

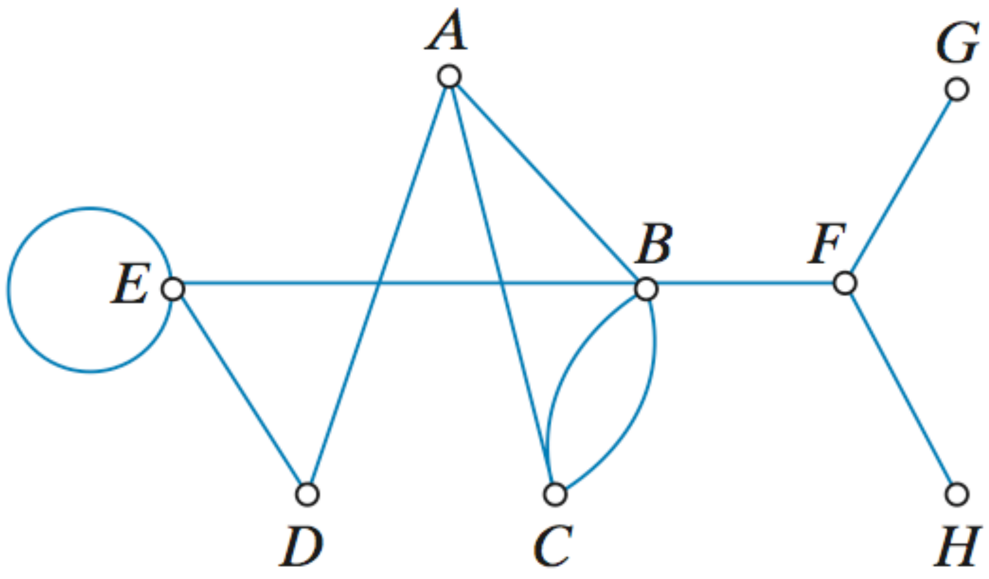
  are two edges connecting *B* and *C*.



Example Paths and Circuits

- *A, C, B, E, E, D* is a *path* of length 5 from *A* to *D*. One of the edges in this path is the loop *EE*.

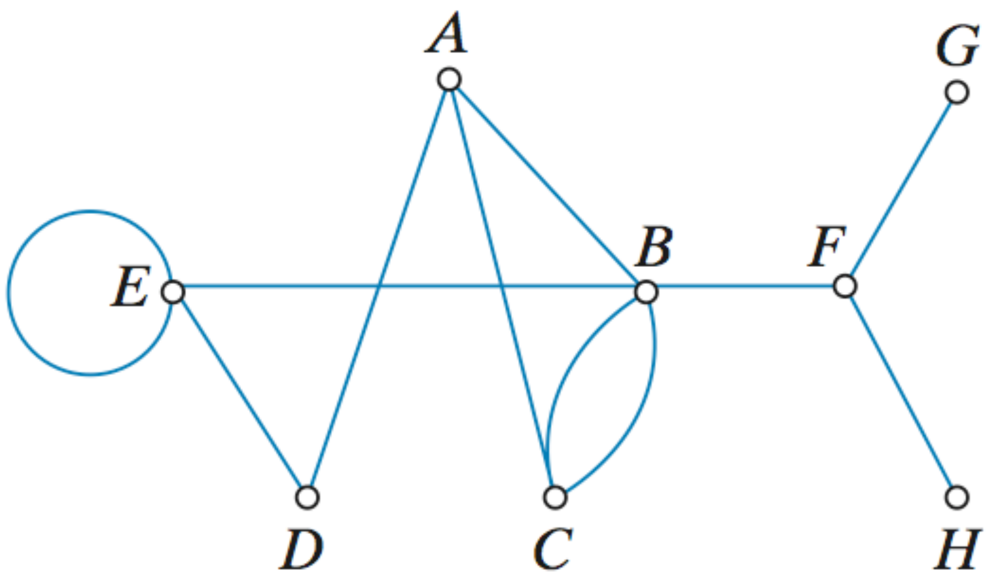
- *A, B, C, B, A, D is not a path because the edge AB is traveled twice.*



Example Paths and Circuits

- *A*, *B*, *C*, *B*, *E*, *E*, *D*, *A*, *C*, *B* is not a path because the edge *BC* is traveled three times.

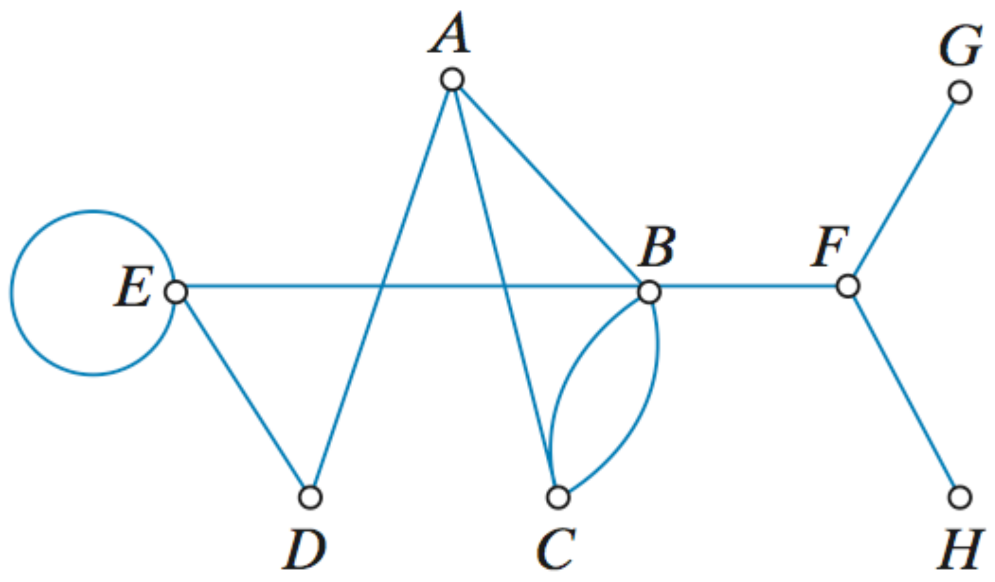
The first two passes are fine, since there are two edges connecting *B* and *C*, but a third pass requires that we travel through one of those edges a second time.



Example Paths and Circuits

- *B*, *C*, *B* is a circuit of length 2. Circuits of length 2 are possible when there are multiple edges.

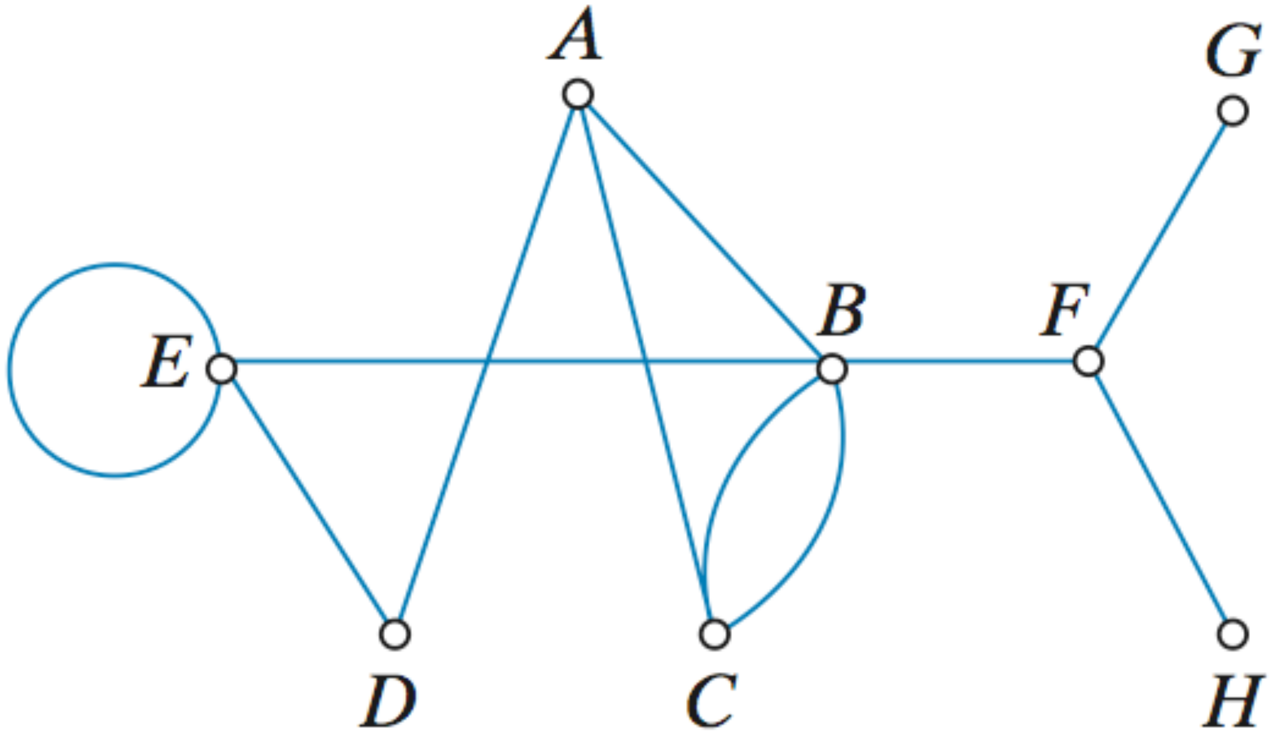
- The *EE* loop is considered to be a circuit of length 1.



Example Connectedness and Bridges

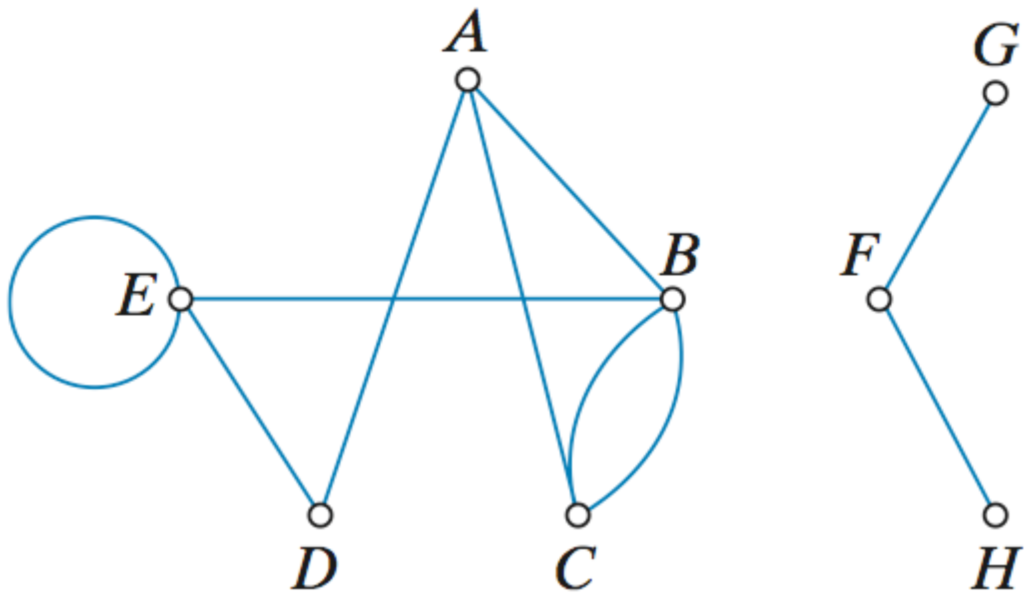
A graph is **connected** if you can get from any vertex to any other vertex along a path. Essentially, this means that the graph is all in one piece.

The graph shown in here is a connected graph.



Example Connectedness and Bridges

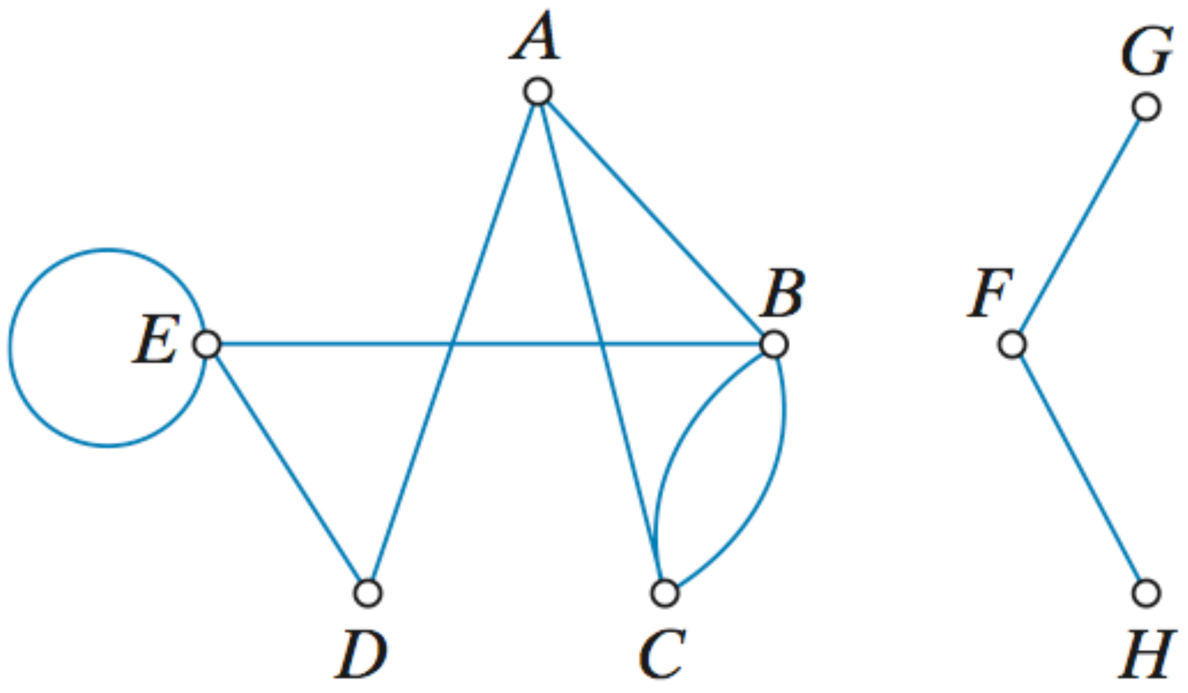
Whereas the graphs shown in here is a **disconnected** graphs.



Example Connectedness and Bridges

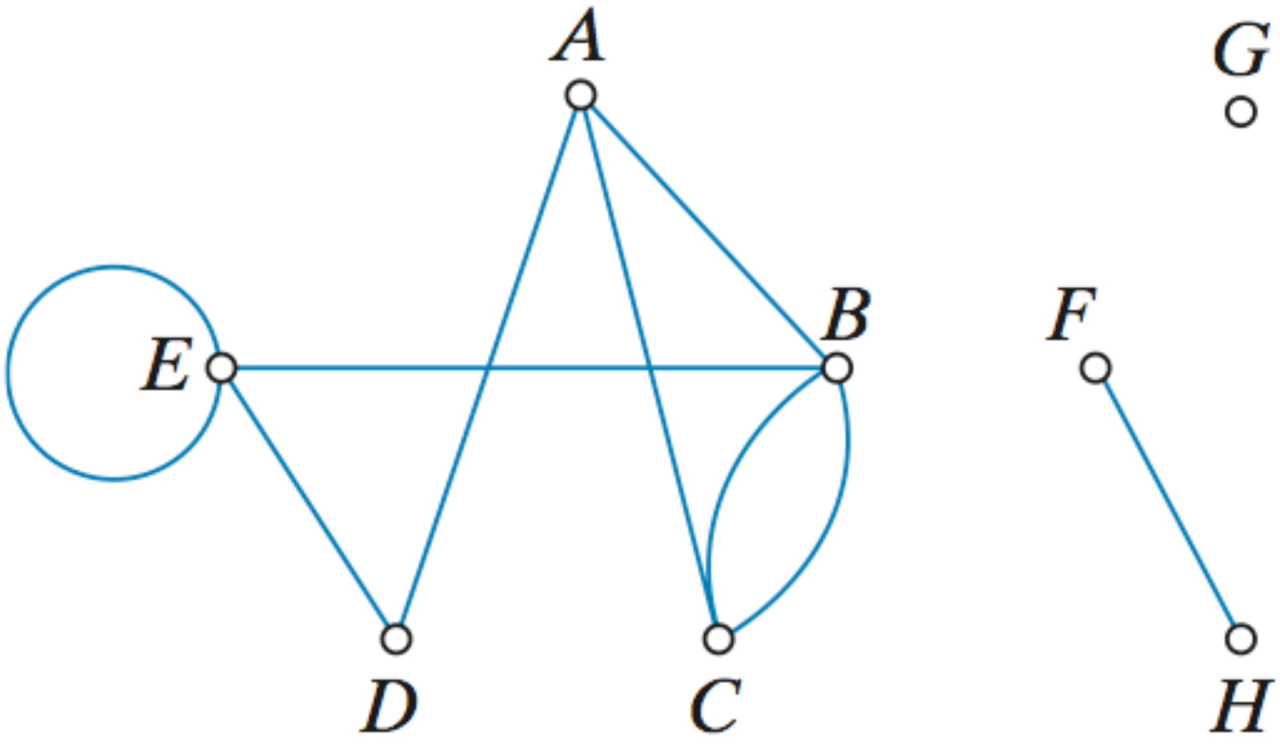
A disconnected graph is made up of separate connected **components**.

The graph on the right is a disconnected graph with two components.



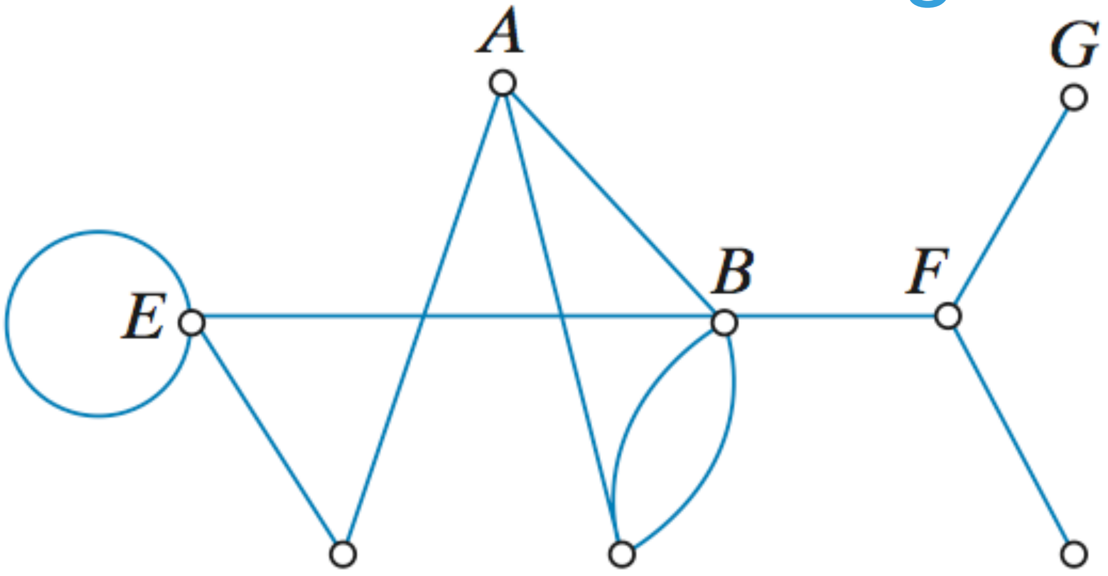
Example Connectedness and Bridges

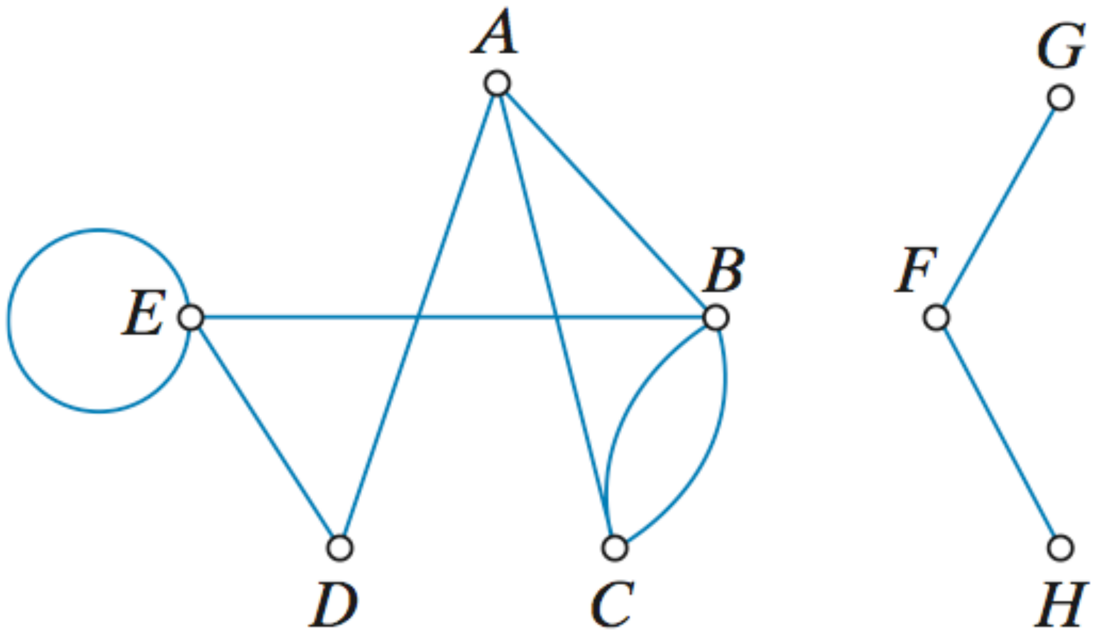
Here we show a disconnected graph with three components (an isolated vertex is a component in and of itself).



Example Connectedness and Bridges

Below is the graph we get when we remove the edge *BF* from the graph above.

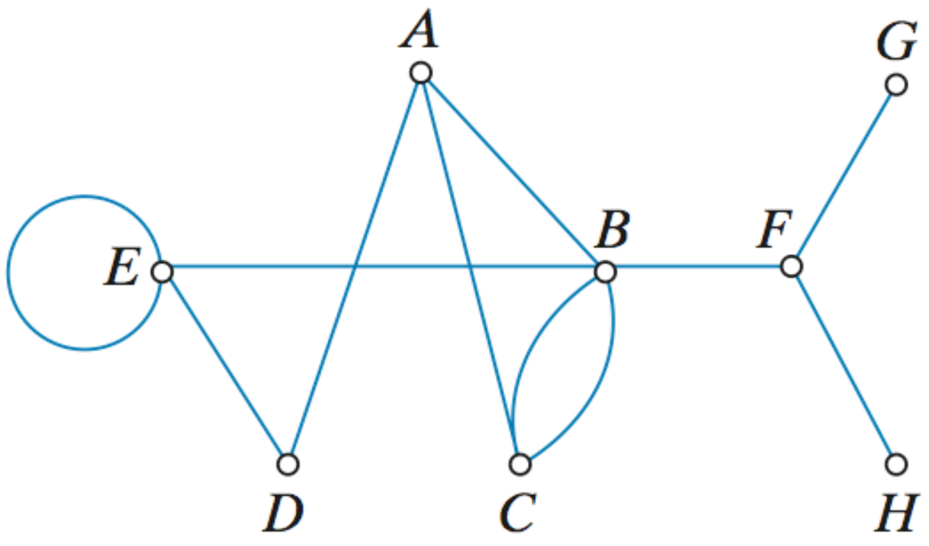




Example Connectedness and Bridges

The removal of just one edge from a connected graph can sometimes make the graph disconnected. An edge whose removal makes a connected graph disconnected is called a bridge.

Thus, we say that *BF* is a bridge in the graph on the previous slide. The graph has two other bridges–
FG and *FH*.



Example   Euler Paths and Euler Circuits

An **Euler path** in a connected graph is a path that travels through *all* the edges of the graph.

Being a path, edges can only be traveled once, so in an Euler path every edge of the graph is traveled *once and only once.*

By definition, only a connected graph can have Euler paths, but, of course, just being connected is not enough.

Example Euler Paths and Euler Circuits

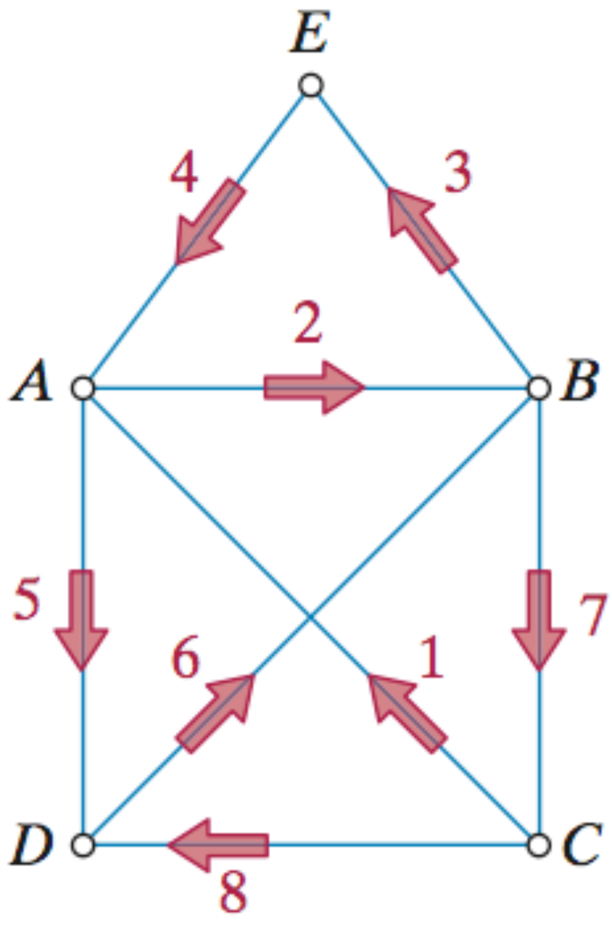
Much like Euler paths, we can also define Euler circuits. An **Euler circuit** is a circuit that travels through every edge of a connected graph. Being a circuit, the trip must end where it started and travel along every edge of the graph once and only once.

A connected graph cannot have both an Euler path and an Euler circuit–it can have one or the other or neither.

Example Euler Paths and Euler Circuits

In the graph shown here, the path C, A, B, E, A, D, B, C, D travels along each of the eight edges of the graph and is therefore an Euler path.

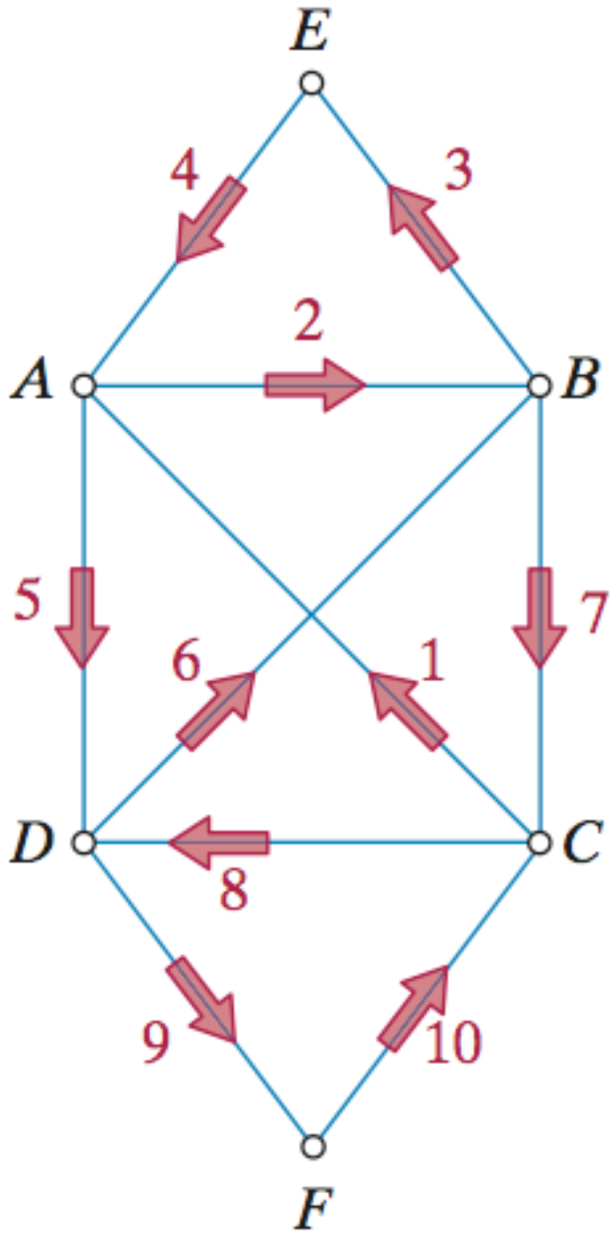
This graph has several other Euler paths– you may want to try to find one that does not start at *C*.



Example Euler Paths and Euler Circuits

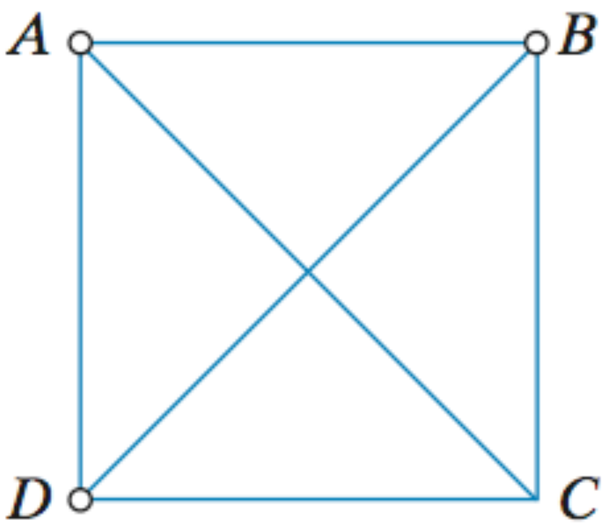
In the graph shown here, the circuit C, A, B, E, A, D, B, C, D, F, C is an Euler circuit (one of many).

You may want to find a different one. (Remember that traveling the edges in the same sequence but using a different starting point is cheating–you are rewriting the same circuit.)



Example Euler Paths and Euler Circuits

The graph shown here has neither an Euler path nor an Euler circuit. (We will learn how to tell that a graph has neither an Euler path nor an Euler circuit later.)



Modeling

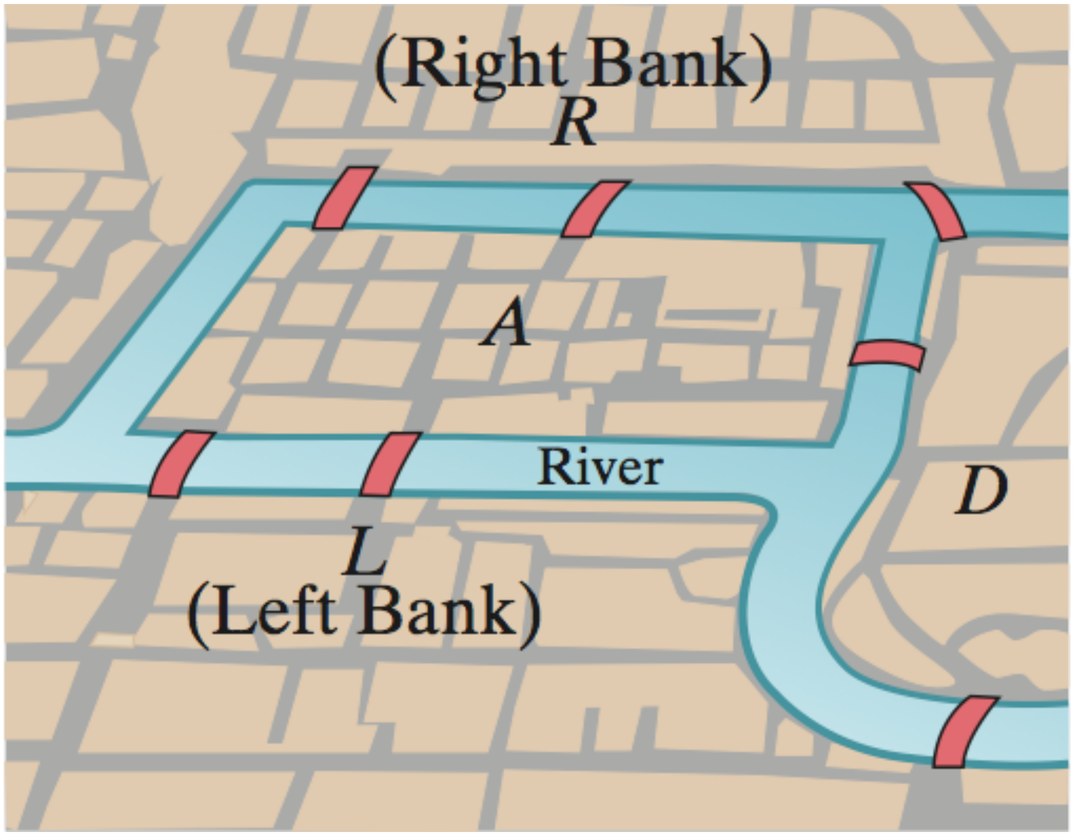
Certain types of problems can be conveniently rephrased as graph problems.

The notion of using a mathematical concept to describe and solve a real-life problem is one of the oldest and grandest traditions in mathematics. It is called *modeling*.

Example: The Seven Bridges of Königsberg

Here is a map of the seven bridges (shown in red) of a city called Königsberg.

Is it possible to take a stroll through the city of Königsberg and cross each of the seven bridges once and only once.

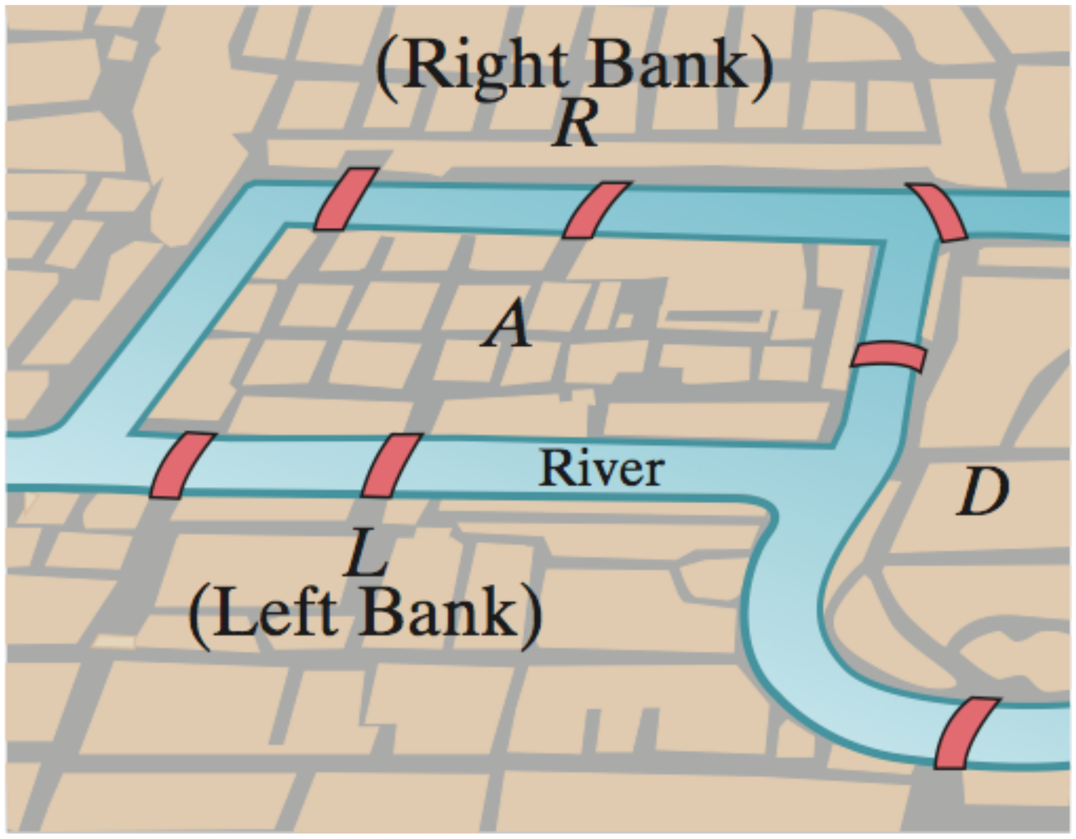


Example: The Seven Bridges of Königsberg

This map is not entirely accurate–the drawing is not to scale and the exact positions and

angles of some of the bridges are changed.

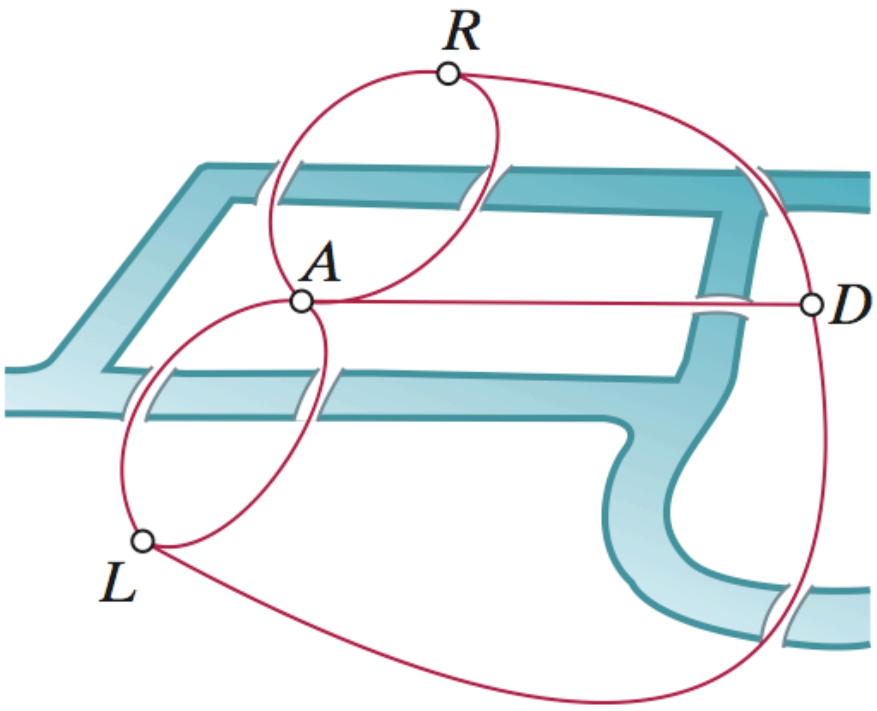
Does it matter?



Example: The Seven Bridges of Königsberg

The only thing that truly matters to the solution of this problem is the relationship between land masses (islands and banks) and bridges. Which land masses are

connected to each other and by how many bridges? This information is captured by the red edges.

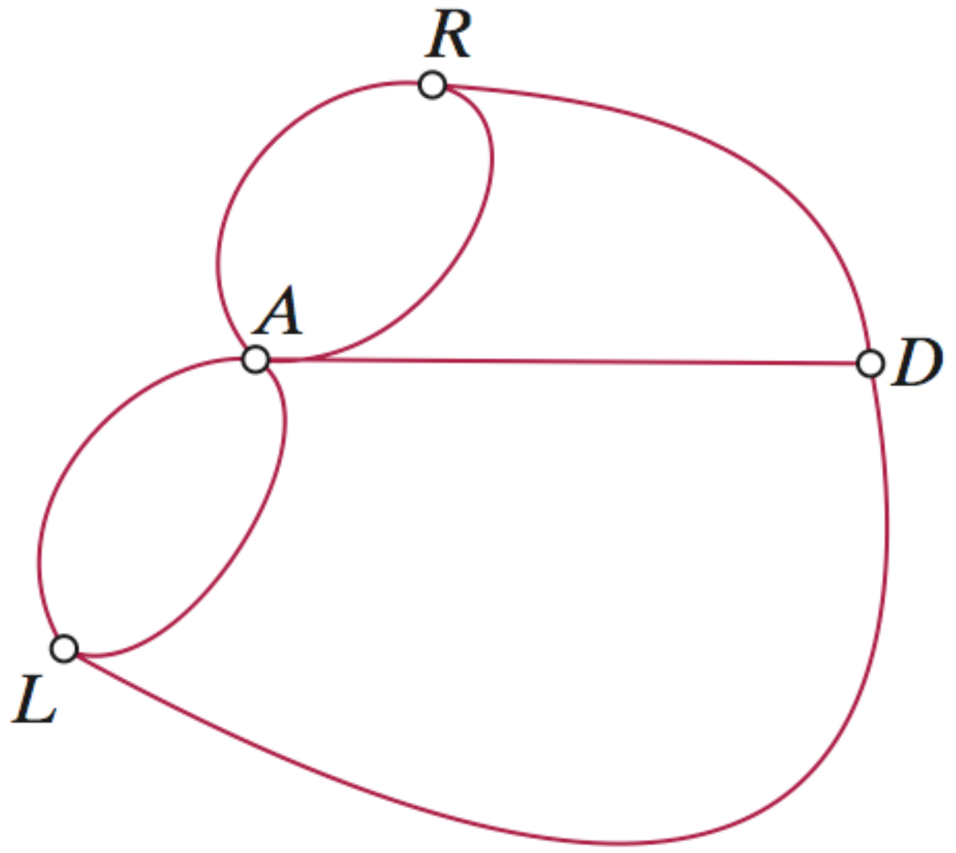


Example: The Seven Bridges of Königsberg

Thus, when we strip the map of all its superfluous information, we end up with the

graph model shown here.

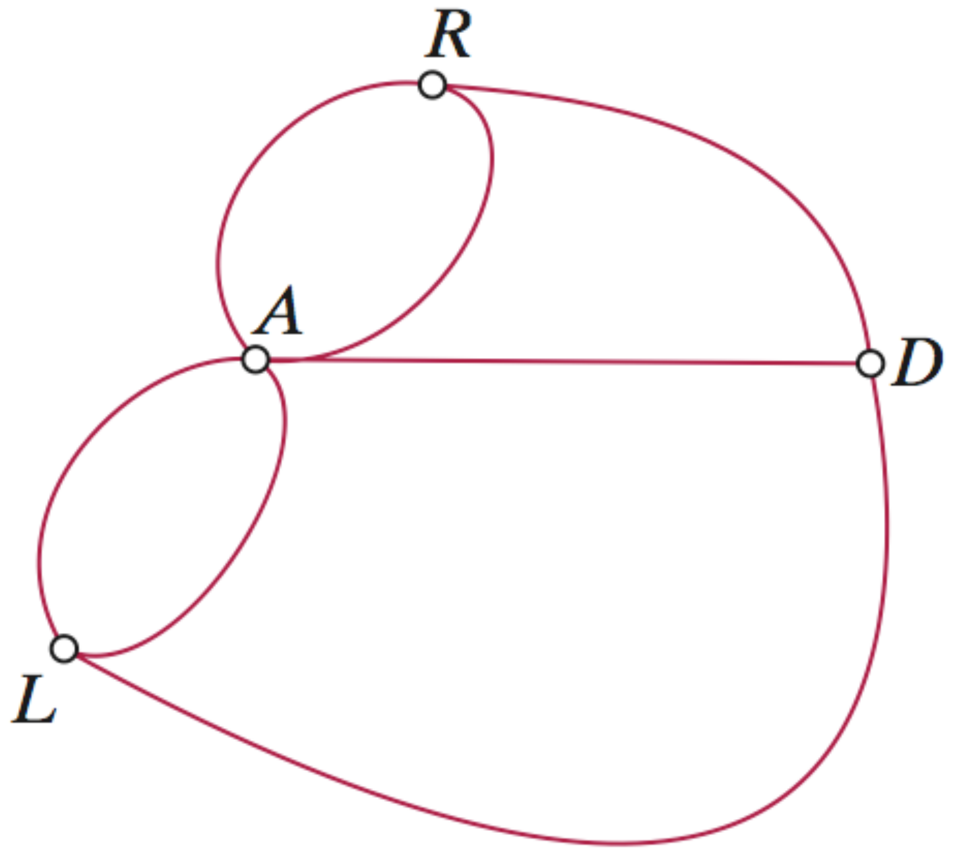
The four vertices of the graph represent each of the four land masses; the edges represent the seven bridges.



Example: The Seven Bridges of Königsberg

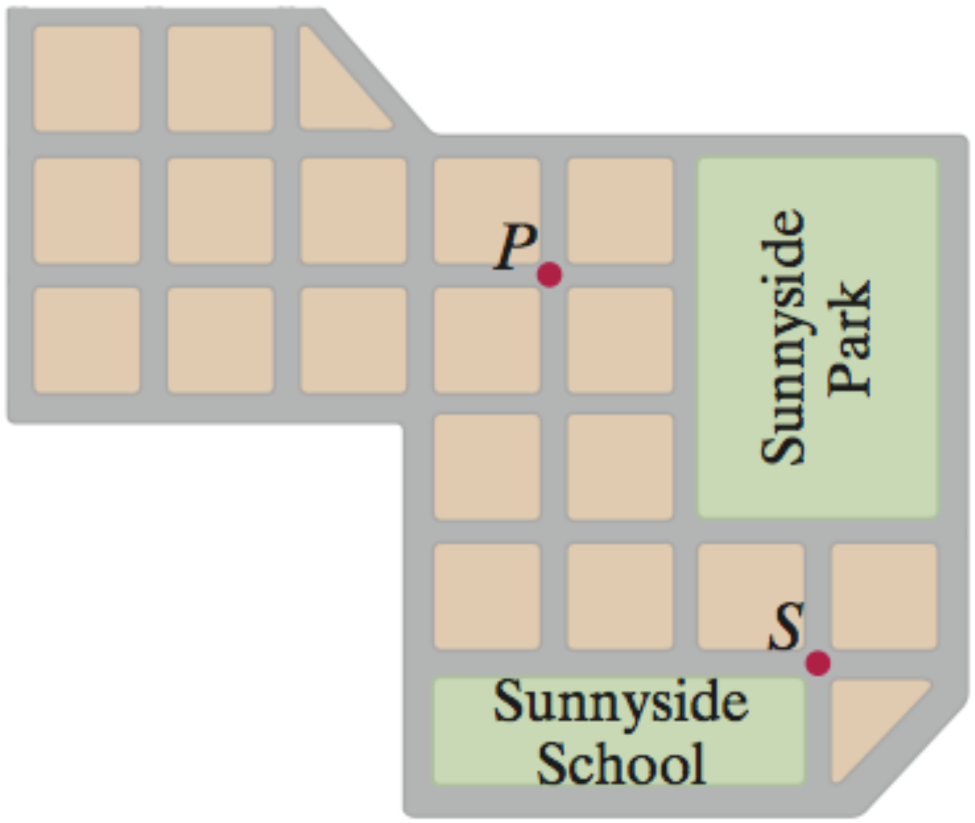
In this graph an Euler circuit would represent a stroll around the town that crosses each

bridge once and ends back at the starting point; an Euler path would represent a stroll that crosses each bridge once but does not return to the starting point.



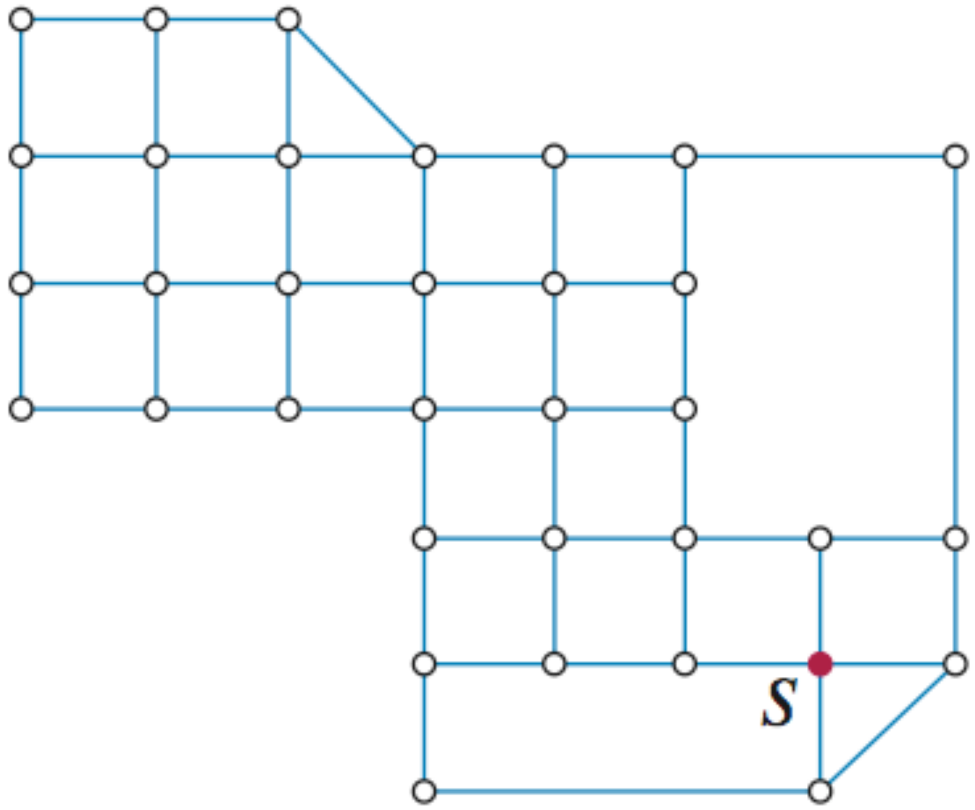
Example Walking the 'Hood: Part 2

Recall the problem of the security guard who needs to walk the streets of the Sunnyside neighborhood.



Example Walking the ‘Hood: Part 2

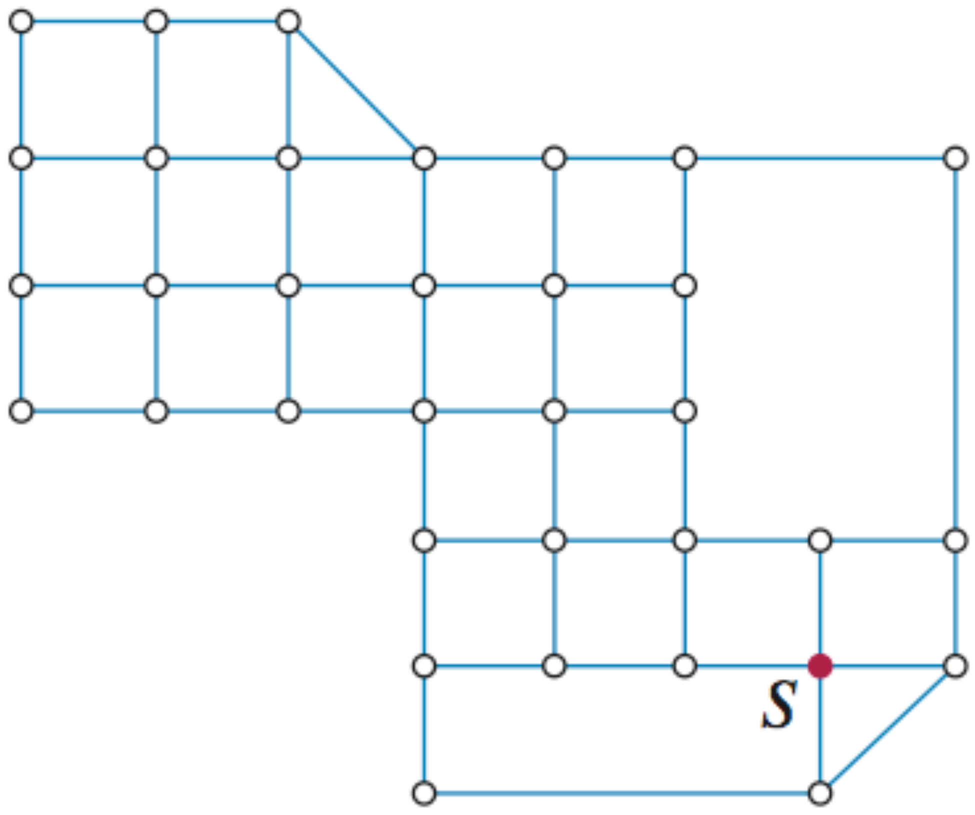
The graph– where each edge represents a block of the neighborhood and each vertex an intersection–is a graph model of this problem.



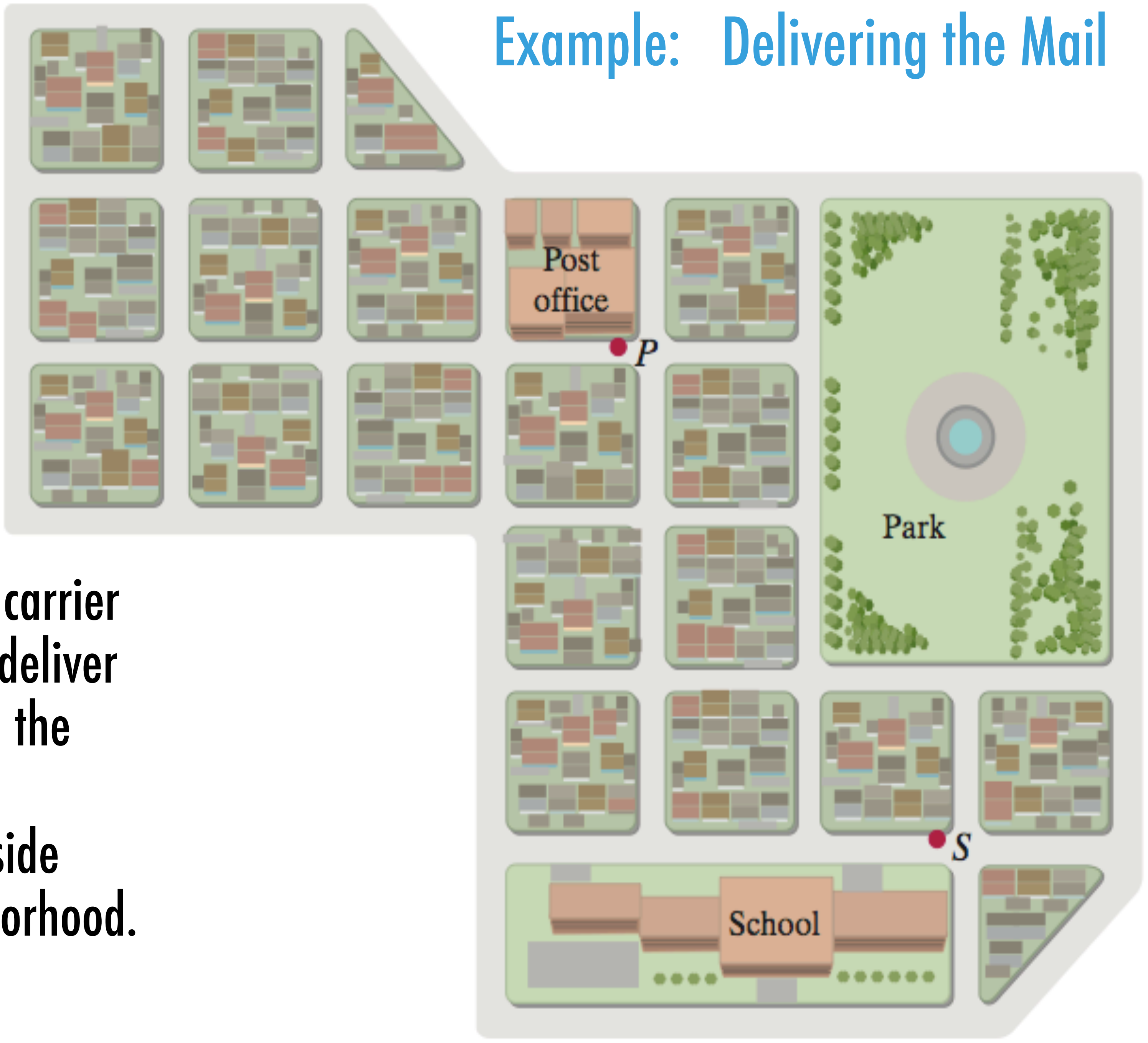
Example Walking the 'Hood: Part 2

Does the graph have an Euler circuit? An Euler path? Neither?

(These are relevant questions that we will learn how to answer in the next section.)



Example: Delivering the Mail



A mail carrier has to deliver mail in the same Sunnyside neighborhood.

Example Delivering the Mail

The difference between the mail carrier’s route and the security guard’s route is that the mail carrier must make two passes through blocks with houses on both sides of the street and only one pass through blocks with houses on only one side of the street;

and where there are no homes on either side of the street, the mail carrier does not have to walk at all.

Example   Delivering the Mail

Recall that unlike the security guard, the mail carrier must make two passes through every block that has homes on both sides of the street(she has to physically place the mail in the mailboxes), must make one pass through blocks that have homes on only one side of the street, and does not have to walk along blocks where there are no houses.

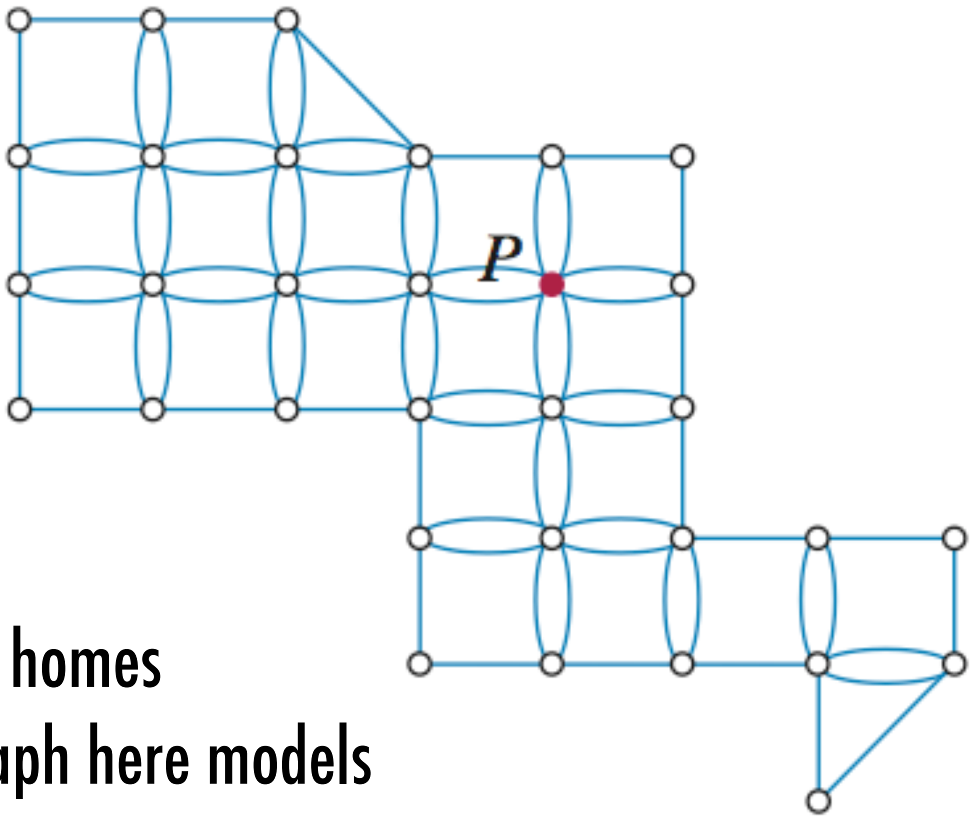
Example Delivering the Mail

In this situation an appropriate graph model requires two edges on the blocks that have homes on both.

sides of the street, one edge for the blocks that have homes on only one side of the street,

and no edges for blocks having no homes

on either side of the street. The graph here models this situation.



Euler’ s Theroems

In this section we are going to develop the basic theory that will allow us to determine if a graph has an Euler circuit, an Euler path, or neither.

This is important because, as we saw in the previous section, what are Euler circuit or Euler path questions in theory are real-life routing questions in practice.

The three theorems we are going to see next (all thanks to Euler) are surprisingly simple and yet tremendously useful.

EULER' S CIRCUIT THEOREM

If a graph is *connected and every vertex is even, then it has an Euler circuit* (at least one, usually more).

- If a graph *has any odd vertices, then it does not have an Euler circuit.*

How to Use the Theorem

Here is how we can use Euler's circuit theorem.

First we make sure the graph is connected. (If it isn't, then no matter what else, an Euler circuit is impossible.)

If the graph is connected, then we start checking the degrees of the vertices, one by one.

As soon as we hit an odd vertex, we know that an Euler circuit is out of the question. If there are no odd vertices, then we know that the answer is yes–the graph does have an Euler circuit!

Illustration using the Theorem

This graph cannot have an Euler circuit for the simple reason that it is disconnected.

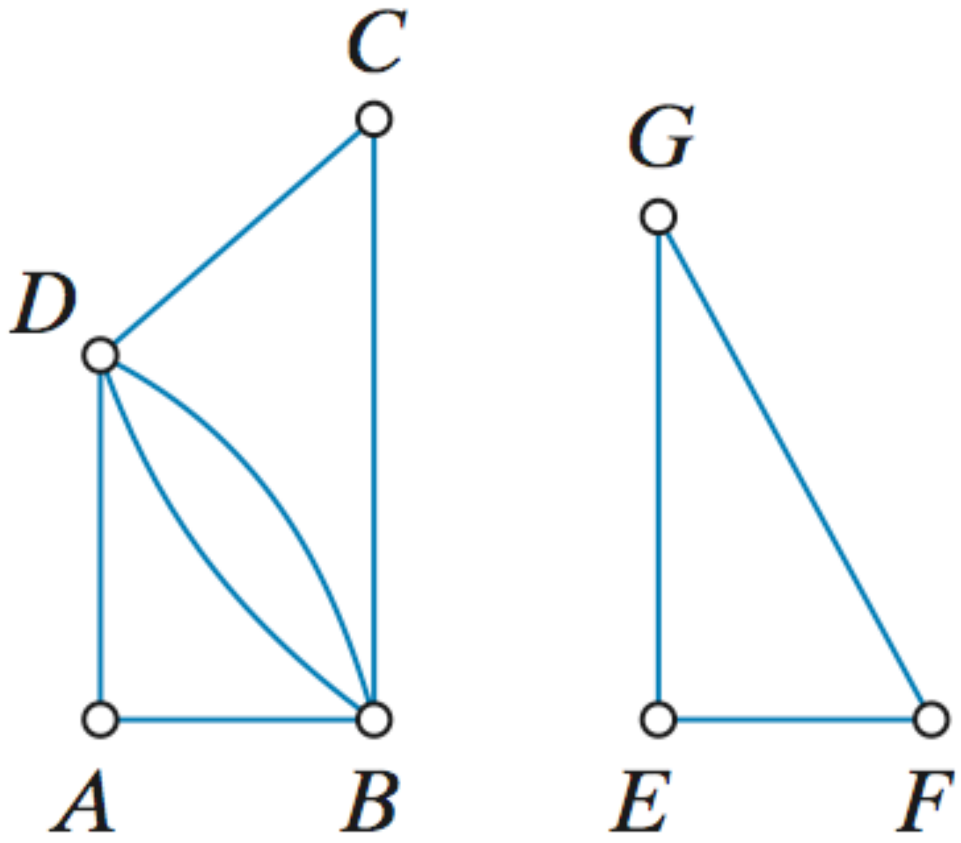


Illustration using the Theorem

This graph is connected, but we can quickly spot odd vertices (*C* is one of them; there are others). Thus graph has no Euler circuits.

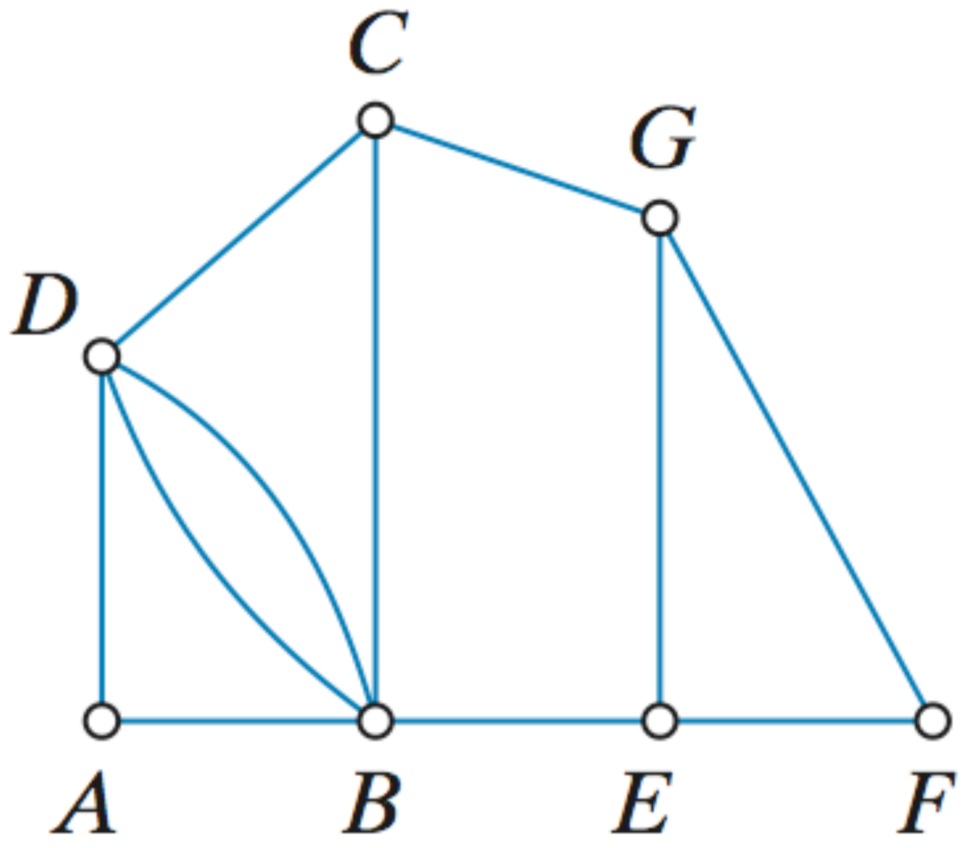
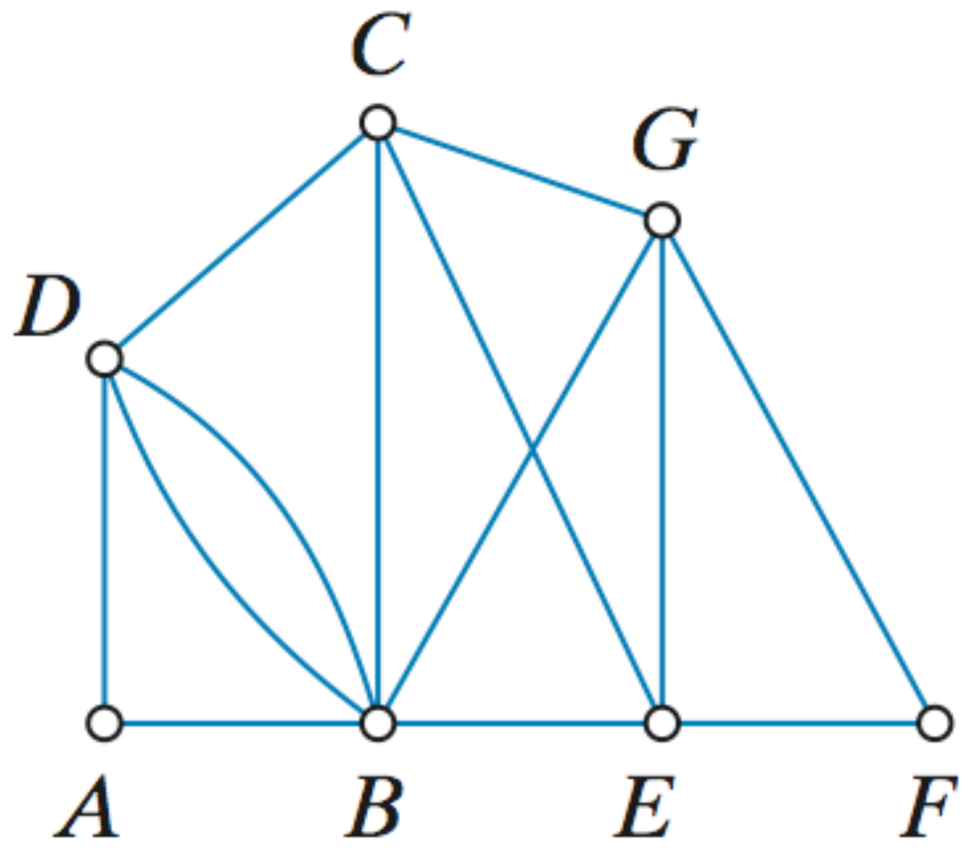


Illustration using the Theorem

This graph is connected and all the vertices are even. Thus this graph does have Euler circuits.



Summary of the Theorem

The basic idea behind Euler’s circuit theorem is that as we travel along an Euler circuit, every time we go through a vertex we use up two different edges at that vertex– one to come in and one to go out.

We can keep doing this as long as the vertices are even. A single odd vertex means that at some point we are going to come into it and not be able to get out.

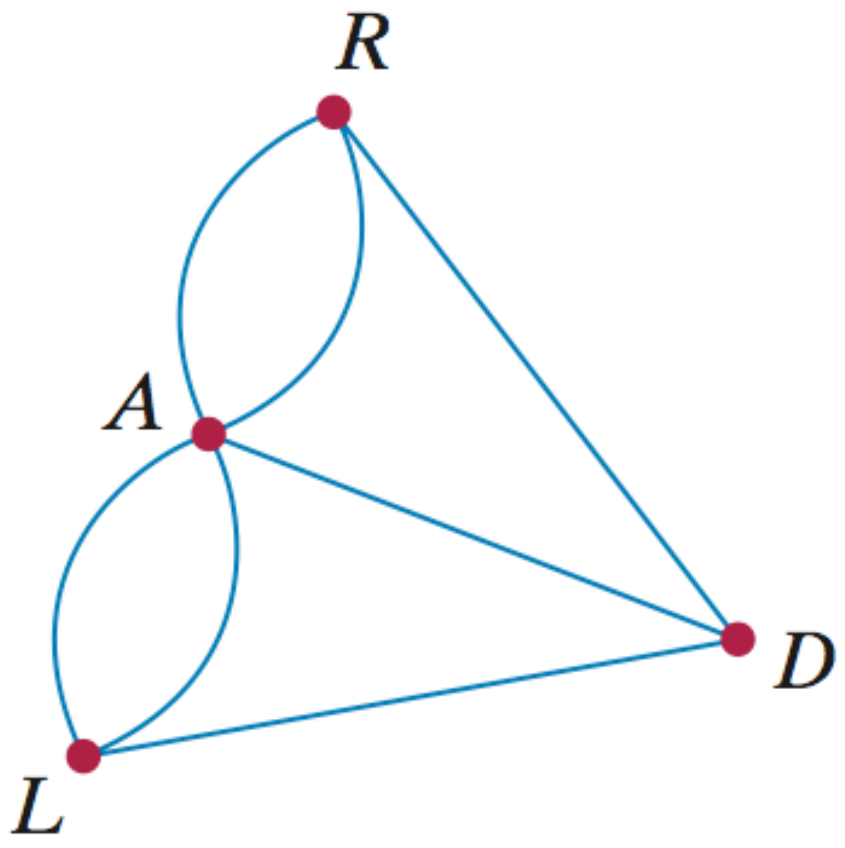
EULER' S PATH THEOREM

- If a graph is *connected and has exactly two odd vertices, then it has an Euler path* (at least one, usually more). *Any such path must start at one of the odd vertices and end at the other one*.
- *If a graph has more than two odd vertices, then it cannot have an Euler path.*

Example The Seven Bridges of Königsberg

Back to the Königsberg bridges problem. Previously, we were able to show that the layout of the

bridges in the old city can be modeled by the graph shown here.



This graph has *four odd vertices; thus, neither an Euler circuit nor an Euler path can exist.*

Example The Seven Bridges of Königsberg

We now have an unequivocal answer to the puzzle: There is no possible way anyone can walk across all the bridges without having to recross some of them!

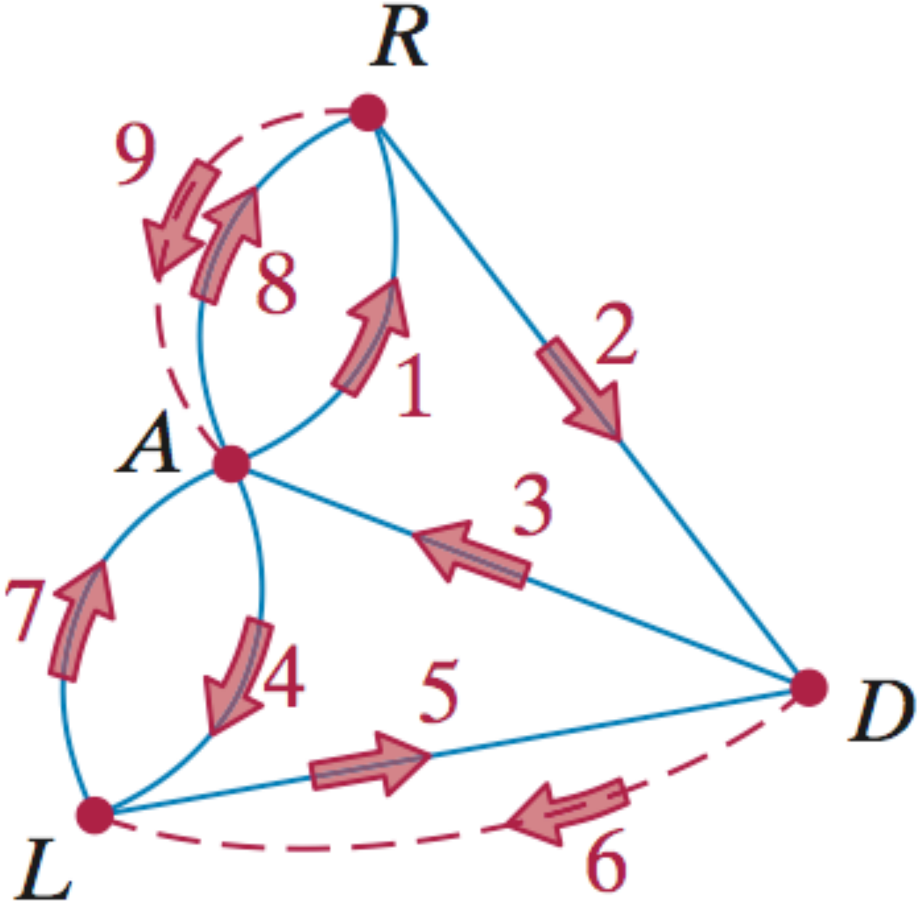
How many bridges will need to be recrossed?

It depends. If we want to start and end in the same place, we must recross at least two of the bridges.

Example The Seven Bridges of Königsberg

One of the many possible routes is shown in here.

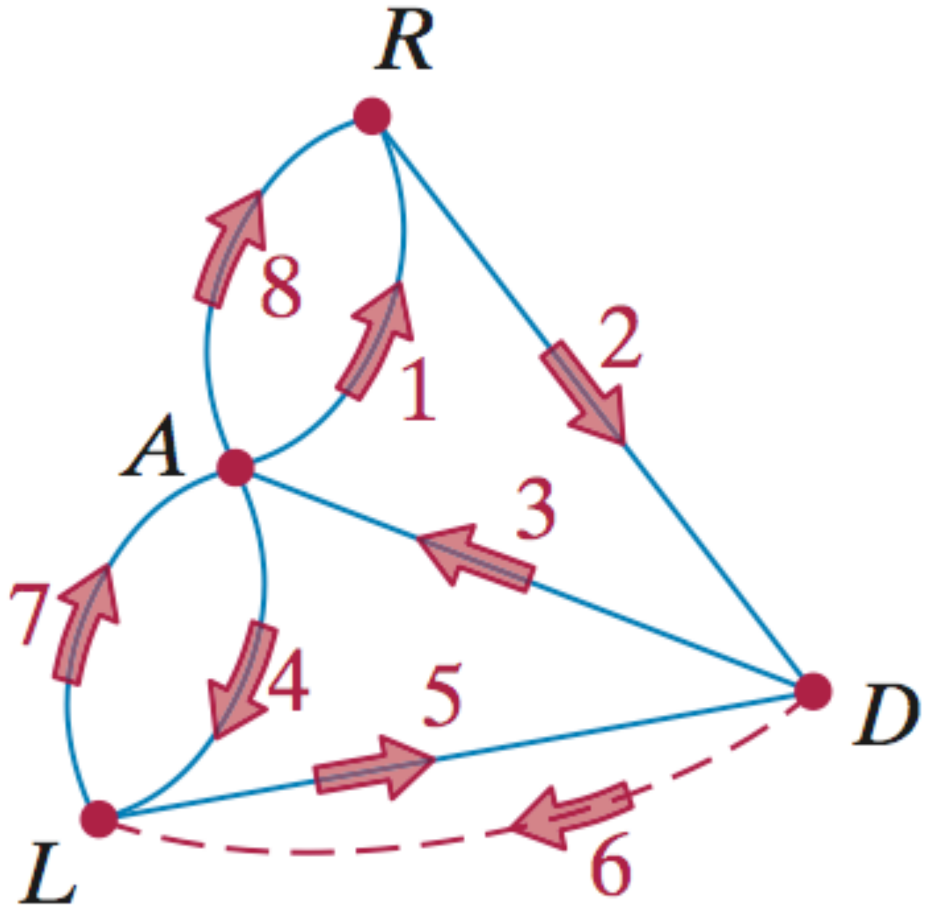
In this route the bridge connecting *L* and *D* is crossed twice, and so is one of the two bridges connecting *A* and *R*.



Example The Seven Bridges of Königsberg

If we are allowed to start and end in different places, we can do it by recrossing just one of the bridges.

One possible route is starting at *A*, crossing bridge *LD* twice, and ending at *R*.



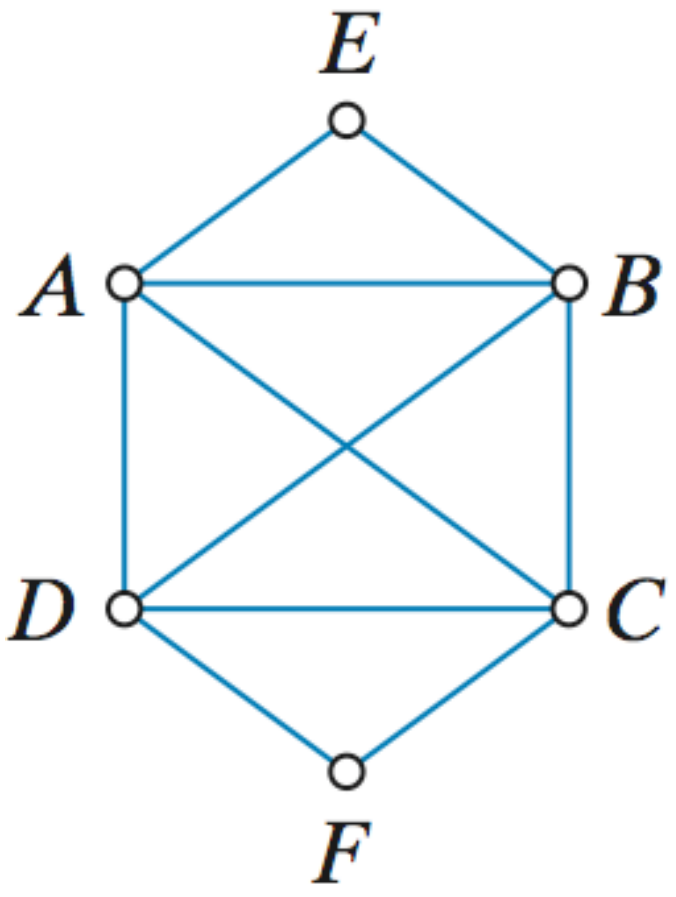
Example Child’s Play

Here we recall a graph from the child’s play example we discussed earlier.

This graph is connected, and

the vertices are all even.

By Euler’s circuit theorem we know that the graph has an Euler circuit, which implies that the original line drawing has a closed unicursal tracing.

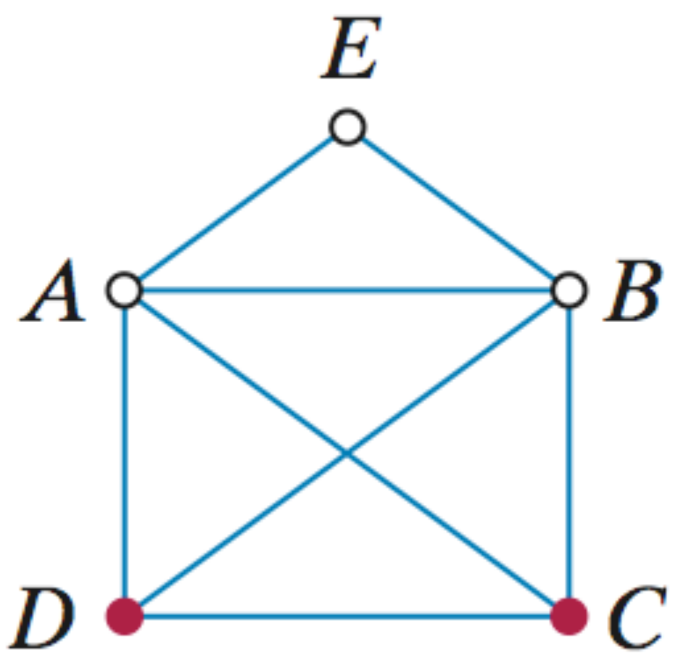


Example Child’s Play

Here is another graph from the child’s play example. This graph is connected and

has exactly two odd vertices (*C* and *D*).

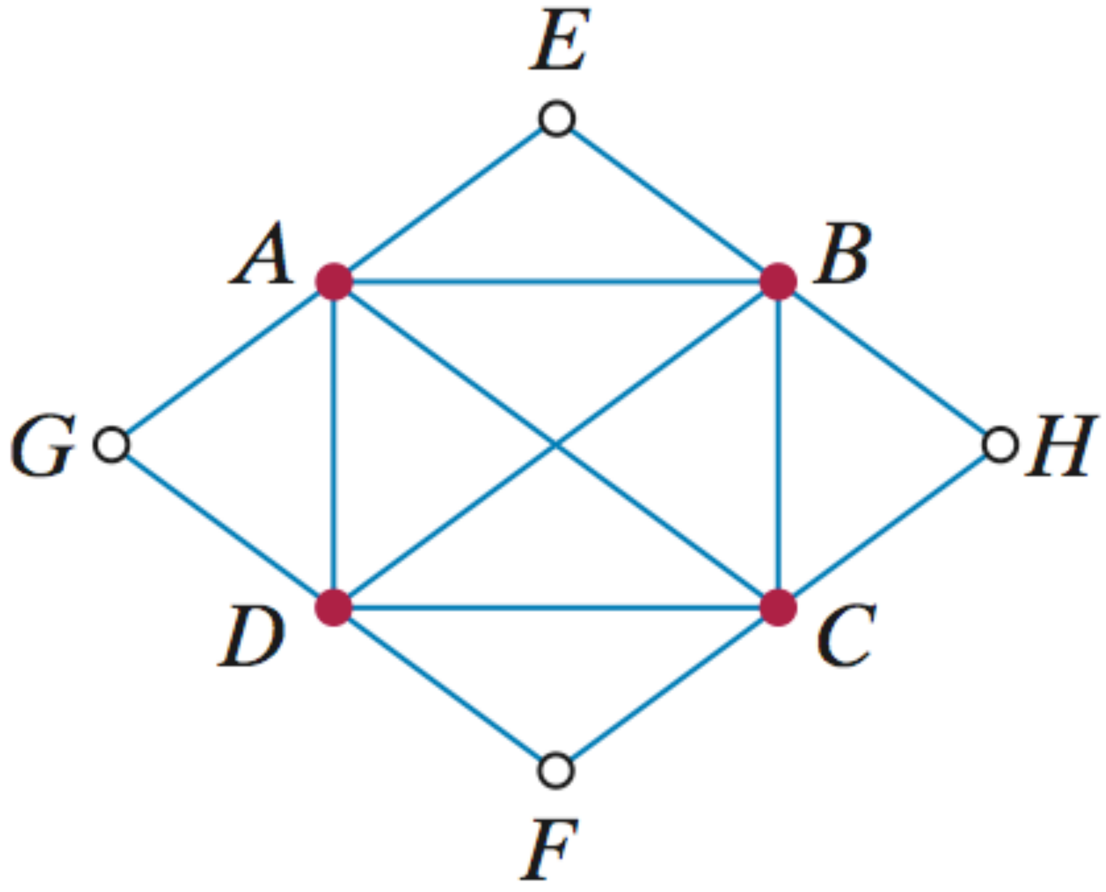
By Euler’s path theorem,the graph has an Euler path (open unicursal tracing). Moreover, we now know that the path has to start at *C* and end at *D*, or vice versa.



Example Child’s Play

This graph has four odd vertices (*A*, *B*, *C*, and *D*),

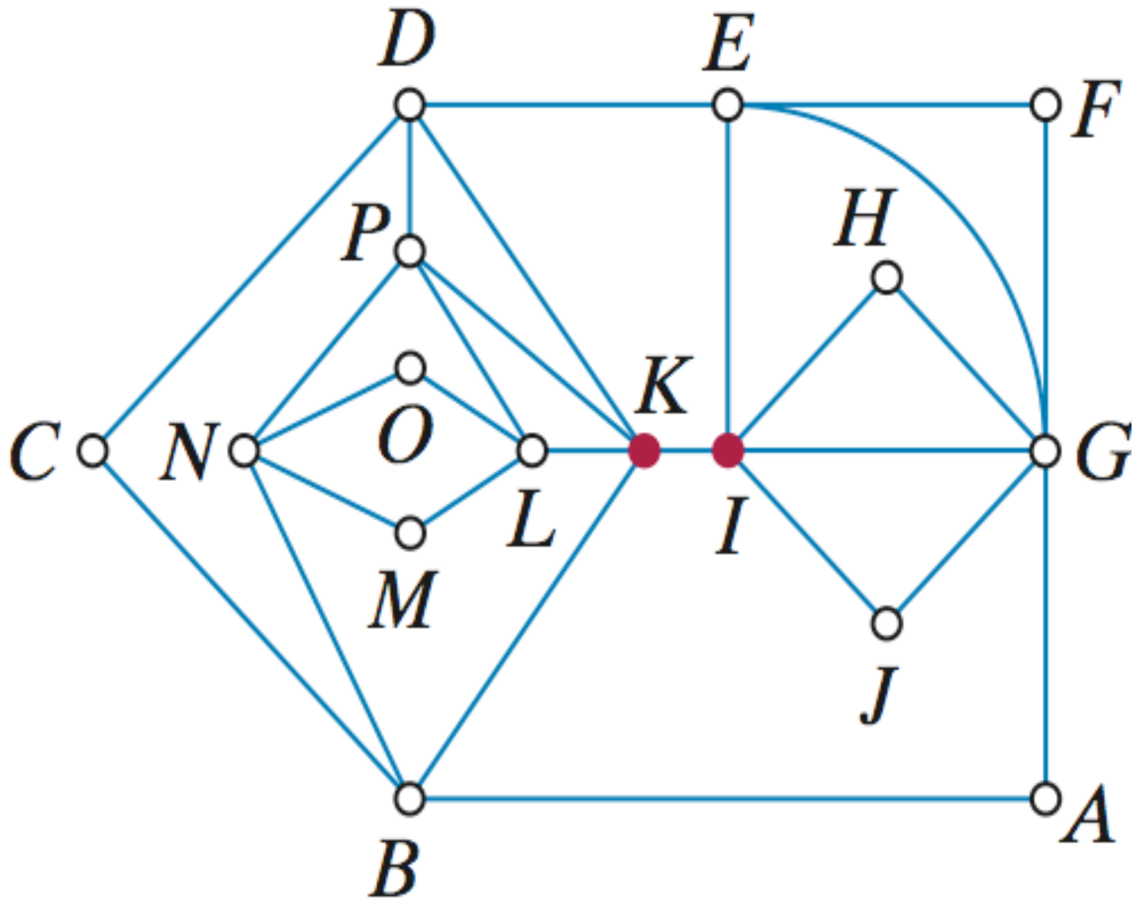
so it has neither an Euler path nor an Euler circuit.



Example Child’s Play

The full power of Euler’s theorems is best appreciated when the graphs get bigger.

This graph is not extremely big, but we can no longer “eyeball” an Euler circuit or path.



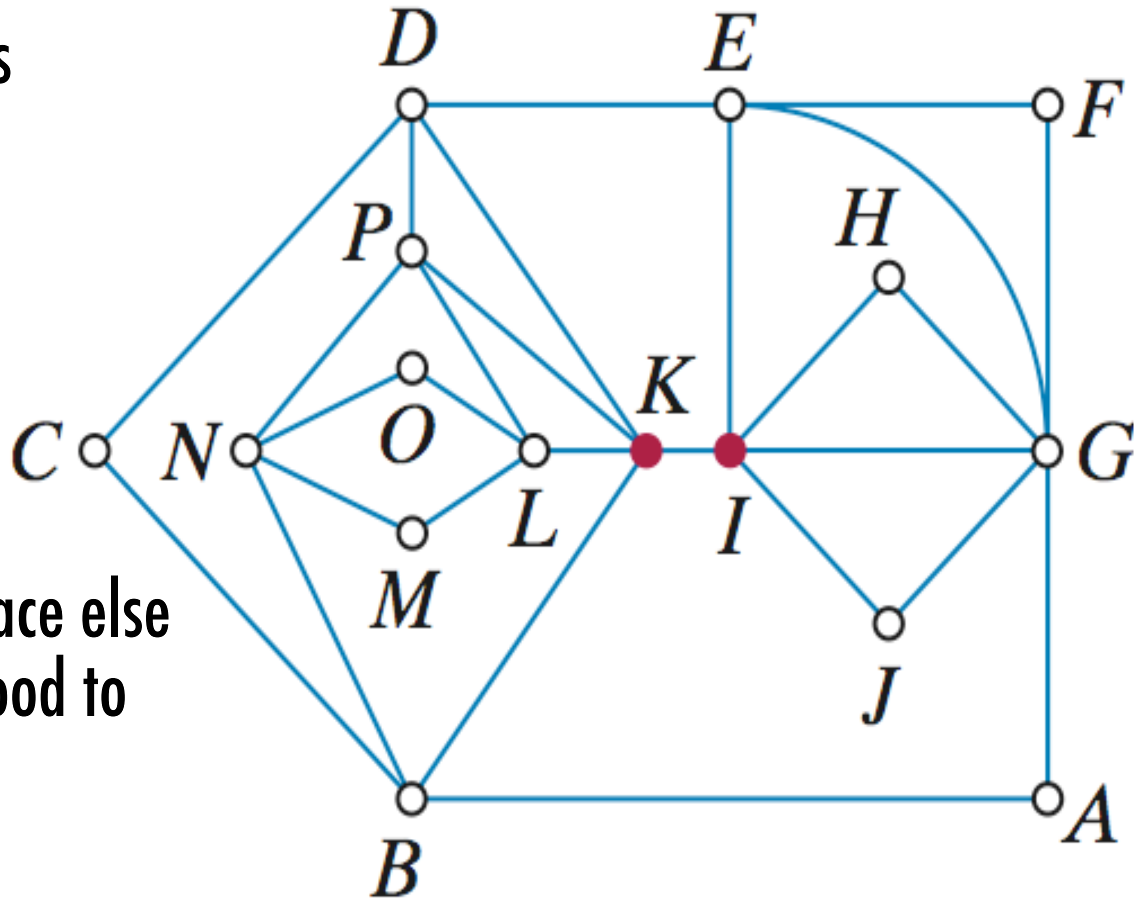
Example Child's Play

On the other hand, a quick check of the degrees of the vertices shows that *K* and *I* are odd vertices and all the rest are even.

An open unicursal tracing is possible as

long as we start it at *K* or *I* (and end it at the

other one). Starting anyplace else will lead to a dead end. Good to know!



Euler' s Third Theorem

Euler's circuit theorem deals with graphs with zero odd vertices, whereas Euler's Path Theorem deals with graphs with two or more odd vertices.

The only scenario not covered by the two theorems is that of graphs with just one odd vertex.

Euler's third theorem rules out this possibility–a graph cannot have just one odd vertex. In fact, Euler's third theorem says much more.

EULER' S SUM OF DEGREES THEOREM

- The sum of the degrees of all the vertices of a graph equals twice the number of edges (and therefore is an even number).
- A graph always has an even number of *odd* vertices.

Euler's Sum of Degrees Theorem

Euler's sum of degrees theorem is based on the following basic observation:

Take any edge–let's call it *XY*. The edge contributes once to the degree of vertex *X* and once to the degree of vertex *Y*, so, in all, that edge makes a total contribution of 2 to the sum of the degrees.

Thus, when the degrees of all the vertices of a graph are added, the total is twice the number of edges.

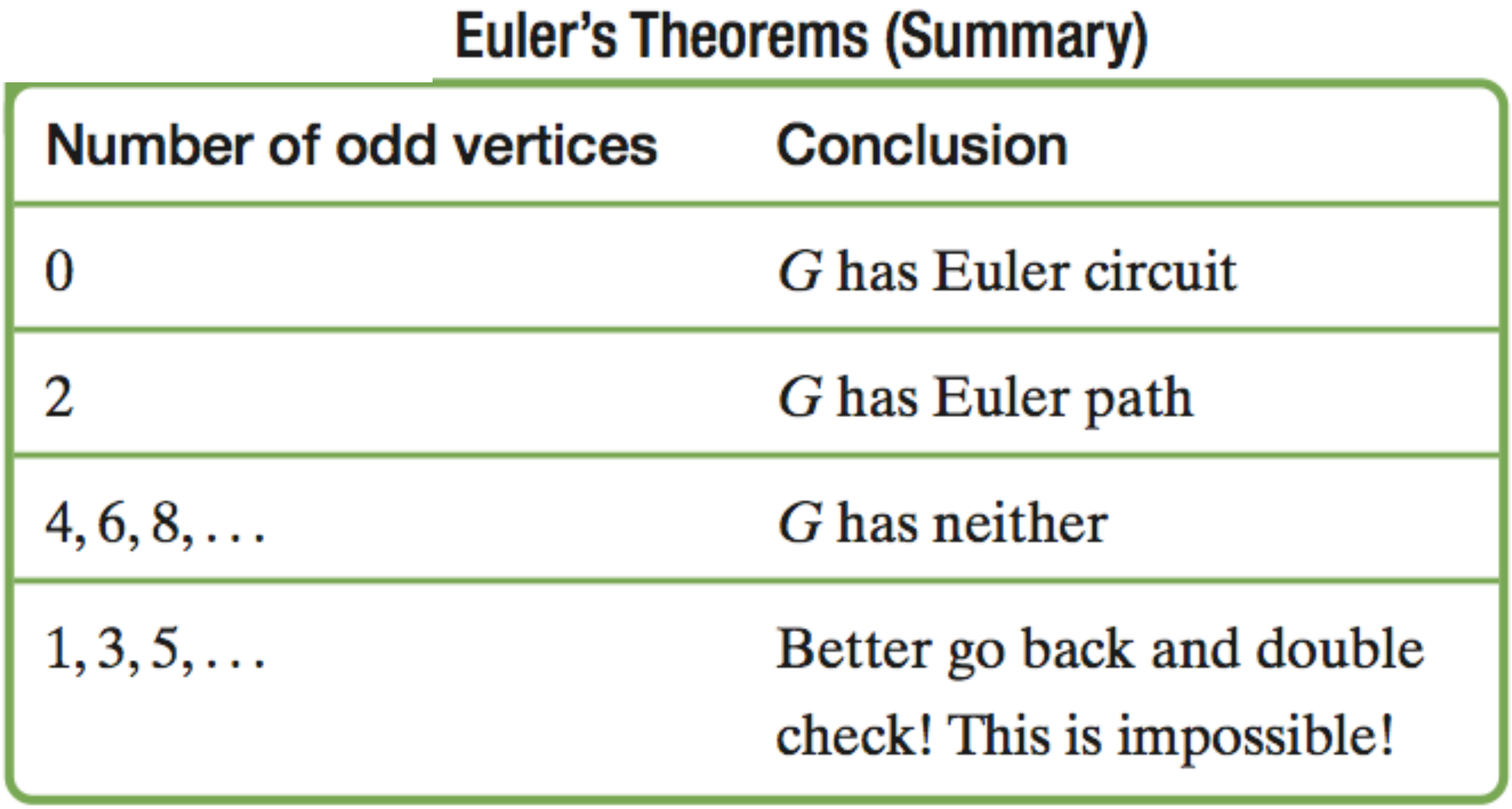
Euler’ s Sum of Degrees Theorem

Since the total sum is an even number, it is impossible to have just one odd vertex, or three odd vertices, or five odd vertices, and so on.

To put it in a slightly different way, the odd vertices of a graph always come in twos.

In the next slide, we summarize Euler’s three theorems.

Euler's Sum of Degrees Theorem



Euler's Theorems (Summary)

Number of odd vertices	Conclusion
0	G has Euler circuit
2	G has Euler path
$4, 6, 8, \ldots$	G has neither
$1, 3, 5, \ldots$	Better go back and double check! This is impossible!

Algorithms

When the graph has an Euler circuit or path, how do we find it?

For small graphs, simple trial-and-error usually works fine, but real-life applications sometimes involve graphs with hundreds, or even thousands, of vertices.

In these cases a trial-and-error approach is out of the question, and what is needed is a systematic strategy that tells us how to create an Euler circuit or path. In other words, we need an algorithm.

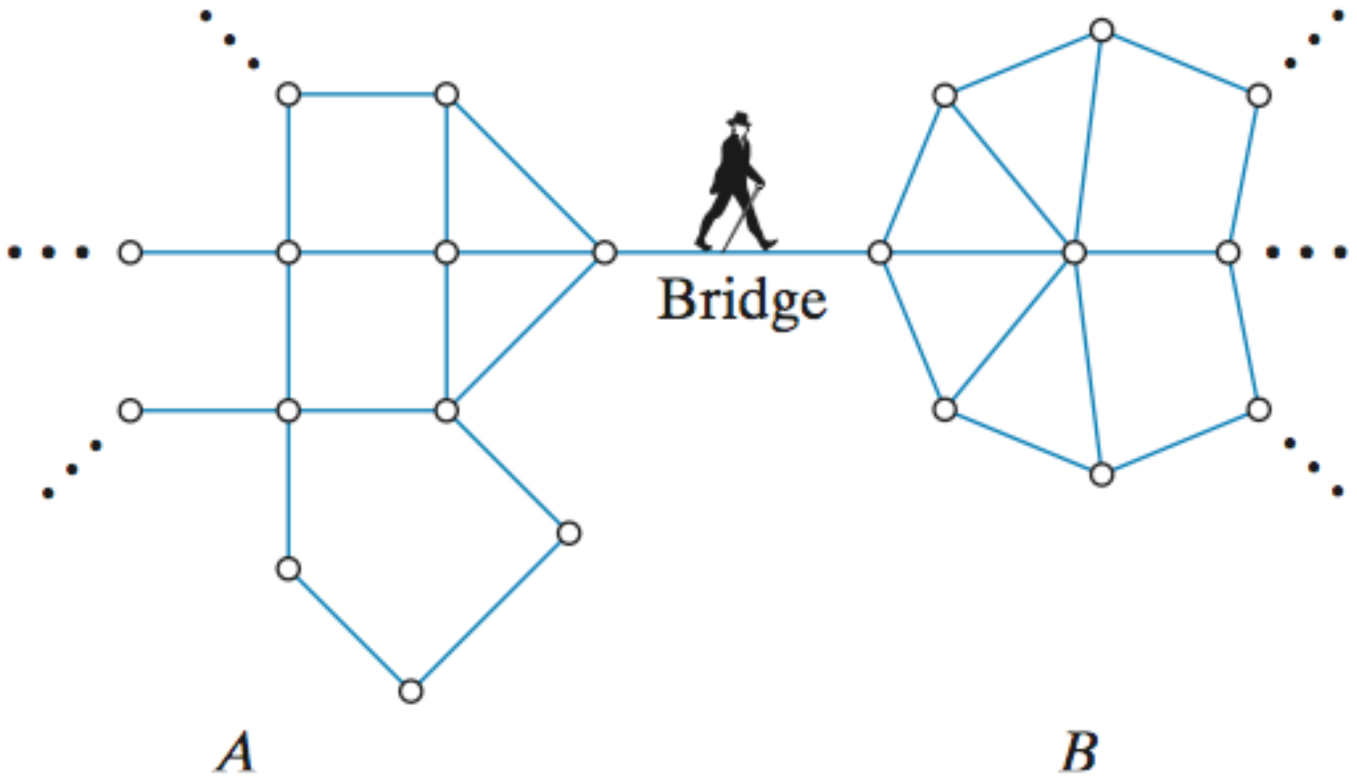
Fleury's Algorithm

We will now turn our attention to an algorithm that finds an *Euler circuit* or an *Euler path* in a connected graph. Technically speaking, these are two separate algorithms, but in essence they are identical, so they can be described as one.

The idea behind Fleury's algorithm can be paraphrased by that old piece of folk wisdom: *Don't burn your bridges behind you.*

Fleury's Algorithm

In graph theory the word bridge has a very specific meaning–*it is the only edge connecting two separate sections* (call them *A* and *B*) of a graph.



Fleury' s Algorithm

Thus, Fleury's algorithm is based on a simple principle:

To find an Euler circuit or an Euler path, *bridges are the last edges you want to cross.*

Our concerns lie only on how we are going to get around the *yet-to-be-traveled* part of the graph. Thus, when we talk about bridges that we want to leave as a last resort, we are really referring to *bridges of the to-be-traveled part of the graph.*

FLEURY’S ALGORITHM FOR FINDING AN EULER CIRCUIT (PATH)

- **Preliminaries.** Make sure that the graph is connected and either (1) has no odd vertices (circuit) or (2) has just two odd vertices (path).
- **Start.** Choose a starting vertex. [In case (1) this can be any vertex; in case (2) it must be one of the two odd vertices.]

FLEURY’ S ALGORITHM FOR FINDING AN EULER CIRCUIT (PATH)

- **Intermediate steps.** At each step, if you have a choice, don’t choose a bridge of the yet-to-be-traveled part of the graph. However, if you have only one choice, take it.
- **End.** When you can’t travel any more, the circuit (path) is complete. [In case (1) you will be back at the starting vertex; in case (2) you will end at the other odd vertex.]

Fleury’ s Algorithm Bookkeeping

In implementing Fleury’s algorithm it is critical to separate the past (the part of the graph that has already been traveled) from the future (the part of the graph that still needs to be traveled).

While there are many different ways to accomplish this (you are certainly encouraged to come up with one of your own), a fairly reliable way goes like this: Start with two copies of the graph.

Copy 1 is to keep track of the “future”; copy 2 is to keep track of the “past.”

Fleury' s Algorithm Bookkeeping

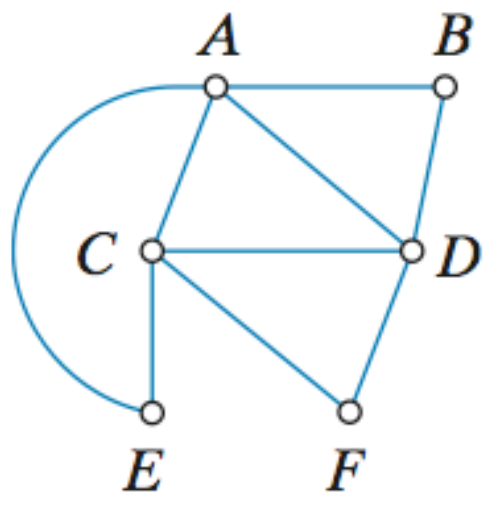
Every time you travel along an edge, erase the edge from copy 1, but mark it (say in red) and label it with the appropriate number on copy 2.

As you move forward, copy 1 gets smaller and copy 2 gets redder. At the end, copy 1 has disappeared; copy 2 shows the actual Euler circuit or path.

Example Implementing Fleury's Algorithm

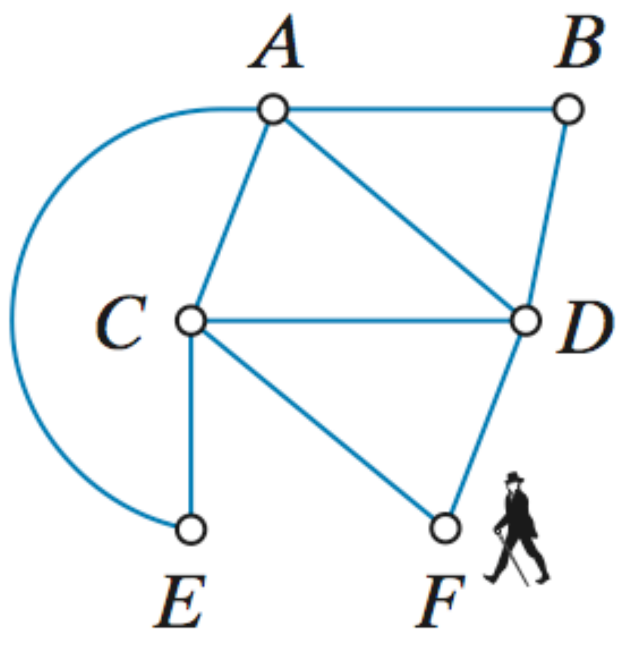
This graph is a very simple graph – it would be easier to find an Euler circuit just by trial-and-error than by using Fleury's algorithm. Nonetheless, we will do it using Fleury's algorithm.

The real purpose of the example is to see the algorithm at work.

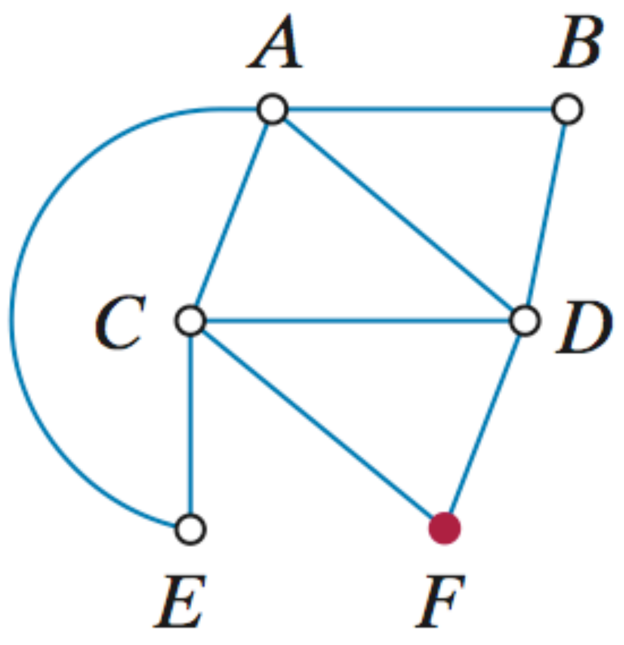


Example Implementing Fleury’s Algorithm

Start: We can pick any starting point we want. Let’s say we start at *F*.



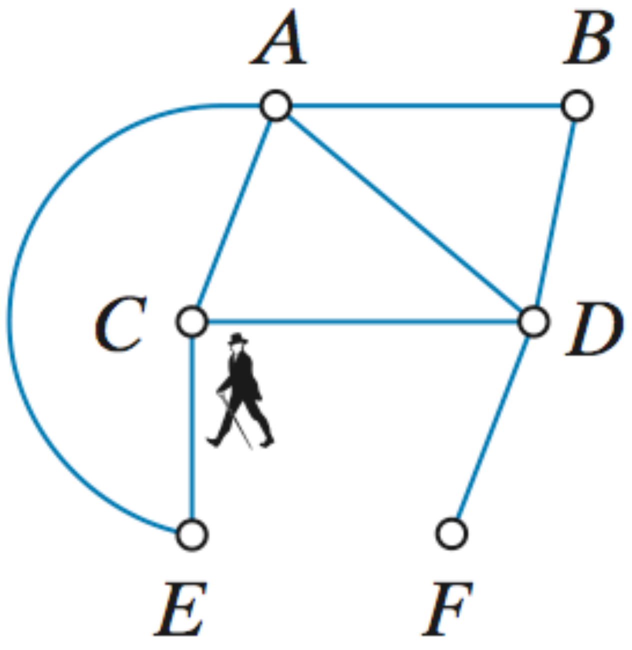
Copy 1



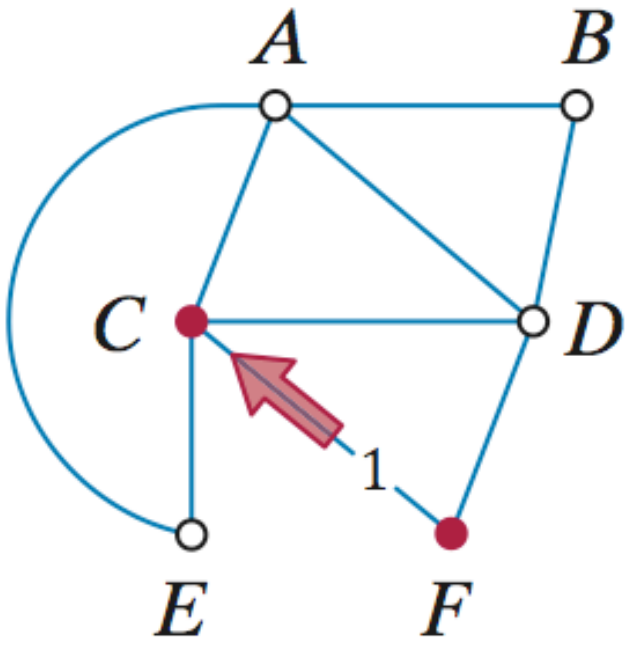
Copy 2

Example Implementing Fleury' s Algorithm

Step 1: Travel from *F* to *C*. (Could have also gone from *F* to *D*.)



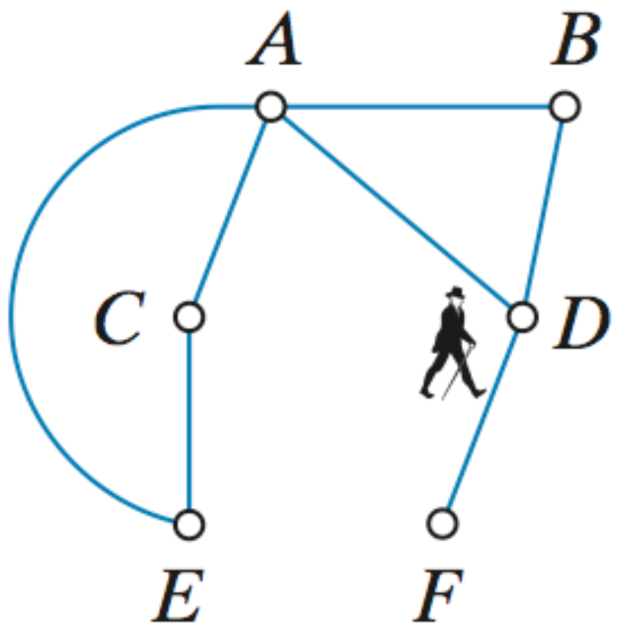
Copy 1



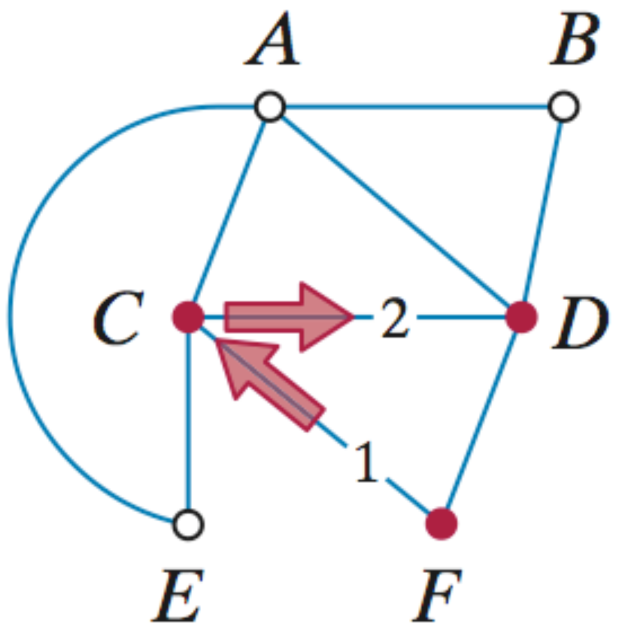
Copy 2

Example Implementing Fleury's Algorithm

Step 2: Travel from *C* to *D*. (Could have also gone to *A* or to *E*.)



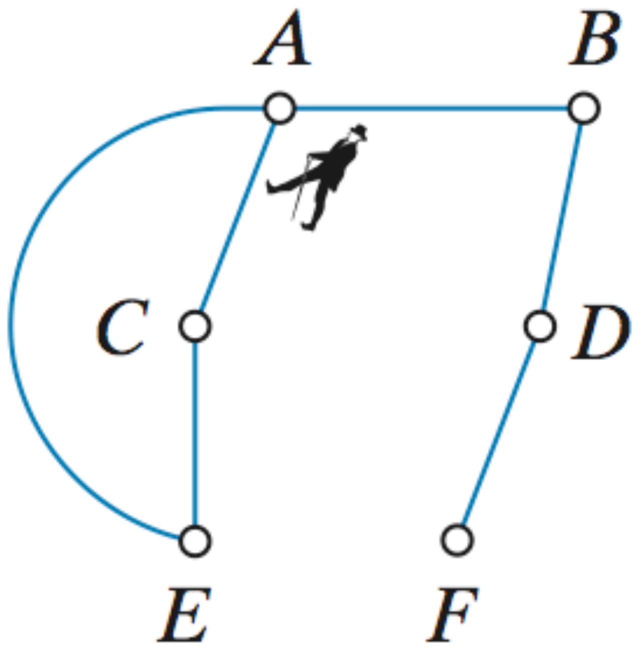
Copy 1



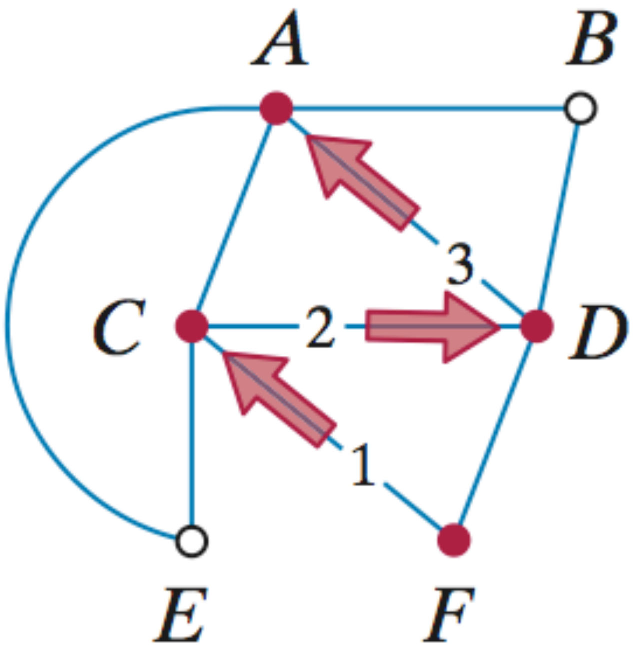
Copy 2

Example Implementing Fleury' s Algorithm

Step 3: Travel from *D* to *A*. (Could have also gone to *B* but not to *F* – *DF* is a bridge!)



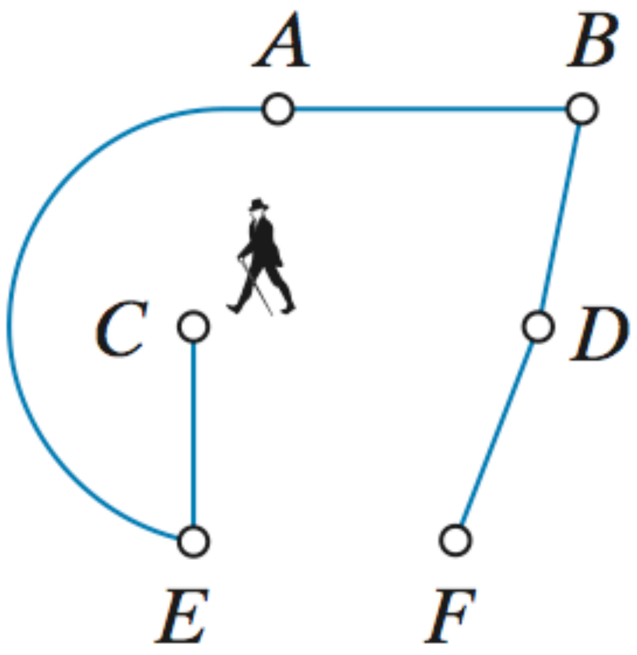
Copy 1



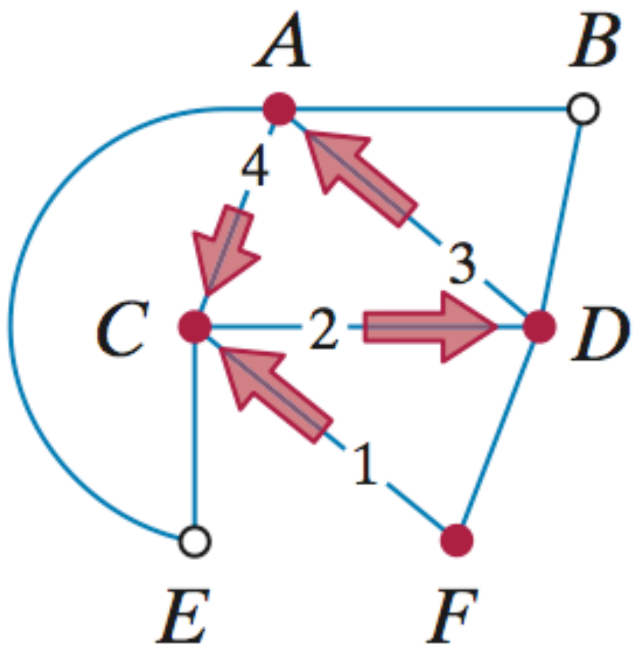
Copy 2

Example Implementing Fleury's Algorithm

Step 4: Travel from *A* to *C*. (Could have also gone to *E* but not to *B* – *AB* is a bridge!)



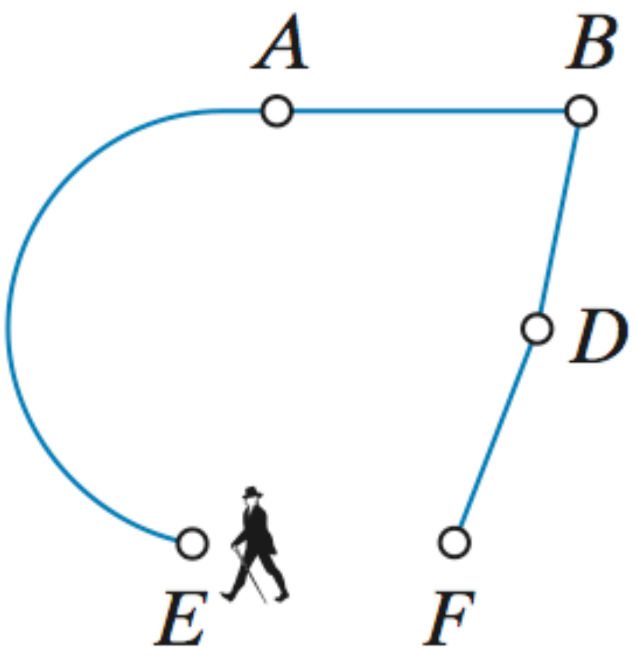
Copy 1



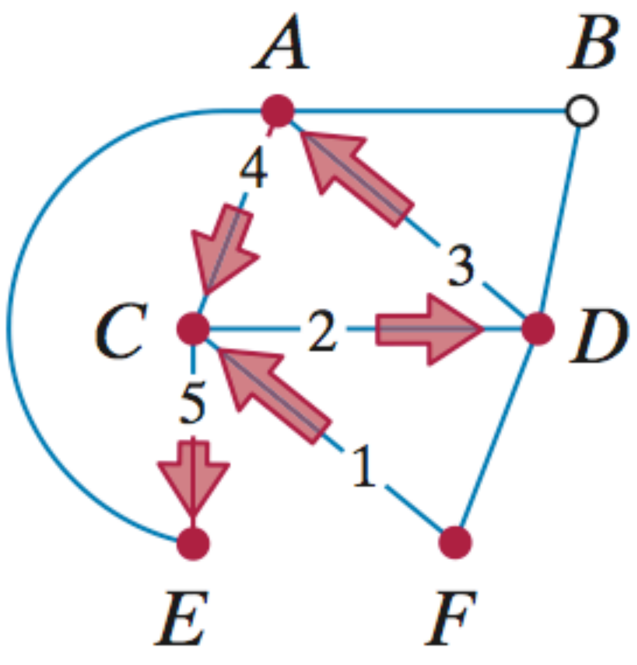
Copy 2

Example Implementing Fleury’s Algorithm

Step 5: Travel from *C* to *E*. (There is no choice!)



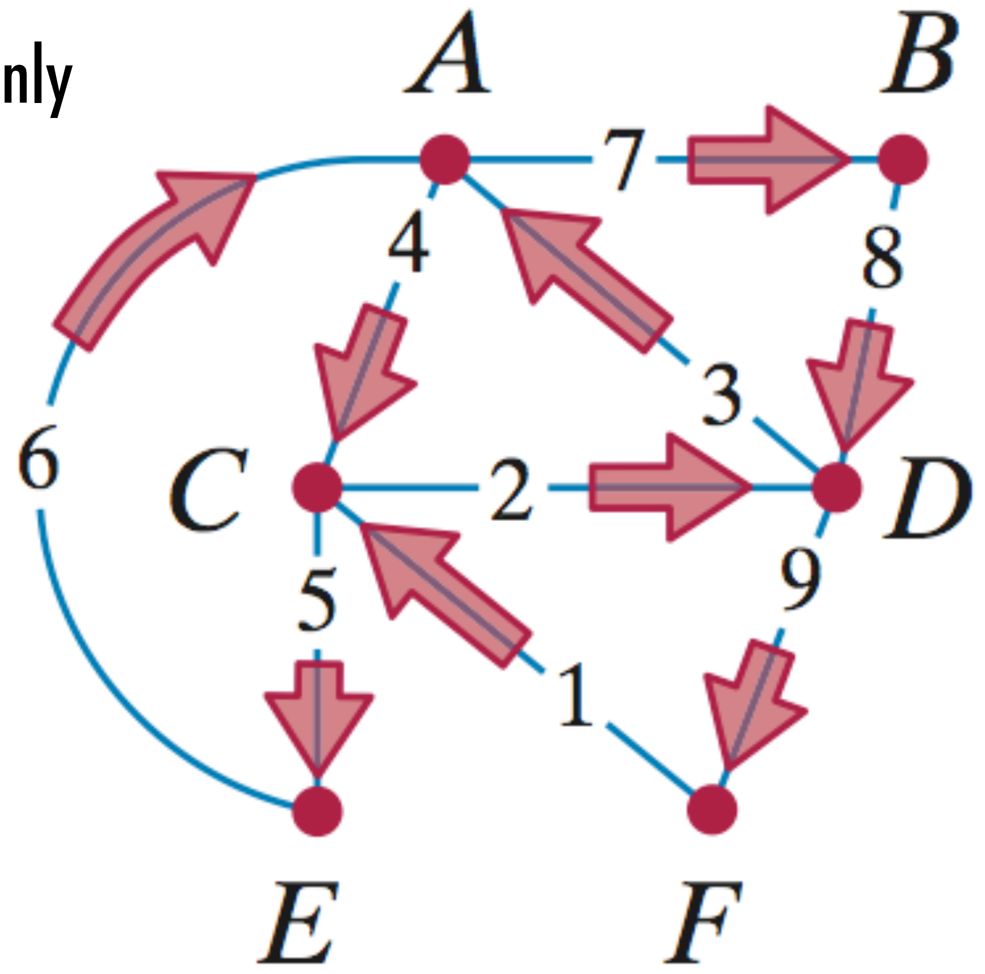
Copy 1



Copy 2

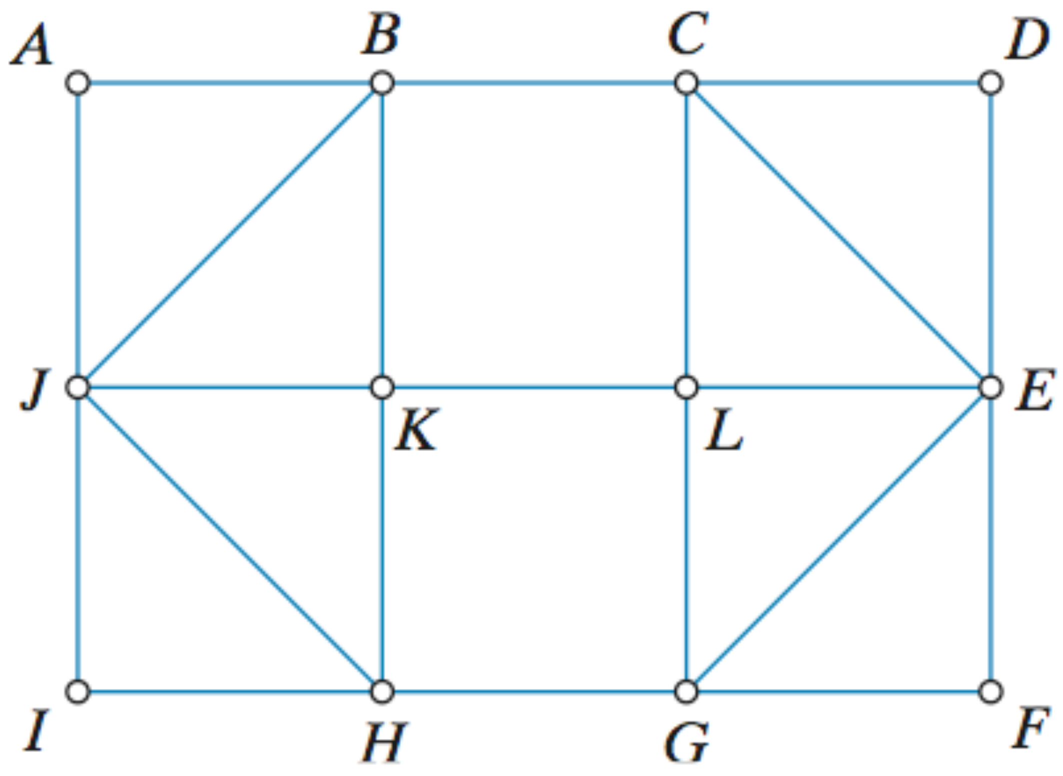
Example Implementing Fleury's Algorithm

Steps 6, 7, 8, and 9: Only one way to go at each step.



Example Fleury’s Algorithm for Euler Paths

We will apply Fleury’s algorithm to the graph below.



Example Fleury' s Algorithm for Euler Paths

Since it would be a little impractical to show each step of the algorithm with a separate picture as we did in previous example, you are going to have to do some of the work. Start by making two copies of the graph.

- **Start.** This graph has two odd vertices, *E* and *J*. We can pick either one as the starting vertex. Let's start at *J*.

Example Fleury's Algorithm for Euler Paths

- **Step 1.** From *J* we have five choices, all of which are OK. We'll randomly pick *K*. (Erase *JK* on copy 1, and mark and label *JK* with a 1 on copy 2.)

- **Step 2.** From *K* we have three choices (*B*, *L*, or *H*). Any of these choices is OK. Say we choose *B*. (Now erase *KB* from copy 1 and mark and label *KB* with a 2 on copy 2.)

Example Fleury' s Algorithm for Euler Paths

- **Step 3.** From *B* we have three choices (*A*, *C*, or *J*). Any of these choices is OK. Say we choose *C*. (Now erase *BC* from copy 1 and mark and label *BC* with a 3 on copy 2.)

- **Step 4.** From *C* we have three choices (*D*, *E*, or *L*). Any of these choices is OK. Say we choose *L*. (EML–that's shorthand for erase, mark, and label.)

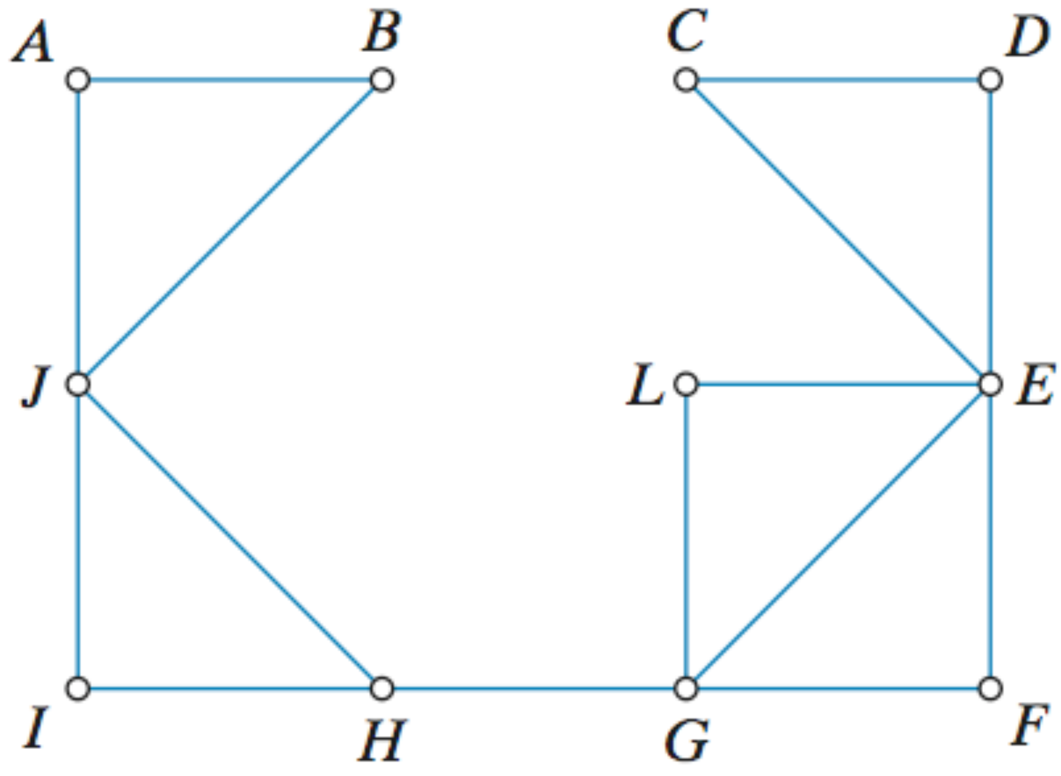
Example Fleury's Algorithm for Euler Paths

- **Step 5.** From L we have three choices (*E*, *G*, or *K*). Any of these choices is OK. Say we choose *K*. (EML.)

- **Step 6.** From *K* we have only one choice– to H. We choose *H*. (EML.)

Example Fleury's Algorithm for Euler Paths

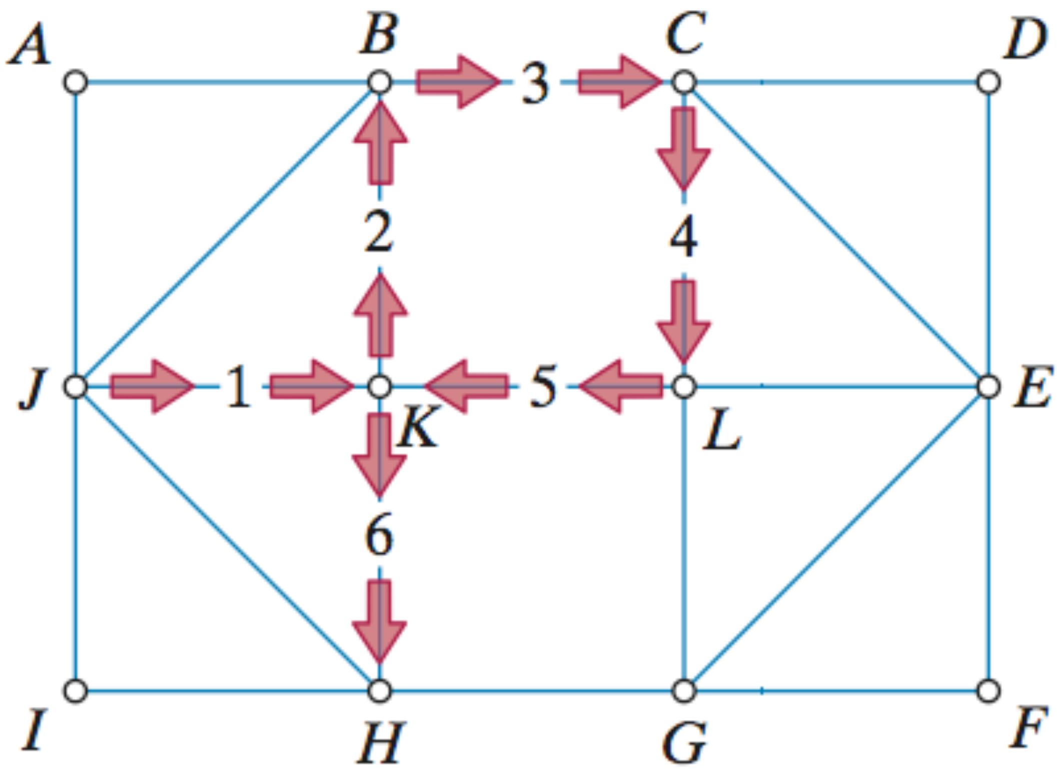
- **Step 7.** From H we have three choices (*G*, *I*, or *J*). We



Copy 1 at Step 7

Example Fleury's Algorithm for Euler Paths

- **Step 7.** Either of the other two choices is OK. Say we



Copy 2 at Step 7

Example Fleury's Algorithm for Euler Paths

- **Step 8.** From *J* we have three choices (*A*, *B*, or *I*), but we should not choose *I*, as *JI* has just become a bridge. Either of the other two choices is OK. Say we choose *B*. (EML)

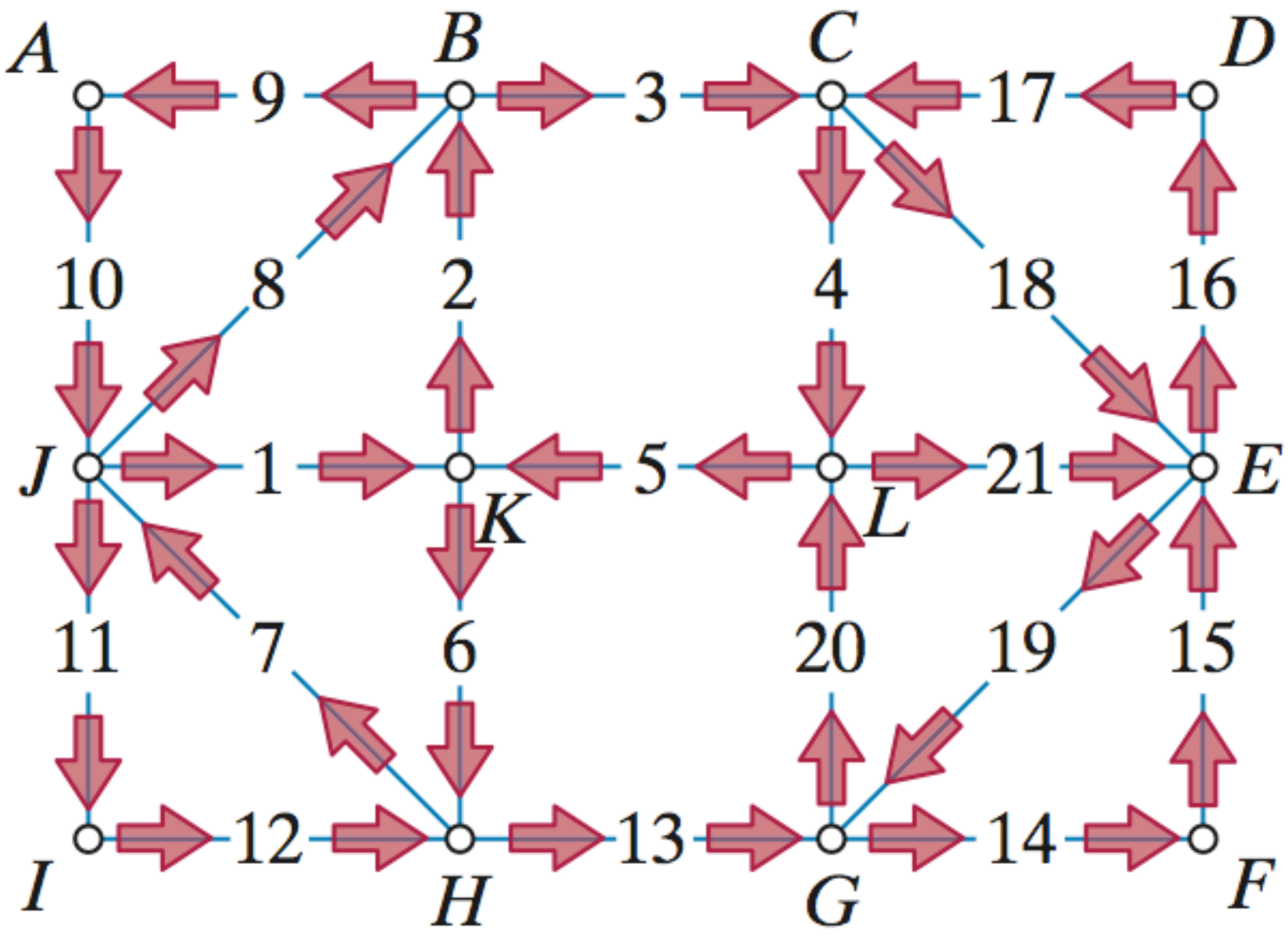
- **Step 9 through 13.** Each time we have only one choice. From *B* we have to go to *A*, then to *J*, *I*, *H*, and *G*.

Example Fleury’s Algorithm for Euler Paths

- **Step 14 through 21.** Not to belabor the point, let’s just cut to the chase. The rest of the path is given by *G*, *F*, *E*, *D*, *C*, *E*, *G*, *L*, *E*. There are many possible endings, and you should find a different one by yourself.

The completed Euler path (one of hundreds of possible ones) is shown in the next slide.

Example Fleury’s Algorithm for Euler Paths



Exhaustive Routes

We will use the term **exhaustive route** to describe a route that travels along the edges of a graph and passes through *each and every edge* of the graph *at least once.*

Exhaustive Routes

Such a route could be an Euler circuit (if the graph has no odd vertices) or an Euler path (if the graph has two odd vertices), but for graphs with more than two odd vertices, an exhaustive route will have to recross some of the edges. This follows from Euler's theorems.

We are interested in finding exhaustive routes that *recross* the fewest number of edges. Why?

Exhaustive Routes

In many applications, each edge represents a unit of cost.

The more edges along the route, the higher the cost of the route. In an exhaustive route, the first pass along an edge is a necessary expense, part of the requirements of the job.

Any additional pass along that edge represents a wasted expense (these extra passes are often described as *deadhead* travel).

Thus, an exhaustive route that minimizes cost (optimal exhaustive route) is one with the fewest number of deadhead edges.

Eulerizing Graphs

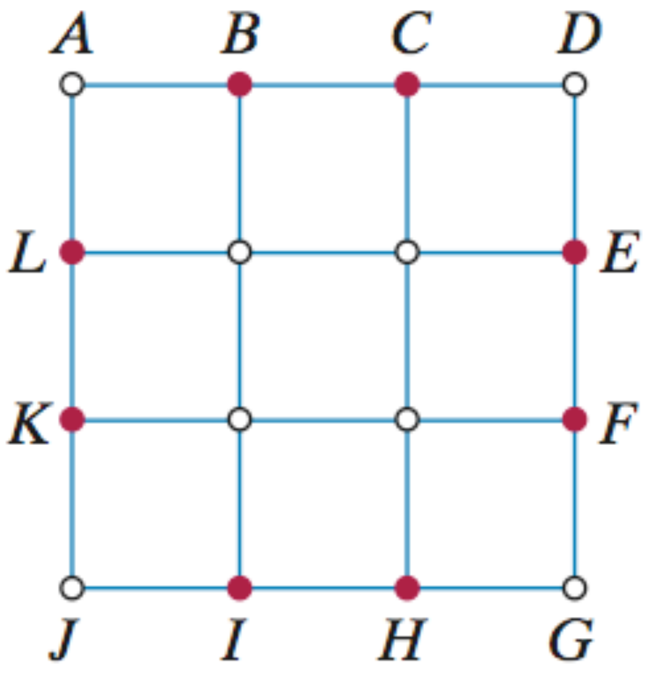
We are now going to see how the theory developed in the preceding sections will help us design optimal exhaustive routes for graphs with many (more than two) odd vertices.

The key idea is that we can turn odd vertices into even vertices by adding "duplicate" edges in strategic places. This process is called **eulerizing** the graph.

Example Covering a 3 by 3 Street Grid

The graph represents a 3 block by 3 block street grid consisting of 24 blocks.

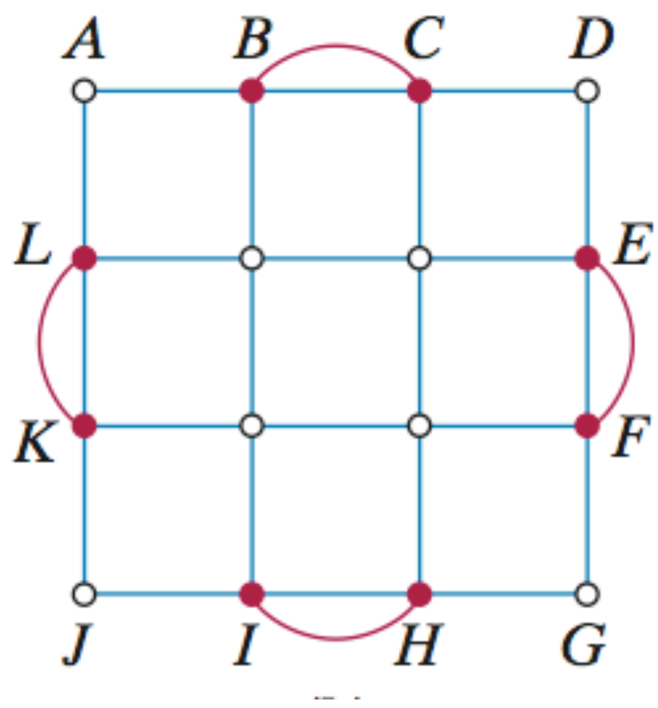
How can we find an optimal route that covers all the edges of the graph and ends back at the starting vertex? Our first step is to identify the odd vertices. This graph has eight odd vertices (*B*, *C*, *E*, *F*, *H*, *I*, *K*, and *L*), shown in red.



Example Covering a 3 by 3 Street Grid

When we add a duplicate copy of edges *BC*, *EF*, *HI*, and *KL*, we get this graph. This is a *eulerized* version of the original graph–its

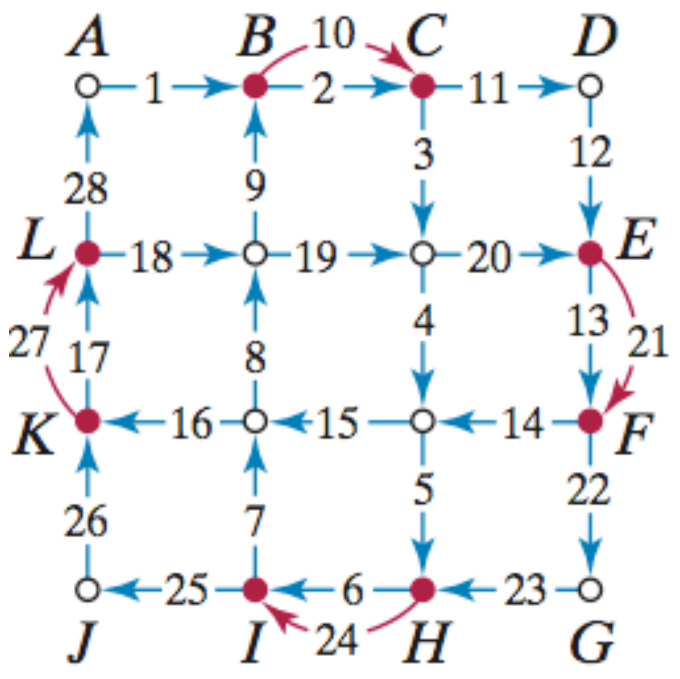
vertices are all even, so we know it has an Euler circuit. Moreover, it's clear we couldn't have done this with fewer than four duplicate edges



Example Covering a 3 by 3 Street Grid

This figure shows one of the many possible Euler circuits, with the edges numbered in the order they are traveled. The Euler circuit represents an exhaustive closed route along

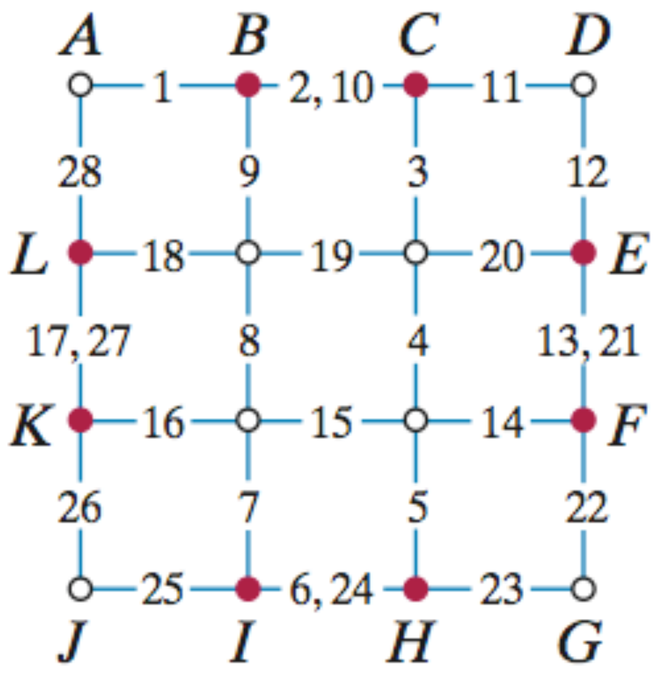
the edges of the original graph, with the four duplicate edges (*BC*, *EF*, *HI*, and *KL*) indicating the deadhead blocks where a second pass is required



Example Covering a 3 by 3 Street Grid

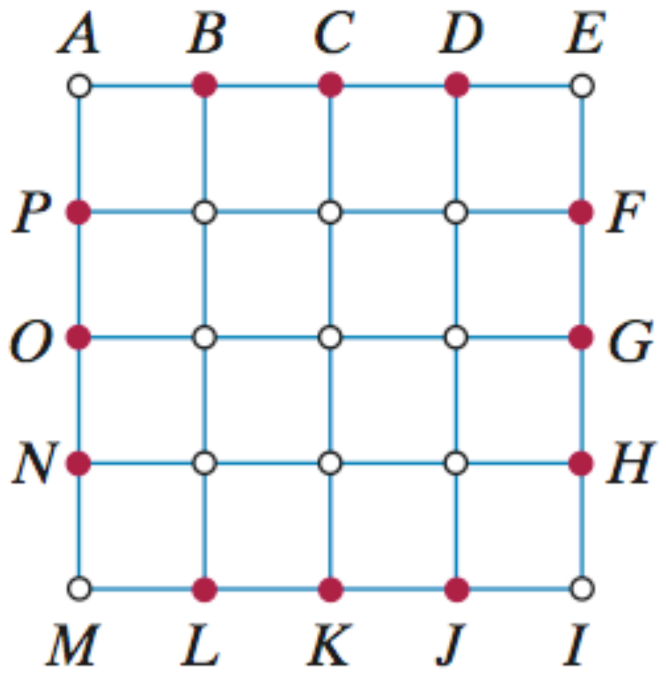
The total length of this route is 28 blocks (24 blocks in the grid plus 4 deadhead blocks), and this route is optimal–no matter how clever you are or how hard you try,

if you want to travel along each block of the grid and start and end at the same vertex, you will have to pass through a minimum of 28 blocks!



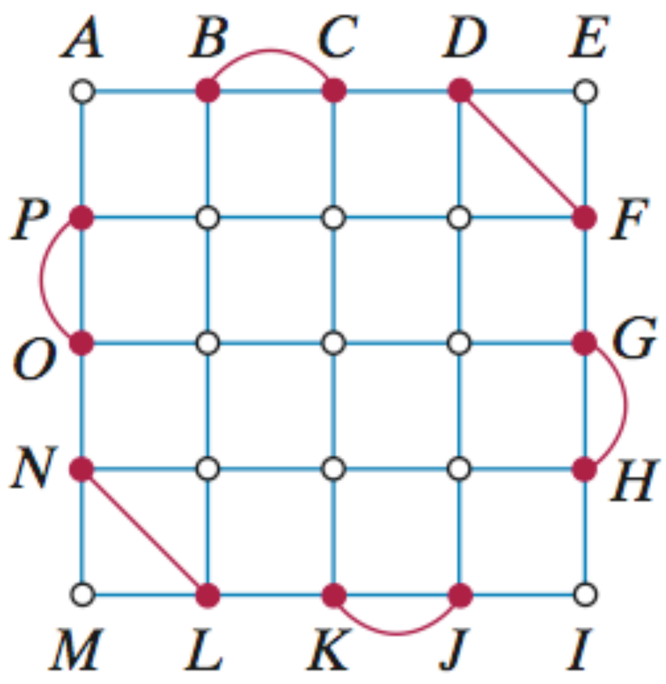
Example Covering a 4 by 4 Street Grid

The graph represents a 4 block by 4 block street grid consisting of 40 blocks. The 12 odd vertices in the graph are shown in



Example Covering a 4 by 4 Street Grid

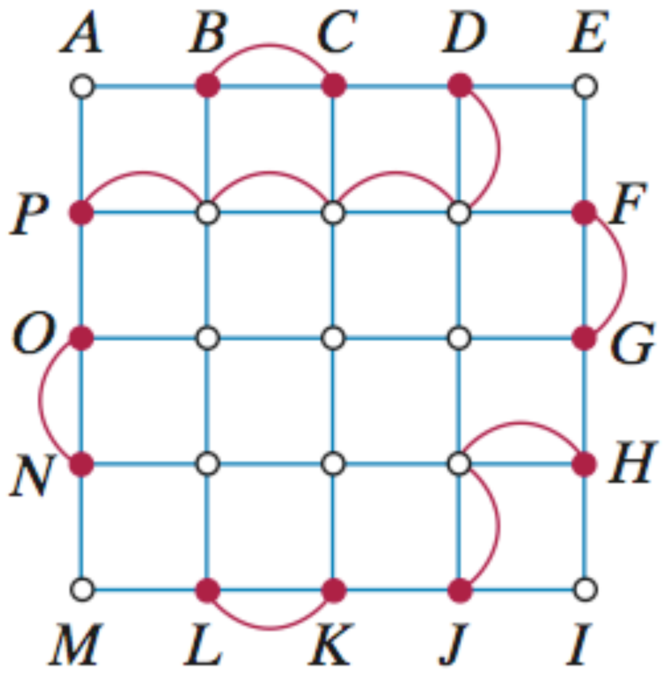
This figure shows how not to do it! This graph violates the cardinal rule of eulerization–you can only duplicate edges that are part of



Example Covering a 4 by 4 Street Grid

This figure shows a legal eulerization, but it is not optimal, as it is obvious that we could have accomplished the same thing by

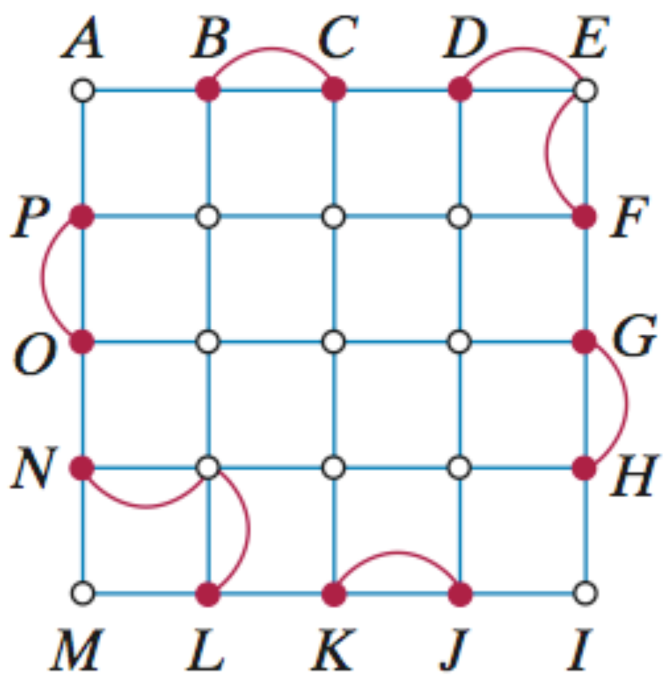
adding fewer duplicate edges.



Example Covering a 4 by 4 Street Grid

This figure shows an *optimal eulerization* of the original graph–one of several possible. Once we have an optimal eulerization, we have the blueprint for an optimal exhaustive

closed route on the original graph. Regardless of the specific details, we now know that the route will travel along 48 blocks–the 40 original blocks in the grid plus 8 deadhead blocks.



Open Route

In some situations we need to find an exhaustive route, but there is no requirement that it be closed–the route may start and end at different points.

In these cases we want to leave two odd vertices on the graph unchanged and change the other odd vertices into even vertices by duplicating appropriate edges of the graph.

Semi-eulerization

This process is called a **semi-eulerization** of the graph.

When we strategically choose how to do this so that the number of duplicate edges is as small as possible, we can obtain an *optimal exhaustive open route.*

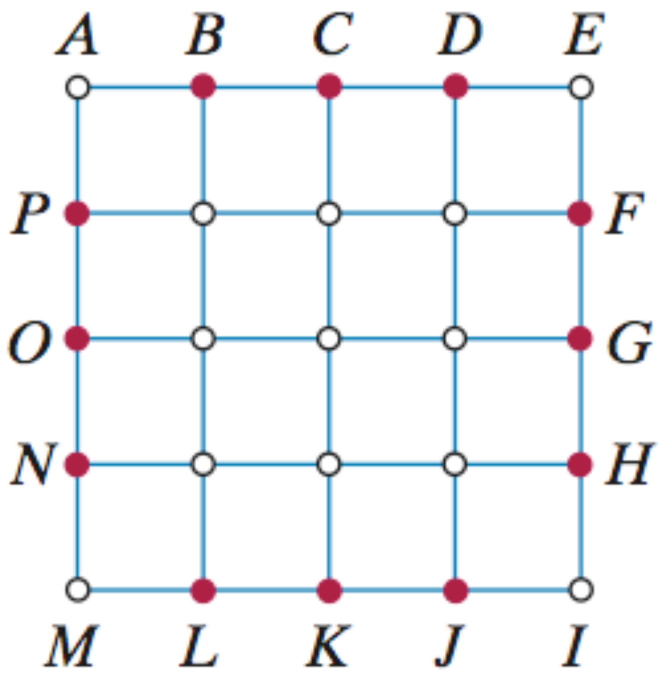
In this case the route will start at one of the two odd vertices and end at the other one.

Example Parade Route

Let's consider once again the 4 by 4 street grid. Imagine that your job is to design a good route for a Fourth of July parade that must pass through each of the 40 blocks of the street grid.

The key is that when routing a parade, you do not want the parade to start and end in the same place.

In fact, for traffic control it is usually desirable to keep the starting and ending points of a parade as far from each other as possible.

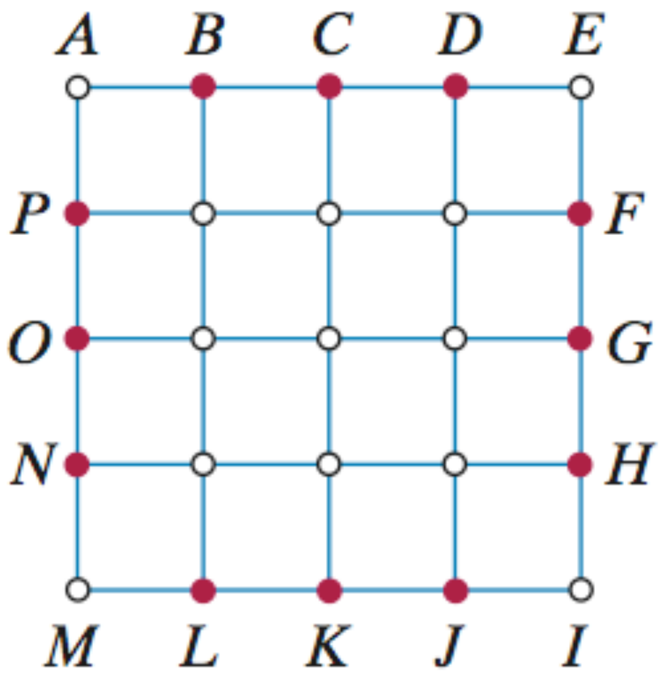


Example Parade Route

The fire department has added one additional requirement to the parade route: The parade has to start at *B*.

Your task, then, is to find a semi-eulerization of the graph that leaves *B* and one more odd vertex unchanged

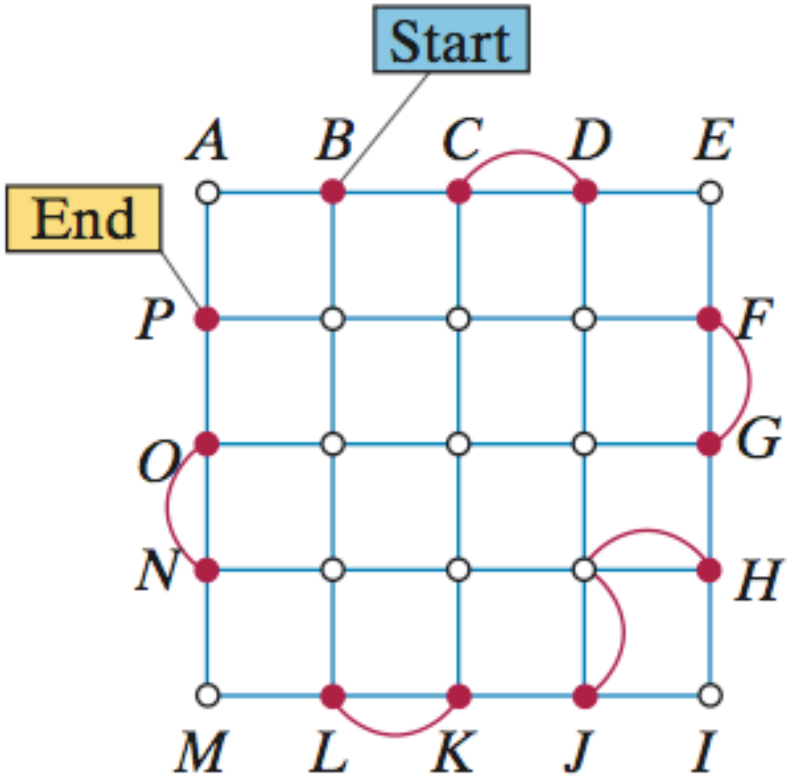
(preferably a vertex far from *B*) and that changes all the other odd vertices into even vertices.



Example Parade Route

This semi-eulerization is *optimal* because it required only six duplicate edges, and this is as good as one can do. The optimal parade route could be found by finding an Euler path.

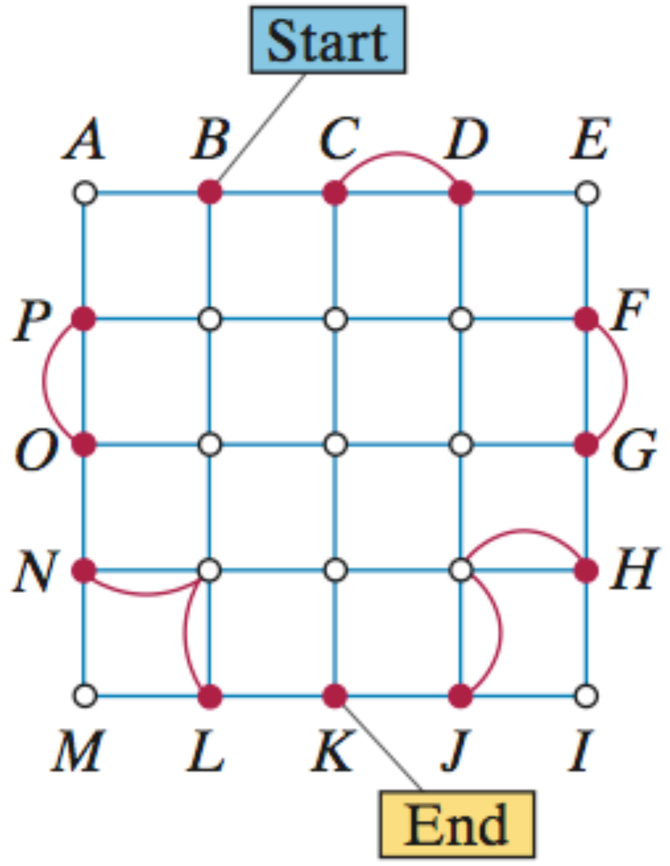
The only bad thing about this route is that the parade would end at *P*, a point a bit too close to the starting point, and the traffic control people are unhappy about that.



Example Parade Route

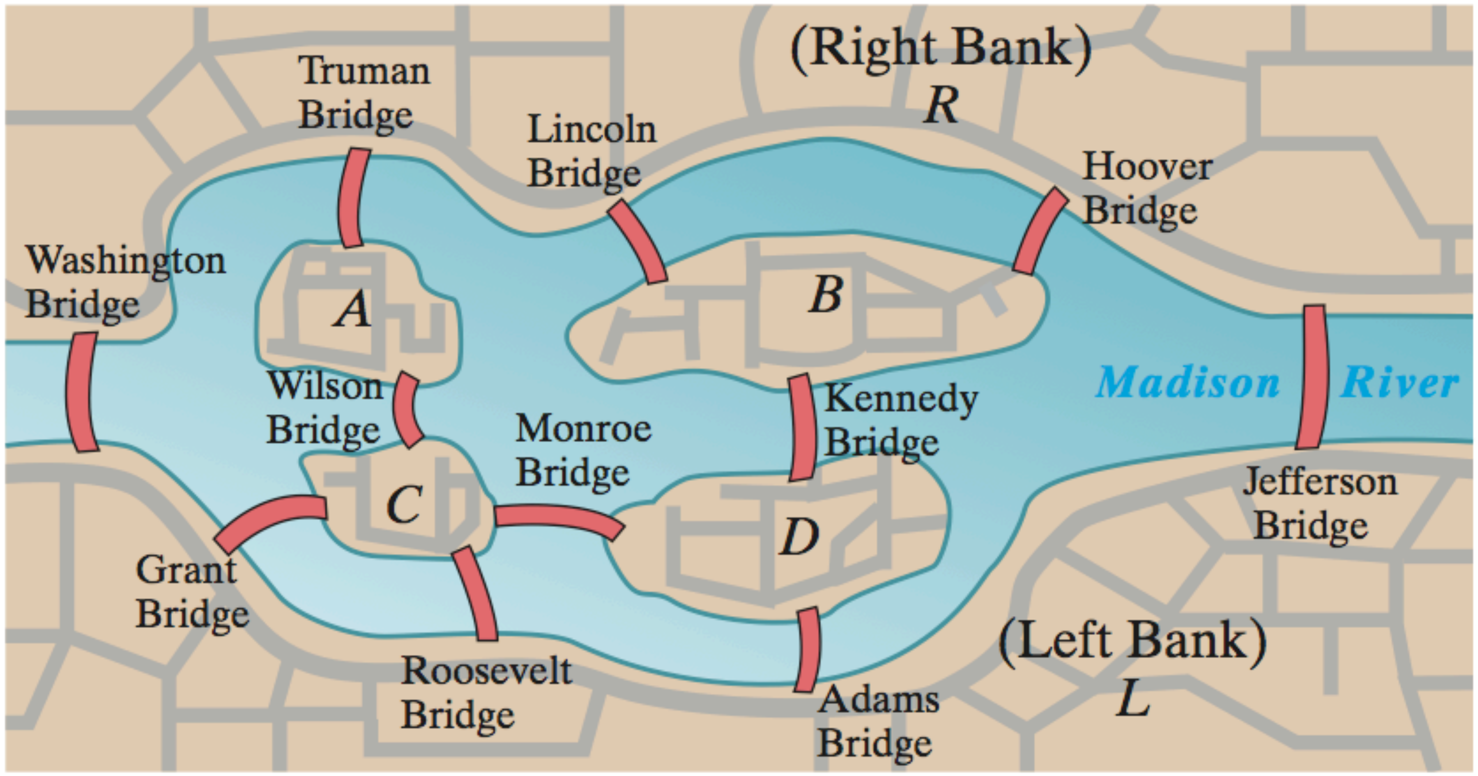
A different semi-eulerization is shown here. The parade route in this case would not be optimal (it has seven deadhead blocks),

but because it ends at *K*, it better satisfies the requirement that the starting and ending points be far apart. The traffic control folks are happy now.



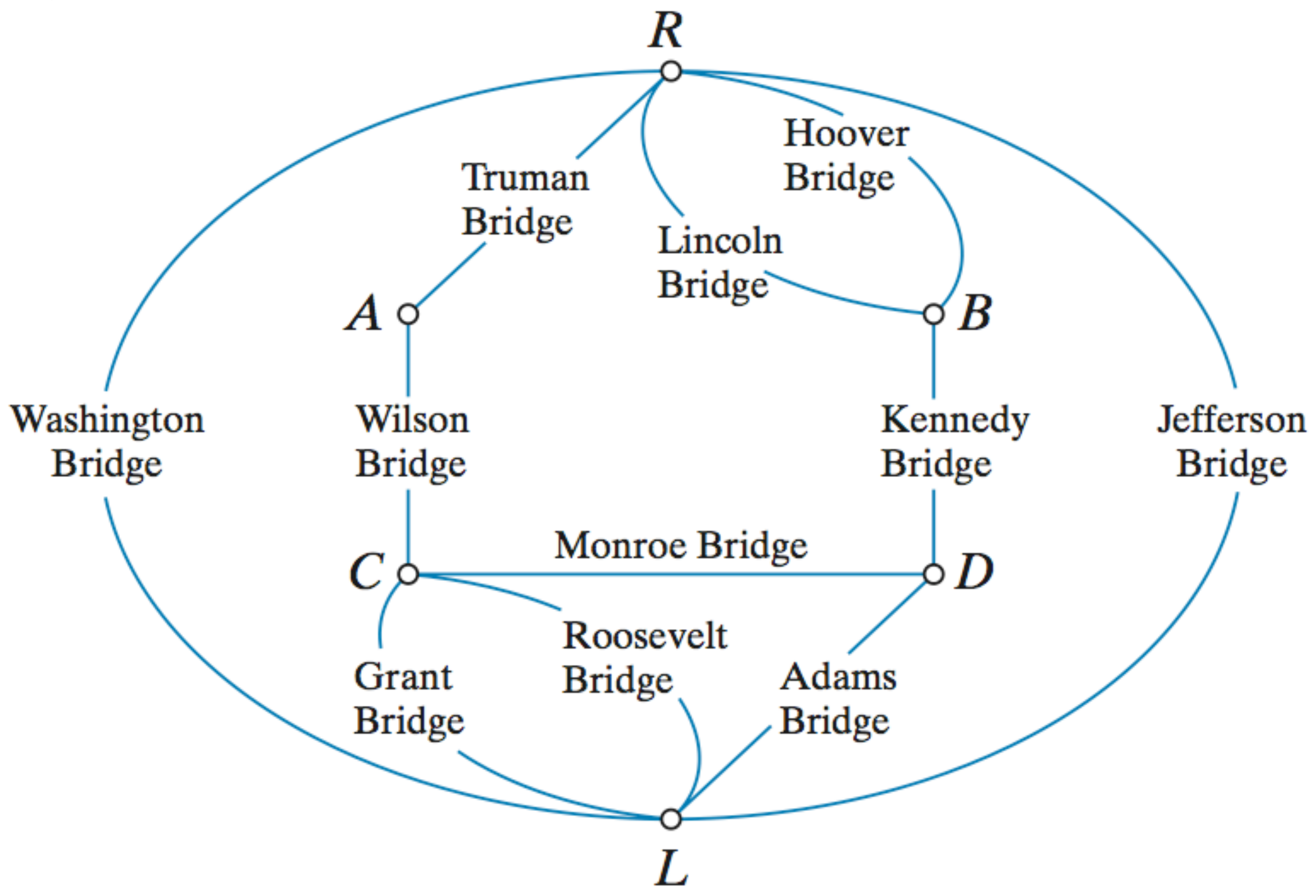
Example The Bridges of Madison County

A photographer needs to take photos of each of the 11 bridges in Madison County.



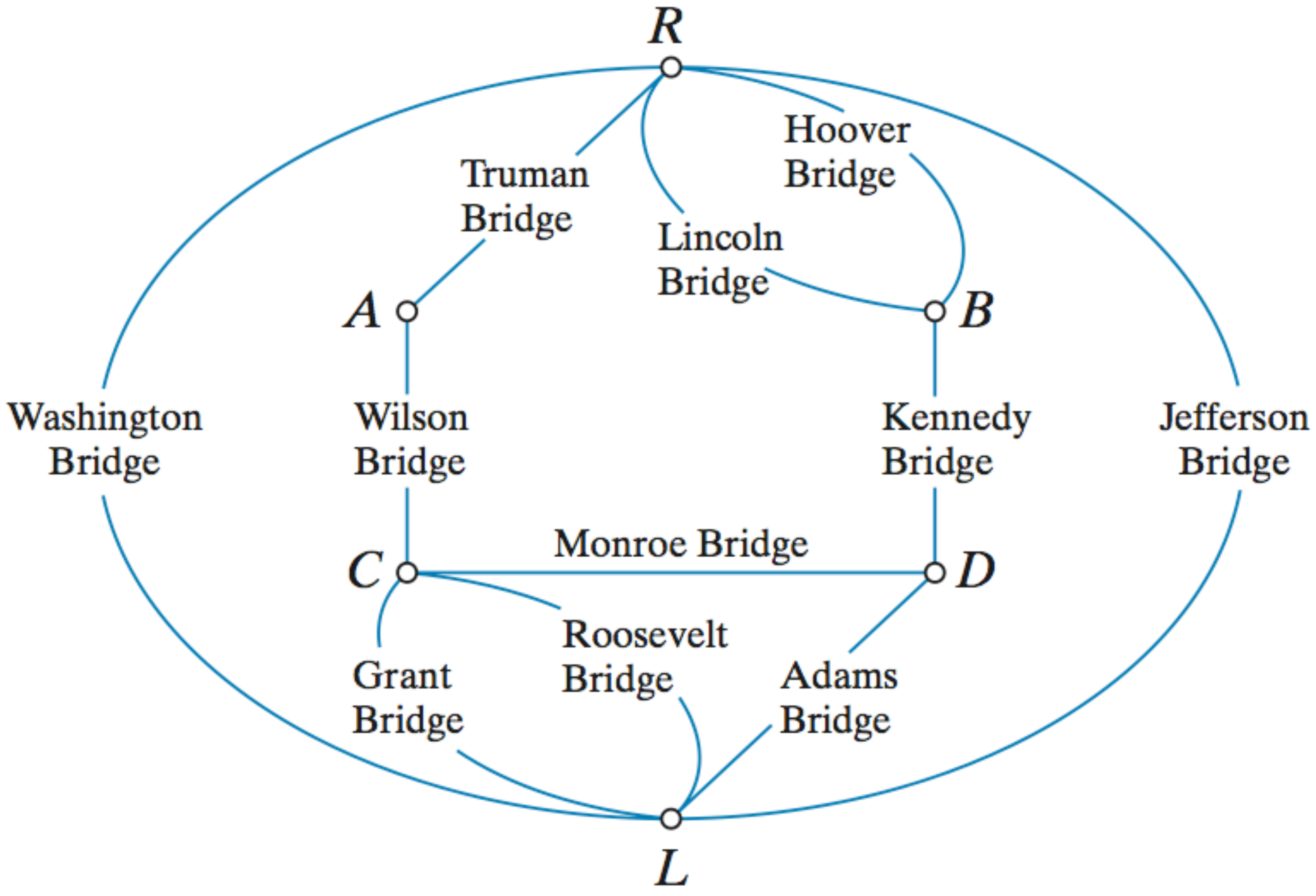
Example The Bridges of Madison County

A graph model of the layout (vertices represent land masses, edges represent bridges) is shown.



Example The Bridges of Madison County

The graph has four odd vertices (*R*, *L*, *B*, and *D*), so some bridges are definitely going to have to be recrossed.



Example The Bridges of Madison County

How many and which ones depends on the other parameters of the problem. (It costs $25 in toll fees to cross a bridge, so the baseline cost for crossing the 11 bridges is $275.

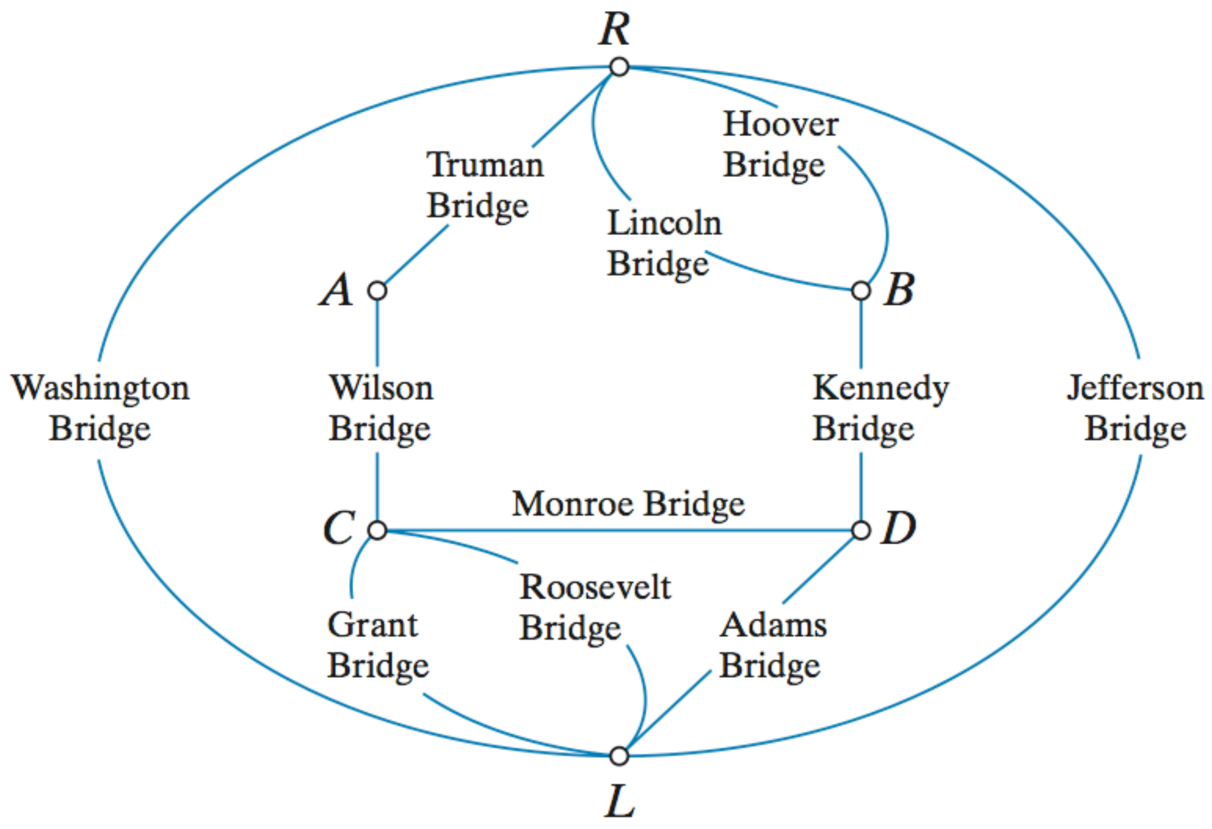
Each recrossing is at an additional cost of $25.) The following are a few of the possible scenarios one might have to consider.

Each one requires a different eulerization and will result in a different route.

Example The Bridges of Madison County

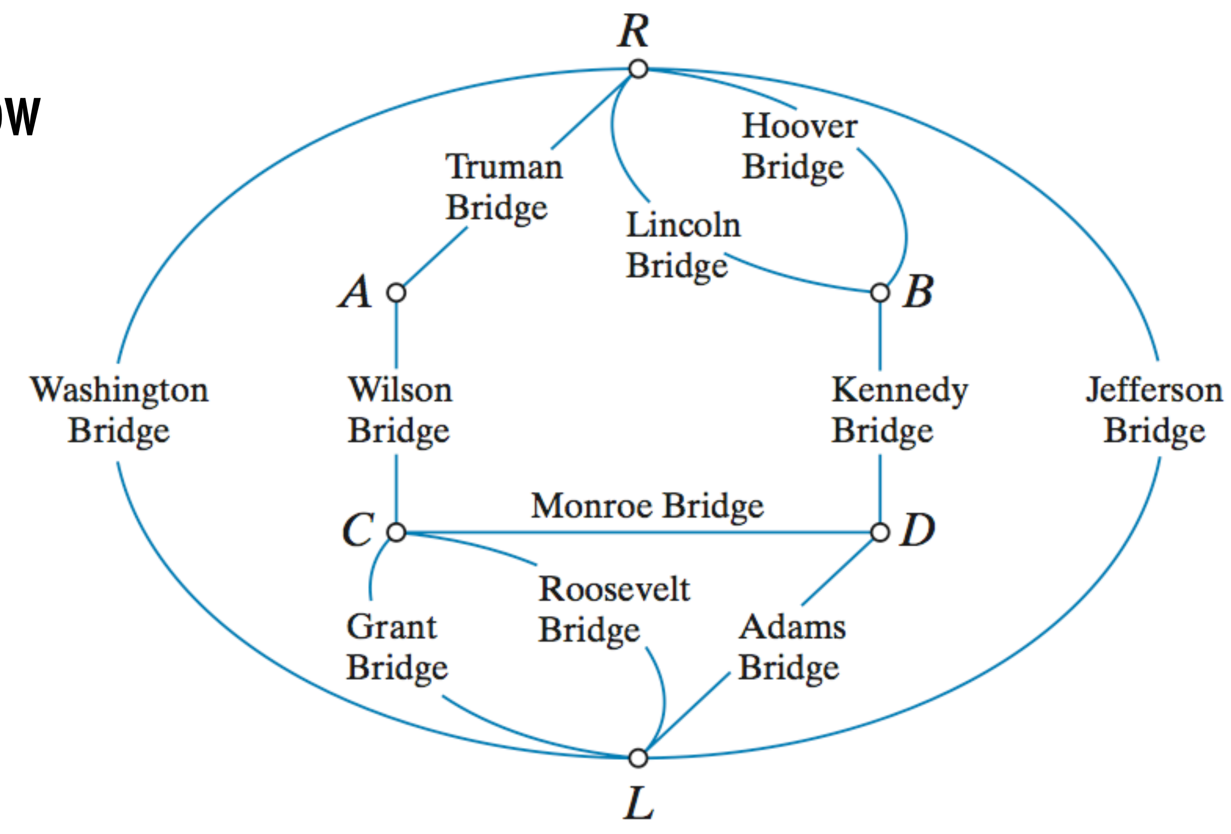
The photographer needs to start and end his trip in the same place. This scenario requires an optimal eulerization of the graph.

This is not hard to do, and an optimal route can be found for a cost of $325.



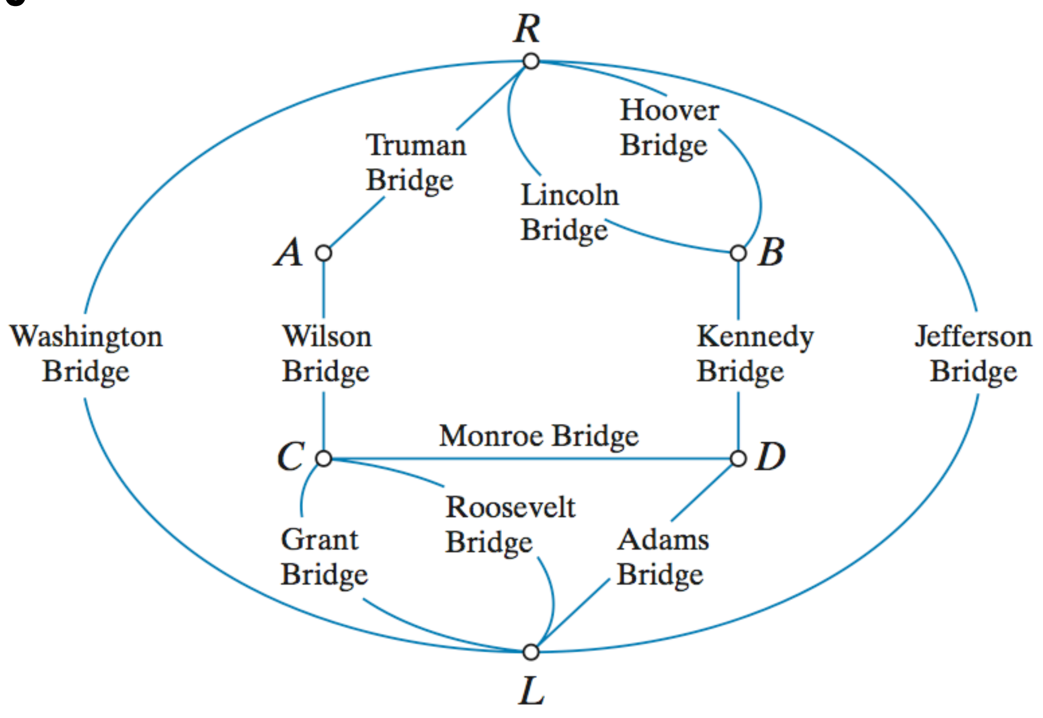
Example The Bridges of Madison County

The photographer has the freedom to choose any starting and ending points for his trip. In this case we can find an optimal semi-eulerization of the graph, requiring only one duplicate edge. Now an optimal route is possible at a cost of $300.



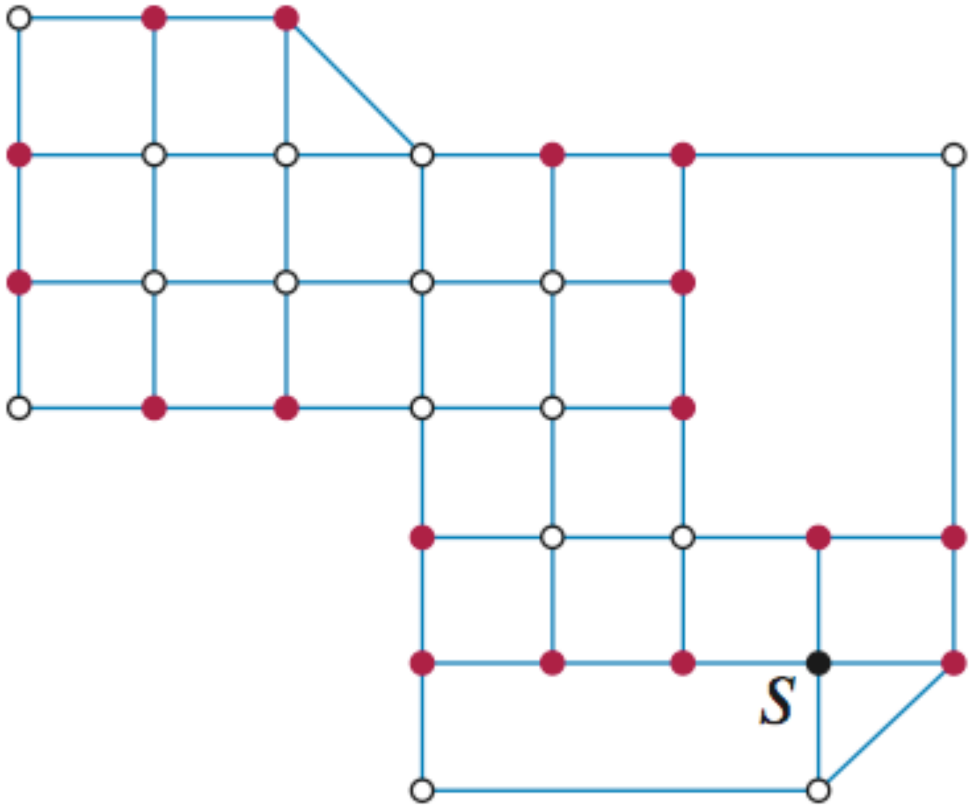
Example The Bridges of Madison County

The photographer has to start his trip at *B* and end the trip at *L*. In this case we must find a semi-eulerization of the graph where *B* and *L* remain as odd vertices and *R* and *D* become even vertices. It is possible to do this with just two duplicate edges and thus find an optimal route that will cost $325.



Example The Exhausted Patrol and the Grateful No Deadhead

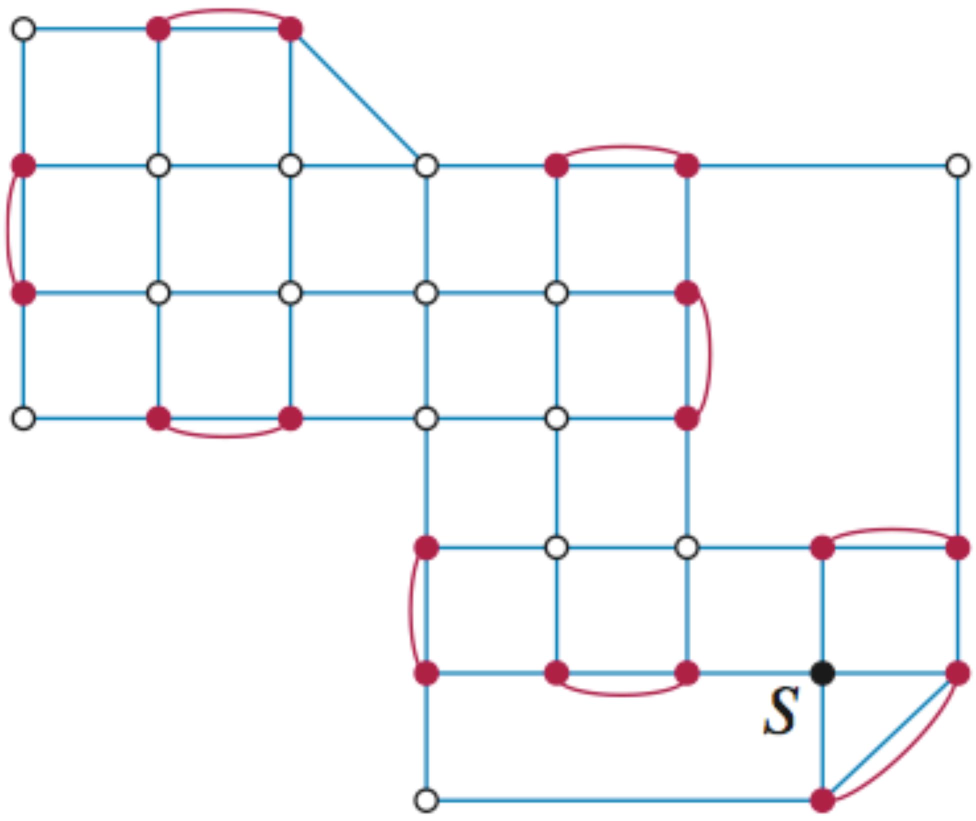
Earlier we raised the question of finding an optimal exhaustive closed route for a security guard hired to patrol the streets of the Sunnyside subdivision and we created the graph model for this problem.



Example The Exhausted Patrol and the Grateful No Deadhead

The graph has 18 odd vertices, shown in red. We now know that the name of the game is to find an optimal eulerization of this graph.

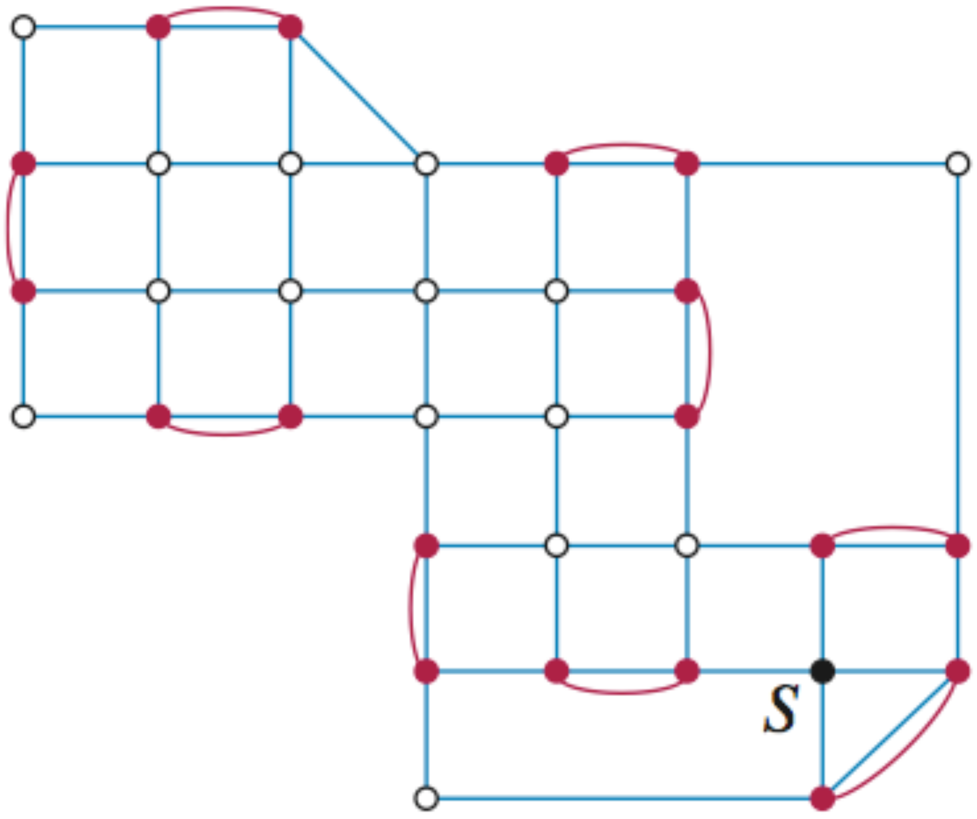
In this case the odd vertices pair up beautifully, and the optimal eulerization requires only nine duplicate edges, shown here.



Example The Exhausted Patrol and the Grateful No Deadhead

All the answers to the security guards questions can now be answered: An optimal route will require nine deadhead blocks.

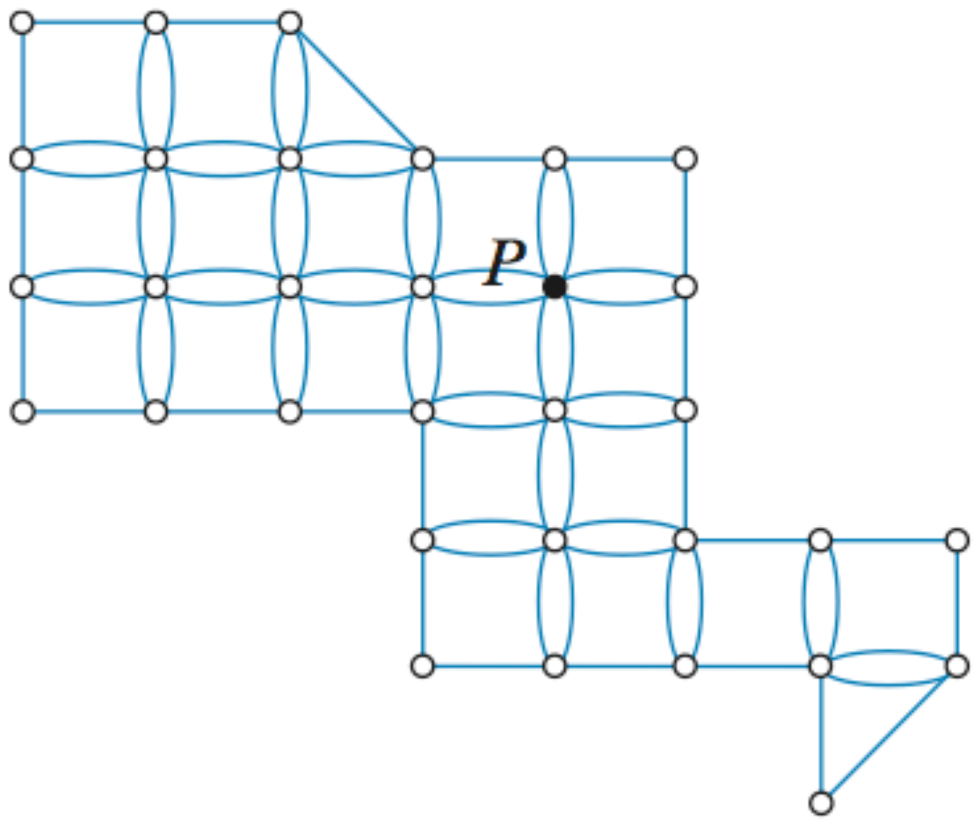
The actual route can be found using trial and error or Fleury's algorithm.



Example The Exhausted Patrol and the Grateful No Deadhead

A slightly different problem is the one facing the mail carrier delivering mail along the streets of the Sunnyside subdivision.

Much to the mail carrier's pleasant surprise, in the graph that models her situation all the vertices are even.



Example The Exhausted Patrol and the Grateful No Deadhead

This means that the optimal route is an Euler circuit, which can be found once again using Fleury’s algorithm (or trial and error if you

prefer). This mail carrier will not have to deadhead.