Chapter 5 – The Mathematics of Getting Around

Notes on Graph Theory

Rayan Ibrahim

Summer 2020

Contents

5 The Mathematics of Getting Around

5.1 The Seven Bridges of Königsberg

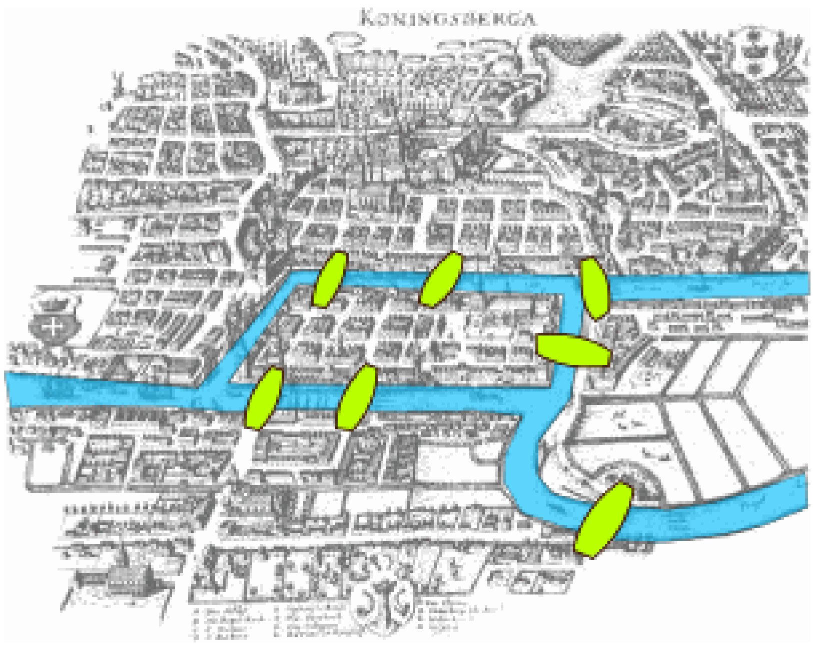


Figure 1: The seven bridges of Königsberg connect land masses and span the Pregel River. This image can be found in the wikipedia article, and credits are attributed to Bogdan Giuşcă.

Königsberg was a city in Prussia in the 1700s. The main city was divided by the Pregel River, with land masses in between, connected by seven bridges (Figure 1).

The Bridges of Königsberg problem first appeared in the 1700s when the following question was posed: "Is there a walking route through the city which crosses each bridge once and only once?" Now of course there are some rules to this. For example, you cannot take a boat or swim in the river, and you must cross a bridge starting at one end and reaching the other.

This problem was solved by Leonhard Euler, a prominent mathematician, and eventually would be the starting point for an entire field of mathematics known as graph theory. You'd be hard pressed to find any graph theory book that fails to mention this problem and its historical significance. Try to solve the problem for yourself (that is, try to draw a route that crosses each bridge once and only once in Figure 1.)

5.2 Graph Theory

There are many problems that are similar to the bridges of Königsberg. For example, one might have a paper route that starts and ends at the same location, and they would like to know what the most optimal route is. You might imagine that this sort of *optimization* scales up, for example national mail and package carriers. As with all problems, we design tools that we can use to solve them.

Definition (Graph)

A *graph* is a mathematical object made up of **vertices** and **edges** that connect those vertices. In a drawing of a graph, vertices are drawn as dots, while we draw a line between two vertices to represent an edge.

Let's look at an example of graphs.

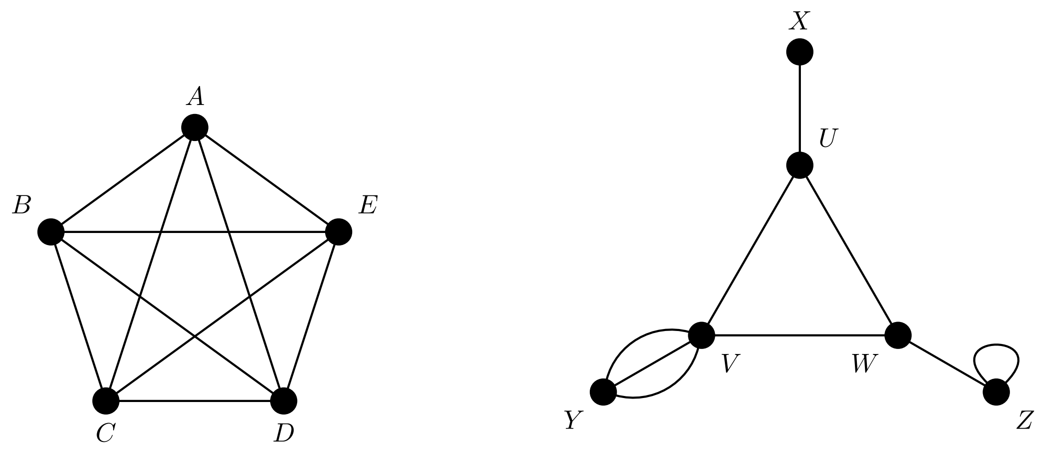


Figure 2: On the left we have what is known as K_5, the *complete* graph on 5 vertices (it's called complete since every possible pair of vertices is connected by an edge). On the right is some random graph.

Note that graphs come in many varieties. In Figure 2, you may think that there are two "graphs." This may the case, but it is also valid to consider the two objects together as one graph. This notion will be important later. We have vertices (black dots), labeled A, B, C, D, E on the left and U, V, W, X, Y, Z on the right. The edges connecting the vertices are referred to by their endpoints.

For example, on the left the vertices A and B are connected by the edge AB. We can also say edge "BA", but the ordering is only important if we impose some sort of direction (for example, one way streets.) Two vertices connected directly by an edge are said to be **adjacent**, while two edges that share an endpoint are said to be **incident** (or adjacent, such as BD and DE.) Vertices may have **multiple edges** between them, which are sometimes referred to as **parallel edges** (think multiple bridges or tunnels connecting two land masses.) We see these above with VY, where V and Y have three edges between them. Finally there are loop edges, or edges that connect a vertex to itself (think of a cul-de-sac or race track). Since the endpoints of such an edge are the same, we can call the loop edge above on the right ZZ.

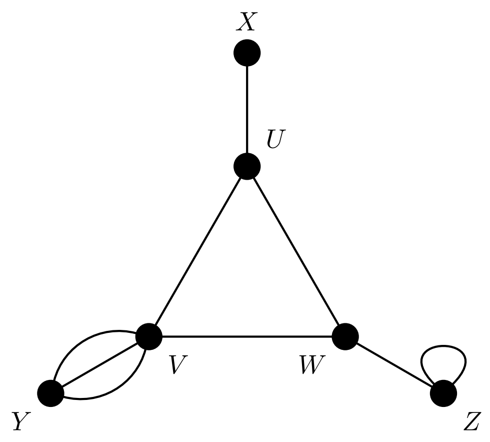
Now we have defined some useful terminology. Let's now introduce the notion of a vertex degree.

Definition (Degree)

The *degree* of a vertex v in a graph, denoted $\deg(v)$ is the number of edges incident to v (that is, the number of edges "touching" v). If the graph has no parallel edges or loops, the degree indicates the number of **neighbors**, or vertices adjacent to v. Note that a loop adds 2 to the degree of a vertex.

A vertex is said to be odd (even) if it's vertex degree is odd (even).

Let's look at one of our graphs.



Here in Figure **??** we have $\deg(U) = 3$, since the edges UX, UV, and UW are all the edges touching U. A complete list of degrees:

$$\begin{aligned} \deg(U) &= 3 \qquad & \deg(X) &= 1 \\ \deg(V) &= 5 \qquad & \deg(Y) &= 3 \\ \deg(W) &= 3 \qquad & \deg(Z) &= 3 \end{aligned}$$

Notice that Z has a loop, so the loop ZZ counts as 2, and along with ZW we have $\deg(Z) = 3$. Additionally, parallel edges each count as 1.

Now that we have some graph terminology and a small understanding under our belts, let's define one of the more basic qualities of a graph.

Definition (Connected)

A graph is said to be *connected* if, for every two vertices u and v, there is a sequence of edges that can be traveled from u to v. If there is at least one vertex that cannot be reached from another, we say the graph is *disconnected.*

Let's look at an example.

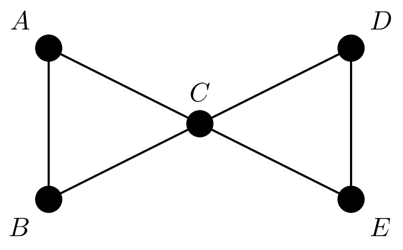


Figure 3: The bow-tie graph.

This graph in Figure 3 is known as the bow-tie graph. In the field of graph theory there is a vast amount of graphs, some named after those who discover (or construct) them, while others are simply named after their appearance. It wouldn't take long for one to see that the bow-tie graph is connected, that is, you can travel from one vertex to another by a series of edges. For example, to get to E from A we would traverse the edge AC, then CE. Let's remove some edges from our bow-tie, effectively disconnecting the graph.

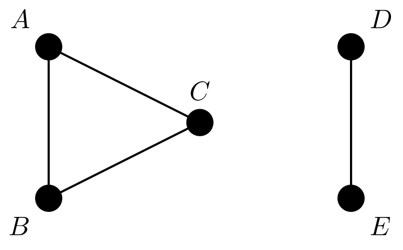


Figure 4: A modified bow-tie.

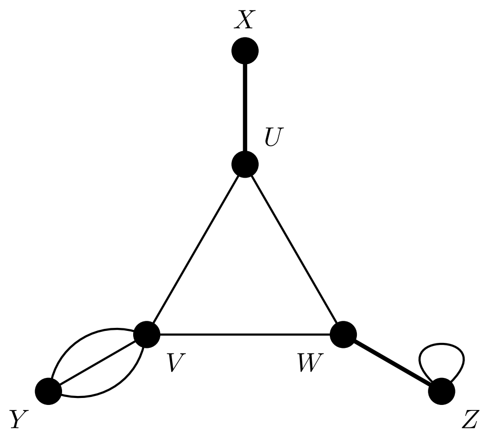
In Figure 4, we removed edges CD and CE from the original bow-tie. This new graph is disconnected, since D and E cannot travel to A, B, or C by a sequence of edges. However, we still have two connected "pieces", which are referred to as *components.*

The notion of removing edges from a graph to disconnect it is a big topic in graph theory, known as connectivity. This brings us to a new definition.

Definition (Bridge)

A *bridge* is an edge in a graph that, if removed, causes the resulting graph to be disconnected.

Let's once again revisit our previous graph in Figure **??**.



Here we can see that the edges XU and WZ (thick) are both bridges. That is, removing those edges from the graph will disconnect it (specifically, disconnect X and Z respectively.) Note that not all graphs have bridges. For example, the bow-tie in Figure 3 has no bridge, since it takes the removal of at least two edges to disconnect it. This is also true of K_5 in Figure 2.

We've been eluding a bit when talking about "traveling" from one vertex to another. It is time yet again for another definition.

Definition (Paths and Circuits)

A *path* is a sequence of edges that starts at one vertex and ends at a different vertex without repeating any edges. (This is really known as a trial in other contexts, but we will use the word path here.)

A *circuit* is a sequence of edges that starts and ends at the **same** vertex without repeating any edges.

We can talk about specific paths and circuits by giving a sequence of edges or vertices traveled to that can be used to trace out the path or circuit on the graph.

Now that we have many definitions, let's wrap up with an example highlighting them all, including the most recently mentioned paths and ciruits.

Example. Consider the following graph.

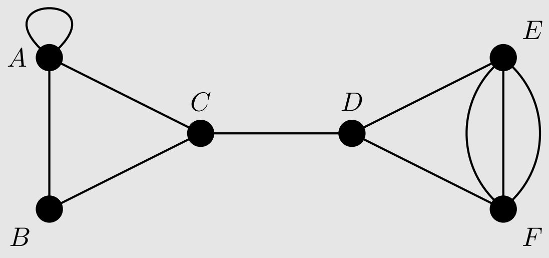


Figure 5: A graph very similar to the bow-tie graph.

Let's first start by listing our vertices and edges. We have

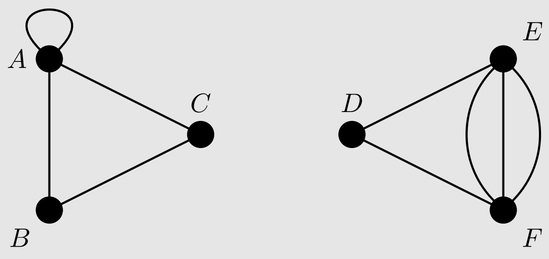
$$\text{Vertices: } \{A, B, C, D, E, F\}$$

$$\text{Edges: } \{AA, AB, AC, BC, CD, DE, \overbrace{EF}^{3}, DF\}$$

Notice that we didn't say EF three times above, even though there are three edges between E and F. There is no general agreed upon way to distinguish multiple edges in a listing of edges of a graph, but one could write a small 3 above EF.
Now we will list the degree of each vertex, i.e. the number of edges incident to each vertex.

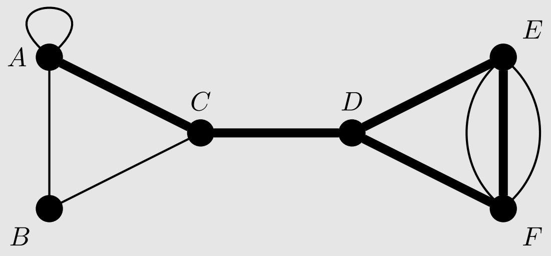
$$\begin{aligned} \deg(A) &= 4 & \deg(D) &= 3 \\ \deg(B) &= 2 & \deg(E) &= 4 \\ \deg(C) &= 3 & \deg(F) &= 4 \end{aligned}$$

Notice that A has a loop, which counts as 2 towards the degree. So while A is adjacent to 2 vertices other than itself, B and C, it's degree is 4. Similarly with E and F, their degree is 4 but they are adjacent to two other vertices.
The graph is connected, since every vertex can travel to every other vertex by a sequence of edges. There is one bridge, namely CD, which if removed disconnects the graph like so:

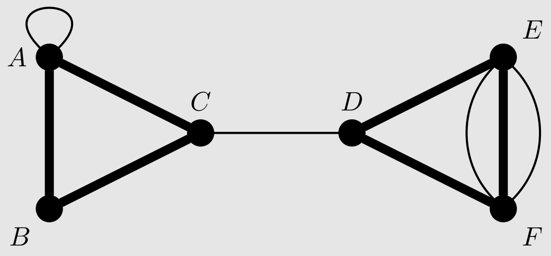


Note that CD is incident to edges AC, BC, DE, and EF.

An example of a path in the graph is $\{AC, CD, DE, EF, FD\}$, as shown below in bold. Another way to convey this path is $ACDEFD$.



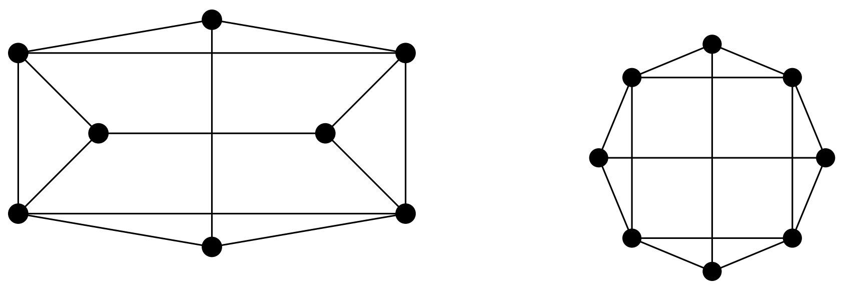
Examples of circuits in the graph are $\{AB, BC, AC\}$ and $\{DE, EF, DF\}$, as shown below in bold. Another way to convey the circuits are ABC and DEF.



Can you find another circuit? (Use the parallel edges).

The best way to get acquainted with graphs is to draw and analyze them. It is not hard to find numerous examples of graphs online, or to simply draw your own. There are also many graphs which are visually pleasing or have special structures which may capture your interest. The reader is encouraged to explore various graphs and to try to find circuits, bridges, the degree of the vertices, etc.

One last point to make is that the same graph can be drawn in several different ways. Here is an example below:



We can also draw our edges all curvy or jagged, it really doesn't matter. What matters is the underlying structure of the graph (the degrees, which vertices are adjacent or nonadjacent.)

Here is a nice list from the book summarizing the basic graph terminology.

Vertices

- **adjacent:** any two vertices connected by an edge.
- **vertex set:** the set of vertices in a graph.
- **odd (even):** an vertex is odd (even) if its degree is odd (even).
- **isolated:** a vertex with no edges connecting to it is isolated.

Edges

- **edge list:** a list of all the edges of a graph.
- **adjacent:** two edges that share a vertex.
- **loop:** an edge that connects a vertex with itself.
- **multiple edges:** more than one edge connecting the same two vertices.
- **bridge:** an edge in a *connected* graph that when removed disconnects the graph

Paths and Circuits

- **path:** a sequence of edges each adjacent to the next, with no edge included more than once, and starting and ending at different vertices.
- **circuit:** same as a path, but starting and ending at the same vertex.
- **Euler path:** a path that covers all the edges of the graph.
- **Euler circuit:** a circuit that covers all the edges of the graph.
- **length:** the number of edges in a path or circuit.

Graphs

- **simple:** a graph with no loops or multiple edges.
- **connected:** there is a path going from any vertex to any other vertex.
- **disconnected:** not connected; consisting of two or more *components.*

5.3 Euler Paths and Circuits

Earlier we mentioned general paths and circuits in a graph. There are special paths and circuits which will prove to be useful to us. Make sure to be familiar with the definitions of paths and circuits before moving on.

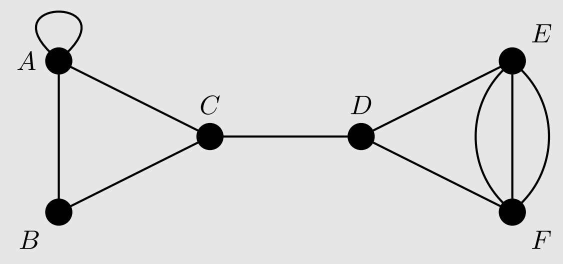
Definition (Euler Path and Euler Circuit)

An *Euler path* is a path that travels **every** edge of a connected graph **exactly one time**.

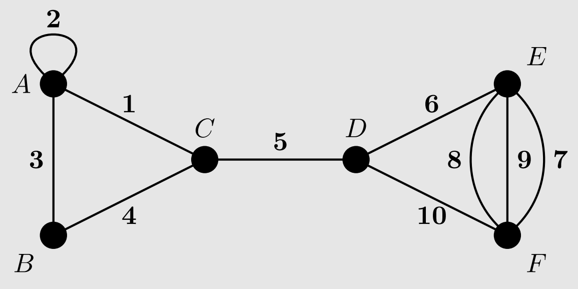
An *Euler circuit* is a circuit is a circuit that travels **every** edge of a connected graph **exactly one time**.

Notice that we have "connected graph" in our definition. This is important. We cannot have a path or circuit that traverses every edge exactly once if our graph is disconnected.

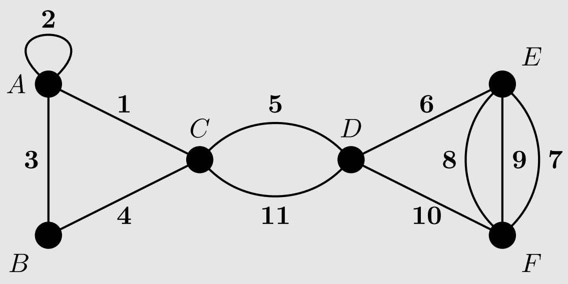
Example. Consider the following graph we are familiar with.



An Euler path is a path that traverses every edge exactly once in the graph. Recall that a path starts and ends at a different vertex. An Euler path for the above graph is (here we will care about the order of the edges to indicate direction): $CA, AA, AB, BC, CD, DE, EF, FE, EF, FD$. Here are the edges labeled in order, starting from C.



It is important to note that not all graphs have Euler paths or Euler circuits. The above graph is an example where there is an Euler path, but no circuit. However, we can modify the graph a bit to guarantee an Euler circuit. Suppose we added another edge between C and D. Then we can extend the previous Euler path into a circuit, as it would begin and end at C, and traverse each edge exactly once.



As we may tell from the example, not all graphs are guaranteed to have an Euler path or circuit (the easiest example to see a graph with no circuit would be to draw 2 vertices and an edge between them as your graph.) Graphs must meet a certain criteria in order to guarantee the existence of an Euler path or circuit.

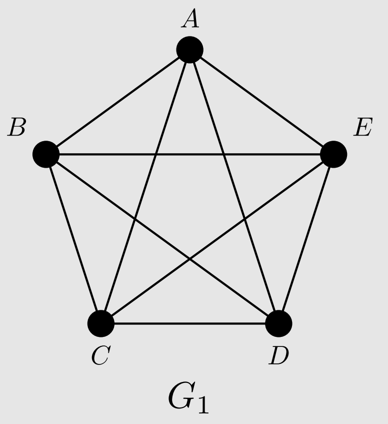
Theorem (Euler Paths and Circuit Requirements)

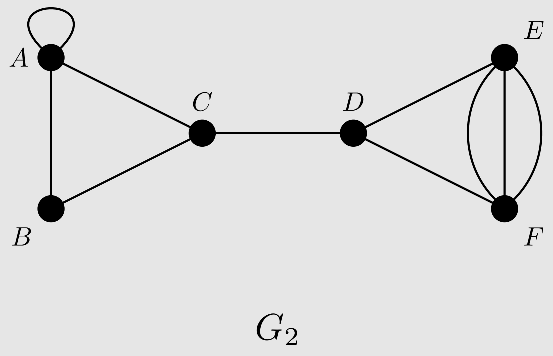
If a graph is connected and all of the vertices have **even degree**, the graph will have an Euler circuit. Any vertex can be the start (end) point of the circuit.

If a graph is connected and has exactly two odd vertices, then it has an Euler path. The path must start at one of the odd vertices and end at the other.

This theorem is very powerful. All we need to do is list the vertex degrees of a graph to be able to tell whether Euler paths or circuits exist in the graph. Note that if you have even one odd vertex, there cannot be an Euler circuit. Why? An Euler circuit must traverse each edge of a graph exactly once, and start and end on the same vertex. Traveling along the circuit, every time you enter a vertex, you leave through a **different** edge. That is, every time you visit a certain vertex along your circuit, exactly 2 edges are used up. In order to use up every edge in the graph exactly once, each vertex must have even degree (since we're using 2 for every visit.) Similar arguments hold for Euler path requirements.

Example. Consider the two graphs, G_1 and G_2 below.

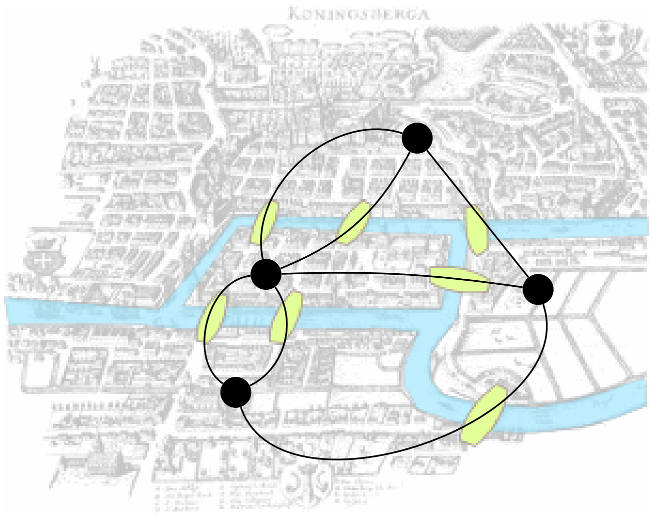




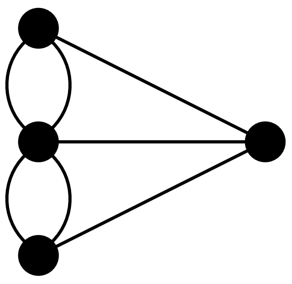
Looking at G_1, the graph on the left, one can quickly realize that each vertex has degree 4. In other words, all the vertices are even, and hence there exists an Euler circuit. Try to find an Euler circuit in G_1.

As for G_2, the graph on the right, we already saw that there is an Euler path by finding one explicitly. We see that C and D are the only vertices that are odd (both have degree 3), and the Euler path we previously found does in fact start at C and end at D. Try testing out the criteria on your own by drawing some graphs.

Let's return to the original problem mentioned at the beginning, The Seven Bridges of Königsberg. Now we can use our graph theory tools to solve the problem. First we must create a graph to model the problem. Let our vertices represent land masses and our edges represent the bridges between the land masses.



Now let's forget about the original picture and just focus on the graph (that's the convenience of it all!)



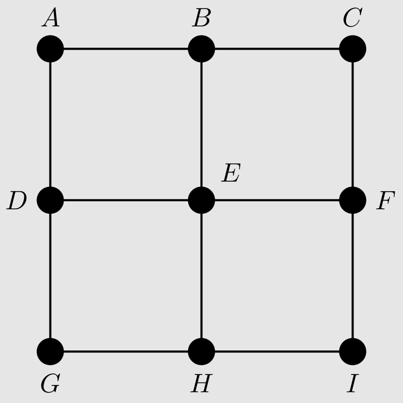
Now the original question asked if it is possible to walk every bridge exactly once. Now that we have a graph, we can use what we know. Notice that every vertex is odd. Specifically we have three vertices with degree 3 and one vertex with degree 5. This graph does not satisfy any of the criteria, and hence there is no Euler path or circuit. In other words, there is no way to walk along the bridges such that we traverse each bridge exactly once.

5.4 Eulerization

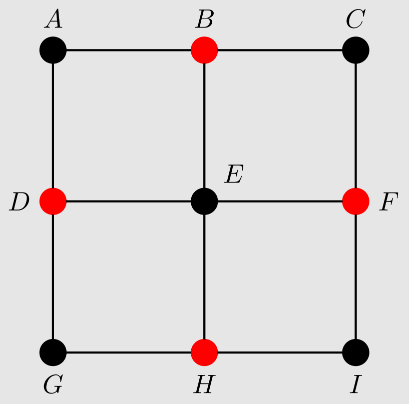
As we've seen with the Königsberg bridge problem, we would like to use graphs to model certain real life scenarios. By translating a landscape into a graph, we were able to reduce the problem, analyze a graph, and come up with a result.

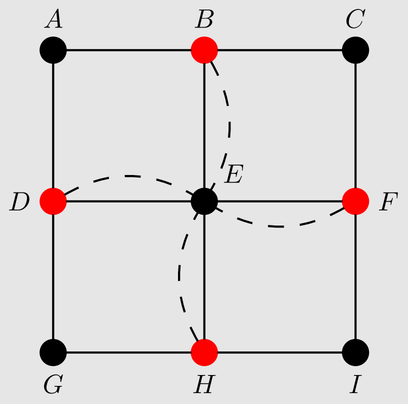
One important application of graphs is optimizing street routes. Imagine that you are a national mail carrier needing to deliver mail across a section of a city. You would want to plan a route that would be the quickest and most cost effective (think wages, fuel costs). It is considered *wasteful* to revisit multiple streets on a given route. This is where graphs come in. We can model scenarios using graphs, then find *optimal* routes by finding Euler circuits in the graphs. If the graph model does not contain an Euler circuit, then we can use a process called *eulerization* in order to obtain a new graph that does. Observe the following example.

Example.
Consider the following 3 by 3 grid.

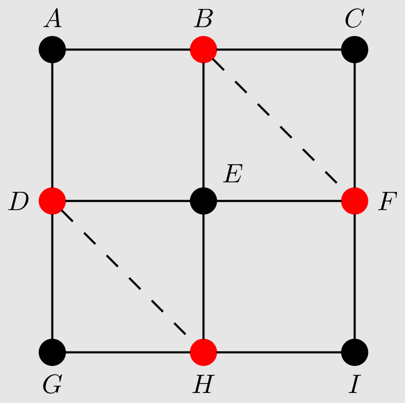


You can think of each edge as a block of the city and each vertex as an intersection (so this is a 2 block by 2 block grid). If we write out the degrees of each vertex, we'll see that there are four vertices with odd degree, namely B, D, F, and H. So, according to Euler's theorems, this graph does not have an Euler circuit nor an Euler path. Now suppose we were a city sanitation worker trying to map out an efficient street cleaning route. How would we plan a route that recrosses the least number of streets as possible? This is where eulerization comes in. First we highlight the vertices giving us trouble, i.e. the odd vertices. Then we **duplicate** edges (dashed) in order to turn the odd degree vertices even.

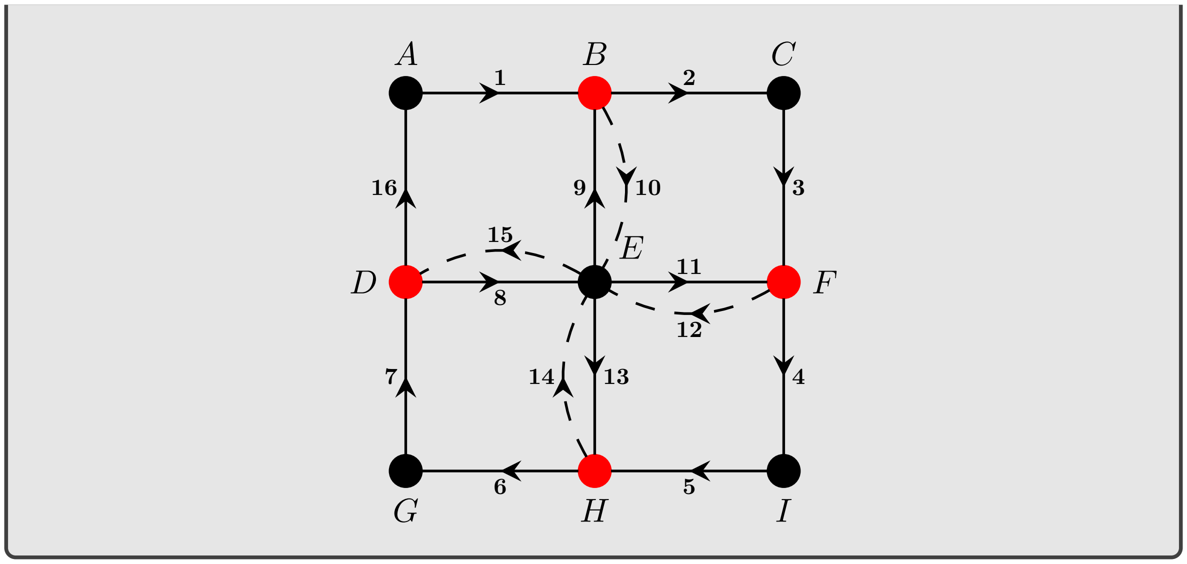




The new graph (on the right) now has all even vertices, and thus has an Euler circuit. Now we must be very careful in which edges we duplicate. Notice that vertex E had even degree prior to any duplicating. Afterwards, since we duplicated four edges adjacent to E, it remains even, while all the odd vertices increase by 1, changing from odd to even. These duplicated edges signify our city sanitation crew traveling the same street more than once, **not** that there is a new street added. Furthermore, we can only duplicate existing edges. For example, the following is **not** an eulerization.



So with our proper eulerization, we can find an Eulerian circuit.



Note that the eulerization in the example is *optimal*. That is, we cannot find an eulerization with 3 or fewer edges. Why is this the case? We have four odd vertices, none of which are adjacent to each other. So we need to duplicate edges adjacent to the odd vertices to bump their degree up from odd to even, and we must do this four times, once for each vertex (1 edge 1 vertex.) Now if there was a pair of odd vertices that were adjacent, we could simply duplicate the edge connecting them (1 edge 2 vertices.)

For small graphs it is easier to tell whether or not we have created an optimal eulerization. The general strategy is

1. If two odd vertices are adjacent, duplicate the edge connecting them. This uses up two vertices while only duplicating one edge.
2. If we can no longer do 1. then we duplicate edges adjacent to the odd vertices, while taking care to not change the degree of even vertices to odd.

Admittedly 2. is the difficult part. In the example above, each of the odd vertices were adjacent to the same vertex, E, and so we were able to duplicate edges in a way that we had 1 duplication per odd vertex, **and** E remained with even degree.