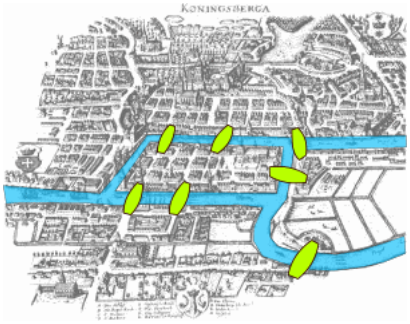
Chapel Hill Math Circle: Graph Theory, Part 1

October 25, 2025

1 An Idea

Consider the following problem: it's 1736 in Königsberg (now Kaliningrad) and you want to have a parade. You'd like to travel across each bridge without retracing any steps. Is this at all possible?



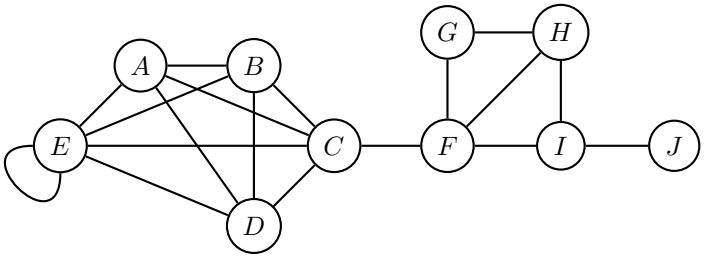
Courtesy of Wikipedia

Much like Leonhard Euler himself, you may find this to be very difficult. To answer this question, we will need to introduce a new mathematical tool called a graph.

2 What is a Graph?

Definition 1. *A Graph is a collection of points, called vertices, and the lines between them, called edges.*

Consider the graph below:



The vertices here are the points $\{A, B, C, D, E, F, G, H, I, J\}$ and the edges are the lines between them. It is sometimes helpful to refer to the edges by noting which vertices they connect, such as AD or EE for the edges connecting vertices A and D or the edge connecting E to itself.

Definition 2. *If two vertices have an edge between them, we called the vertices adjacent. An edge connecting a vertex to itself is called a loop.*

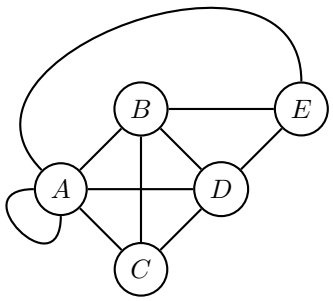
2.1 Exercises

1. Draw a graph with three vertices and four edges, at least one of which is a loop.

2.2 Properties of Graphs

Definition 3. *The number of edges that meet at a vertex X is called the degree of X and is denoted by* $\deg(X)$.

Consider the following graph:



Notice that $\deg(A) = 6$ (loops add 2 to the degree) and $\deg(E) = 3$. Compute $\deg(B)$, $\deg(C)$, and $\deg(D)$.

Definition 4. *Vertices with even degree shall be called Even and those with odd degree shall be called Odd.*

As such, in the preceding graph, notice that A is an Even vertex and E is and Odd vertex. Is there anything that we can say about the sum of all degrees in a Graph? Why is this the case?

2.3 Exercises

1. Can you draw a Graph with four vertices such that all of the vertices are Odd?

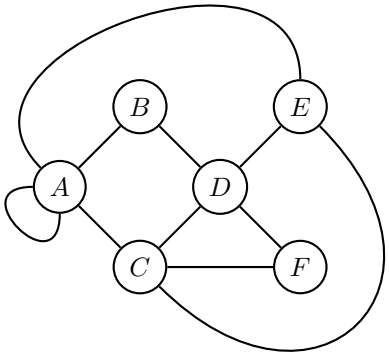
2. Can you draw a Graph with four vertices such that all of them are Even?

3. Challenge: Can you draw a Graph with three vertices such that all of them are Odd? Why or why not?

2.4 Paths and Circuits

Definition 5. *We say that two edges are adjacent edges when they share a common vertex. A sequence of distinct edges, each adjacent to the next, is called a path. Note that we can also represent a path by the sequence of vertices of each edge.*

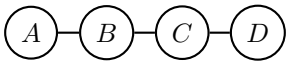
Consider the following graph:



We have a path starting at A and ending at A via the sequence AB, BD, DE, EC, AC, AA. This is not the only such path. Try to find a different one. Can you find one that travels to each vertex exactly once?

Definition 6. *The number of edges in a path is called the length of the path.*

In our above example, we have a path of length 6.



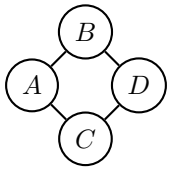
We sometimes refer to this graph as a path of length 3, and similarly we can define paths of any length.

Definition 7. *A circuit is a path that starts and ends at the same vertex.*

Notice that the example above in Definition 5 is a circuit! Can you find a circuit on this graph which visits all of the vertices exactly once?

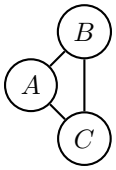
Definition 8. *A cycle is a circuit in which no vertex appears more than once, except the first and last vertex.*

Is the circuit in the example in Definition 5 a cycle? Why/why not? Is the circuit AA a cycle?



We sometimes refer to this graph as a cycle of size 4.

Here is an example of a cycle of size 3:



2.5 Exercises

1. In all of the previously seen graphs, which have circuits that visit every vertex?

2. Do they still have circuits if we also require you to only visit each vertex exactly once?

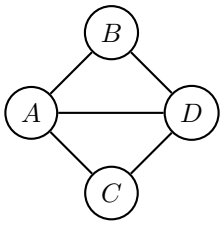
3. We say that a graph is a tree if it is connected[1] and has no cycles. Can you try to draw all possible trees with 4 vertices?

4. What is the minimum number of edges you need in a tree with 5 vertices? Generalize/prove it!

2.6 Euler Paths and Circuits

Definition 9. *An Euler Path is a path that covers all the edges of the graph. An Euler Circuit is a circuit that covers all the edges of the graph.*

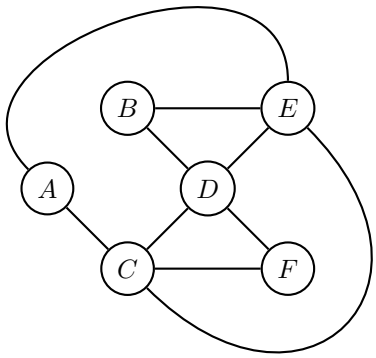
Consider the following Graph:

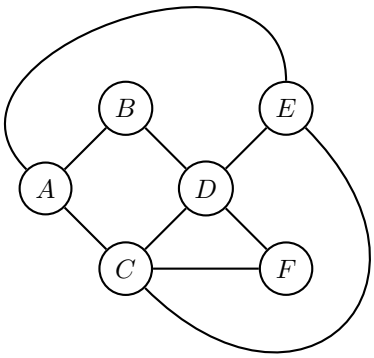


[1] A connected Graph is one in which every vertex can reach any other vertex via the edges.

One Euler Path we could build starts at A and goes AB, BD, DC, AC, AD. Are there others? Try to see if you can find at least one more, if it's possible.

We might be interested in knowing when such Circuits and Paths are actually possible. Consider the following graphs:





Only one of these graphs has an Euler Circuit. Which one is it? What is the Circuit (is it the only one)? Why does it have an Euler Circuit; why doesn't the other one?

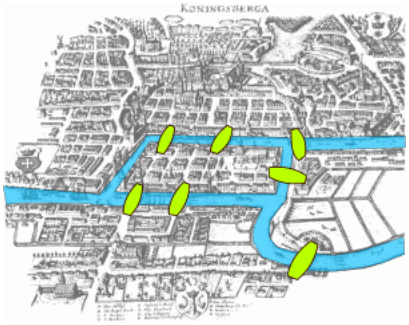
Do you have a conjecture about when an Euler Circuit will occur? If so, what is it?

Theorem 1 (Euler's Circuit Theorem)**.** *If a graph is connected (meaning every vertex is accessible via another using edges) and every vertex is Even, then it has at least one Euler circuit. If the graph has any odd vertices, then it does not have a single Euler circuit.*

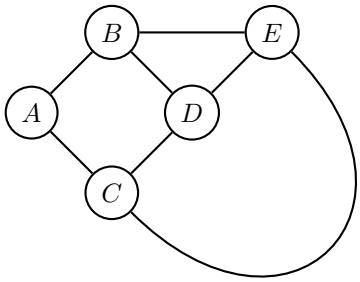
Theorem 2 (Euler's Path Theorem)**.** *If a Graph is connected and has exactly two Odd vertices, then it has an Euler Path. If it has more than two Odd vertices, then it has no Euler Path.*

2.7 Exercises

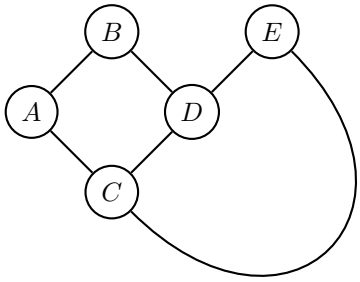
1. Let's think back to our original problem! Can we walk across every bridge in Königsberg exactly once? Why or why not? Hint: Turn the city into a graph!



2. Consider the following graph. Does it have an Euler Circuit? If so, describe a Circuit. If not, does deleting an edge or adding an edge make it have one?



3. Consider the following Graph. Does it have an Euler Path? If no, does deleting or adding an edge force it to have one? If yes, what is the path? Notice anything about the starting and ending vertices?



2.8 Algorithms for Paths and Circuits

It's now easy to tell when a Graph has (or does not have) an Euler Circuit or an Euler Path. But how does one go about finding out what the actual path or circuit is?

Fleury's Algorithm

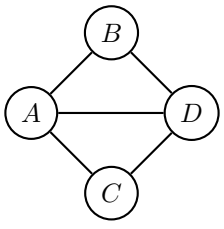
Preliminaries: Make sure that the graph is connected and either has no odd vertices (circuit) or has just two odd vertices (path).

Start: Choose a starting vertex. In the case of a circuit, any vertex will do! In the case of a path, you must choose one of the two odd vertices.

Intermediate steps: At each step, if you have a choice, don't choose an edge of the yet-to-be-traveled part of the graph. However, if you have only one choice, you must take it.

End: When you can't travel any more, the circuit or path is complete. In the circuit case, you should return to the starting vertex. In the path case, you will end at the other odd vertex.

Let's look at an example. The following graph has an Euler path:

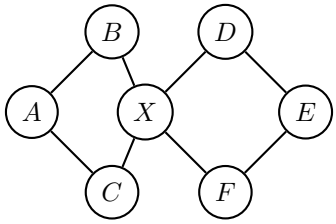


If we want to follow Fleury's algorithm, we need to start at an odd vertex: A or D. Let's pick A. We first go to an unvisited part of the graph, so let's go to B. Thus, the first edge we visit is AB. We have no choice from here, so we go BD. Next, we should not visit a place we haven't been, so we go to DA. There is no choice from here but to do AC followed by CD. Thus, we have constructed an Euler path that starts at A and ends at D. This is not the only Euler path available on this graph (find another one!)

Is this the only such algorithm? No! We could find others. To define a new one called Hierholzer's algorithm, we need to introduce more tools first.

Definition 10. *When two circuits in a graph have no edges in common but share a common vertex v, we say the two circuits are kissing at v.*

In the example below, the circuits defined by AB, BX, XC, CA and DE, EF, FX, XD form two kissing circuits because they kiss at X.



Hierholzer's Algorithm

Step 1: Start with an arbitrary circuit, C_0.

Step 2: Find a kissing circuit to C_0. If there are no kissing circuits, then you are finished. C_0 is itself an Euler circuit of the graph. If there is a kissing circuit to C_0, call it K_0 and let V denote the vertex at which the two circuits kiss.

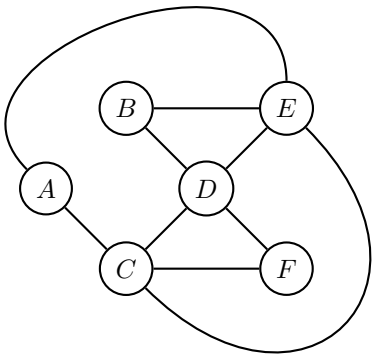
Step 3: Let C_1 denote the circuit obtained by "patching" K_0 to C_0 at vertex V. Now find a kissing circuit to C_1. Again, if there are none, you are done. If there is a kissing circuit to C_1, call it K_1 and let W denote the vertex at which the two circuits kiss.

Step 4: Continue this process forever until there are no more kissing circuits left.

If we consider our above example. C_0 can be the circuit formed by the vertices A, B, X, C. Then K_0 is formed by D, E, F, X where $V = X$. Hence C_1 is the circuit AB, BX, XD, DE, EF, FX, XC, and CA. Since there are no more kissing circuits left, we are done!

2.9 Exercises

1. Consider the following graph and use Fleury's algorithm to find either an Euler Circuit or an Euler Path:



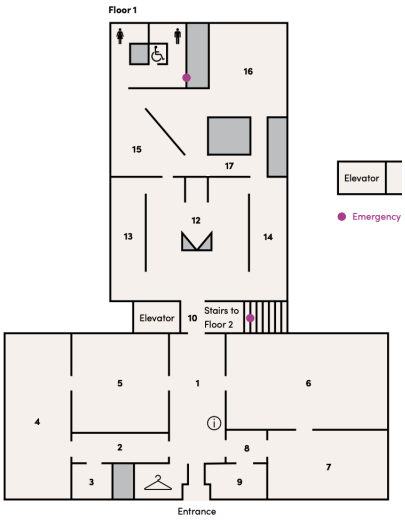
2. Using the same Graph as above, use Hierholzer's algorithm to find an Euler circuit. Do you get the same circuit as you did for the preceding question? Why or why not? Should you expect to get the same answer?

3. Suppose G is a connected graph and every vertex is even. Explain why the following statement is true: If a circuit in G has no kissing circuits, then that circuit must be an Euler circuit.

2.10 Challenge Problems

1. Suppose that you're in a room with a large number of people. You all decide to randomly shake hands and count each shake you perform (you are allowed to shake the same person's hand multiple times and you are allowed to shake your own hand, but you must double count that one!) Why is it the case that the number of people who shake an odd number of hands be an even number?

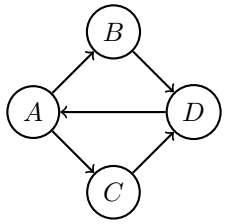
2. Consider this map of the Ackland Art Museum. Is it possible to visit each exhibit exactly once?



3 Network Flows

Definition 11. *A directed graph is defined in the same way as a graph (a collection of vertices and edges), except that the edges now come with one of two orientations: An edge joining A and B can be oriented from A to B, or from B to A, except for loops (i.e edges from a vertex to itself), which just have one possible orientation.*

The following is a graph from earlier that is now directed:

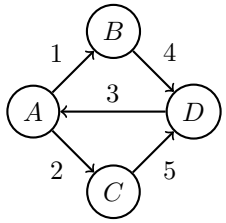


Definition 12. *A network is a directed graph in which we assign a nonnegative number $c(e)$ to each edge (the **capacity** of the edge), and there are 2 distinct vertices s (the source) and t (the sink).*

A flow f is an assignment of a nonnegative number $f(e)$ to each edge, and we call the flow feasible if $f(e) \leq c(e)$, and the incoming flow to each vertex is equal to the outgoing flow for each vertex (excluding the source and sink)

The **value** of a flow is the net flow into the sink i.e the total flow entering the sink - the total flow exiting the sink.

Consider the network above, where the edge e is labeled by $f(e)$:



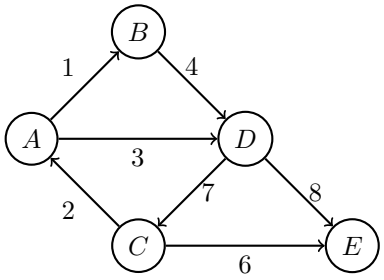
The value of the flow of vertex D is 6, since the flow entering the sink, D is $5 + 4 = 9$ and the flow exiting is 3, hence $9 - 3 = 6$.

3.1 Exercises

1. For the network above (with source A and sink D), compute the flow of each vertex

2. What is the sum of all the total flows? Is this special to this network? Can you show that it is true for all networks?

3. Consider the network below (with source A and sink D) with its labeled capacity. Find any feasible flow that is distinct from the zero flow.



Definition 13. *Given a network with source s and sink t, a **cut** is a partition of the set of vertices of the network into 2 sets S and T so that $s \in S$ and $t \in T$.*

Given such a partition, let $[S, T]$ denote the set of edges with tail in the set S, and head in the set T.

Definition 14. *The **capacity** of the cut defined by the sets S and T is the sum of all capacities of edges in $[S, T]$*

Note that the capacity of any cut is unaffected by edges that start at T and end at S!

4 Exercises

1. In the network above (source A, sink E): find as many examples of cuts as you can, and find their capacities.

2. Try to find a cut with the lowest possible capacity. Can you show that this cut has the lowest possible capacity?

3. Find feasible flows with the highest possible value. What do you observe?

We can make a similar statement for any such network due to:

Theorem 3. *(Ford–Fulkerson) For any network, the maximum possible value of a feasible flow is equal to the minimum possible capacity of a cut.*

There is also an explicit algorithm to find a cut with the minimum possible value due to Ford and Fulkerson, and we will introduce this in 2 steps:

First, given a feasible flow f, we say what it means to have an f-**augmenting path** in the network. Given such an augmenting path, we can change the flow f to get a feasible flow f' so that the value of f' is higher than the value of f. If there are no more f-augmenting paths, then the flow f can be shown to be a maximum flow (using tools from optimization), and the Ford–Fulkerson algorithm returns a cut with capacity = value(f).

Given a flow f on a network, an $f-$augmenting path P is a path from the source to the sink in the underlying graph (Since each network is a directed graph, the underlying graph is obtained by forgetting about the orientations of the directed edges) of the network so that each edge e of the network in P satisfies:

1. If the path P follows e in the same direction, then $f(e) < c(e)$.
2. If the path P follows e in the opposite direction, then $f(e) > 0$.

We claimed that any f-augmenting path P leads to a feasible flow f' so that the value of f' is greater than the value of f. To get this new flow, we assign numbers $g(e)$ to every edge in the path as follows:

1. If P follows e in the forward direction, let $g(e) := c(e) - f(e)$.
2. If P follows e in the backward direction, let $g(e) = f(e)$.

Let x be the minimum value of $g(e)$ for every edge e in P. This value is called the **tolerance** of the path.

The new flow f' is obtained by:

1. If e is not on the path P, then $f'(e) := f(e)$
2. If P follows e in the forward direction, then $f'(e) := f(e) + x$
3. If P follows e in the opposite direction, then $f'(e) := f(e) - x$

This new flow f' is feasible and

$$\text{value}(f') = \text{value}(f) + x$$

Ford–Fulkerson Algorithm (Challenging)

Preliminaries: A feasible flow f.

Step 0: Define two sets of vertices: One called R and one called S, where R contains a single element s, and S is empty.

Step 1: Pick any vertex v in the network that is in the set R but not S.

Step 2: Pick an edge of the form vw, where the edge is from v to w. If there is an edge vw with $f(vw) < c(vw)$ (i.e the flow is less than the capacity) and w is not in R, then add w to R.

Step 3: Repeat Step 2 for every edge of the form vw.

Step 4: Pick an edge of the form uv, where the edge is from u to v. If $f(uv) > 0$, then add u to R.

Step 5: Repeat Step 4 for every edge of the form uv.

Step 6: Add v to S.

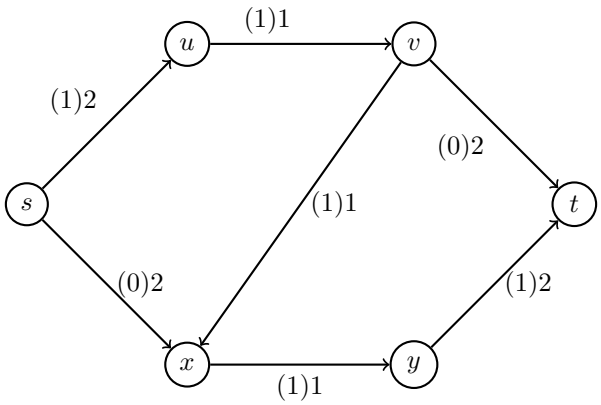
Step 7: If t is in R, then trace the path from t backwards to s: this can be shown to be an $f-$augmenting path. Then terminate the algorithm.

Step 8: If $R = S$, then the cut defined by S and all the vertices not in S can be shown to be a minimum cut. Then terminate the algorithm.

Step 9: If neither of the above 2 cases happens, then start again from Step 1.

This algorithm can be shown to terminate in finitely many steps, and the algorithm can only terminate at step 7 or step 8. In each of these steps, we end up with a minimum cut, or an augmenting path. If the algorithm terminates at step 8, we have a minimum cut, and the capacity of this cut is the value of the flow. If the algorithm terminates at step 7, we have an augmenting path for the flow. By augmenting the flow as detailed above, we get a new feasible flow and we can run the Ford–Fulkerson algorithm repeatedly, till the output is a minimum cut.

Exercise: Run the Ford–Fulkerson algorithm on the following network (the source is s, the sink is t), the non-parenthesized terms at each edge indicate the capacity, and the parenthesized quantities are the flows for each edge.



5 Extra Space