Queen's University
School of Computing

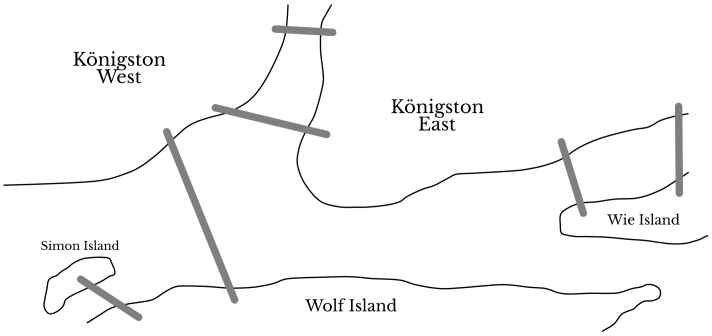
CISC 203: Discrete Mathematics for Computing II
Module 12: Graph Theory
Fall 2021

This module corresponds to the following sections from your textbook:

47. Fundamentals of Graph Theory
48. Subgraphs
49. Connection
50. Trees
51. Eulerian Graphs

1 Traveling Around Königston

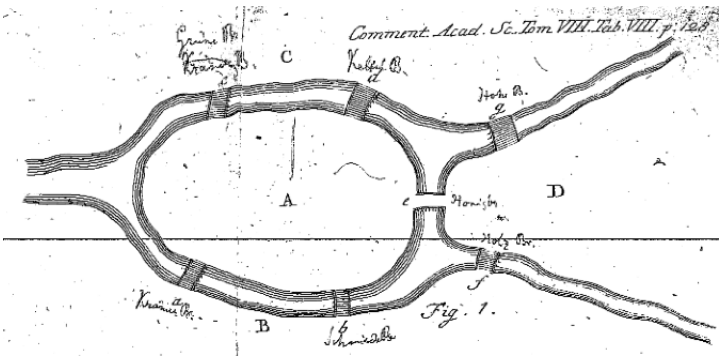
Imagine that, one day, you take a trip to the City of Königston. (This city looks a lot like Kingston, but it has a large German population, hence the name.) Königston consists of five regions: Königston West, Königston East, Wolf Island, Wie Island, and Simon Island. They are arranged as follows:



To move between these five regions, you can take either a bridge or a ferry. Since you want to get the full Königston experience, but you don't want to stay out too late, you decide to cross each bridge and ride each ferry exactly once.

During your trip, the local government announces a new ferry service between Wolf Island and Wie Island. Naturally, you feel the need to redo your tour. However, with this new ferry, it is no longer possible for you to cross each bridge and ride each ferry exactly once! How could such a small change have such a big effect?

In 1736, the Swiss mathematician Leonhard Euler thought about a remarkably similar problem called the "Bridges of Königsberg". The downtown area of the City of Königsberg (now known as Kaliningrad, Russia) is divided into four parts by a river, and each of these parts are connected to one another by a total of seven bridges. In his paper, Euler drew the following map to illustrate the layout of the city:



Interestingly, Euler found that, no matter what path you take across the bridges, you cannot cross each of the bridges exactly once. With this simple puzzle, Euler inadvertently kicked off the entire study of **graph theory**—something that affects both mathematicians and tourists to this day.

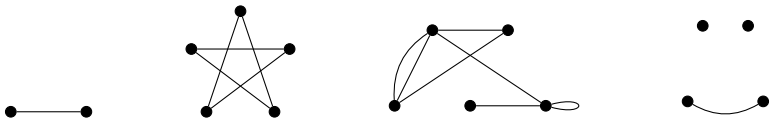
2 Graphs

At its core, a **graph** is not a complicated concept. In fact, graphs are only made up of two sets.

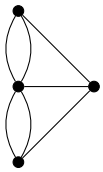
Definition 1 (Graph)**.** A graph $G = (V, E)$ consists of a nonempty finite set of vertices V and a set of edges E, where each element of E is a pair $\{u, v\}$ of vertices $u, v \in V$.

In other words, a graph is just a set of **vertices** and a set of **edges** that link those vertices.

Example 2. Each of the following are graphs:



Example 3. We can model the Bridges of Königsberg problem with the following graph:



Example 4. Let

$$G = \Big(\{1, 2, 3, 4, 5, 6, 7\}, \big\{\{1, 2\}, \{1, 3\}, \{2, 3\}, \{3, 4\}, \{5, 6\}\big\}\Big).$$

Here, the set of vertices V is $\{1, 2, 3, 4, 5, 6, 7\}$ and the set of edges E contains five two-element subsets of V: $\{1, 2\}$, $\{1, 3\}$, $\{2, 3\}$, $\{3, 4\}$, and $\{5, 6\}$.

Given a graph G, we denote its set of vertices as $V(G)$ and its set of edges as $E(G)$.

2.1 Properties of Graphs

You may have noticed that Definition 1 is worded in a very general way, and it allows for strange things that might make our graphs complicated, like multiple edges between the same vertices or edges joining a vertex to itself (called **loops**). If we like, we can refine our definition to consider restricted graphs that don't allow for such strange things. We call such graphs **simple**; these are the kinds of graphs we will be dealing with most of the time.

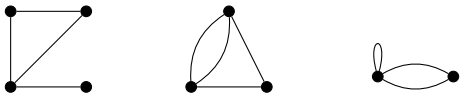
Definition 5 (Simple graph)**.** A graph $G = (V, E)$ is simple if,

1. given two edges $\{u_1, v_1\}, \{u_2, v_2\} \in E$, either $u_1 \neq u_2$ or $v_1 \neq v_2$ (no multiple edges); and
2. if $\{u, v\} \in E$, then $u \neq v$ (no loops).

Note that the graph in Example 4 is a simple graph.

In the literature, graphs with multiple edges are called **multigraphs** and graphs with both multiple edges and loops are called **pseudographs**. However, unless mentioned otherwise, in these notes when discussing graphs we refer to simple graphs by default.

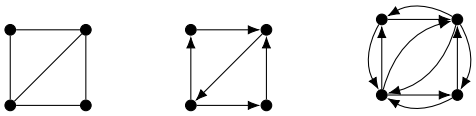
Example 6. The first graph is a simple graph. Neither the second nor third graphs are simple graphs. (The second graph is a multigraph and the third graph is a pseudograph.)



Both Definitions 1 and 5 assume that the set of edges E consists of unordered pairs, so if an edge exists between vertices u and v, then another edge implicitly exists between v and u (think of a road with two opposing lanes). If we instead require that E consists of ordered pairs, then the existence of an edge between vertices u and v does not necessarily imply the existence of an edge between v and u. We say that such graphs are **directed**, because the direction of each edge in the set E matters when we move between vertices (think about one-way roads, for example). Directed graphs are not discussed in the textbook.

As you would expect, we say that graphs where the direction of edges does not matter are **undirected**.

Example 7. Out of the following graphs, the first graph is undirected and both the second and third graphs are directed. The third graph is the directed equivalent of the first graph.



From this point onward, when we mention graphs we will refer exclusively to undirected graphs.

2.2 Properties of Vertices and Edges

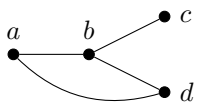
Now that we have established terminology for talking about graphs in general, let's look at some terminology for talking about particular properties of a graph. In this section, assume that we are given an arbitrary graph $G = (V, E)$.

Definition 8 (Adjacent, Neighbourhood)**.** Given vertices $u, v \in V$, if $\{u, v\} \in E$ (that is, if an edge exists between u and v), then we say that u and v are **adjacent**, and we can express this with the notation $u \sim v$. We also say that the edge $\{u, v\}$ is **incident** to the vertices u and v. The set of all vertices adjacent to some vertex u is called the **neighbourhood** of u.

Definition 9 (Degree)**.** We say that a vertex u is of **degree** k if it is adjacent to k other vertices in the graph. We denote the degree of a vertex u by $\deg(u)$. We may formally define the degree of a vertex u as

$$\deg(u) = |\{v \in V \mid \{u, v\} \in E\}|.$$

Example 10. In the following graph, we see that vertex a is adjacent to vertices b and d; vertex b is adjacent to vertices a, c, and d; vertex c is adjacent to vertices b and c; and vertex d is adjacent to vertices a and b. From this, we see that $\deg(a) = 2$, $\deg(b) = 3$, and $\deg(d) = 2$.



Definition 11 (Regular graphs)**.** If each vertex of a graph G has the same degree, then G is a **regular** graph. A regular graph G is sometimes called ***k*-regular** if the degree of each vertex is k.

Using what we know about vertices, edges, and degrees, we can now prove our first small result in graph theory: if we sum the degrees of every vertex in a graph G, our total will be exactly twice the number of edges of G.

Theorem 12 (Handshake lemma)**.** *Given a graph $G = (V, E)$,*

$$\sum_{u \in V} \deg(u) = 2 \cdot |E|.$$

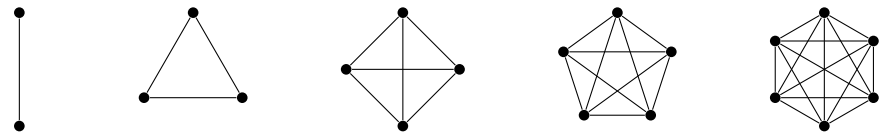
Proof. Each edge $\{u, v\} \in E$ contributes one to the degrees of the vertices u and v. In the case where the edge is not a loop, adding one to both $\deg(u)$ and $\deg(v)$ double-counts that edge. In the case where the edge is a loop, adding two to $\deg(u)$ also double-counts that edge. Therefore, the summation of the degrees of all vertices in G will be exactly twice the number of edges in G. □

We call the previous result the "handshake lemma" because of the fun observation that, if n people shake hands with one another at a party, then exactly $2n$ hands will be shaken. (As an exercise, verify this at the next party you attend after the pandemic.)

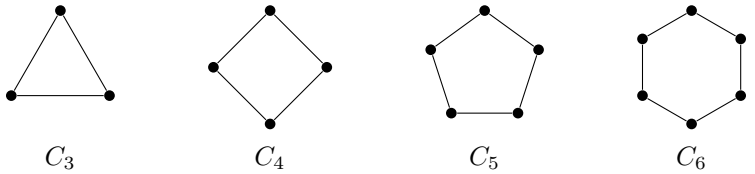
2.3 Special Graphs

Throughout our study of graph theory, we will see certain special graphs come up often in definitions, theorems, and examples. Let us look at a few of these special graphs now.

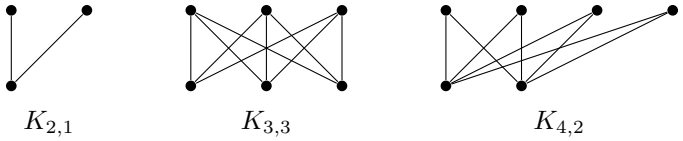
Definition 13 (Complete graph)**.** The **complete graph** on n vertices, denoted K_n, consists of n vertices and $\frac{n(n-1)}{2}$ edges joining each vertex to every other vertex exactly once.



Definition 14 (Cycle graph)**.** The **cycle graph** on $n \geq 3$ vertices, denoted C_n, consists of n vertices and n edges that join each vertex u_i to the vertex u_{i+1} for all $1 \leq i \leq n$. The vertex u_n is joined to the vertex u_1 to complete the cycle.



Definition 15 (Bipartite graph)**.** A **bipartite graph** is a graph whose vertex set V can be partitioned into two subsets V_1 and V_2 such that each edge joins a vertex in V_1 to a vertex in V_2. If V_1 contains m vertices and V_2 contains n vertices, and if each vertex in V_1 is joined to every vertex in V_2, then we say that the graph is a **complete bipartite graph**, denoted $K_{m,n}$.

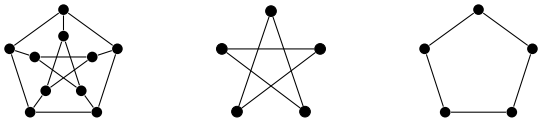


3 Subgraphs

Definition 16 (Subgraph)**.** If we have two graphs $G = (V, E)$ and $H = (V', E')$ where $V' \subseteq V$ and $E' \subseteq E$, then we say that H is a **subgraph** of G.

In other words, if H is a subgraph of G, it means that H is a graph consisting of vertices and edges from the graph G.

Example 17. The leftmost graph, known as the **Petersen graph**, is used frequently as an example or counterexample when proving results in graph theory. We will see the Petersen graph again later in these notes, but for now, observe that the Petersen graph contains the two rightmost graphs as subgraphs.



Definition 18 (Spanning Subgraphs)**.** If H is obtained from G by deleting only edges, then H is called a **spanning subgraph** of G.

If we delete an edge from a graph G, we denote that graph as $G - e$. We see that this does not affect the number of vertices in the graph, so $V(G-e) = V(G)$ (and trivially, $E(G-e) = E(G)-\{e\}$). More generally, if H is a spanning subgraph of G, this means that $V(G) = V(H)$.

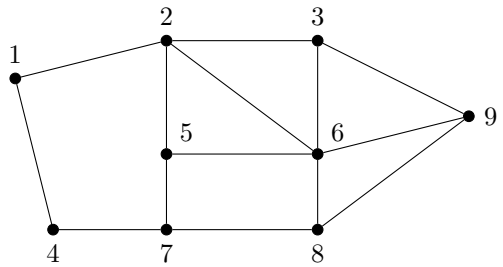
Deleting a vertex from a graph, which we denote as $G-v$ is less straightforward. We can see that $V(G-v) = V(G) - v$. But when removing a vertex, we also need to remove any edges that are incident on that vertex. So, $E(G - v) = \{e \in E(G) : v \notin e\}$. In other words, the edges in $G - v$ are all the edges of G that are not incident on v.

Definition 19 (Induced Subgraphs)**.** Let H be a subgraph of G where the $V(H)$ is a subset of $V(G)$. Furthermore, let $E(H)$ be the set of all legally allowable edges from $E(G)$, i.e., all the edges that connect two vertices in $V(H)$. Then H is called an **induced subgraph** of G.

Subgraphs are particularly useful when we discuss certain problems in graph theory, since we may be able to reduce a given graph to a subgraph that is easier to use or understand.

Definition 20 (Clique, clique number)**.** Let G be a graph. A subset of vertices $S \subseteq V(G)$ is called a **clique** provided any two distinct vertices in S are adjacent. The **clique number** of G, denoted $\omega(G)$, is the size of a largest clique.

Example 21. Consider the following graph.



Here are two cliques: $\{1, 4\}$, $\{2, 5, 6\}$. Can you find more? What is the clique number of this graph? Compare your answers with Example 48.8 of the textbook.

Definition 22 (Independent set)**.** Let G be a graph. A subset of vertices $S \subseteq V(G)$ is called an independent set provided no two vertices in S are adjacent. The **independence number** of G, denoted $\alpha(G)$, is the size of a largest independent set.

Example 23. Consider the graph in Example 21. Here are two independent sets: $\{1, 3, 5\}$, $\{4, 6\}$, $\{4\}$. Can you find more? What is the independence number of this graph? Compare your answers with Example 48.10 of the textbook.

Definition 24 (Complement)**.** Let G be a graph. The **complement** of G, denoted $\overline{G}$, is the graph formed by removing all the edges of G and replacing them by all possible edges that are not in G.

Exercise

Draw the complement of the graph in Example 21. Compare with the graph G at the top of p. 341 of the textbook.

The following theorem clarifies the relationship between cliques and independent sets.

Theorem 25. *Let G be a graph. A subset of $V(G)$ is a clique of G if and only if it is an independent set of $\overline{G}$. Furthermore, $\omega(G) = \alpha(\overline{G})$ and $\alpha(G) = \omega(\overline{G})$.*

Proof. If the reader is sufficiently alert, they will notice that this follows from Definition 20, Definition 22, and Definition 24. □

4 Connectedness

Up to this point, we have used terms like "joined" to denote the property of two vertices u and v being related by an edge $\{u, v\}$. Why didn't we use a more natural term instead, like "connected"? As it turns out, in graph theory, the term "connected" has a very specific meaning, so we must take care not to overload that word with too many meanings.

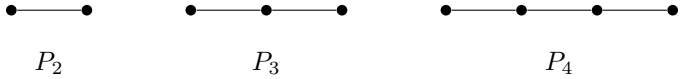
In this section, we will discuss how to model certain problems for which graphs are highly suitable: namely, problems that involve finding a way from one location to another location. In a graph, such problems reduce from finding a sequence of edges from one vertex to another vertex, and such a sequence is called a **path**.

Definition 26 (Walk)**.** A **walk** in a graph is a list of vertices such that each vertex in the list is adjacent to the next vertex in the list. The length of a walk $W = (v_0, v_1, \ldots, v_l)$ is l. Note that there are $l + 1$ vertices on a walk of length l.

Definition 27 (Concatenation)**.** Suppose $W_1 = (v_0, v_1, \ldots, v_l)$ and $W_2 = (w_0, w_1, \ldots, w_k)$ and suppose that $v_l = w_0$. The **concatenation** of W_1 and W_2 is $W_1 + W_2 = (v_0, v_1, \ldots, v_l = w_0, w_1, \ldots, w_k)$.

Definition 28 (Path)**.** A **path** in a graph is a walk in which no vertex is repeated.

The notion of a path gives rise to another class of special graphs called **path graphs** on n vertices, denoted P_n, which consist of n vertices and $n - 1$ edges that join each vertex u_i to the vertex u_{i+1} for all $1 \leq i \leq n$. In a path graph with $n \geq 2$ vertices, exactly two vertices have degree 1 and the remaining $n - 2$ vertices each have degree 2.



Recalling the motivation for this section, we now investigate the reason why we don't refer to two vertices joined by an edge as "connected". **Connectedness** in a graph is a very particular property. In a networked computer lab, each of the computers can send and receive data with one another since they are all connected to the same network. Likewise, a GPS unit can find driving routes between any two cities that are connected to the highway system.

In general, connectedness in a graph implies that you can reach any vertex from any other vertex in the graph. We define it more formally as follows.

Definition 29 (Connectedness)**.** A graph $G = (V, E)$ is connected if, for every pair of vertices $u, v \in V$, there exists a walk in G between u and v.

Thus, the graphs depicted in Examples 34 and 35 are both connected.

Although Definition 29 only tells us that there exists a walk between any pair of distinct vertices of a connected graph, we can in fact strengthen this result via a small proof.

Theorem 30. *Given a graph $G = (V, E)$, if G is connected, then there exists a path between every pair of distinct vertices $u, v \in V$.*

Proof. Since G is connected, there exists at least one walk from u to v. Let $\{e_1, e_2, \ldots, e_n\}$ denote the sequence of edges that form such a walk.

If this walk is a path, then we are done. Otherwise, there exists some subsequence of repeated edges $\{e_i, \ldots, e_{j-1}\}$ where $e_i = e_j$. We may delete this subsequence of repeated edges to obtain a new sequence of edges $\{e_1, e_2, \ldots, e_{i-1}, e_j, \ldots, e_n\}$ that gives a shorter walk, and we may repeat this process until the resultant walk is a path. □

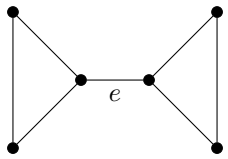
4.1 Connected Components

Given an arbitrary vertex u of a graph, we say that all of the vertices reachable from u belong to the same **connected component** of the graph. Graphs may contain one or more connected components as subgraphs, but the graph itself is connected only if it consists of exactly one connected component.

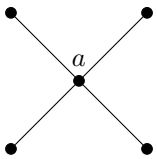
If we take a graph G with m connected components, and if removing a vertex from G produces a new graph G' with more than m connected components, then we call that vertex a **cut vertex**. If the same outcome occurs after removing an edge, then we call that edge a **cut edge**.

The problem of determining cut vertices and cut edges in a connected graph is important, since it tells us quite a bit about the structure of the graph. For instance, determining the cut edges in a graph that models a highway system will reveal weak points in the system; if the road corresponding to a cut edge becomes impassable, then traffic won't be able to get from one point to another point.

Example 31. The following graph is connected, since there exists a path between every pair of vertices. If we delete edge e from the graph, then the graph will consist of two connected components, but the graph itself will no longer be connected.



Example 32. The following graph is connected. If we delete vertex a from the graph, then each edge will disappear and the graph will consist of four connected components: one for each vertex.



4.2 Applications of Connectedness

A number of famous problems (or infamous problems, depending on who you ask) in computer science relate closely to the notions of paths, connectedness, and cuts in graphs. A selection of the most well-known are as follows.

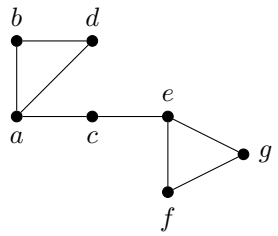
- The **graph reachability** problem asks whether a path exists from one given vertex to another given vertex within a graph. We can solve this problem by simply finding the connected components of the graph.
- The **shortest path** and **longest path** problems, as the name suggests, ask you to find the shortest path and longest path, respectively, between two vertices in a graph. Here, the "length" of a path may be measured by the number of edges or by weights associated with each edge. An abundance of algorithms exist for solving the shortest path problem, including Dijkstra's algorithm (which runs in time proportional to $|V|\log(|V|) + |E|$) and the Bellman-Ford algorithm (which runs in time proportional to $|V| \cdot |E|$). Interestingly, however, the longest path problem has no known efficient solution. These algorithms are very important for routing in computer networks.
- The **min-cut** and **max-cut** problems ask you to find the minimum number of edges and maximum number of edges, respectively, that partition the vertices of a graph into two subsets. To find a minimum cut, we can use techniques such as the Edmonds-Karp algorithm, which runs in time proportional to $|V| \cdot |E|^2$. Unfortunately, we are not so lucky when it comes to finding a maximum cut: this problem once again has no known efficient solution.

5 Trees

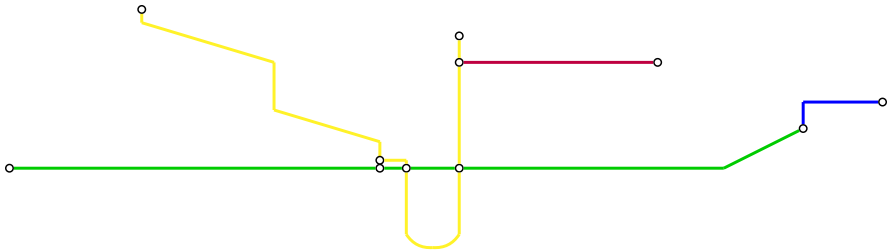
Reading Definition 28, you might have implicitly assumed that paths must be linear or "open". However, nothing in our definition requires that to be the case. Imagine, for example, a graph representing an electrical circuit, where vertices denote components and edges denote wires. In order for the electrical circuit to work, it must be closed; that is, starting from one component, we must be able to traverse each of the wires and end up at the same component. Traversing the wires is equivalent to following a path through a graph and, appropriately enough, we call such a "closed path" a **cycle** (also sometimes called a **circuit**).

Definition 33 (Cycle)**.** A walk of length $n > 2$ from a vertex u to a vertex v is a cycle if $u = v$ and no other vertex is repeated.

Example 34. In the following graph, we see that (among others) there exist paths b–a–c–e, d–a–b, and f–e–c; and cycles a–b–d–a and e–f–g–e.



Example 35. The following graph depicts a simplified version of the Toronto subway system, where vertices denote terminal stations and stations that allow for transfers between lines.



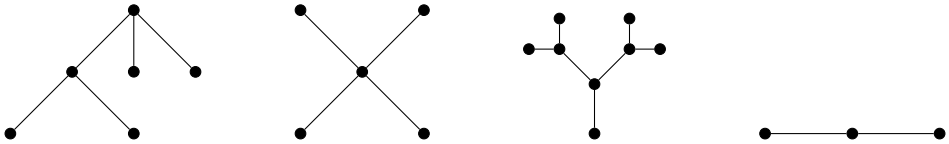
In this graph, each edge colour represents a path between terminal stations on a particular line. A number of other paths exist between different lines as well. The graph also contains a cycle.

Definition 36 (Acyclic, forest)**.** A graph containing no cycles is called **acyclic**, or a **forest**.

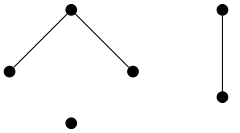
Definition 37 (Tree)**.** A **tree** is a connected, acyclic graph.

Observe from the above definitions that a tree is a connected forest.

Example 38. Each of the following graphs are trees.



The following graph, containing three connected components, is a forest.

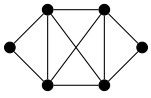


Definition 39 (Leaf)**.** A **leaf** of a graph is a vertex of degree 1.

Definition 40 (Spanning tree)**.** Let G be a graph. A **spanning tree** of G is a spanning subgraph of G that is a tree.

A spanning tree gets its name from the fact that it spans the vertices of a graph.

Example 41. Consider the following graph:



Some spanning trees contained in this graph are as follows. (This is not a complete list!)

Observe that we don't refer to a spanning tree of a graph as *the* spanning tree. This is because spanning trees need not be unique; a graph may contain many different spanning trees, and there are methods of counting the number of spanning trees a given graph contains.

Before we continue, it is worth mentioning a small result that relates spanning trees to a certain property of their associated graph.

Theorem 42. *A graph G is connected if and only if G contains a spanning tree.*

Proof. Omitted. See proof of Theorem 50.11 on p. 354 of the textbook. □

Below, we give a number of equivalent characterizations of a tree.

Theorem 43. *Given a graph T, the following statements are equivalent.*

1. *T is a tree.*
2. *There exists exactly one path between any pair of vertices in T.*
3. *T is connected and every edge of T is a cut edge.*
4. *T is acyclic and the addition of any edge to T creates a cycle.*

Proof. To prove the equivalence of all of these statements, we will show that one statement implies the next statement in a chain of proofs.

$1 \Rightarrow 2$: Let T be a tree, and suppose that there exists some pair of vertices u and v that are connected by two distinct paths, say p_1 and p_2. We can reach v from u by following path p_1, and we can return to u from v by following path p_2. However, this implies that T contains a cycle, which contradicts our assumption that T is a tree. Therefore, no two vertices of T can be connected by more than one path.

$2 \Rightarrow 3$: Let T be a graph where every pair of vertices is connected by exactly one path. Then T is connected. Let e be an arbitrary edge of T, and let $T - e$ be the tree produced by removing e from T. If $T - e$ is connected, then there exists a path between the vertices that were previously incident to e. Since e itself was also a path between these vertices, then there must have existed more than one path between some pair of vertices in T, which contradicts our assumption. Therefore, $T - e$ is not connected and, since e is an arbitrary edge, any edge removed from T is a cut edge.

$3 \Rightarrow 4$: Let T be a connected graph where every edge is a cut edge. Then every pair of vertices in T is connected by exactly one path, so T is acyclic. Suppose we add an edge e to T between two arbitrary vertices u and v. Since T is connected, a path already exists between u and v, so the addition of e to T creates a cycle between u and v. Since u and v are arbitrary vertices, the addition of any edge to T creates a cycle.

$4 \Rightarrow 1$: Let T be an acyclic graph where the addition of any edge to T creates a cycle. To show that T is a tree, all we need to do is show that T is connected. Suppose that T is not connected. Then there exists some pair of vertices u and v such that no path exists between these vertices. However, this means that we can add an edge between u and v without creating a cycle, which contradicts our assumption. Therefore, T must be connected, and thus T is a tree. □

There exist other equivalent characterizations of a tree beyond the four characterizations we proved here. For example, it is possible to show that:

- A graph T is a tree if and only if T is connected and T contains n vertices and $n - 1$ edges; or
- A graph T is a tree if and only if T is acyclic and T contains n vertices and $n - 1$ edges.

Here, we will prove a statement simpler than the two above. First, however, we require a small intermediate result.

Lemma 44. *If T is a tree and v is a leaf of T, then $T - v$ is a tree.*

Proof. To show that $T - v$ is a tree, we must show that $T - v$ is both connected and acyclic.

We can see immediately that $T - v$ is acyclic, since if it were not, then our original tree T would contain a cycle. Since T is acyclic, $T - v$ must also be acyclic.

Next, consider two arbitrary vertices u and w from $T - v$. If $T - v$ is connected, then there exists a path between u and w. Moreover, this path cannot contain the removed vertex v; if it did, then v would have to be located somewhere between u and w on the path, meaning that v would have degree 2. This contradicts

our assumption that v is a leaf. Therefore, for any pair of arbitrary vertices in $T - v$, there exists a path between those vertices that does not include v, and so $T - v$ is connected. □

With this lemma, we can prove the aforementioned statement.

Theorem 45. *A tree T with n vertices contains $n - 1$ edges.*

Proof. We will prove the claim using the principle of mathematical induction on the number of vertices of T. Let $P(n)$ be the statement "a tree T with n vertices contains $n - 1$ edges".

When $n = 1$, the tree contains one vertex and thus has no edges. Therefore, $P(1)$ is true.

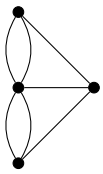
Assume that $P(k)$ is true for some $k \in \mathbb{N}$, i.e., a tree with k vertices contains $k - 1$ edges.

We now show that $P(k + 1)$ is true; that is, a tree with $k + 1$ vertices contains k edges. Suppose v is a leaf of T. By Lemma 44, removing v from T produces a tree $T - v$ with k vertices. By our inductive hypothesis, $T - v$ contains $k - 1$ edges. It follows that T must contain k edges when we add v back into the tree.

Therefore, $P(k + 1)$ is true. By the principle of mathematical induction, $P(n)$ is true for all $n \in \mathbb{N}$. □

6 Eulerian Graphs

Let's return to the example that motivated our study of graph theory: the Bridges of Königsberg problem. Recall that if we abstracted away the notions of land, water, and bridges, we were left with a graph that looked like the following:



When Euler studied this problem, he wanted to know whether there existed a route through Königsberg that traversed each of the seven bridges exactly once. In graph terminology, Euler wanted to find a path through the given graph that included each edge exactly once.

If we generalize by asking the same question of any graph, then we obtain the following definition.

Definition 46 (Eulerian trail, tour)**.** Let G be a graph. A walk in G that traverses every edge exactly once is called an **Eulerian trail**. If, in addition, the trail begins and ends at the same vertex, we call the walk an **Eulerian tour**. Finally, if G has an Eulerian tour, we call G **Eulerian**.

Now, the introduction to these notes unfortunately spoiled the answer to the Bridges of Königsberg problem, since we already saw that Euler could not find a path that crosses every bridge exactly once. But some questions still remain: how did Euler determine that no such path existed, and how can we determine whether an Eulerian path or Eulerian circuit exists in some given graph?

In his paper on the Bridges of Königsberg problem, Euler gave the following condition for a graph to contain an Eulerian tour (also referred to as an Eulerian cycle or circuit).

Theorem 47. *A connected graph G contains an Eulerian cycle if and only if every vertex of G is of even degree.*

Proof. ($\Rightarrow$): Assume that G contains an Eulerian cycle. Every time such a cycle reaches a vertex u of G, it contributes two to $\deg(u)$ (one for entering u and one for leaving u). Since we are dealing with a cycle, the initial and final vertices are identical, so this vertex also has degree 2. Therefore, every vertex of G is of even degree.

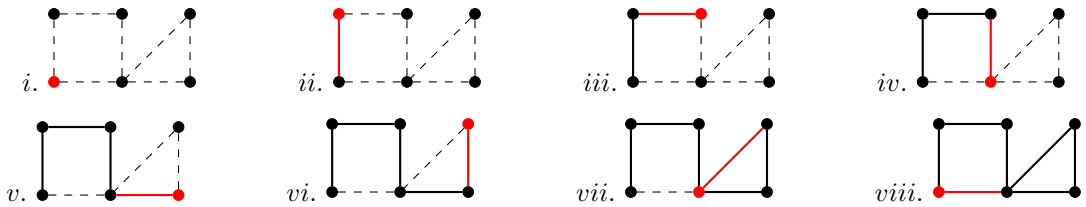
($\Leftarrow$): Assume that every vertex of G is of even degree. Choose an arbitrary vertex, say u, and take this vertex to be the initial vertex of a cycle in G. Arbitrarily choose an edge $\{u, w_1\}$ incident to u, and then continue arbitrarily choosing edges at each subsequent vertex. Once our procedure chooses an edge $\{w_i, u\}$, our cycle is complete. We may arbitrarily choose edges since the degree of each vertex is even, so every time we enter a vertex we are assured to have an edge from which we may leave that vertex. Moreover, this process is guaranteed to terminate since the edge set is finite.

In order for our cycle to be Eulerian, it must contain every edge of G. If this is the case, then we are done. Otherwise, construct the subgraph H of G consisting of all edges not included in the cycle. Every vertex of H is of even degree, so we may repeat the process of arbitrarily choosing edges from an arbitrary initial vertex v that is common between H and the first cycle we found in G, until a cycle is completed in H. Then, "patch" the cycle in H into the cycle in G at the common vertex v. See this YouTube video for a helpful illustration of how this is done.

Repeating this process until all edges in G have been included, we obtain an Eulerian cycle. □

The process used in the second part of the preceding proof is known as Hierholzer's algorithm. This algorithm is named for the German mathematician Carl Hierholzer, who was also the first person to complete the proof of Euler's claim. (Euler only managed to show that the condition given in Theorem 47 was necessary, not sufficient.)

Example 48. Given the following graph consisting of 6 vertices and 7 edges, running Hierholzer's algorithm results in finding the following Eulerian cycle. The edge selected at each step is highlighted in red. (Note that, since the algorithm makes arbitrary choices, this sequence of steps may not be the same each time the algorithm is run on the graph.)



Now that we have a condition for Eulerian cycles, let's try to modify this condition in order to say something about when a graph contains an Eulerian trail. Since the existence of an Eulerian trail is a weaker property than the existence of an Eulerian cycle, we would expect the condition on the graph to be weakened as well. Indeed, the condition is weakened in the sense that we may now have a certain number of odd-degree vertices in our graph.

Theorem 49. *A connected graph G contains an Eulerian trail with vertices u and v as endpoints of the trail if and only if G contains exactly two vertices u and v of odd degree.*

Proof. ($\Rightarrow$): Assume that G contains an Eulerian trail. Every time such a trail reaches a vertex w of G, it contributes two to $\deg(w)$ (one for entering w and one for leaving w). Since we are dealing with a walk (not a cycle), the initial and final vertices u and v are distinct, so both vertices have degree 1 and they are, in fact, the only vertices in G of odd degree.

($\Leftarrow$): Assume exactly two vertices of G are of odd degree; say, u and v. If we join u and v by an edge, then every vertex in G will be of even degree. By Theorem 47, G contains an Eulerian cycle. If we remove this added edge from G, then the Eulerian cycle becomes an Eulerian trail that starts at u and ends at v. □

With this result, we can finally conclude why no path through Königsberg exists that crosses each bridge exactly once. Note that, of the four vertices in our Königsberg graph, three of the vertices have degree 3 and one vertex has degree 5. So, since it has more than two vertices of odd degree, Theorem 49 tells us that no Eulerian trail can exist.

Example 50. Consider the two graphs below. The graph on the left has four vertices that each have an odd number of vertices incident on them. Therefore, it is impossible to draw this figure without lifting your pencil or drawing a line more than once. However, the graph on the right only has two vertices with an odd number of edges incident on them. These two vertices must be the first/last points in a drawing, to avoid lifting your pencil or drawing the same line more than once. Try it!

i. 

ii.