3 Paths and Circuits

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.

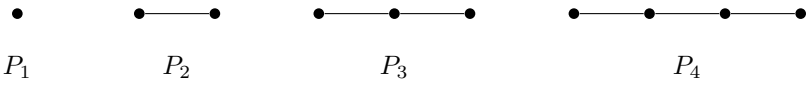
In this section, we will discuss how to model certain problems for which graphs are highly suitable: namely, problems that involve finding a way from one location to another location. In a graph, such problems reduce from finding a sequence of edges from one vertex to another vertex, and such a sequence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

We formally define the notion of a path as follows.

Definition 15 (Path)**.** Given an undirected graph $G = (V, E)$, two vertices $u, v \in V$, and a natural number n, a path of length n from vertex u to vertex v is a sequence of edges $\{e_1, e_2, \ldots, e_n\}$ where $e_1, \ldots, e_n \in E$ and where $e_1 = \{u, w_1\}$, $e_2 = \{w_1, w_2\}$, $\ldots$, $e_n = \{w_n, v\}$ where $w_1, \ldots, w_n \in V$.

Note that a path of length 0 is simply a vertex by itself. We may adapt our definition of a path to apply to directed graphs in the appropriate way by changing our notation for edges from unordered pairs to ordered pairs.

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. (The path graph on 1 vertex is just the singleton graph.)

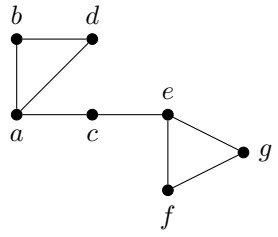


Reading Definition 15, you might have implicitly assumed that paths must be linear. However, nothing in our definition requires that to be the case. Imagine, for example, a graph representing an electrical circuit, where vertices denote components and edges denote wires. In order for the electrical circuit 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 **circuit**.

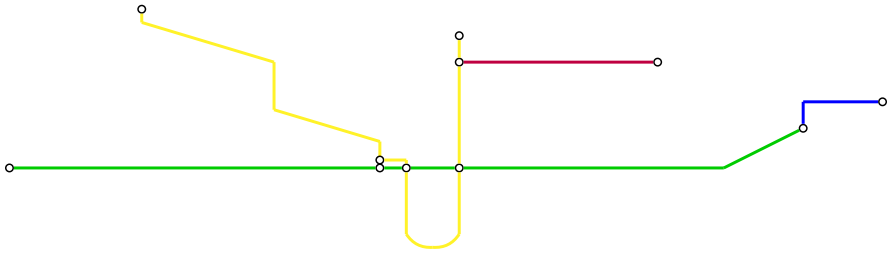
Definition 16 (Circuit)**.** A path of length n from a vertex u to a vertex v is a circuit if $u = v$ and if $n > 0$.

Note that, in Definitions 15 and 16, we do not require each of $e_1, e_2, \ldots, e_n$ to be distinct. Indeed, a path or circuit may include an edge more than once, and this will not lead to any problems. However, if no edge in the path or circuit is included more than once, then we say that the path or circuit is **simple**.

Example 17. In the following graph, we see that (among many others) there exist paths a–b, a–c–e–g, c–e–c–a, and f–g–f; simple paths b–a–c–e, d–a–b, and f–e–c; and circuits a–b–d–a and e–f–g–e.



Example 18. 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 circuit.

Using path graphs and cycle graphs, we can characterize when a graph contains either a simple path or a simple circuit of length n via the following small result.

Lemma 19. *An undirected graph G contains a simple path of length n if and only if P_{n+1} is a subgraph of G. Likewise, G contains a simple circuit of length n if and only if C_n is a subgraph of G.*

Proof. Immediate. □

3.1 Connectedness

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 some arbitrary vertex in the graph.

3.1.1 Undirected Graphs

For the case of undirected graphs, the definition of connectedness is straightforward.

Definition 20 (Connectedness)**.** An undirected graph $G = (V, E)$ is connected if, for every pair of vertices $u, v \in V$, there exists a path in G between u and v.

Thus, the graphs depicted in Examples 17 and 18 are both connected.

Although Definition 20 only tells us that there exists a path between any pair of distinct vertices of a connected graph, we can in fact strengthen this result via a small proof.

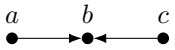
Theorem 21. *Given an undirected graph $G = (V, E)$, if G is connected, then there exists a simple path between every pair of distinct vertices $u, v \in V$.*

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

If this path is simple, then we are done. Otherwise, if this path is not simple, then 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 path, and we may repeat this process until the resultant path is simple. □

3.1.2 Directed Graphs

As with many other graph concepts we've seen thus far, when it comes to directed graphs, we must revise our definition of connectedness. It is not enough to call a directed graph "connected" if some edge exists between all vertices. Consider, for example, the following graph:



Clearly, we can reach vertex b from both vertices a and c. However, we can't reach either vertex a or c from vertex b, since the edges are directed. Therefore, we can't call this graph "connected" in the sense of Definition 20.

What we must do instead is create different definitions of connectedness for directed graphs. Our definitions will vary depending on whether we care more about the directions of each edge (leading to **strong connectedness**) or about the underlying structure of the graph (leading to **weak connectedness**).

Definition 22 (Strong connectedness)**.** A directed graph $G = (V, E)$ is strongly connected if, for every pair of vertices $u, v \in V$, there exist two directed paths in G from u to v and from v to u.

Definition 23 (Weak connectedness)**.** A directed graph $G = (V, E)$ is weakly connected if, for every pair of vertices $u, v \in V$, there exists an undirected path in G between u and v.

In other terms, a directed graph G is weakly connected if, after removing the directions from each edge, the underlying undirected graph G' is connected. Returning to our small example, we established that our graph was not strongly connected (since we cannot reach either a or c from b), but it is weakly connected.

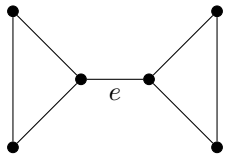
3.1.3 Connected Components

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.

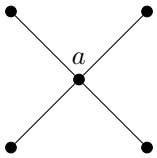
If we take a graph G with m connected components, and if removing a vertex from G produces a new graph G' with more than m connected components, then we call that vertex a **cut vertex**. If the same outcome occurs after removing an edge, then we call that edge a **cut edge**.

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 24. 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 25. 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.



3.1.4 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.

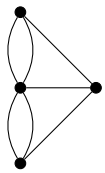
- The **graph reachability** problem asks whether a path exists from one given vertex to another given vertex within a graph. In an undirected graph, we can solve this problem by simply finding the connected components of the graph. In a directed graph, a number of approaches exist to solve the problem; it is possible to find the solution in time proportional to $|V|^3$ by using the Floyd-Warshall algorithm. (Other algorithms are more efficient, but require some preprocessing work.)
- 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.
- 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.

3.2 Special Paths and Circuits

Having established the definitions of paths and circuits, we can move on to discussing a couple of specific graph properties that tell us about the kinds of paths or circuits a given graph contains. These properties are named after the two famous mathematicians who studied them: Leonhard Euler and William Rowan Hamilton.

3.2.1 Eulerian Paths and Circuits

Let's begin by returning 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 (that is, a simple path).

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

Definition 26 (Eulerian path/circuit)**.** A graph $G = (V, E)$ contains an Eulerian path if there exists a simple path in G that contains every edge in E. Similarly, G contains an Eulerian circuit if there exists a simple circuit in G that contains every edge in E.

If a graph contains an Eulerian path or an Eulerian circuit, then we call the graph itself **Eulerian**. Some authors refer to graphs that contain an Eulerian path but not an Eulerian circuit as **semi-Eulerian**.

We may easily adapt Definition 26 to work for directed graphs by considering directed paths or directed circuits.

Remark. It's easy to confuse the notion of Eulerian paths and Eulerian circuits with similar notions we will see in the next section. To remember Definition 26, observe that **E**ulerian paths and circuits traverse the **e**dges of a graph.

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 circuit.

Theorem 27. *A connected graph G contains an Eulerian circuit if and only if every vertex of G is of even degree.*

Proof. ($\Rightarrow$): Assume that G contains an Eulerian circuit. Every time such a circuit 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 circuit, the initial and final vertices are identical, so this vertex also has degree 2. Therefore, every vertex of G is of even degree.

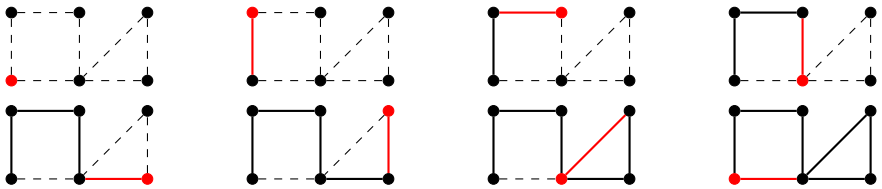
($\Leftarrow$): Assume that every vertex of G is of even degree. Choose an arbitrary vertex, say u, and take this vertex to be the initial vertex of a circuit in G. Arbitrarily choose an edge $\{u, w_1\}$ incident to u, and then continue arbitrarily choosing edges at each subsequent vertex. Once our procedure chooses an edge $\{w_i, u\}$, our circuit 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 circuit 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 circuit. Every vertex of H is of even degree, so we may repeat the process of arbitrarily choosing edges from an arbitrary initial vertex v until a circuit is completed in H. Then, attach the circuit in H to the circuit in G at the vertex v.

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

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 27 was necessary, not sufficient.)

Example 28. Given the following graph consisting of 6 vertices and 7 edges, running Hierholzer's algorithm results in finding the following Eulerian circuit. (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 circuits, let's try to modify this condition in order to say something about when a graph contains an Eulerian path. Since the existence of an Eulerian path is a weaker property than the existence of an Eulerian circuit, 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 29. *A connected graph G contains an Eulerian path if and only if G contains exactly two vertices of odd degree.*

Proof. ($\Rightarrow$): Assume that G contains an Eulerian path. Every time such a path 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 path, the initial and final vertices are distinct, so both vertices have degree 1 and they are, in fact, the only vertices in G of odd degree.

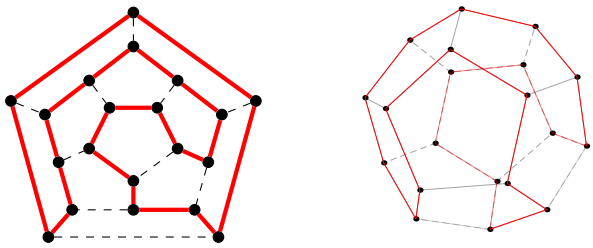
($\Leftarrow$): Assume exactly two vertices of G are of odd degree; say, u and v. If we join u and v by an edge, then every vertex in G will be of even degree. By Theorem 27, G contains an Eulerian circuit. If we remove this added edge from G, then the Eulerian circuit becomes an Eulerian path. □

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. Since none of the vertices are of even degree, Theorem 29 tells us that no Eulerian path can exist.

3.2.2 Hamiltonian Paths and Circuits

In 1856, William Rowan Hamilton invented a mathematical puzzle called the "icosian game". Given an dodecahedron (a three-dimensional solid with twelve faces), the objective of the icosian game is to find a path along the edges of the solid that simultaneously touches each of the 20 corners exactly once and finishes at the starting corner.

The commercially-sold version of the puzzle consisted of a wooden base, upon which was inscribed 20 holes and 30 lines, as well as 20 numbered pegs. The following figures show a solution to the icosian game in both two dimensions (which most closely resembles the commercial puzzle) and three dimensions.



Hamilton found a solution to his puzzle by developing an entirely-new mathematical structure known as the icosian calculus. In turn, his work spurred interest in finding paths in other graphs that visit every vertex, and this interest led to the following definition.

Definition 30 (Hamiltonian path/circuit)**.** A graph $G = (V, E)$ contains a Hamiltonian path if there exists a path in G that visits every vertex exactly once. Similarly, G contains a Hamiltonian circuit if there exists a circuit in G that visits every vertex aside from the initial/final vertex exactly once and visits the initial/final vertex exactly twice.

If a graph contains a Hamiltonian circuit, then we call the graph itself **Hamiltonian**. If a graph contains a Hamiltonian path but not a Hamiltonian circuit, then we call the graph **traceable**.

We may easily adapt Definition 30 to work for directed graphs by considering directed paths or directed circuits.

Since we have previously seen conditions that tell us when a graph contains an Eulerian path or an Eulerian circuit, we might assume that there exists another set of conditions that allow us to determine if a graph contains a Hamiltonian path or a Hamiltonian circuit. After all, the only difference between the concepts is that we're considering vertices instead of edges. As it turns out, however, we aren't so lucky; no efficient

method is known that solves the problem of determining whether a given graph contains a Hamiltonian path or a Hamiltonian circuit.

There are some basic observations that allow us to conclude immediately that a given graph has a Hamiltonian circuit; for instance,

- every complete graph K_n, $n \geq 3$, contains a Hamiltonian circuit;
- every cycle graph C_n contains a Hamiltonian circuit; and
- every graph G that contains an Eulerian path contains a Hamiltonian circuit in the corresponding **line graph** $L(G)$, which is the graph representing adjacencies between *edges* of G.

Don't worry if the definition of a line graph isn't immediately clear to you; we only mention the last observation to illustrate that a connection exists between Eulerian paths and Hamiltonian circuits.

Moreover, a pair of theorems exist that relate the existence of Hamiltonian circuits to the degrees of vertices in a graph. These theorems, which we will not prove here, were originally proved by the mathematicians Gabriel Dirac and Øystein Ore. Both theorems essentially state that a graph is Hamiltonian if it contains enough edges.

Theorem 31 (Dirac's theorem)**.** *If a simple graph $G = (V, E)$ with $n \geq 3$ vertices is such that, for every vertex $u \in V$, $\deg(u) \geq n/2$, then G contains a Hamiltonian circuit.*

Proof. Omitted. □

The result by Ore is a strengthening of the result by Dirac.

Theorem 32 (Ore's theorem)**.** *If a simple graph $G = (V, E)$ with $n \geq 3$ vertices is such that, for every pair of nonadjacent vertices $u, v \in V$, $\deg(u) + \deg(v) \geq n$, then G contains a Hamiltonian circuit.*

Proof. Omitted. □

Keep in mind that all of the conditions we have seen in this section are only sufficient for proving the existence of a Hamiltonian circuit. If some graph does not meet these conditions, then that does not necessarily imply that the graph does not contain a Hamiltonian circuit. Currently, no conditions are known that are both necessary and sufficient for proving the existence of a Hamiltonian path or Hamiltonian circuit.

3.2.3 Applications of Special Paths and Circuits

Recall from previous notes that we discussed the **traveling salesperson** problem in the context of permutations. The traveling salesperson problem asks, given a list of n cities and a list of distances between each city, what is the shortest route that both visits all cities and ends in the origin city?

If we model an instance of the traveling salesperson problem as a graph, where vertices are cities and (weighted) edges are distances between cities, then it becomes evident that this problem is a variant of the Hamiltonian circuit problem. Unfortunately, we already know that no efficient method is known to solve the Hamiltonian circuit problem, so consequently we have no efficient method of solving the traveling salesperson problem either.

So, if we have no efficient method of finding Hamiltonian paths or Hamiltonian circuits, could we instead try to augment a given graph to make it Hamiltonian? This is known as the **Hamiltonian completion** problem, which asks for the minimum number of edges we must add to a graph to ensure it contains a Hamiltonian circuit. Both Dirac's and Ore's theorems give conditions on the number of edges required for a simple graph to contain a Hamiltonian circuit, and the Hamiltonian completion problem has been solved for particular classes of graphs. However, in the general case, we still have no efficient method for determining this minimum number.