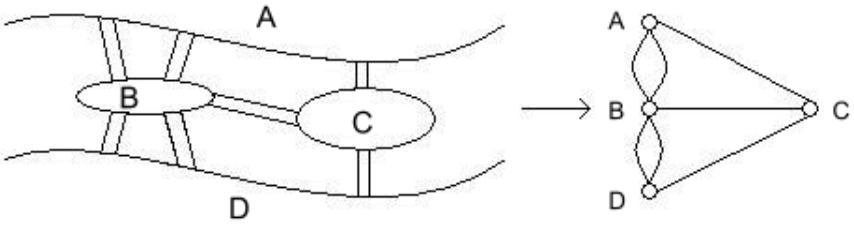
# An Introduction to Graphs

A few centuries ago, residents of the city of Königsberg, Prussia were interested in a certain problem. They wanted to know whether it was possible to walk through their city, represented in the image below, by crossing each bridge only once.




This problem is called The Seven Bridges of Königsberg, and in 1736 the brilliant mathematician Leonhard Euler proved that such a task was impossible. Euler did this by modeling the arrangement of bridges as a graph (or, more accurately, a *multigraph* since there can be multiple edges between the same pair of vertices). For this reason, Euler is generally hailed as the inventor of graph theory.

The relevant features of the seven bridge problem are the two islands ($B$ and $C$), the two banks ($A$ and $D$), and the bridges between these areas. All of these features can be represented by a *graph*, as shown on the right in the image above. The two islands and each bank are represented by the 4 little circles, which are called *vertices*. The bridges are represented by the lines between the vertices, called the *edges* of the graph. (Include labels for vertices/edges?) This graph happens to be *undirected*, since the bridges can be crossed in either direction. If we instead had one-way bridges, the graph would be *directed*.

More formally, a directed graph $G(V,E)$ consists of a finite set of vertices $V$ and a set of edges $E$. $E$ is a subset of $V \times V$, where $V \times V$ is the *Cartesian product* of $V$ with itself.[1] An edge $(v,w)$in a directed graph is usually indicated by drawing a line between $v$ and $w$, with an arrow pointing towards $w$. *Undirected graphs* may be regarded as special kinds of directed graphs, in which $(u,v) \in E$ if and only if $(v,u) \in E$. Thus in an undirected graph the directions of the edges are unimportant, so an edge of an undirected graph is an *unordered* pair of vertices $\{u,v\}$ and is indicated by a line between $u$ and $v$ with no arrow.

As we have defined them, graphs are allowed to have *self-loops*; i.e., edges of the form $(u,u)$ that go from a vertex $u$ to itself. Usually, however, graphs are assumed to have no self-loops unless otherwise stated, and we will assume this from now on. We will also be working only with graphs, and not multigraphs such as the one representing the bridges of Königsberg.

A *path* in a directed graph $G = (V,E)$ is a sequence of edges $(v_1,v_2),(v_2,v_3),...,(v_{n-2},v_{n-1}),(v_{n-1},v_n)$. In this case we say that there is a path *between* $v_1$ *and* $v_n$. A path in an undirected graph is defined similarly. Usually a path is assumed to be *simple*, meaning $v_1,\ldots,v_n$ are distinct. A path with repeated vertices will be called a *walk*. A *cycle* (or *circuit*) is a sequence of edges $(v_1,v_2),(v_2,v_3),,(v_{n-2},v_{n-1}),(v_{n-1},v_n),(v_n,v_1)$, where $v_1,\ldots,v_n$ are distinct. A *tour* is a walk which starts and ends at the same vertex. A graph is said to be

[1] Recall that the Cartesian produce of two sets $U$ and $V$ is defined as $U \times V = \{(u,v) : u \in U \text{ and } v \in V\}$.

*connected* if there is a path between any two distinct vertices.

We say that an edge $(u,v)$ is *incident* to vertices $u$ and $v$, and vertices $u$ and $v$ are *neighbors* or *adjacent*. If $G = (V,E)$ is an undirected graph then the *degree* of vertex $u \in V$ is the number of edges incident to $u$, i.e., $degree(u) = |\{v \in V : \{u,v\} \in E\}|$. A vertex $u$ whose degree is 0 is called an *isolated* vertex, since there is no edge which connects $u$ to the rest of the graph. In a directed graph, the *in-degree* of a vertex $u$ is the number of edges from other vertices to $u$, and the *out-degree* of $u$ is the number of edges from $u$ to other vertices.

# Eulerian Walks and Tours

We can now restate the problem above in the language of graphs: is there a walk in the graph that uses each edge exactly once? This type of walk is called an *Eulerian walk*. An *Eulerian tour* is a tour that uses each edge exactly once.

Let an even degree graph be a graph in which all vertices have even degree. The next theorem gives necessary and sufficient conditions for a graph to have an Eulerian tour.

**Euler's Theorem:** An undirected graph $G = (V,E)$ has an Eulerian tour if and only if the graph is connected (except possibly for isolated vertices) and even degree.

**Proof** ($\Longrightarrow$): Assume that the graph has an Eulerian tour. This means every vertex that has an edge adjacent to it (i.e., every non-isolated vertex) must lie on the tour, and is therefore connected with all other vertices on the tour. This proves that the graph is connected (except for isolated vertices).

Next note that, for each vertex in the tour except the first (and last), the tour leaves it in the next step after entering it. Thus, every time the tour visits a vertex, it traverses exactly two edges incident to it. Since the Eulerian tour uses each edge exactly once, this implies that every vertex except the first has even degree. And the same is true of the first vertex $v$ as well because the first edge leaving $v$ can be paired up with the last edge returning to $v$. This completes the proof of this direction.

**Proof** ($\Longleftarrow$): Assume that $G = (V,E)$ is connected and even degree. We will show how to construct an Eulerian tour in $G$. We will begin walking from a vertex $u$, never repeating edges. Since the graph is even degree, each time the walk enters a vertex, it must be able to exit as well. Our walk can therefore only end on $u$; it will be a tour. The claim below formalizes this intuition:

> **Claim 1** In an even degree graph, a walk starting from a vertex $u$ can only get stuck at $u$.

Let's call the resulting tour $A$. Is it Eulerian? Not necessarily; our tour may not encounter all edges in the graph. Consider the image below- our algorithm could have resulted in tour $A$, which does not traverse edges along $B$ and is therefore not Eulerian.


To continue, we will remove $A$ from $G$ and create a new tour $B$ on the remaining edges. We can do this because the remaining graph (consisting of the remaining edges) still has even degree:

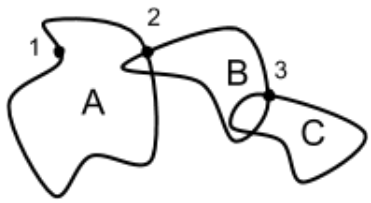
**Claim 2** Removing a tour from an even degree graph will result in an even degree graph.

However, since our eventual goal is to create a single Eulerian tour. So we must splice tours $A$ and $B$ together into a single tour that traverses each of them. We can only do this if they intersect — there must be some edge of $B$ such that one of its endpoints lies on $A$. The following claim says there is always an untraversed edge "hanging" from $A$ (we will use this edge as our starting point to discover tour $B$):

**Claim 3** Let $A$ be a tour in a connected (except for isolated vertices) graph $G$. If $A$ does not contain all edges in $G$, there exists an edge $\{u, v\}$ such that $A$ passes through $u$ but does not contain $\{u, v\}$.

We can now begin walking from $u$, using only edges that did not occur in $A$. By Claim 2, the removal of edges of $A$ results in an even degree graph and therefore by Claim 1, this new walk will get stuck at $u$, creating tour $B$. We can combine $A$ with $B$ by splicing the two together: we start walking along $A$ until we reach $u$, and then walk along $B$ until we return to $u$, and then return to walking along $A$ until we finish. If this new tour does not include all edges in $G$, Claim 3 implies that there exists an untraversed edge which is connected to the tour. We can then repeat the process.

Here is a possible scenario:




If the above image is the graph in question, our first tour could be tour $A$. Our second tour could be tour $B$. We splice the first two tours together by starting at point 1, walking along $A$ until we reach point 2, walking along $B$ until we return to point 2, and finishing our walk along $A$, ending back at point 1. Call this tour $T$. We then create another tour $C$. To splice $T$ and $C$ together, we walk along $T$ until reaching point 3, then walk along $C$ until returning to point 3, and then finish our walk along $T$. The final walk would be an Eulerian tour.

We can now prove all the claims above.

# Proof of Claim 1

First consider a walk from $u$ to $v$. For a vertex $w$, let $n(w)$ be the number of edges on the walk incident to $w$. Let's say we get stuck at $v$. Then $n(v)$ is odd; we have entered $v$ but not exited. However, since $v$ is even degree, there must be at least one unused edge incident to $v$, which we can use to exit $v$. Therefore, we cannot get stuck at $v$.

# Proof of Claim 2

If we removed the edges traversed by a tour, we are decreasing the degree of each vertex $w$ by $n(w)$. Since a tour exits each vertex as often as it enters and our tour does not have any repeated edges, $n(w)$ must be even for all $w$. Therefore, the degree of each vertex remains even.

# Proof of Claim 3

Suppose $A$ does not contain all edges in the graph $G$. Let $\{x,y\}$ be an edge not in the tour. If $x$ is on $A$ then we are done. Otherwise, since the graph is connected, there is a path from vertex $a$ on tour $A$ to vertex $x$. This path starts with a vertex on $A$ and ends with a vertex not on $A$. As we traverse the path from $a$ to $x$, there must be a first time it touches a vertex $v$ not on the tour $A$ (you can formally prove this by induction!). The previous vertex $u$ must be on the tour, and so $\{u,v\}$ is the desired edge.

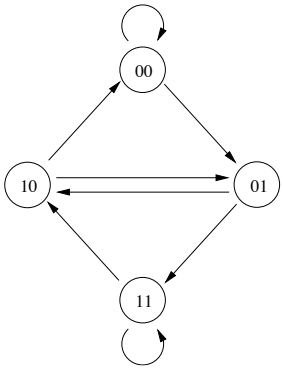
# de Bruijn Graphs

A *de Bruijn sequence* is a $2^n$-bit circular sequence such that every string of length $n$ occurs as a contiguous substring of the sequence exactly once. For example, the following is a de Bruijn sequence for the case $n = 3$:


Notice that there are eight substrings of length three, each of which corresponds to a binary number from 0 to 7. It turns out that such sequences can be generated from a *de Bruijn graph* $G = (V,E)$, where $V$ is the set of all $n-1$ bit strings (i.e., $V = \{0,1\}^{n-1}$). Each vertex $a_1a_2...a_{n-1}$ has two outgoing edges: $(a_1a_2...a_{n-1}, a_2a_3...a_{n-1}0)$ and $(a_1a_2...a_{n-1}, a_2a_3...a_{n-1}1)$. Therefore, each vertex also has two incoming edges: $(0a_1a_2...a_{n-2}, a_1a_2...a_{n-1})$, $(1a_1a_2...a_{n-2}, a_1a_2...a_{n-1})$. For example, if we have the vertex 110, then the two outgoing edges would be directed toward the vertices 100, and 101. Note that these are directed edges, and self-loops are permitted.

The de Bruijn sequence is generated by an Eulerian tour in this graph. Euler's theorem above can be modified to work for directed graphs: all we need to modify is the second condition, which should now say: "for every vertex $v$ in $V$, the in-degree of $v$ equals the out-degree of $v$". Clearly, the de Bruijn graph satisfies this condition, and therefore it has an Eulerian tour.

To actually generate the sequence, starting from any vertex, we walk along the tour and add the corresponding bit which was shifted in from the right as we traverse each edge. Here is the de Bruijn graph for $n = 3$. Exercise: Find the Eulerian tour of this graph that generates the de Bruijn sequence given above.



# Hypercubes

Imagine there is only one road connecting cities in southern California with cities in northern California. If that road is blocked, drivers will have no way of traveling through California. Is there a better network design which would avoid this issue?

We can model this problem using a graph; the vertices will represent destinations and the edges will represent roads. The property we want is that no two large sets of cities can be disconnected by failure of a few roads. Of course, you could achieve this by a complete graph, where there is a road between every possible pair of cities, but this is hardly practical. Can we achieve something similar while bounding the total number of edges (or equivalently the degree of each vertex)? The hypercube is a canonical example of a low degree, well connected graph.

Recall that the set of all $n$-bit strings is denoted by $\{0,1\}^n$. The vertex set of an $n$-dimensional hypercube is $\{0,1\}^n$ (i.e., there are exactly $2^n$ vertices, each labeled with a distinct $n$-bit string), and with an edge between vertices $x$ and $y$ iff $x$ and $y$ differ in exactly one bit position. I.e., if $x = x_1x_2\ldots x_n$ and $y = y_1y_2\ldots y_n$, then there is an edge between $x$ and $y$ iff there is an $i$ such that $\forall j \neq i, x_j = y_j$ and $x_i \neq y_i$.

There is another equivalent recursive definition of the hypercube:

The $n$-dimensional hypercube consists of two copies of the $n-1$-dimensional hypercube (the 0-subcube and the 1-subcube), and with edges between corresponding vertices in the two subcubes. i.e., there is an edge between vertex $x$ in the 0-subcube (also denoted as vertex $0x$) and vertex $x$ in the 1-subcube (denoted $1x$).

**Claim:** The total number of edges in an $n$-dimensional hypercube is $n2^{n-1}$.

**Proof:** Each vertex has $n$ edges incident to it, since there are exactly $n$ bit positions that can be toggled to get an edge. Since each edge is counted twice, once from each endpoint, this yields a grand total of $n2^n/2$.

**Alternative Proof:** By the second definition, it follows that $E(n) = 2E(n-1) + 2^{n-1}$, and $E(1) = 1$. A straightforward induction shows that $E(n) = n2^{n-1}$.

It is impossible to break the $n$-dimensional hypercube into two equal parts, each consisting of $2^{n-1}$ vertices, without cutting at least $2^{n-1}$ edges. We will prove something stronger: consider how many edges must be cut to separate a subset $S$ of vertices from the remaining vertices $V - S$. Assume that $S$ is the smaller piece; i.e. $|S| \leq |V - S|$. Then $|S| \leq 2^{n-1}$. Let $E_S$ denote the set of edges connecting vertices in $S$ to vertices in $V - S$. Note that $E_S = E_{V-S}$.

**Theorem:** $|E_S| \geq |S|$.

**Proof:** By induction on $n$. The base case $n = 1$ is trivial.

For the induction step, let $S_0$ be the vertices from the 0-subcube in $S$, and $S_1$ be the vertices in $S$ from the 1-subcube.

**Case 1:** $|S_0| \leq 2^{n-1}/2$ **and** $|S_1| \leq 2^{n-1}/2$

If $|S_0| \leq 2^{n-1}/2$ and $|S_1| \leq 2^{n-1}/2$ then applying the induction hypothesis to each of the subcubes shows that the number of edges between $S$ and $V - S$ even without taking into consideration edges that cross between the 0-subcube and the 1-subcube, already exceeds $|S_0| + |S_1| = |S|$.

**Case 2:** $|S_0| > 2^{n-1}/2$

Then $|S_1| \leq 2^{n-1}/2$ since $|S| \leq 2^{n-1}$.

By the inductive hypothesis applied to $|S_1|$, there must also be at least $|S_1|$ edges in the 1-subcube crossing between $S$ and $V - S$. What about the 0-subcube? There we have $|S_0| > 2^{n-1}/2$, so we cannot apply the

induction hypothesis. Instead, if we let $V_0$ be all the vertices in the 0-subcube, we can apply the inductive hypothesis to $\bar{S}_0 = V_0 - S_0$, since this set has $2^{n-1} - |S_0| < 2^{n-1}/2$ elements. Applying the induction hypothesis, we note that the number of edges between $S_0$ and $V_0 - S_0$ is at least $2^{n-1} - |S_0|$. So between the 0-subcube and the 1-subcube we have a grand total of $2^{n-1} - |S_0| + |S_1|$ crossing edges between $S$ and $V - S$. We are still short of our goal.

We can add to our total by noting that there must be at least $|S_0| - |S_1|$ edges in $E_S$ that cross between the two subcubes. This is because there is an edge between every vertex of the form $0x$ and the corresponding vertex $1x$. This gives us a grand total of at least $2^{n-1} - |S_0| + |S_1| + |S_0| - |S_1| = 2^{n-1} \geq |S|$ So $|E_S| \geq |S|$, as required. This completes the proof of the inductive step in Case 2.