EECS 70 Discrete Mathematics and Probability Theory
Summer 2020

Note 6

1 Graph Theory: An Introduction

One of the fundamental ideas is the notion of *abstraction*: capturing the essence or the core of some complex situation by a simple model. This sense of abstraction as the heart of approximate modeling is something that we use all the time[1] in EECS-related fields, but it is also something that we see in Physics, etc. Some of the largest and most complex entities we might deal with include the internet, circuits, the brain, maps, and social networks. In each case, there is an underlying "network" or *graph* that captures the important features that help us understand these entities more deeply. In the case of the internet, this network or graph specifies how web pages link to one another. For circuits, the graph specifies the connection between nodes by circuit elements. In the case of the brain, it is the interconnection network between neurons. We can describe the connection structure in the beautiful framework of *graph theory*, which is the subject of this lecture note.

Remarkably, the modern approach to graph theory has its origins in a simple evening pastime of the residents of Königsberg, Prussia (nowadays Kaliningrad, Russia) a few centuries ago. Through their city ran the Pregel river, depicted in Figure 1(a) below, separating Königsberg into two banks A and D and islands B and C. Connecting the islands to the mainland were seven bridges. As the residents of the city took their evening walks, many would try to solve the challenge of picking a route that would cross each of the seven bridges precisely once and return to the starting point.

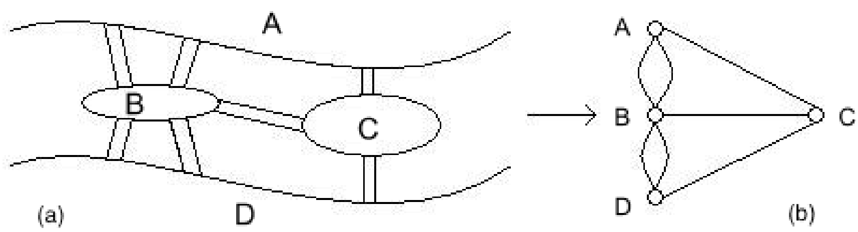


Figure 1: (a) The city of Königsberg. (b) The (multi-)graph modeling the bridge connections in Königsberg.

In 1736, the brilliant mathematician Leonhard Euler proved this task to be impossible. How did he do it? The key is to realize that, for the purpose of choosing such a route, Figure 1(a) can be replaced by Figure 1(b), where each land mass A, B, C, and D is replaced by a small circle, and each bridge by a line segment. With this abstraction in place, the task of choosing a route can be restated as follows: trace through all the line segments and return to the starting point without lifting the pen, and without traversing any line segment more than once. The proof of impossibility is simple. Under these tracing rules, the pen must enter each small circle as many times as it exits it, and therefore the number of line segments incident to that circle must be even. But in Figure 1(b), each circle has an odd number of line segments incident to it, so it is

[1]In fact, you've seen this already in the last lecture note where we introduced the idea of matchings. They're not just useful for modeling getting people jobs! In earlier courses, you have seen how linear algebra provides a very general language for many such abstractions — one of our core goals in 70 is to augment your conceptual vocabulary with more discrete abstractions that complement the fundamentally continuous modeling perspectives that linear algebra and calculus have given you access to already. It is a very beautiful aspect of our field that in later courses, you will see how these seemingly different perspectives actually come together in powerful ways as linear-algebraic thinking illuminates aspects of graphs and vice-versa!

impossible to carry out such a tracing. Actually Euler did more. He gave a precise condition under which the tracing can be carried out. For this reason, Euler is generally hailed as the inventor of graph theory.

1.1 Formal definitions

Formally, an (undirected) graph $G = (V,E)$ is defined by a set of *vertices* V and a set of *edges* E. Each edge is a pair$\{u,v\}$ of vertices. We represent the vertices of the graph as points, and the edges as line segments (or curves) connecting them. For example, in the graph G_2 in Figure 2 below, the vertex set is $V = \{1,2,3,4,5\}$ and the edges are $E = \{\{1,2\},\{2,3\},\{3,4\},\{4,5\},\{5,1\}\}$. Note that Figure 1(b) is a slightly more general graph, in which we allow multiple edges between the same pair of vertices: technically, E is then a *multiset* rather than a set. With the exception of this one classical example, in our graphs we will not allow multiple edges between the same two vertices.

More generally, we can also define a *directed* graph. A directed graph $G = (V,E)$ again consists of a set of vertices V and a set of edges E, with the difference that now an edge is an *ordered* pair of vertices (u,v). In other words, E is a subset of the Cartesian product $V \times V$. (Recall that the Cartesian product $U \times V$ of sets U and V is defined as $U \times V = \{(u,v) : u \in U \text{ and } v \in V\}$.) In contrast to the undirected case, the edge (u,v) is *not* the same as the edge (v,u). A directed graph is also represented as a set of points V, and the edge (u,v) is now shown as a *directed* line segment, or *arrow*, from u to v. The graph G_1 in Figure 2 below has vertex set $V = \{1,2,3,4\}$ and edge set $E = \{(1,2),(1,3),(1,4)\}$. Note that (e.g.) $(1,2)$ is an edge of G_1, but $(2,1)$ is not.

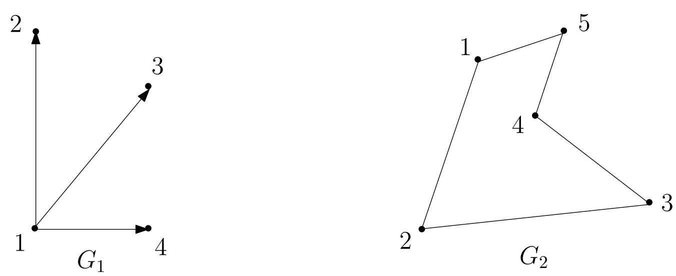


Figure 2: Examples of directed and undirected graphs, respectively.

Directed graphs are more general than undirected ones, in the following sense: we can think of an undirected graph as a directed graph in which every edge (u,v) is always accompanied by its reversal (v,u): in other words, every arrow in the graph always "goes in both directions." A more folksy interpretation is to view an undirected graph as a system of two-way streets, and a directed graph as a system of one-way streets; a two-way street can then be seen as two one-way streets going in opposite directions.

What kind of relationships do directed and undirected graphs model? One example might be different kinds of transportation networks, where the vertices are cities on a map. An undirected graph may be used to model freeway connections between the cities (which are of course two-way), whereas a directed graph may model direct airline flights (which are not necessarily two-way). Our second example is drawn from *social networks*, an area in which graph theory plays a fundamental role. Suppose you wish to model a social network in which vertices correspond to people, and an edge between (say) Alex and Bridget indicates that the two people know each other: this would be an undirected graph. On the other hand, if you wish to model the fact that Alex recognizes Bridget, you would use a directed graph (since this does not necessarily imply that Bridget recognizes Alex).

Moving on with our definitions, we say that edge $e = \{u,v\}$ (or $e = (u,v)$) is *incident* on vertices u and v, and that u and v are *neighbors* or *adjacent*. If G is undirected, 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.

Sanity check! What does the degree of a vertex represent in an *undirected* social network in which an edge $\{u,v\}$ means u and v know each other? How should we interpret an isolated vertex in such a graph?

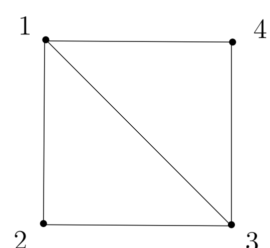
A directed graph, on the other hand, has two different notions of degree due to the directions on the edges. Specifically, the *in-degree* of a vertex u is the number of edges from other vertices to u, and the *out-degree* of u is the number of edges from u to other vertices.

Sanity check! What do the in-degree and out-degree of a vertex represent in a *directed* social network in which an edge (u,v) means that u recognizes v?

Finally, our definition of a graph thus far allows edges of the form $\{u,u\}$ (or (u,u)), i.e., *self-loops*. In most situations, however, this gives us no interesting information (e.g., in the social network setting it means that person A knows him/herself!). Thus, in general in these notes, we shall assume that our graphs have no self-loops, unless stated otherwise. As mentioned earlier, we shall also not in general allow multiple edges between a pair of vertices (except in the case of the Seven Bridges of Königsberg). (Graphs that allow self-loops and multiple edges are often referred to as *multigraphs*.)

Walks, tours, paths and cycles. Let $G=(V,E)$ be an undirected graph. A *walk* in G is a sequence of edges $(e_1,\ldots,e_{n-1})$ for which there is a sequence of vertices $(v_1,\ldots,v_n)$ such that $e_i=\{v_i,v_{i+1}\}$ for $i=1,2,\ldots,n-1$. A walk with no repeated edges that begins and ends with the same vertex is called a *tour*. For example, suppose the graph G_3 below models a small communication network in which each vertex corresponds to a node, and two nodes u and v are neighbors if there exists a direct communication link between u and v. Then $\{3,2\},\{2,1\},\{1,4\},\{4,3\}$ is a tour in G_3.

A walk with vertice sequence $v_1,\ldots,v_n$ is a *path* if all vertices (and hence all edges) are distinct; in this case we say that there is a path *between* v_1 *and* v_n.



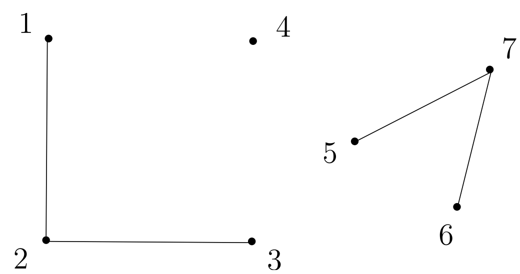
G_3

Sanity check! What is the shortest path from node 1 to node 3 in G_3? How about the longest path?

A *cycle* is a tour where the only repeated vertex is the start and end vertex.

Sanity check! Give a cycle starting at node 1 in G_3.

Connectivity. Much of what we discuss in this note revolves around the notion of connectivity. A graph is said to be *connected* if there is a path between any two distinct vertices. For example, the network G_3 above is connected, since one can send a message from any node to any other node via *some* sequence of direct links. On the other hand, the network below is *disconnected*.



Sanity check! Why would you not want a communication network to be disconnected?

Note that *any* graph (even a disconnected one) always consists of a collection of *connected components*, i.e., sets $V_1,\ldots,V_k$ of vertices, such that all vertices in a set V_i are connected. For example, the graph above is not connected, but nevertheless consists of three connected components: $V_1 = \{1,2,3\}$, $V_2 = \{4\}$, and $V_3 = \{5,6,7\}$.

2 Complete Graphs, Trees and Hypercubes

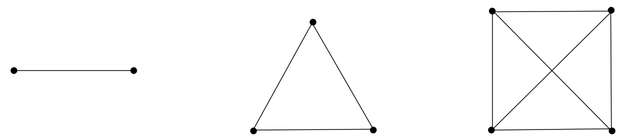
As we have seen, graphs are an abstract and general construct allowing us to represent relationships between objects, such as people in social networks or places in transportation networks. In practice, certain classes of graphs prove especially useful. For example, imagine that our graph represents the interconnections between routers on the internet. To send a packet from one node to another, we need to find a path in this graph from our source to our destination. Therefore, to be able to send a packet from any node to any other node, we need the graph to be connected. A minimally connected graph is called a *tree*, and it is the most efficient graph (i.e., with the minimum number of edges) that we can use to connect any set of vertices.

But now suppose the connections between some routers are not very reliable, so sometimes we can lose an edge between two vertices in the graph. If our graph is a tree, then removing an edge from it immediately results in a disconnected graph, which is a disaster for the network. To avoid this bad case, we want some sort of redundancy or robustness in the graph connectivity. Clearly the most connected graph is the *complete* graph, in which all nodes are connected to all other nodes. However, as we shall see below, the complete graph uses a huge number of edges, which makes it impractical for large-scale problems.

There is also a nice family of graphs called *hypercubes*, which in some sense combine the best of both worlds: they are robustly connected, but do not use too many edges. In this section, we study these three fundamental classes of graphs in more detail.

2.1 Complete graphs

We start with the simplest class of graphs, the *complete* graphs. Why are such graphs called "complete"? Because they contain the *maximum* possible number of edges: i.e., every pair of (distinct) vertices u and v is connected by an edge $\{u,v\}$. For example, here are the complete graphs on $n = 2,3,4$ vertices, respectively.



Here, the notation K_n denotes the *unique* complete graph on n vertices. Formally, we can write $K_n = (V, E)$ for $|V| = n$ and $E = \{\{v_i, v_j\} \mid v_i \neq v_j \text{ and } v_i, v_j \in V\}$.

Sanity check!

1. Can you draw K_6, the complete graph on $n = 6$ vertices?
2. What is the degree of every vertex in K_n?
3. How many edges are there in K_n? (Answer: $n(n-1)/2$.) Verify that the K_6 you drew above has this many edges.

Next, let us return to the theme of connectivity. A complete graph is special in that each vertex is connected by an edge to every other vertex. Thus, such a graph is very "strongly connected" in that a large number of edges must be removed before we disconnect the graph into two components. Obviously this is a desirable property to have in (say) a communication network.

Sanity check! What is the minimum number of edges which must be removed from K_n to disconnect it? (Hint: how many edges do you need to remove to obtain an isolated vertex?)

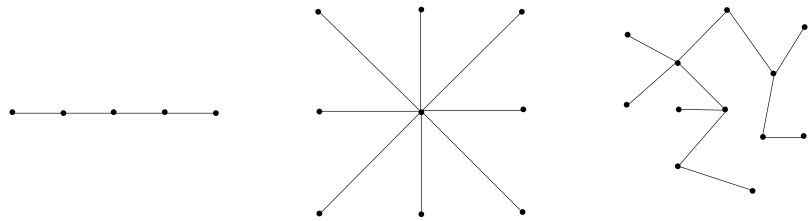
Finally, though it is much less common, we can naturally define a complete *directed* graph as one in which, for any pair of distinct vertices u, v, both the edges (u, v) and (v, u) are present.

2.2 Trees

In this section we discuss trees, which we briefly mentioned in our proof of Euler's formula in the previous section. If complete graphs are "maximally connected," then trees are the opposite: Removing just a single edge disconnects the graph! Formally, there are a number of equivalent definitions for a graph $G = (V, E)$ to be a tree, including:

1. G is connected and contains no cycles.
2. G is connected and has $n-1$ edges (where $n = |V|$ is the number of vertices).
3. G is connected, and the removal of any single edge disconnects G.
4. G has no cycles, and the addition of any single edge creates a cycle.

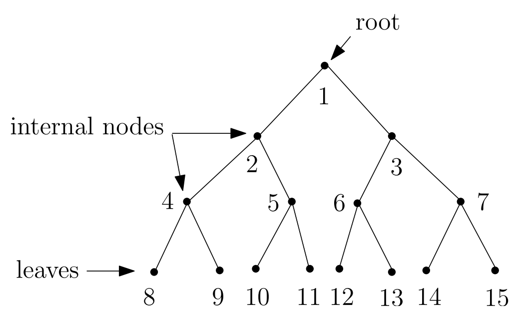
Here are three examples of trees:



Sanity check!

1. Convince yourself that the three graphs above satisfy all four equivalent definitions of a tree.
2. Give an example of a graph which is *not* a tree.

Why would we want to study such funny-looking graphs? One reason is that many graph-theoretical problems which are computationally intractable on arbitrary graphs, such as the Maximum Cut problem, are easy to solve on trees. Another reason is that they model many types of natural relationships between objects. To demonstrate, we now introduce the concept of a *rooted* tree, an example of which is given below.



In a rooted tree, there is a designated node called the *root*, which we think of as sitting at the top of the tree. The bottom-most nodes are called *leaves*, and the intermediate nodes are called *internal nodes*. The *depth* d of the tree is the length of a longest path from the root to a leaf. Moreover, the tree can be thought of as grouped into layers, or *levels*, where the k-th level for $k \in \{0, 1, \dots, d\}$ is the set of vertices which are connected to the root via a path consisting of precisely k edges.

Sanity check!

1. What is the depth of the tree above?
2. Which vertices are on level 0 of the tree above? How about on level 3?

Where do rooted trees come in handy? Consider, for example, the setting of bacterial cell division. In this case, the root might represent a single bacterium, and each subsequent layer corresponds to cell division in which the bacterium divides into two new bacteria. Rooted trees can also be used to allow fast searching of ordered sets of elements, such as in *binary search trees*, which you may already have encountered in your studies.

One of the nice things about trees is that induction works particularly well in proving things about them. Let us demonstrate by proving that the first two definitions of a tree given above are indeed equivalent.

Theorem 6.1. *The statements "G is connected and contains no cycles" and "G is connected and has $n-1$ edges" are equivalent.*

Proof. We proceed by showing the forward and converse directions separately.

Forward direction. We prove using strong induction on n that if G is connected and contains no cycles, then G is connected and has $n-1$ edges. Assume $G=(V,E)$ is connected and contains no cycles.

Base case ($n=1$): In this case, G is a single vertex and has no edges. Thus, the claim holds.

Inductive hypothesis: Assume the claim is true for $1 \leq n \leq k$.

Inductive proof: We show the claim for $n=k+1$. Remove an arbitrary vertex $v \in V$ from G along with its incident edges, and call the resulting graph G'. Clearly, *removing* a vertex cannot create a cycle; thus, G' contains no cycles. However, removing v may result in a *disconnected* graph G', in which case the induction hypothesis cannot be applied to G' as a whole. Thus, we have two cases to examine — either G' is connected, or G' is disconnected. Here, we show the former case, as it is simpler and captures the essential proof ideas. The latter case is left as an exercise below.

So, assume G' is connected. But now G' is a connected graph with no cycles on k vertices, so we can apply the induction hypothesis to G' to conclude that G' is connected and has $k-1$ edges. Let us now add v back to G' to obtain G. How many edges can be incident on v? Well, since G' is connected, then if v is incident on more than one edge, G will contain a cycle. But by assumption G has no cycles! Thus, v must be incident on one edge, implying G has $(k-1)+1=k$ edges, as desired.

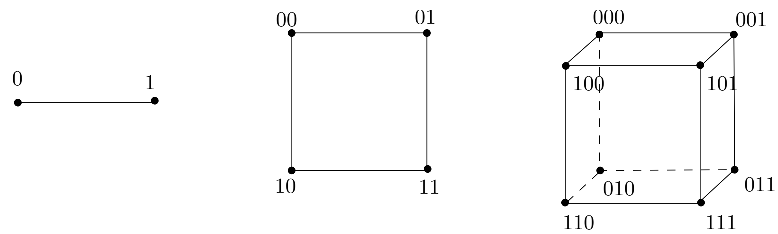
Converse direction. We prove using contradiction that if G is connected and has $n-1$ edges, then G is connected and contains no cycles. Assume G is connected, has $n-1$ edges, and contains a cycle. Then, by definition of a cycle, removing any edge in the cycle does not disconnect the graph G. In other words, we can remove an edge in the cycle to obtain a new connected graph G' consisting of $n-2$ edges. However, we claim that G' must be disconnected, which will yield our desired contradiction. This is because in order for a graph to be connected, it must have at least $n-1$ edges. This is a fact that you have to prove in the exercise below. This completes the proof of the converse direction. ☐

2.3 Hypercubes

We have discussed how complete graphs are a class of graphs whose vertices are particularly "well-connected." However, to achieve this strong connectivity, a large number of edges is required, which in many applications is infeasible. Consider the example of the *Connection Machine*, which was a massively parallel computer made by the company Thinking Machines in the 1980s. The idea of the Connection Machine was to have a million processors working in parallel, all connected via a communications network. If you were to connect each pair of such processors with a direct wire to allow them to communicate (i.e., if you used a complete graph to model your communications network), this would require on the order of 10^{12} wires! What the builders of the Connection Machine thus decided was to instead use a 20-dimensional *cube* to model their network, which still allowed a strong level of connectivity, while limiting the number of neighbors of each processor in the network to 20. This section is devoted to studying this particularly useful class of graphs, known as *hypercubes*.

The vertex set of the n-dimensional hypercube $G=(V,E)$ is given by $V=\{0,1\}^n$, where recall $\{0,1\}^n$ denotes the set of all n-bit strings. In other words, each vertex is labeled by a unique n-bit string, such as $00110\cdots0100$. The edge set E is defined as follows: Two vertices x and y are connected by edge $\{x,y\}$ if and only if x and y differ in exactly one bit position. For example, in the 4-dimensional hypercube, $x=0000$ and $y=1000$ are neighbors but $x=0000$ and $y=0011$ are not. More formally, $x=x_1x_2\ldots x_n$ and $y=y_1y_2\ldots y_n$

are neighbors if and only if there is an $i \in \{1, \ldots, n\}$ such that $x_i \neq y_i$ and $x_j = y_j$ for all $j \neq i$. To help you visualize the hypercube, we depict the 1-, 2-, and 3-dimensional hypercubes below.



There is an alternative and useful way to define the n-dimensional hypercube via recursion, which we now discuss. Define the 0-subcube (respectively, 1-subcube) as the $(n-1)$-dimensional hypercube with vertices labeled by $0x$ for $x \in \{0,1\}^{n-1}$ (respectively, $1x$ for $x \in \{0,1\}^{n-1}$). Then, the n-dimensional hypercube is obtained by placing an edge between each pair of vertices $0x$ in the 0-subcube and $1x$ in the 1-subcube.

Sanity check! Where are the 0- and 1-subcubes in the 3-dimensional hypercube depicted above? Can you use these along with the recursive definition above to draw the 4-dimensional hypercube? (Hint: See Figure 3(vii).)

Exercise. Prove that the n-dimensional hypercube has 2^n vertices. Hint: Use the fact that each bit has two possible settings, 0 or 1.

We began this section by singing the praises of the hypercube based on its connectivity properties; we now investigate these claims formally. Let us begin by giving two proofs of a simple property of the hypercube. Each proof relies on one of our two equivalent (direct and recursive, respectively) definitions of the hypercube.

Lemma 6.1. *The total number of edges in an n-dimensional hypercube is $n2^{n-1}$.*

Proof 1. The degree of each vertex is n, since n bit positions can be flipped in any $x \in \{0,1\}^n$. Since each edge is counted twice, once from each endpoint, this yields a total of $n2^n/2 = n2^{n-1}$ edges. $\square$

Proof 2. Let $E(n)$ denote the number of edges in the n-dimensional hypercube. By the second (recursive) definition of the hypercube, it follows that $E(n) = 2E(n-1) + 2^{n-1}$, where the term 2^{n-1} counts the number of edges between the 0-subcube and the 1-subcube. The base case is $E(1) = 1$. A straightforward proof by induction now shows that $E(n) = n2^{n-1}$. $\square$

Exercise. Use induction to show that, in Proof 2 above, $E(n) = n2^{n-1}$ for all $n \geq 1$.

Let us now focus on the question of connectivity, and prove that the n-dimensional hypercube is well-connected in the following sense: To disconnect any subset $S \subseteq V$ of vertices from the rest of the graph, a large number of edges must be removed. In particular, we shall see that the number of removed edges must scale with $|S|$. In the theorem below, recall that $V - S = \{v \in V : v \notin S\}$ is the set of vertices that are not in S.

Theorem 6.2. *Let $S \subseteq V$ be such that $|S| \leq |V - S|$ (i.e., that $|S| \leq 2^{n-1}$), and let E_S denote the set of edges connecting S to $V - S$, i.e.,*

$$E_S := \{\{u,v\} \in E \mid u \in S \text{ and } v \in V - S\}.$$

Then, it holds that $|E_S| \geq |S|$.

Proof. We proceed by induction on n.

Base case ($n = 1$): The 1-dimensional hypercube graph has two vertices 0 and 1, and one edge $\{0,1\}$. We also have the assumption $|S| \leq 2^{1-1} = 1$, so there are two possibilities. First, if $|S| = 0$, then the claim trivially holds. Otherwise, if $|S| = 1$, then $S = \{0\}$ and $V - S = \{1\}$, or vice versa. In either case we have $E_S = \{0,1\}$, so $|E_S| = 1 = |S|$.

Inductive hypothesis: Assume the claim holds for $1 \leq n \leq k$.

Inductive step: We prove the claim for $n = k + 1$. Recall that we have the assumption $|S| \leq 2^{n-1} = 2^k$. Let S_0 (respectively, S_1) be the vertices from the 0-subcube (respectively, 1-subcube) in S. We have two cases to examine: Either S has a fairly equal intersection size with the 0- and 1-subcubes, or it does not.

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

   In this case, we can apply the induction hypothesis separately to the 0- and 1-subcubes. This says that restricted to the 0-subcube itself, there are at least $|S_0|$ edges between $|S_0|$ and its complement (in the 0-subcube), and similarly there are at least $|S_1|$ edges between $|S_1|$ and its complement (in the 1-subcube). Thus, the total number of edges between S and $V - S$ is at least $|S_0| + |S_1| = |S|$, as desired.

2. **Case 2:** $|S_0| > 2^{k-1}$

   In this case, S_0 is unfortunately too large for the induction hypothesis to apply. However, note that since $|S| \leq 2^k$, we have $|S_1| = |S| - |S_0| \leq 2^{k-1}$, so we *can* apply the hypothesis to S_1. As in Case 1, this allows us to conclude that there are at least $|S_1|$ edges in the 1-subcube crossing between S and $V - S$.

   What about the 0-subcube? Here, we cannot apply the induction hypothesis directly, but there is a way to apply it after a little massaging. Consider the set $V_0 - S_0$, where V_0 is the set of vertices in the whole 0-subcube. Note that $|V_0| = 2^k$ and $|V_0 - S_0| = |V_0| - |S_0| = 2^k - |S_0| < 2^k - 2^{k-1} = 2^{k-1}$. Thus, we *can* apply the inductive hypothesis to the set $V_0 - S_0$. This yields that the number of edges between S_0 and $V_0 - S_0$ is at least $2^k - |S_0|$. Adding our totals for the 0-subcube and the 1-subcube so far, we conclude there are at least $2^k - |S_0| + |S_1|$ crossing edges between S and $V - S$. However, recall our goal was to show that the number of crossing edges is at least $|S|$; thus, we are still short of where we wish to be.

   But there are still edges we have not accounted for — namely, those in E_S which cross between the 0- and 1-subcubes. Since there is an edge between every vertex of the form $0x$ and the corresponding vertex $1x$, we conclude there are at least $|S_0| - |S_1|$ edges in E_S that cross between the two subcubes. Thus, the total number of edges crossing is at least $2^k - |S_0| + |S_1| + |S_0| - |S_1| = 2^k \geq |S|$, as desired.

□

3 Revisiting Königsberg: Eulerian Tours

With a formal underpinning in graph theory under our belts, we are ready to revisit the Seven Bridges of Königsberg. What exactly is this problem asking? It says: Given a graph G (in this case, G is an abstraction of Königsberg), is there a walk in G that uses each edge exactly once? We call any such walk in a graph an *Eulerian walk*. (In contrast, by definition a walk can normally visit each edge or vertex as many times as desired.) Moreover, if an Eulerian walk is closed, then it is called an *Eulerian tour*. Thus, the Seven Bridges of Königsberg asks: Given a graph G, does it have an Eulerian tour?[2] We now give a precise characterization of this in terms of simpler properties of the graph G. For this, define an *even degree* graph as a graph in which all vertices have even degree.

Theorem 6.3 (Euler's Theorem (1736))**.** *An undirected graph $G = (V,E)$ has an Eulerian tour if and only if G is even degree, and connected (except possibly for isolated vertices).*

Proof. To prove this, we must establish two directions: "if", and "only if".

Only if. We give a direct proof for the forward direction, i.e., if G has an Eulerian tour, then it is connected and has even degree. Assume that G 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, we prove that each vertex has even degree by showing that all edges incident to a vertex can be paired up. Notice that every time the tour enters a vertex along an edge it exits along a different edge. We can pair these two edges up (they are never again traversed in the tour). The only exception is the start vertex, where the first edge leaving it cannot be paired in this way. But notice that by definition, the tour necessarily ends at the start vertex. Therefore, we can pair the first edge with the last edge entering the start vertex. So all edges adjacent to any vertex of the tour can be paired up, and therefore each vertex has even degree.

If. This direction is trickier to prove. We give a recursive algorithm that, given a graph G satisfying the condition in the theorem, finds an Eulerian tour, and prove by induction that it correctly outputs such a tour.

We start with a useful subroutine, $\textsc{FindTour}(G,s)$, which finds a tour (not necessarily Eulerian) in G. FINDTOUR is very simple: it just starts walking from a vertex $s \in V$, at each step choosing any untraversed edge incident to the current vertex, until it gets stuck because there is no more adjacent untraversed edge. We now prove that FINDTOUR must in fact get stuck at the original vertex s.

Claim: $\textsc{FindTour}(G,s)$ must get stuck at s.

Proof of claim: An easy proof by induction on the length of the walk shows that when FINDTOUR enters any vertex $v \neq s$, it will have traversed an odd number of edges incident to v, while when it enters s it will have traversed an even number of edges incident to s. Since every vertex in G has even degree, this means every time it enters $v \neq s$, there is at least one untraversed edge incident to v, and therefore the walk cannot get stuck. So the only vertex it can get stuck at is s. The formal proof is left as an exercise. □

The algorithm $\textsc{FindTour}(G,s)$ returns the tour it has traveled when it gets stuck at s. Note that while $\textsc{FindTour}(G,s)$ always succeeds in finding a tour, the tour it finds may not be Eulerian because it may not traverse all the edges of G.

We now give a recursive algorithm $\textsc{Euler}(G,s)$ that outputs an Eulerian tour starting and ending at s. $\textsc{Euler}(G,s)$ invokes another subroutine $\textsc{Splice}(T,T_1,\dots,T_k)$ which takes as input a number of *edge dis-*

[2] Technically, the problem asks this question about a multigraph G, in which multiple edges between pairs of vertices are allowed. However, as we shall see, our answer to the question is equally valid in this slightly more general setting: i.e., our proofs will work even if there are multiple edges.

joint tours $T, T_1, \ldots, T_k$ ($k \geq 1$), with the condition that the tour T intersects each of the tours $T_1, \ldots, T_k$ (i.e., T shares a vertex with each of the T_i's). The procedure $\text{SPLICE}(T, T_1, \ldots, T_k)$ outputs a single tour T' that traverses all the edges in $T, T_1, \ldots, T_k$, i.e., it splices together all the tours. The combined tour T' is obtained by traversing the edges of T and, whenever a vertex s_i that intersects another tour T_i is reached, taking a detour to traverse T_i from s_i back to s_i again, and then continuing to traverse T.

The algorithm $\text{EULER}(G, s)$ is given as follows:

Function $\text{EULER}(G, s)$
$T = \text{FINDTOUR}(G, s)$
Let $G_1, \ldots, G_k$ be the connected components when the edges in T are removed from G, and let s_i be the first vertex in T that intersects G_i
Output $\text{SPLICE}(T, \text{EULER}(G_1, s_1), \ldots, \text{EULER}(G_k, s_k))$
end EULER

We prove by induction on the size of G that $\text{EULER}(G, s)$ outputs an Eulerian Tour in G. The same proof works regardless of whether we think of size as number of vertices or number of edges. For concreteness, here we use number of edges m of G.

Base case: $m = 0$, which means G is empty (it has no edges), so there is no tour to find.

Induction hypothesis: $\text{EULER}(G, s)$ outputs an Eulerian Tour in G for any even degree, connected graph with at most $m \geq 0$ edges.

Induction step: Suppose G has $m+1$ edges. Recall that $T = \text{FINDTOUR}(G, s)$ is a tour, and therefore has even degree at every vertex. When we remove the edges of T from G, we are therefore left with an even degree graph with less than m edges, but it might be disconnected. Let $G_1, \ldots, G_k$ be the connected components of the remaining graph. Each such connected component has even degree and is connected. Moreover, T intersects each of the G_i, and as we traverse T there is a first vertex where it intersects G_i. Call this vertex s_i. By the induction hypothesis $\text{EULER}(G_i, s)$ outputs an Eulerian tour of G_i. Now according to its definition, SPLICE splices the individual tours together into one large tour whose union is all the edges of G, and hence an Eulerian tour. □

Sanity check! Why does Theorem 6.3 imply that the answer to the Seven Bridges of Königsberg is no?

---

4 Planar Graphs

A graph is *planar* if it can be drawn in the plane without crossings. For example, the first four graphs (i)-(iv) shown in Figure 3 below are planar. Notice that graphs (i) and (ii) are the same, but drawn differently. Even though the second drawing has crossings, the graph is still considered planar since it is possible to draw it without crossings.

The other three graphs (v)–(vii) in Figure 3 are not planar. Graph (v) is the infamous "three houses-three wells graph," also called $K_{3,3}$. Graph (vi) is the complete graph with five nodes, or K_5. Graph (vii) is the four-dimensional cube. We shall soon see how to prove that all three of these latter graphs are non-planar.

Planar graphs are one of the most widely-studied and important sub-classes of graphs. One reason for their importance is that there are many algorithms that work much more efficiently for problems on planar graphs than for the same problems on general graphs.

When a planar graph is drawn in the plane, one can distinguish, besides its vertices and edges, its *faces*

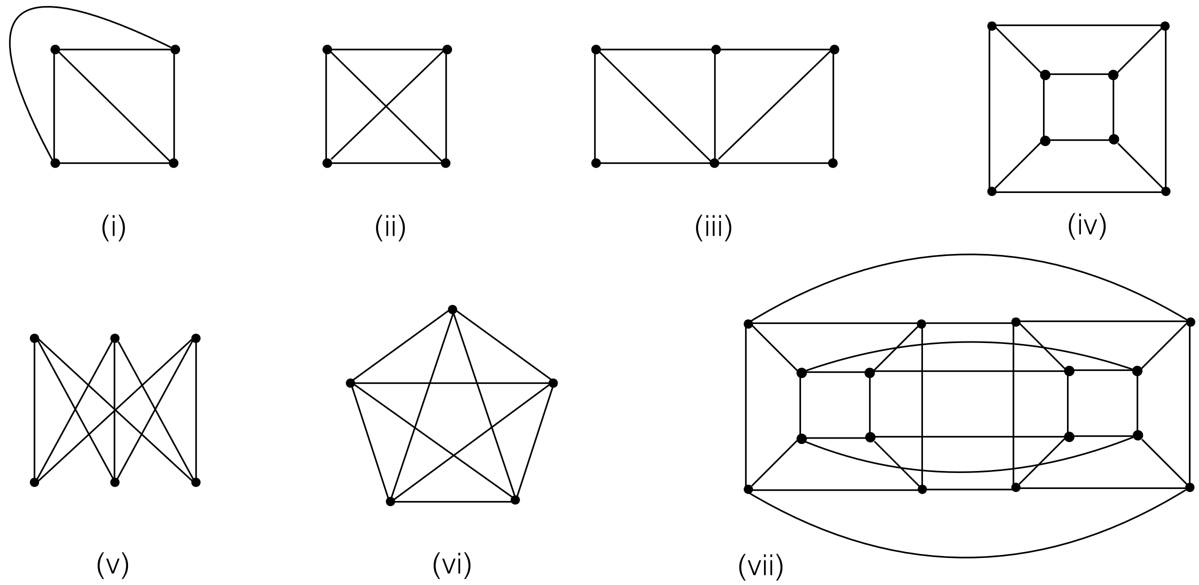


Figure 3: Examples of planar and non-planar graphs.

(more precisely, these are faces of the drawing, not of the graph itself). The faces are the regions into which the graph subdivides the plane. One of them is infinite, and the others are finite. We will denote the number of faces by f; in addition, we'll denote the number of vertices by v and the number of edges by e. For example, for graph (i) above $v = 4$, $e = 6$ and $f = 4$. For graph (iv) (the cube), $v = 8$, $e = 12$ and $f = 6$.

One basic and important fact about planar graphs is *Euler's formula*, $v + f = e + 2$ (check it for graphs (i), (iii) and (iv) above). It has an interesting story. The ancient Greeks knew that this formula held for all polyhedra (check it for the cube, the tetrahedron, and the octahedron, for example), but could not prove it. How do you do induction on polyhedra? How do you apply the induction hypothesis? What is a polyhedron minus a vertex, or an edge? In the 18th century Euler realized that this is an instance of the inability to prove a theorem by induction *because it is too weak*, something that we saw several times when we were studying induction. To prove the theorem, one has to generalize polyhedra. And the right generalization is *planar graphs*.

Exercise. Can you see why planar graphs generalize polyhedra? Why are all polyhedra (without "holes") planar graphs?

Theorem 6.4. *(Euler's formula) For every connected planar graph,* $v + f = e + 2$

The proof will be a straightforward induction, but the base case will make use of the concept of a *tree*. This is a special kind of graph which is connected and acyclic (i.e., has no cycles). Trees are connected graphs with the smallest possible number of edges, and we will look at them in more detail in the next section. One fact we'll need here about trees is that any tree with v vertices always has exactly $e = v - 1$ edges.

Proof. By induction on e. The formula certainly holds when $e = 0$ since then $v = f = 1$.

Now consider any connected planar graph with $e > 1$. There are two cases:

- If the graph is a tree, then $f = 1$ (drawing a tree in the plane does not subdivide the plane), and $e = v - 1$ (see the next section of these notes).

- If the graph is not a tree, it must have a cycle. Take any cycle and delete any edge of the cycle. This amounts to reducing both e and f by one, without changing v. By induction, the formula is true in the smaller graph, and so it must be true in the original one as well.

This completes the inductive proof. □

Exercise. What happens when the graph is not connected? How does the number of connected components enter the formula?

Now let's see how to use Euler's formula to decide whether some graphs are planar or not. Take a planar graph with f faces, and consider one face. It has a number of *sides*, that is, edges that bound it. Note that an edge may be counted twice, if it has the same face on both sides, as happens for example in a tree (such edges are called *bridges*). Denote by s_i the number of sides of face i. Now, if we add the s_i's we must get $2e$, because each edge is counted exactly twice, once for the face on its right and once for the face on its left (these may coincide if the edge is a bridge). We conclude that, in any planar graph,

$$\sum_{i=1}^{f} s_i = 2e. \tag{1}$$

Now notice that, since we don't allow parallel edges between the same two vertices, and if we assume that there are at least two edges in the graph (so there are at least three vertices), every face has at least three sides, or $s_i \geq 3$ for all i. It follows that $3f \leq 2e$. Solving for f and plugging into Euler's formula we get

$$e \leq 3v - 6.$$

This is an important fact. First it tells us that planar graphs are *sparse*, i.e., they cannot have too many edges. For example, a 1,000-vertex connected graph can have anywhere between 999 and roughly half a million edges. This inequality tells us that for *planar* 1,000-vertex graphs the range is much smaller: between 999 and $2,994$ edges.

It also tells us that K_5 is not planar: Just notice that it has five vertices and ten edges.

$K_{3,3}$ has $v = 6$, $e = 9$ so it passes the planarity test with flying colors. However, recall that "a planar graph satisfies $e \leq 3v - 6$" doesn't mean that "if a graph satisfies $e \leq 3v - 6$, then it is a planar graph." We must think a little harder to show that $K_{3,3}$ is non-planar. Notice that, if we had drawn it in the plane, there would be no triangles: this is because in any triangle either two wells or two houses would have to be connected, but that is not possible. So, by the same reasoning as before, Equation (1) now gives us $4f \leq 2e$, and solving for f and plugging into Euler's formula, $e \leq 2v - 4$, which shows that $K_{3,3}$ is non-planar.

So, we have established that K_5 and $K_{3,3}$ are both non-planar. Actually, there is something deeper going on here: in some sense, these are *the only non-planar graphs*. This is made precise in the following famous result, due to the Polish mathematician Kuratowski (this is what "K" stands for).

Theorem 6.5. *A graph is non-planar if and only if it contains K_5 or $K_{3,3}$.*

"Contains" here means that one can identify nodes in the graph (five in the case of K_5, six in the case of $K_{3,3}$) which are connected as in the corresponding graph through paths (possibly single edges), and such that no two of these paths share a vertex (except of course for their endpoints). For example, the 4-cube shown below (graph (vii) from earlier) is non-planar because it contains $K_{3,3}$, as shown below.

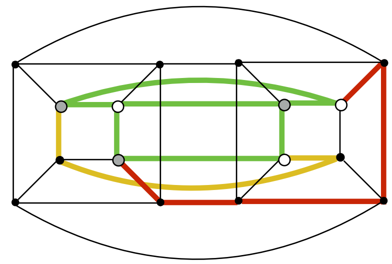


Figure 4: Copy of $K_{3,3}$ in the 4-cube. The vertices of $K_{3,3}$ are shown as white and grey points, respectively. Green lines are single-edge paths between these vertices. Red and amber lines show multi-edge paths.

Exercise. Can you find K_5 in the same graph?

One direction of Kuratowski's theorem is obvious: If a graph contains one of these two non-planar graphs, then of course it is itself non-planar. The other direction, namely that in the absence of these graphs we can draw any graph in the plane, is difficult. If you are interested in reading a proof, you may want to type "proof of Kuratowski's theorem" into your favorite search engine.

4.1 Duality and Coloring

There is an interesting *duality* between planar graphs. For example, the Greeks knew that the octahedron and the cube are "dual" to each other, in the sense that the faces of one can be put in correspondence with the vertices of the other (think about it). The tetrahedron is self-dual. And the dodecahedron and the icosahedron (look for images in the web if you don't know these pretty things) are also dual to one another.

What does this mean? Take a planar graph G, and assume it has no bridges and no degree-two nodes. Draw a new graph G^*: Start by placing a node on each face of G. Then draw an edge between two faces if they touch at an edge — draw the new edge so that it crosses that edge. The result is G^*, also a planar graph. Notice now that, if you construct the dual of G^*, it is the original graph: $(G^*)^* = G$.

Duality is a convenient consideration when thinking about planar graphs. Also, it tells us that "coloring a political map so that no two countries who share a border have the same color" is the same problem as "coloring the vertices of a planar graph (the dual of the political map) so that no two adjacent vertices have the same color."

A famous theorem states that four colors are always enough! (Search for "four color theorem".)

We shall prove something weaker:

Theorem 6.6. *Every planar graph can be colored with five colors.*

Before proceeding with the proof, we consider alternative legal colorings of a graph. We first take the subset of vertices of two colors, say 1 and 2, and compute the connected components. We note that we can switch the two colors in a single connected component; consider any edge that is not in the connected component is clearly fine, any edge in the connected component has both endpoints switched which is also fine.

Proof. Induction on v. The base case is not worth talking about, so we go directly to the inductive step. Let G be a planar graph. I claim there is a node of degree five or less. In proof, consider the inequality $e \leq 3v - 6$. If all vertices had degree six or more, then e would be at least $3v$.

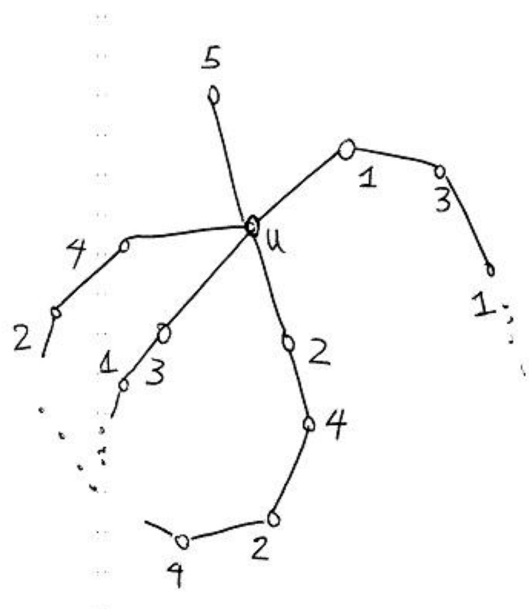
So, consider a node u of degree five or less. If it has degree four or less, we are done: Remove u, color the remaining graph with 5 colors (by induction), and then put u back in and color it by a color that is missing from its neighbors.

So, u must have 5 vertices, and in the coloring of $G-u$ they all got different colors. Look at them clockwise around u, and call them u_1, u_2, u_3, u_4, u_5, and their colors $1, 2, 3, 4, 5$.

Now try to change the color of u_2 to color 4 by determining the connected component containing u_2 and consisting of vertices that are colored 2 or 4. If u_4 is in the connected component, switching 2 and 4 is not helpful. But, we do know that there is a path between u_2 and u_4 colored 2 and 4.

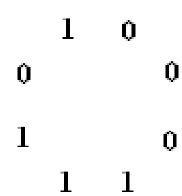
Similarly, we can try to change the color of u_1 to 3, and this will succeed unless there is a path between u_1 to u_3 colored 1 and 3.

If both of these attempts fail, then we get two paths: one from u_1 to u_3 colored 1 and 3, and the other from u_2 to u_4 colored 2 and 4. But planarity says that these two paths must intersect at some vertex. What is the color of this vertex? □



5 Practice Problems

1. 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 such as 000, 001, 010, etc. It turns out that such sequences can be generated from the *de*

Bruijn graph, which is a directed graph $G = (V,E)$ on the vertex set $V = \{0,1\}^{n-1}$, i.e., the set of all $n-1$ bit strings. Each vertex $a_1a_2...a_{n-1} \in V$ has two outgoing edges:

$$(a_1a_2...a_{n-1},\ a_2a_3...a_{n-1}0) \in E \quad \text{and} \quad (a_1a_2...a_{n-1},\ a_2a_3...a_{n-1}1) \in E.$$

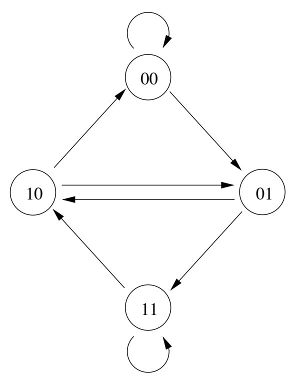
Therefore, each vertex also has two incoming edges:

$$(0a_1a_2...a_{n-2},\ a_1a_2...a_{n-1}) \in E \quad \text{and} \quad (1a_1a_2...a_{n-2},\ a_1a_2...a_{n-1}) \in E.$$

For example, for $n = 4$, the vertex 110 has two outgoing edges directed toward 100 and 101, and two incoming edges from 011 and 111. Note that these are directed edges, and self-loops are permitted.

The de Bruijn sequence is generated by an Eulerian tour in the de Bruijn graph. Euler's theorem (Theorem 6.3) 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 $n = 3$.



Find the Eulerian tour of this graph that generates the de Bruijn sequence given above.

2. In this question, we complete the induction component of the proof of Theorem 6.1.

   (a) Suppose in the proof that G' has two distinct connected components G'_1 and G'_2. Complete the inductive step to show that G is connected and has $n-1$ edges. (Hint: Argue that you can apply the induction hypothesis to G'_1 and G'_2 separately. Note that this requires strong induction!)

   (b) More generally, G' may have $t \geq 2$ distinct connected components G'_1 through G'_t — generalize your argument above to this setting in order to complete the proof of Theorem 6.1.

Life lesson. This question teaches you a general paradigm for solving problems, be it in research or everyday life. Specifically, when faced with a difficult problem (such as the proof of Theorem 6.1), first try to solve it in the simplest case possible (in this case, when G' is connected). Then, extend your solution to handle more difficult cases until you establish the general claim (in this case, when G' has t connected components).

---

3. Complete the proof of the converse direction in Theorem 6.1 by proving, using induction on the number of vertices n, that any connected graph must have at least $n-1$ edges.