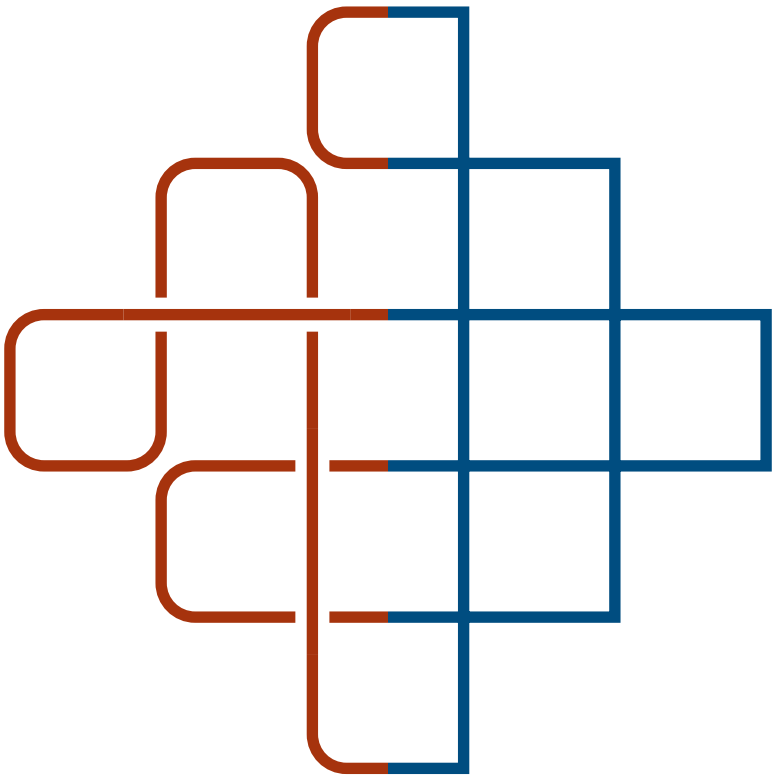
Mikhail Lavrov

Start Doing Graph Theory

Part II: Vertex Degrees



available online at https://vertex.degree/

Contents

About this document

This is Part II of *Start Doing Graph Theory*. It covers the properties of degree sequences in graph theory, and some of the things we can learn from them, along with an introduction to multigraphs and directed graphs.

Right now, only parts I through IV are ready. When the book is finished, I plan to provide a variety of ways to read it: a single PDF of the whole book, several smaller PDFs like this one, and eventually an HTML version. For now, think of this document as a preview!

5 Regular graphs

The purpose of this chapter

This chapter introduces the graphic sequence problem, which we will look at in more detail in Chapter 6. If your goal is to become more comfortable with proof-writing in graph theory, then the proof of Theorem 5.1 is worth reading with additional care: it is our first existence proof, where our goal is not to prove something about all graphs of some time, but to prove that a graph satisfying a certain condition exists.

This is always a difficult kind of proof to read, since constructions often feel highly specific and unmotivated when you first encounter them. You should, first of all, keep in mind that alternate proofs of Theorem 5.1 exist which present completely different constructions! The best way to learn would be to skip the proof of the theorem at first reading and to try to find your own direction by looking at small examples and trying to generalize—but that is much harder. Construction proofs will get easier with more experience, though, after you've seen many different ideas that you can borrow from.

In this chapter, you might also notice a design decision of mine: I chose *not* to have a chapter introducing many "boring" definitions like those of graph unions and graph complements, but to wait to give them until we had a problem that would motivate them. Similarly, there are many notable graph families, which I have only presented as needed.

Aside from these definitions, much of the second half of this chapter is skippable in that it's not a necessary prerequisite for what comes next; I have tried to give a general feeling of the variety of graphs that are out there, and the ways we can think about classifying them.

5.1 Degree sequences

To state the graphic sequence problem, we need some definitions first.

Definition 5.1. *If G is a graph, the* ***degree sequence of G*** *is a sequence of numbers that gives all the degrees of all the vertices of G.*

Despite the word "sequence", these don't come in any particular order, because the vertices of a graph don't have a particular order. (There might be a natural order you're tempted to use based on the way we drew a graph, but we don't want the properties of graphs to depend on how we draw them!) It would have been more reasonable to talk about the "degree multiset" of a graph: a **multiset** keeps track of how often its elements appear, but not their order, which is exactly what we want here.

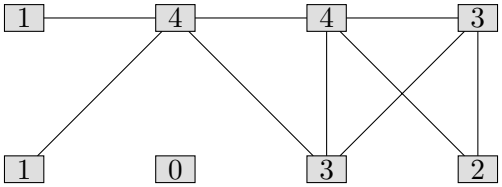


Figure 5.1: An 8-vertex graph labeled with the degrees of its vertices

Instead, the convention is to write the degree sequence of a graph in descending order.[1] We do this to avoid the temptation of looking for meaning in the order of the numbers, and because it will be convenient for working with the degree sequence. For example, we would say that Figure 5.1 shows a graph with degree sequence $4, 4, 3, 3, 2, 1, 1, 0$.

It is a relatively quick and painless process to examine a graph, count the number of lines poking out of each point, and in doing so compute the degree sequence. For this reason, we often look to the degree sequence first to try to learn about a graph; even if it does not always tell us what we want to know, it is a quick way to start! This is, for example, a good way to determine if two graphs are isomorphic.

The inverse problem is harder. That is the problem of taking a sequence of nonnegative integers in descending order, and determining if it is the degree sequence of some graph. Some terminology:

Definition 5.2. *A sequence $d_1, d_2, \ldots, d_n$ where $d_1 \geq d_2 \geq \cdots \geq d_n$ is a **graphic sequence** if there exists a graph G with that degree sequence: a graph G with $V(G) = \{x_1, x_2, \ldots, x_n\}$ such that $\deg(x_i) = d_i$ for $i = 1, 2, \ldots, n$.*

Thus, the inverse problem is to ask: how do we tell when a sequence of numbers is a graphic sequence? For example, $4, 4, 3, 3, 2, 1, 1, 0$ is a graphic sequence, and the graph in Figure 5.1 is a proof of this. (There are other graphs with the same degree sequence, too.)

Here are a few quick puzzles about graphic sequences that illustrate some things you already know about them, and some things you have yet to learn.

Problem 5.1. *Is the sequence $8, 8, 8, 8, 8, 8, 8, 8$ graphic?*

Answer to Problem 5.1. No: the terms are too big. This sequence would like to be the degree sequence of an 8-vertex graph, but the largest possible degree in an 8-vertex graph is 7, achieved when a vertex is adjacent to all 7 other vertices. □

Problem 5.2. *Is the sequence $5, 5, 5, 3, 3, 3, 1, 1, 1$ graphic?*

Answer to Problem 5.2. No: this sequence violates Corollary 4.2 to the Handshake Lemma. A graph cannot have an odd number of vertices with an odd degree. □

[1] We must allow for the possibility of ties. By convention, when mathematicians call something a "decreasing sequence", it means that every term is strictly less than the previous, which is generally false for degree sequences; when two consecutive terms can be equal, this is called a "non-increasing sequence." This is awkward, so I will say that a sequence in "descending order", which does not have a standard meaning, may also contain equal terms.

Problem 5.3. *Is the sequence* $4, 3, 2, 1, 0$ *graphic?*

Answer to Problem 5.3. No: suppose for contradiction that G is a graph with degree sequence $4, 3, 2, 1, 0$. Let x be the vertex with degree 4 and let y be the vertex with degree 0. Then x is adjacent to all 4 other vertices, so in particular $xy \in E(G)$; however, y is adjacent to 0 other vertices, so in particular $xy \notin E(G)$. Therefore it is impossible for G to exist. □

Question:	How do the solutions to Problem 5.1 and Problem 5.2 generalize?
Answer:	They give us two rules: in an n-term graphic sequence, the values must be integers between 0 and $n-1$, and an even number of them must be odd.

Question:	Are these two rules sufficient?
Answer:	No, and that's why Problem 5.3 is there. It's a 5-term sequence in which the values are integers between 0 and 4, and 2 of them are odd; however, the sequence is not graphic.

We should think of the rules we deduced from the solutions to Problem 5.1 and Problem 5.2 as simple preliminary tests we can carry out before thinking about a sequence very hard. If a sequence fails one of the two preliminary tests, it's definitely not graphic, but even if it passes both preliminary tests, it still might not be graphic. We will eventually develop a comprehensive theory of graphic sequences, which will encompass the argument we used to solve Problem 5.3 as a special case.

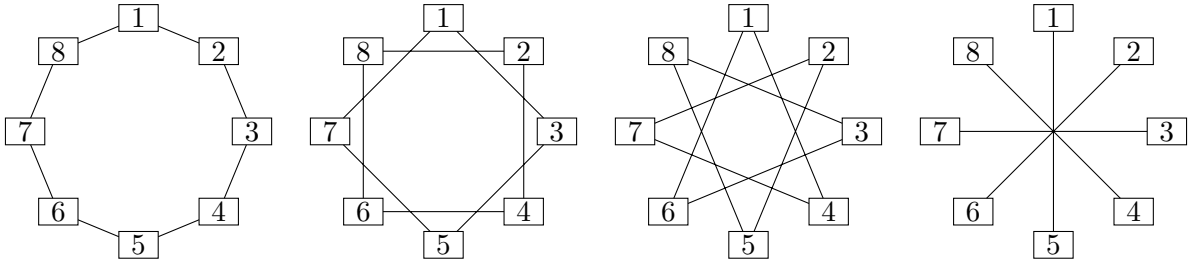
5.2 Regular graphs

As a special case, we can consider the graphic sequence problem for a sequence in which all terms are equal:

Definition 5.3. *A* ***regular graph*** *is a graph in which every vertex has the same degree. More specifically, an* ***r-regular graph*** *is a graph in which every vertex has degree r. The degree sequence of such a graph is $r, r, \ldots, r$.*

Making a definition does not guarantee that any object satisfying the definition exists. So when do regular graphs exist?

Question:	Suppose the sequence $r, r, \ldots, r$ with n terms is graphic. What do our two simple preliminary tests tell us about the relationship between r and n?
Answer:	We must have $0 \leq r \leq n-1$, and if r is odd, then n must be even.



(a) $I_{8,1}$ (b) $I_{8,2}$ (c) $I_{8,3}$ (d) $I_{8,4}$

Figure 5.2: Several examples of ingredient graphs

In fact, the existence problem for regular graphs is much easier than the general case: these are the only two conditions!

Theorem 5.1. *An r-regular graph on n vertices exists whenever $0 \leq r \leq n-1$ and at least one of r or n is even.*

Proof. To prove that an r-regular graph on n vertices exists, we need to construct one.

For most values of r and n, there are many many options available. We just have to show how to construct one of them. The hard part about this proof is that we have to give a general rule based on the values of r and n. (For specific values of r and n, the proof would be very short and just look like "here, look at the graph I drew.") Like a cooking recipe, our proof has to adjust to the size of the output we want to produce.

Our recipe for constructing an r-regular graph on n vertices begins by constructing several "ingredient graphs". These will all have n vertices called $1, 2, \ldots, n$, and we will draw them spaced evenly around a circle.

The **ingredient graph $I_{n,k}$** is defined[2] to be the graph with an edge between every pair of vertices that are "k steps apart" around the circle. Formally, it has an edge ij whenever $i - j \equiv \pm k \pmod{n}$.[3] Figure 5.2 shows several examples.

Many of the ingredient graphs $I_{n,k}$ are isomorphic to the cycle graph C_n: this is true, for instance, of $I_{8,1}$ (Figure 5.2a) and $I_{8,3}$ (Figure 5.2c). We do care about the difference between $I_{8,1}$ and $I_{8,3}$, though, even though they're isomorphic: we will want to combine our ingredients, so the exact edge sets matter for our proof.

In other cases, such as $I_{8,2}$ (Figure 5.2b), $I_{n,k}$ has several connected components isomorphic to cycle graphs. Since all cycle graphs are 2-regular, $I_{n,k}$ will usually also end up being 2-regular, whether it has one component or many. This can be seen from the definition: vertex i is adjacent to vertex j when $j \equiv i + k \pmod{n}$ and when $j \equiv i - k \pmod{n}$, giving it two neighbors.

The only exceptional cases occur when n is even and $k = n/2$, such as for example with $I_{8,4}$ (Figure 5.2d). Here, the vertex k steps around the circle from vertex i is the same no matter

[2] Defined only by me, as far as I know. Unlike most definitions in this textbook, the term "ingredient graph" is not standard terminology.

[3] If you're unfamiliar with modular arithmetic, this notation—which I will use in this textbook almost always to define graphs with rotational symmetry, as is the cases here—just means that our arithmetic wraps around, so we treat $n + 1$ the same as 1, $n + 2$ the same as 2, and so on.

which way we go: $i+k \equiv i-k \pmod{n}$. So each vertex only has 1 neighbor, and the ingredient graph $I_{n,n/2}$ is 1-regular.

There is one more observation to make about the ingredient graphs. For any two vertices i and j in the set $\{1, 2, \ldots, n\}$, there is exactly one graph $I_{n,k}$ containing the edge ij. This value of k is equal to either $|i-j|$ or to $n-|i-j|$, whichever is smaller: the number of steps around the circle from i to j in the direction where that number of steps is shorter.

This last observation means that we can build an r-regular graph for any r by taking the union of several ingredient graphs. What do we mean by the "union" of two graphs? In this case, we're considering graphs with the same set of vertices. For two graphs G and H with $V(G) = V(H)$, we define the **union**, denoted $G \cup H$, to be the graph with the same set of vertices as G and H had, and all edges that are contained in either G or H. That is, $V(G \cup H) = V(G) = V(H)$, and $E(G \cup H) = E(G) \cup E(H)$.

Now that we've laid out the ingredients for our recipe, here is the recipe itself: the rule by which we can get an r-regular graph on n vertices. We assume the two conditions in the statement of Theorem 5.1: $0 \leq r \leq n-1$, and if r is odd, then n is even.

- If r is even, take the union $I_{n,1} \cup I_{n,2} \cup \cdots \cup I_{n,r/2}$.

  For each i, vertex i has 2 incident edges in each of the $r/2$ graphs in the union, and all of these incident edges are different. In total, $\deg(i) = 2+2+\cdots+2 = 2 \cdot \frac{r}{2} = r$.

- If r is odd and n is even, take the union $I_{n,1} \cup I_{n,2} \cup \cdots \cup I_{n,(r-1)/2} \cup I_{n,n/2}$.

  For each i, vertex i has 2 incident edges in the first $(r-1)/2$ graphs in the union, and 1 incident edge in $I_{n,n/2}$. Again, all of these edges are different. In total, $\deg(i) = 2+2+\cdots+2+1 = 2 \cdot \frac{r-1}{2} + 1 = r$.

When $r = 0$, there's nothing to take the union of. We must treat this as a special case: take a graph which still has vertices $1, 2, \ldots, n$, but no edges at all. This graph is 0-regular!

In all cases, we obtain an n-vertex graph in which every vertex has degree r, proving the theorem. □

Question:	What graph do we end up constructing when $r = 1$ (and n is even)?
Answer:	Here, $\frac{r-1}{2}$ is 0, so we don't include any of the 2-regular ingredient graphs, and only take $I_{n,n/2}$.

Question:	What graph do we end up constructing when $r = n-1$?
Answer:	Here, we end up taking the union of all the ingredient graphs we have, and get a result isomorphic to the complete graph K_n.

The graphs we construct in the proof of this theorem have a special name. (Unlike the "ingredient graphs" $I_{n,k}$ used in the proof itself, this notation is at least a little bit standard.)

Definition 5.4. *The r-regular graph on n vertices constructed in the proof of the Theorem 5.1 is called the **Harary graph**, denoted $H_{n,r}$.*

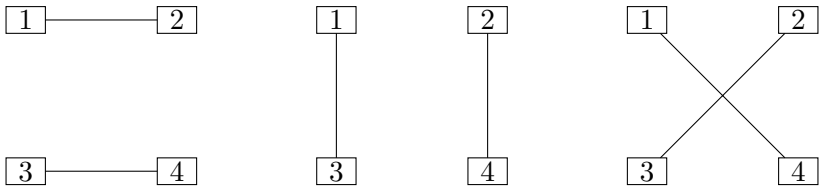
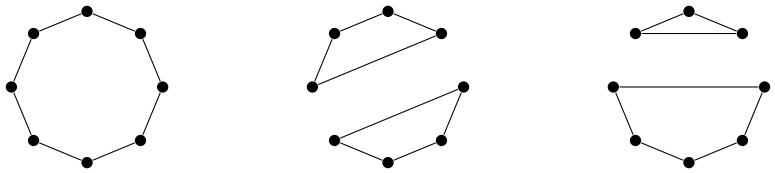


Figure 5.3: Three different(?) 1-regular graphs with vertex set $\{1, 2, 3, 4\}$



(a) C_8 (b) $C_4 \sqcup C_4$ (c) $C_3 \sqcup C_5$

Figure 5.4: Three non-isomorphic 2-regular graphs with 8 vertices

5.3 How many regular graphs are there?

Though Theorem 5.1 proves the existence of r-regular, n-vertex graphs with all compatible values of n and r, we've only constructed one graph for each pair (n, r). How many possibilities exist in total? Let's start with some small values of r and see how the story unfolds.

When $r = 0$, every vertex is an isolated vertex: it has degree 0. There cannot be any edges. For any possible vertex set, there is exactly one graph, sometimes called the **empty graph**.

When $r = 1$, whether we believe in one possibility or multiple possibilities depends on how we count. You see, suppose x is a vertex in a 1-regular graph. Then x has only one neighbor, which we can call y. Vertex y also has only one neighbor, and we already know what it is: x. Thus, x and y form a 2-vertex connected component with each other and nothing else, and the entire graph is split up into $n/2$ such components. (A 1-regular graph, as we already know, is only possible when n is even, so $n/2$ is an integer.)

The counting difficulty comes in when we ask how many ways there are to choose these components. In one sense, there are many options: Figure 5.3 shows three different possibilities in just the 4-vertex case, and the total number of possibilities grows quickly. However, they are all isomorphic to each other!

To capture this sentiment, we might say that there is only one n-vertex 1-regular graph **up to isomorphism**. Alternatively, we can say that there are many **labeled** 1-regular graphs on n vertices when n is even (for a fixed vertex set such as $\{1, 2, \ldots, n\}$) but a unique **unlabeled** graph.

Moving on, let's consider what happens when $r = 2$. We know by Theorem 4.4 that such a graph contains a cycle. In fact, in a k-vertex cycle $x_0, x_1, x_2, \ldots, x_{k-1}, x_0$, each vertex has 2 neighbors in that cycle already, so it can have no other neighbors in the graph: the vertices of the cycle form a connected component!

The difference between the 1-regular and 2-regular case is that the connected components can vary in size. Figure 5.4 shows three non-isomorphic possibilities for an 8-vertex 2-regular graph:

it could be isomorphic C_8 (Figure 5.4a), or the union of two copies of C_4 (Figure 5.4b), or the union of copies of C_3 and C_5 (Figure 5.4c).

By the way, what is this "union"? It is not union of two graphs with the same vertex set, as it was in the proof of Theorem 5.1. (In that sense, $C_4 \cup C_4$ would just be C_4.) Rather, we want to take a **disjoint union**, in which there is no overlap between the vertices coming from one C_4 and the vertices coming from the other.

Here are the complete definitions of the two notions of union that we've seen in this chapter:

Definition 5.5. *If G and H are two graphs, then their **union** $G \cup H$ is the graph with vertex set $V(G) \cup V(H)$ and edge set $E(G) \cup E(H)$.*

Definition 5.6. *The **disjoint union** of G and H, written $G \sqcup H$, is a union $G' \cup H'$ where G' and H' are isomorphic to G and H but $V(G') \cap V(H') = \varnothing$.*

Definition 5.6 only tells us what $G \sqcup H$ is up to isomorphism. It's possible to come up with a specific rule for how G' and H' are created from G and H, but usually the notation is used when the exact vertex set of $G \sqcup H$ does not matter.

To return to our regular graphs, we can conclude that all 2-regular graphs on n vertices are isomorphic to a disjoint union of one or more cycle graphs. To put it another way: up to isomorphism, the only connected 2-regular graph on n vertices is the cycle graph C_n.

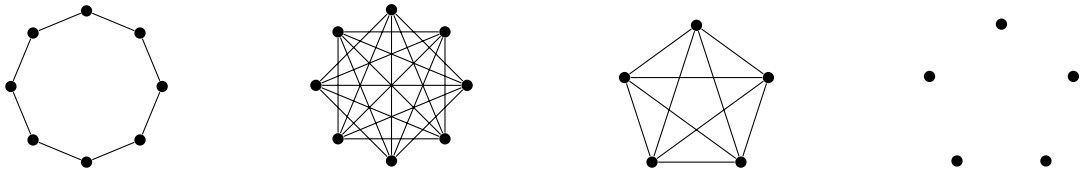
Once $r \geq 3$, any hope of classifying the r-regular graphs on n vertices goes out the window. Even when $r = 3$, there are many examples. We've seen the Wagner graph and the cube graph; by chance, the Harary graph $H_{8,3}$ is isomorphic to the Wagner graph. Both of these examples have a lot of symmetry, and you would be forgiven for thinking that regular graphs *should* be highly symmetric. However, there are many 3-regular graphs, most of which do not look like anything in particular.

The Online Encyclopedia of Integer Sequences (OEIS) lists integer sequences with notable mathematical properties. Entry A2851 of the OEIS (`https://oeis.org/A2851`) gives the number of connected 3-regular graphs on $2n$ vertices, up to isomorphism. I cite an online encyclopedia rather than give a formula because there is no clean formula for the entries of this sequence. However, you can observe that they grow rather quickly from the first few entries:

$$1, 0, 1, 2, 5, 19, 85, 509, 4060, 41301, \ldots$$

You can probably guess that the story for 4-regular graphs, 5-regular graphs, and so on is very similar. We only regain any semblance of order once r gets very close to n. The reason for this is the operation on graphs known as the complement:

Definition 5.7. *If G is a graph, then the **complement of G**, denoted $\overline{G}$, is the graph with the same vertex set as G, and exactly the edges between those vertices which G does not have.*



(a) C_8 (b) $\overline{C_8}$ (c) K_5 (d) $\overline{K_5}$

Figure 5.5: Some graphs and their complements

Figure 5.5 shows some examples of graphs and their complements, such as the cycle graph C_8 (Figure 5.5a) and its complement $\overline{C_8}$ (Figure 5.5b).

If G is an n-vertex graph and a vertex x has degree $\deg_G(x) = d$, then in the complement $\overline{G}$, vertex x has degree $\deg_{\overline{G}}(x) = n - 1 - d$. (There are $n - 1$ other vertices in $V(G) = V(\overline{G})$, and all of them contribute to x's degree either in G or in $\overline{G}$.) In particular, if G is an r-regular n-vertex graph, then $\overline{G}$ is an $(n - 1 - r)$-regular n-vertex graph.

This lets us classify the $(n - 3)$-regular n-vertex graphs as the complements of the 2-regular n-vertex graphs. The $(n - 2)$-regular n-vertex graphs are all complements of the 1-regular n-vertex graphs, which means that up to isomorphism, only one such graph exists for each value of n. Finally, the only possible $(n - 1)$-regular n-vertex graph is the complete graph K_n.

Question:	What is the complement of K_n?
Answer:	It is the empty graph, which was our 0-regular example. (In Figure 5.5c and Figure 5.5d, you can see this for $n = 5$.) For this reason, the empty graph does not have its own notation: it's usually just written $\overline{K_n}$.

Question:	If G has n vertices and m edges, how many vertices and edges does $\overline{G}$ have?
Answer:	$\overline{G}$ still has n vertices, and it has $\binom{n}{2} - m$ edges: all $\binom{n}{2}$ possible edges between its vertices, except for the m that were present in G.

5.4 The Petersen graph and the Kneser graphs

There is a well-known 3-regular graph that, so far, we have only seen in passing in an exercise at the end of a chapter: the Petersen graph.

The Petersen graph is notable in graph theory for many reasons. In this textbook, we will see it several times in its role of providing a small counterexample to several plausible-sounding conjectures in graph theory. There are also several classification theorems (which we will not see) in which it plays a central role.

Two diagrams of the Petersen graph are shown in Figure 5.6, showing some of the symmetry it has: one with 5-fold and one with 3-fold rotational symmetry. In fact, the Petersen graph has much more symmetry than that, which can be seen from its combinatorial definition:

Figure 5.6: Two diagrams of the Petersen graph

Definition 5.8. *The **Petersen graph** is the graph with vertex set*

$$\{12, 13, 14, 15, 23, 24, 25, 34, 35, 45\}$$

consisting of all unordered pairs of elements of $\{1, 2, 3, 4, 5\}$*, and an edge between two vertices whenever they have no elements of* $\{1, 2, 3, 4, 5\}$ *in common. (For example, vertex* 12 *is adjacent to vertices* 34*,* 35*, and* 45*.)*

This definition has a much greater degree of symmetry than either of the diagrams! For any permutation of the set $\{1, 2, 3, 4, 5\}$, we can relabel the vertices of the Petersen graph according to that permutation; this will not change which vertices have no elements of $\{1, 2, 3, 4, 5\}$ in common, so it is an automorphism of the Petersen graph. All $5! = 120$ automorphisms of the Petersen graph can be described in this way.

We can generalize this definition: the Petersen graph is part of an infinite family of regular, highly symmetric graphs known as the Kneser graphs. For all integers n and k with $0 \leq k \leq n$, the vertices of the **Kneser graph $K(n, k)$** are the k-element subsets of the set $\{1, 2, \ldots, n\}$. As in the definition of the Petersen graph, two vertices of $K(n, k)$ are adjacent if they have no elements in common. (The definition is only interesting when $n \geq 2k$; otherwise, any two k-element subsets of $\{1, 2, \ldots, n\}$ are guaranteed to overlap.)

Question:	How many vertices does $K(n, k)$ have, as a function of n and k?
Answer:	There are $\binom{n}{k}$ vertices: the number of ways to choose k unordered elements of $\{1, 2, \ldots, n\}$ without repetition.

Question:	What is the degree of an arbitrary vertex of $K(n, k)$?
Answer:	A vertex X has $\binom{n-k}{k}$ neighbors: the k-element subsets of the set $\{1, 2, \ldots, n\} - X$.

This explains why the Kneser graphs are always regular. For the same reason as the Petersen graph, they have many automorphisms. Though the Petersen graph is by far the most widely encountered of the Kneser graphs, they all have applications to the combinatorics of set families, due to their definition via subsets of $\{1, 2, \ldots, n\}$.

5.5 Practice problems

1. Let's take a closer look at the Harary graphs and their ingredient graphs.

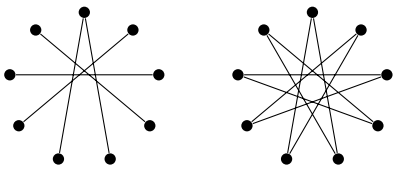
   a) Draw a diagram of $H_{6,0}$ through $H_{6,5}$.

   b) For which n and k is the ingredient graph $I_{n,k}$ isomorphic to C_n? (For an extra challenge, find an explicit automorphism $\varphi\colon V(C_n) \to V(I_{n,k})$.)

   c) For which n and r is the Harary graph $H_{n,r}$ connected? (It might be more straightforward to answer the opposite question: when is $H_{n,r}$ *not* connected?)

2. If n and r are both odd, then an r-regular graph on n vertices does not exist. However, Frank Harary (the inventor, or possibly discoverer, of Harary graphs) defined a Harary graph $H_{n,r}$ in those cases as well. This is a graph on n vertices where $n-1$ of them have degree r, and one has degree $r+1$.

   We can construct this using a slight modification of the construction Theorem 5.1. A new ingredient graph $I_{n,n/2}$ is required, which is nearly 1-regular, but has one vertex of degree 2. For example, here is the "ingredient" graph $G_{9,4.5}$ (on the left), next to a picture of the old "ingredient" graph $G_{9,4}$ that it replaces (on the right):

   a) In general, we define $I_{n,n/2}$ for odd n to be the graph with vertices $\{1, 2, \ldots, n\}$ where vertex i is adjacent to vertex $i + \frac{n-1}{2}$ for $i = 1, 2, \ldots, \frac{n+1}{2}$.

      Prove that $I_{n,n/2}$ is a subgraph of $I_{n,(n-1)/2}$ whose degree sequence is $2, 1, 1, \ldots, 1$.

   b) Use this as an additional ingredient to prove that when r and n are both odd, a Harary graph $H_{n,r}$ exists: a graph on n vertices where $n-1$ of them have degree r, and one has degree $r+1$.

3. Here is a fragment of an alternate proof of the $r = 3$ case of Theorem 5.1.

   ... assume that a 3-regular graph H on $n-4$ vertices exists. Then, we can create a 3-regular graph G on n vertices, just by adding a new connected component to H: four new vertices adjacent to each other and to no other vertices ...

   What kind of a proof is this? What else do we need to do to finish the proof of Theorem 5.1 using this idea?

4. For each diagram in Figure 5.6, show how to label the vertices with elements of the set $\{12, 13, \ldots, 45\}$ (the vertex set of the Petersen graph) so that two vertices are adjacent in the diagram if and only if their labels have no digit in common.

5. a) Prove that the Kneser graph $K(n, k)$ is connected when $n > 2k$. *(With a few exceptions, it is not connected when $n \leq 2k$; what are the exceptions?)*

   b) Find and prove a similar condition that determines when $K(n, k)$ has diameter 2: when the maximum distance between any two vertices of $K(n, k)$ is 2.

6 Graphic sequences

The purpose of this chapter

Most graph theory textbooks present the graphic sequence algorithm differently, by working with numerical sequences rather than directly with graphs. In my opinion, that makes the problem harder and more boring, and takes away class time from discussing edge swaps and the ideas in the proof of Theorem 6.2.

I think that Problem 6.1 and its solution for 8 people is a good way to motivate the graphic sequence algorithm. The proof of Proposition 6.1 could be skipped in a graph theory course covering this material, but I felt that I had to include it to avoid leaving the problem without a solution.

6.1 An unusual party

Let me tell you a story.[1] One time, I was at a very large party—there were 100 people there! Not everybody knew each other, of course. In fact, each person at the party knew a different number of people. In addition, I—

"—That can't be right!" you interrupt. "You're saying that the graph which represents who knew each other at the party had the degree sequence $99, 98, 97, \ldots, 2, 1, 0$? But that's not a graphic sequence!"

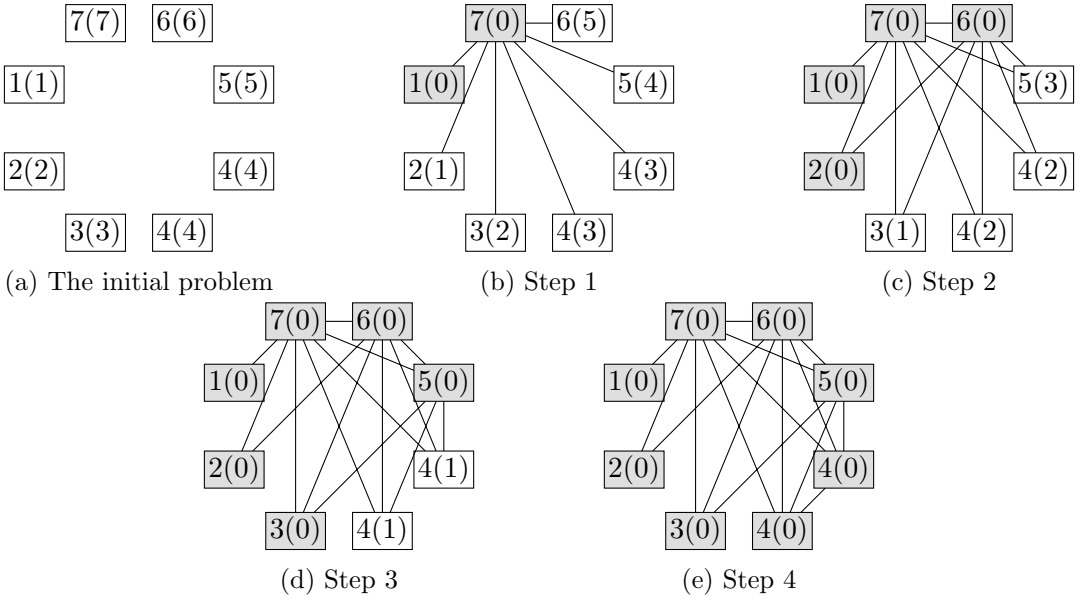
Question:	Why isn't this sequence graphic?
Answer:	The vertex of degree 99 is adjacent to every other vertex, but the vertex of degree 0 is adjacent to none of the other vertices, and these facts contradict each other.

Oh, sorry. My mistake. Let me try again.

Problem 6.1. *One time, I was at a very large party—there were* 100 *people there! Not everybody knew each other, of course. In fact, each person at the party knew a different number of people, with one exception: I personally knew the same number of people as another guest, named Masha. In addition, I can promise you that everyone at this party knew at least one other person.*

Now, given this information, can you tell me how many people I knew at the party? Also, can you tell me whether Masha and I knew each other?

[1] In West's *Introduction to Graph Theory* [6], a version of this story appears as "The Handshake Problem".



(a) The initial problem (b) Step 1 (c) Step 2

(d) Step 3 (e) Step 4

Figure 6.1: Reconstructing the 8-person party

Before we solve this problem, let's consider a smaller case: an 8-person party and its graph of who knows whom. In this graph, the vertex degrees are $7, 6, 5, 4, 3, 2, 1$, but with one of these duplicated. We can't duplicate an odd number, because then the sum of degrees would violate the Handshake Lemma. So we must duplicate the 6, the 4, and the 2. I can tell you the correct option: duplicating the 4 is the only choice that will work.

If the degree sequence is $7, 6, 5, 4, 4, 3, 2, 1$, can we reconstruct the graph? Let me show you a graphic way of thinking about it. In Figure 6.1a, I have drawn the 8 vertices, with labels of the form $a(b)$. These indicate a vertex that needs a neighbors, with b of them left to assign.

For one of these vertices, we can immediately assign the neighbors. The vertex labeled 7(7) needs to be adjacent to every other vertex, so we can draw all of those edges, as shown in Figure 6.1b. Now its label changes to 7(0), and we mark it in grey to show that it's inactive. For all other vertices, the second number decreases by 1, and one more vertex also becomes inactive: the vertex labeled 1(0). The inactive vertices cannot receive any more edges.

We can continue. The vertex labeled 6(5) needs 5 more neighbors, and there are only 5 other active vertices, so it must be adjacent to all of them! We add all of those edges in Figure 6.1c. Now this vertex gets the new label 6(0) and goes inactive. For all other active vertices, the second number decreases by 1, and the vertex now labeled 2(0) also becomes inactive.

Two more steps solve the problem entirely. When we process the vertex labeled 5(3) in Figure 6.1c, we arrive at the diagram in Figure 6.1d. At that point, we only have two active vertices left, and each one wants 1 more neighbor, so we make them adjacent, arriving at the diagram in Figure 6.1e. Now we know all the edges of the graph.

We will return to the ideas in this solution very soon. For now, let me make one more observation: the "leftover" degrees of the active vertices in Figure 6.1b are $5, 4, 3, 3, 2, 1$, which is a 6-vertex version of the same problem. This suggests a solution to Problem 6.1 by induction.

Proposition 6.1. *For all $n \geq 1$, up to isomorphism, there is a unique $2n$-vertex graph which contains a vertex of every degree $1, 2, \ldots, 2n-1$. In that graph, there are two vertices of degree n, and they are adjacent.*

Proof. We induct on n. When $n = 1$, the graph we are looking for is a graph with 2 vertices, at least one of which has degree 1. There most be an edge between the 2 vertices, so we get a graph isomorphic to the path graph P_2. In this graph, there are two vertices of degree 1, and they are adjacent, completing the base case.

Next, for some $n > 1$, assume that the previous case of the proposition holds. That is, there is a unique $2(n-1)$-vertex graph G which contains a vertex of degree $1, 2, \ldots, 2n-3$. In G, there are two vertices of degree $n-1$, and they are adjacent.

Mimicking the way in which the 6-person party sits inside the 8-person party in Figure 6.1b, we extend G to a $2n$-vertex graph H by adding two vertices x and y; we make x adjacent to y and to all vertices of G, and we make y adjacent only x. Here, $\deg_H(x) = 2n-1$ and $\deg_H(y) = 1$; for all vertices $z \in V(G)$, we have $\deg_H(z) = \deg_G(z) + 1$ (because we added the edge xz) and so the vertices of G fill in the degrees $2, 3, \ldots, 2n-2$. The two vertices of degree $n-1$ in G become vertices of degree n in H, and they are adjacent.

This proves that the situation in Proposition 6.1 is *possible*, but not that it is unique. To prove it is unique, let H' be any $2n$-vertex graph that satisfies the conditions. Let x be a vertex of degree $2n-1$ in H', and let y be a vertex of degree 1 in H'. Then x must be adjacent to all $2n-1$ other vertices (including y), which means y is only adjacent to x. Let $G' = H' - x - y$; then in G', every degree k between 1 and $2n-3$ is present, because degree $k+1$ is present in H on a vertex that's neither x nor y. Since G' satisfies the conditions in the proposition, it must be isomorphic to the graph G in the previous paragraph, which forces H' to be isomorphic to the graph H in the previous paragraph.

Question:	How do we know every isomorphism between G and G' extends to an isomorphism between H and H'?
Answer:	To extend an isomorphism $\varphi\colon V(G) \to V(G')$, just define $\varphi(x) = x$ and $\varphi(y) = y$. Since in both H and H', x is adjacent to every vertex and y is adjacent only to x, the extended φ is guaranteed to preserve the edges out of x and y.

This proves uniqueness, which completes the induction step; by induction, the graph exists and is unique up to isomorphism for all n. □

In particular, the answer to Problem 6.1 (the $n = 50$ case of Proposition 6.1) is that Masha and I both knew 50 people at the party, and we did know each other.

6.2 A graphic sequence algorithm

The reconstruction in Figure 6.1 is only possible for very special degree sequences. Our decision at each step was forced, and the reason it was forced is the uniqueness result in Proposition 6.1. Most degree sequences do not have a corresponding uniqueness result, in which case, life is more complicated.

What do we do if we're faced with a diagram like one of the ones in Figure 6.1, but there are no forced deductions to make? It turns out that there's still something we can do. It will feel like making a lucky guess. Later in this chapter, in Theorem 6.2, we will prove that the guess is always justified, at least in one sense. Before we do that, though, let me explain the guess and why it is at least reasonable.

My reasoning is this: in Figure 6.1, we were always living life on the edge. The highest-degree vertex always just barely had enough edges we could place, and that's *why* the result was unique. We would like our graphs to be as little like that, actually: we want to avoid having vertices which need many edges. We also want to avoid creating too many inactive vertices (which don't need any more edges) too quickly.

Putting these together, which vertex should we deal with next, at each step? It should be the vertex with the highest remaining degree. Which neighbors should we give it, if we have the choice? It should be the other vertices with the highest remaining degree—this ensures those vertices need fewer edges later, and also avoids creating inactive vertices too early.

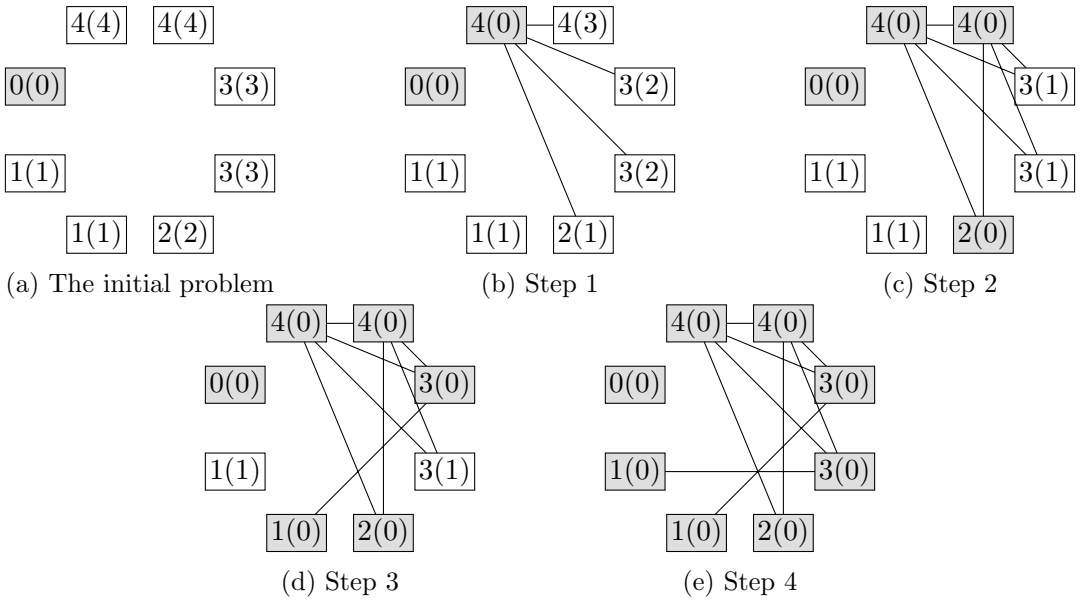
Now, let me present an algorithm which makes all these ideas formal and precise.

Given a sequence of nonnegative integers $d_1, d_2, \ldots, d_n$, our goal is to test if the sequence is graphic, and if it is, to construct a graph whose degree sequence is $a_1, a_2, \ldots, a_n$. We begin our construction with a graph G that has vertex set $V(G) = \{1, 2, \ldots, n\}$ and edge set $E(G) = \varnothing$: no edges to start with. We will add edges to G as we go.

For all $i = 1, \ldots, n$, we intend for vertex i to have degree d_i at the end. We also give vertex i a **demand** representing how many more edges it needs to meet this goal. Initially, we set its demand equal to d_i, but we will update the demand of i over the course of the algorithm. We will call a vertex **active** if its demand is positive, and **inactive** if its demand is 0. In diagrams, we will show vertex i with a label $a(b)$ where $a = d_i$ and b is the demand of i; we will also shade inactive vertices in gray.

We repeat the algorithm below for as long as any active vertices remain. In each iteration, we perform the following steps:

1. Pick a vertex x with the highest possible demand, breaking ties arbitrarily; let k be the demand of x.
2. If there are fewer than k active vertices other than x, stop and declare that the sequence is *not* graphic: we have failed to construct a graph with this sequence.
3. Otherwise, let S be a set of k active vertices, other than x, whose demand is as high as possible (again, breaking ties arbitrarily).
4. Add the k edges xy for all $y \in S$ to the graph G. Set the demand of x to 0 (making it inactive) and decrease the demand of each vertex $y \in S$ by 1.



(a) The initial problem (b) Step 1 (c) Step 2

(d) Step 3 (e) Step 4

Figure 6.2: Finding a graph with degree sequence $4, 4, 3, 3, 2, 1, 1, 0$

Once no more active vertices remain, we stop.

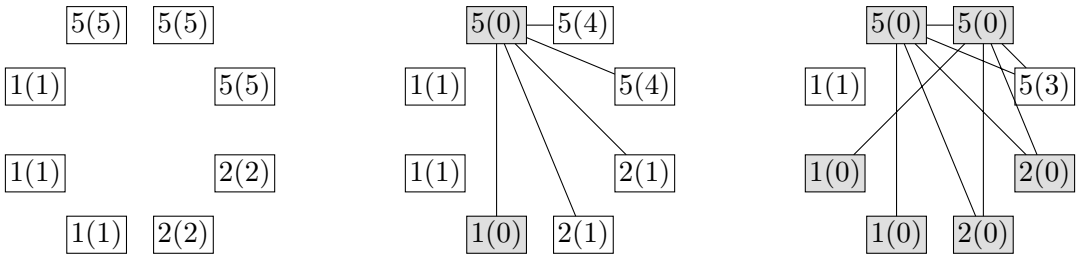
Question:	What is the maximum number of iterations we will need?
Answer:	Since at least one vertex becomes inactive at each step, we can be be certain to stop after n iterations (in an n-vertex graph), though we may stop sooner.

Question:	How do we know that when we construct a graph at the end, it has the right degree sequence?
Answer:	At each step, the degree of each vertex increases by the same amount that its demand decreases. Since the demand of vertex i decreases from d_i to 0 by the end, its degree must have increased from 0 to d_i: the value we wanted.

Let me show you two examples of this algorithm in action. After those examples, it will be more clear what we do and do not need to prove in order to be certain that the algorithm works.

6.3 Two examples

First, let's use the algorithm in the previous section to find a graph with the degree sequence $4, 4, 3, 3, 2, 1, 1, 0$. (If you have a good memory, you can be certain that at least one such graph exists, because we saw such a graph in Chapter 4.)



(a) The initial problem (b) Step 1 (c) Step 2

Figure 6.3: Trying to find a graph with degree sequence $5, 5, 5, 2, 2, 1, 1, 1$

We begin with an empty graph where the demand of each vertex is equal to the degree we want it to have, as shown in Figure 6.2a. (The vertex with label 0(0) starts out inactive; it does not affect the problem in any way.)

We choose one of the vertices with demand 4, and add edges to the 4 other vertices with the highest demands (their demands are 4, 3, 3, and 2). The result is Figure 6.2b, with one fewer active vertex.

Next, we can only choose the vertex with demand 3. The vertices that will become its remaining neighbors must include the two vertices with demand 2, as well as one of the vertices with demand 1; we arbitrarily choose it to be the vertex labeled 2(1). Adding the edges between these vertices and decreasing demand accordingly, we obtain Figure 6.2c.

Here, all four remaining vertices are tied for highest demand: their demands are all 1. Our next step will be to draw an edge between two of its vertices, and any of the ways to do that are fine. Figure 6.2d shows one possible way we could proceed. Finally, in this graph, there are only two active vertices, each with a demand of 1, and the only thing we can do is add an edge between them. This produces the graph in Figure 6.2e. We can check that our algorithm produced the correct result by checking every vertex to see that if it has label $d(0)$, then it has degree d.

In this example, the sequence we started with was graphic; let's consider an example where the sequence we start with is not graphic. I will use the sequence $5, 5, 5, 2, 2, 1, 1, 1$.

The first step, going from Figure 6.3a to Figure 6.3b, is completely normal. We pick one of the vertices labeled 5(5) and join it to: the other two vetices labeled 5(5), the two vertices labeled 2(2), and one of the vertices labeled 1(1). Two vertices become inactive at this step, though if we had made different choices, that number could have been even higher.

Next, we pick one of the vertices with demand 4, and draw 4 edges. One of them has to go to the other vertex with demand 4, and the rest go to vertices with demand 1. There are several ways to draw the edges, but all of them have a similar result to what is shown in Figure 6.3c. Almost all of the vertices have become inactive.

We can no longer proceed! The highest demand in Figure 6.3c is 3, but we do not have 3 other active vertices to connect to. So we declare that the sequence $5, 5, 5, 2, 2, 1, 1, 1$ is not graphic, and give up.

Now, where in this process do we need additional proof?

Nothing more is needed with the first example. In principle, once we have a graph with the right degree sequence, we don't care if we got it by a valid algorithm, or an invalid algorithm,

or if it came to us in a dream: we have the answer, and we can check it! But we are in even better shape than that: we've already shown that if the graphic sequence algorithm produces a graph, then the graph it produces is guaranteed to have the right degree sequence.

The second example is iffier. In a sense, what we're doing is similar to taking a crossword, writing in a few random words without looking at the clues, then pointing to one of the crossing entries and saying, "There's no word with the letters Q, X, blank, O. So there must be a mistake in the crossword." Sure, there could be a mistake—or maybe we just made the wrong guesses?

For the specific degree sequence $5, 5, 5, 2, 2, 1, 1, 1$, we could backtrack and see that if we had made different choices, we'd be equally stuck. However, we don't want a backtracking algorithm; for a longer degree sequence, that would take forever. What we want is an argument that justifies the specific choices our algorithm makes.

But what is it that we want? We don't need to prove that a graph with such a degree sequence *must* have the adjacencies we guess that it does. There could be multiple solutions, after all! What we need to prove is that whenever solutions exist, our guess doesn't eliminate them all: that at least one of the solutions has the structure that the graphic sequence algorithm expects.

6.4 The Havel–Hakimi theorem

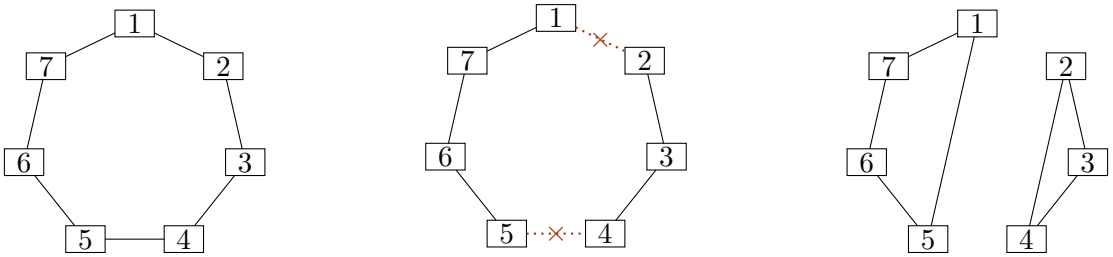
The result we need to prove is known as the Havel–Hakimi theorem: named after Václav Havel, who proved it in 1955 [4], and S. L. Hakimi, who rediscovered the corresponding algorithm in 1962 [3]. Before we prove it, we will restate it in a different way, closer to how it was presented by Havel.

Theorem 6.2. *Let G be a graph with vertex set $x_1, x_2, \ldots, x_n$ such that $\deg(x_1) \geq \deg(v_2) \geq \cdots \geq \deg(x_n)$. Let $k = \deg(x_1)$ and let $S = \{x_2, x_3, \ldots, x_{k+1}\}$.*

Then there is a graph H with $V(H) = V(G)$ and with $\deg_H(x_i) = \deg_G(x_i)$ for all i such that the neighbors of x_1 in H are precisely the vertices of S.

Question:	How does Theorem 6.2 justify the correctness of our algorithm?
Answer:	In the first step, x_1 (up to tiebreaks) is exactly the vertex we will process, and the edges from x_1 to S are exactly the edges we add. Theorem 6.2 tells us that if there are any graphs G with the degree sequence we want, then in particular there is a graph H in which the choices we made are correct.

Question:	That only applies to the first step—what about future steps?
Answer:	In future steps, we can ignore the edges we've already placed and apply Theorem 6.2 by using the demands as the vertex degrees. Again, the theorem tells us that if there is any solution, then in particular there is a solution consistent with the choices the algorithm makes.



(a) The initial graph (b) Replace edges 12 and 54... (c) ... by edges 15 and 24.

Figure 6.4: An example of an edge swap

The advantage of stating Theorem 6.2 in this new form is that it describes a transformation: we want to take graph G and transform it into another graph H, without changing the vertices or their degrees, so that H has the adjacencies we want. We'll do the transformation step by step, so we need a way to make a minor change to a graph without changing its degree sequence. This minor change is an operation called an **edge swap**.

Definition 6.1. *Let G be a graph in which vw and xy are edges but vx and wy are not. Then an **edge swap** in G is the operation of deleting edges vw and xy, then adding new edges vx and wy.*

Figure 6.4 shows an example of an edge swap. The affected vertices v, w, x, and y each lose an edge and gain another edge; thus, the overall degree sequence does not change.

We begin by proving a lemma that essentially says, "we can always use edge swaps to make progress toward what we want." The lemma and its proof use the notation from the statement of Theorem 6.2.

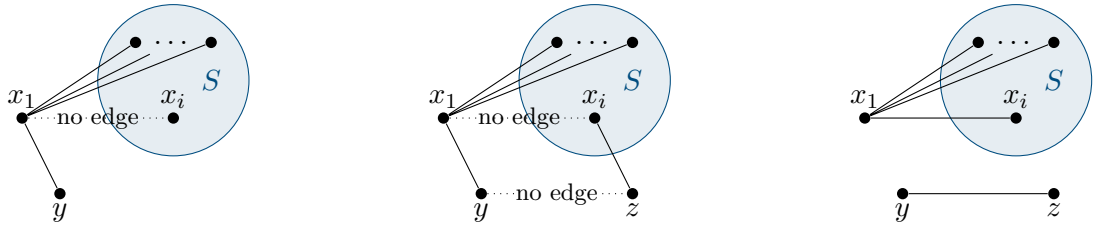
Lemma 6.3. *If not all vertices of S are adjacent to x_1, then we can perform an edge swap in G to increase the number of vertices in S adjacent to x_1.*

Proof. Let $x_i \in S$ be some vertex not adjacent to x_1. Since $|S| = k$, and $\deg(x_1) = k$, in order for x_1 to have k neighbors, it must also be adjacent to some vertex $y \notin S$, as shown in Figure 6.5a.[2]

We would like to perform an edge swap that replaces edge x_1y (which we don't need) by edge x_1x_i (which we do need). To do this, we need to find a fourth vertex: a vertex z such that x_i is adjacent to z, but y is not. (In other words, we are looking for the picture shown in Figure 6.5b.)

Here is where the order of degrees in G comes in. Since the degrees of $x_1, x_2, \ldots, x_n$ are in descending order, we know in particular that the vertices in S have greater or equal degrees than the vertices not in S (except for x_1). In particular, $\deg(x_i) \geq \deg(y)$. Moreover, y already has one neighbor that x_i does not have: the vertex x_1.

[2]Unlike many diagrams in this book, which show you all the vertices and edges of a graph, the diagrams in Figure 6.5 only show you a few vertices and edges: the ones relevant to the proof of Lemma 6.3. This is common when presenting a proof, because if we want the argument to be fully general, we can't just show how it works on one specific graph.



(a) Vertices x_i and y (b) Goal: find z (c) The key edge swap

Figure 6.5: The three steps in the proof of Lemma 6.3.

Therefore it is impossible for y to be adjacent to every neighbor of x_i, and also to x_1: then, we'd have $\deg(x_i) < \deg(y)$. So we can find a neighbor z of x_i such that y is not adjacent to z.

Now perform the edge swap which replaces x_1y and x_iz by x_1x_i and yz (as in Figure 6.5c). After this switch, x_i is adjacent to one more vertex of S. □

Question: Figure 6.5b shows z outside the set S. Is it a problem if $z \in S$?

Answer: No: the argument is identical whether or not $z \in S$. (But this is an example of how diagrams might be misleading, so it's an important question to ask yourself!)

It is important that $x_i \in S$ but $y \notin S$ because we want to compare $\deg(x_i) \geq \deg(y)$, but $\deg(z)$ is never compared to anything, so from this point of view it doesn't matter if $z \in S$.

It is also important that $x_i \in S$ but $y \notin S$ because we want the gain of edge x_1x_i and the loss of edge x_1y to represent progress. But edge x_1z is neither added nor removed by the edge swap, so from this point of view we also don't care if $z \in S$.

I put Lemma 6.3 first because from here, the proof of Theorem 6.2 is essentially just an application of the lemma, together with the extremal principle. It is a good exercise to try to do this yourself before reading ahead.

Proof of Theorem 6.2. Of all graphs with the same vertex set and vertex degrees as G, let H be chosen to maximize the number of neighbors of x_1 in S.

Then actually, all vertices of S must be adjacent to x_1. If not, we could use Lemma 6.3 to increase the number of neighbors of x_1 in S. But H was chosen to have the largest possible number of such neighbors, so this can't happen.

Therefore H is exactly the graph needed to prove the theorem. □

I should say that the graphic sequence algorithm we've described is also known as the Havel–Hakimi algorithm, but it is commonly described in a different way that I consider less intuitive. Instead of putting the demands on the vertices and drawing edges, we can think of the algorithm as modifying a sequence $d_1 \geq d_2 \geq \cdots \geq d_n$ of the intended degrees. If $k = d_1$, the highest

degree, then the algorithm places edges from the first vertex to the k vertices after it. This corresponds to an operation on sequences, turning $d_1, d_2, \ldots, d_n$ into

$$\underbrace{d_2 - 1, d_3 - 1, \ldots, d_{k+1} - 1}_{k \text{ terms}}, d_{k+2}, \ldots, d_n.$$

Repeating this operation eventually produces a sequence of all 0's (in which case the original sequence is graphic) or a sequence which is clearly not graphic (for example, because it contains negative numbers).

6.5 More on degree sequences

We only used edge swaps in the proof of Theorem 6.2, but they have other applications. To begin with, we can show the following theorem:

Theorem 6.4. *If two graphs G and H have the same vertex set V, and $\deg_G(x) = \deg_H(x)$ for all $x \in V$, then we can turn G into H by doing edge swaps.*

Proof. We prove this by induction on the size of V (the common vertex set of G and H).

When $|V| = 1$, there is nothing to show: there is only one possible graph on 1 vertex.

Assume this is possible for all pairs of $(n-1)$-vertex graphs, and let G and H both have n vertices. Let x be a vertex with the highest degree (in both G and H) and let S be the $\deg(x)$ vertices with the highest degrees after x. (This is the same S as in our proof of the Havel–Hakimi theorem and the edge swap lemma.)

By the argument in our proof of Theorem 6.2, we can perform edge swaps on G to get a graph G' in which x's neighbors are the vertices in S. We can do the same to H, getting a graph H' (and so there is also a sequence of edge swaps that turn H' into H, by reversing those edge swaps).

By induction, there is a sequence of edge swaps that turns $G' - x$ into $H' - x$ (both $(n-1)$-vertex graphs on the same vertex set). We can perform these edge swaps on G' instead, and they will work equally well, turning G' into H'. Finally, we know that we can turn H' into H.

This shows that we can turn G into H, and so by induction, this works when G and H have any number of vertices. □

Why is this theorem useful? Well, we know how to answer two possible questions about potential degree sequences...

1. Is this sequence graphic? (Is it the degree sequence of a graph?)
2. How is this sequence graphic? (Find a graph with this degree sequence.)

...but there is a third, harder question we can ask:

3. What is a typical graph with this degree sequence?

That's deliberately vague. However, Theorem 6.4 is the beginning of a possible answer to the last question.

What we can do is start with *some* graph which has that degree sequence, then perform many edge swaps at random, getting a random graph with this degree sequence.[3] We can approximately answer questions like "are any graphs with this degree sequence bipartite" or "what is the average diameter of a graph with this degree sequence" by sampling many random graphs and analyzing the set of results.

We can even approximately answer questions like "how many different graphs have this degree sequence?" To do this, just sample lots of graphs, then count how many repeats you get. Compare this to how many repeats you'd expect, if there were N possible graphs total. This should let you estimate the most likely value of N.

There is one difficulty to the procedure I'm outlining to you so cheerfully and optimistically, and that is the requirement of performing *many* edge swaps. Obviously, just one or two random edge swaps are not enough: the result is guaranteed to be very similar to our non-random starting graph. As the number of edge swaps grows, the dependence on the starting graph slowly decreases. However, it's still not known in general how many edge swaps are necessary to make that dependence negligible. Without that knowledge, the random sampling algorithm cannot be considered reliable.

6.6 Practice problems

1. Using the degree sequence algorithm, or in some other way, determine which of the sequences below are graphic:

   a) $7, 6, 5, 4, 3, 2, 1$.

   b) $4, 4, 3, 3, 3, 2, 2$.

   c) $6, 6, 5, 3, 2, 2, 1, 1$.

   d) $n, n, n, \underbrace{3, 3, \ldots, 3}_{n \text{ times}}$ (for any n).

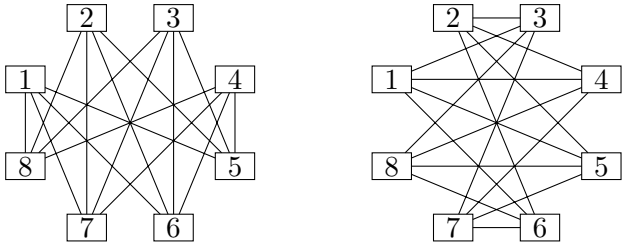
   e) $n-1, n-1, n-1, n-1, \underbrace{3, 3, \ldots, 3}_{n-4 \text{ times}}$ (for any n).

   For the sequences that are graphic, construct graphs with those degree sequences.

2. If we can turn any graph G into any graph H with the same degree sequence using edge swaps, then we can turn $K_{4,4}$ into $K_{4,4}$ using edge swaps.

   Specifically, let G be the graph on the left in the diagram below, and let H be the graph on the right. Corresponding vertices have the same degrees, and both of these graphs are isomorphic to $K_{4,4}$, but they are not the same graph: they have different edges. G has all edges between $\{1, 2, 3, 4\}$ and $\{5, 6, 7, 8\}$, while H has all edges between $\{1, 2, 5, 6\}$ and $\{3, 4, 7, 8\}$.

[3] Fine print: this does not sample a graph uniformly at random. Essentially, it samples a graph with probability proportional to how many edge swaps are possible to do in it. But that's a known bias we can account for.



Find a sequence of edge swaps to turn G into H. *(Hint: four edge swaps are enough.)*

3. Let me tell you another story.

   The other day, I was at another very large party. Once again, each person at the party knew a different number of people, with one exception: I personally knew the same number of people as another guest, named Pasha. In addition, I can promise you that everyone at this party knew at least one other person.

   The surprising thing is this: I did not know Pasha! How could this happen, and why does this not contradict Proposition 6.1?

4. In Problem 6.1, I included the condition that everyone at the party knew at least one other person. If this condition is dropped, then there is another possible graph: how can we find it? (What should its degree sequence be?)

5. There are 7 possible graphs with the vertex set $\{1, 2, 3, 4, 5\}$ such that $\deg(1) = \deg(2) = 3$ and $\deg(3) = \deg(4) = \deg(5) = 2$. Find all of them, and demonstrate directly that we can turn any one of them into any other via edge swaps.

   (Hint: of the 7 graphs, there is a "central" example that is one edge swap away from all 6 others.)

6. There is another proof of Theorem 6.4 that is closer to our argument for Theorem 6.2.

   Specifically, if G and H have the same vertex set, and every vertex has the same degree in G and in H, then we can measure "how far away" G and H are from each other by asking: how many edges does G have that are not also edges of H? Now we can do the following:

   a) Prove that if G and H are not literally the same graph, then there is an edge swap we can perform on G to reduce "how far away" it is from H.

   b) Prove Theorem 6.4 using part (a).

7. Let G be an n-vertex graph whose vertices $1, 2, \ldots, n$ have degrees $d_1, d_2, \ldots, d_n$ respectively, sorted in descending order:

$$d_1 \geq d_2 \geq \cdots \geq d_n.$$

   Then d_1 (the maximum degree of G) can be at most $n - 1$; in fact, it can be at most the number of nonzero degrees among $d_2, \ldots, d_n$.

a) Explain why this is equivalent to the following inequality:

$$d_1 \leq \sum_{i=2}^{n} \min\{d_i, 1\}.$$

Here, $\min\{a, b\}$ is simply the smaller of the two numbers a and b.

b) Prove the following inequality, which is a 2-vertex generalization of the previous one:

$$d_1 + d_2 \leq 2 + \sum_{i=3}^{n} \min\{d_i, 2\}.$$

(Hint: consider the number of edges between vertices 1 *and* 2 *and the rest of the graph. Why is the "*2+*" there?)*

c) In general, we have an inequality of the form

$$\sum_{i=1}^{k} d_i \leq f(k) + \sum_{j=k+1}^{n} \min\{d_j, k\}$$

for each $k \in \{1, \ldots, n\}$. What is the best possible function $f(k)$ that makes this inequality true? (I ask for "best possible" because some incredibly big function like $f(k) = 2^{2^{2^k}}$ will also work, but will not be as useful as the smallest possible number you could put there.)

More is true: a theorem of Erdős and Gallai [2] states that if a sequence $d_1, d_2, \ldots, d_n$ satisfies the inequality in part (c) for every k, and the sum $d_1 + d_2 + \cdots + d_n$ is even, then the sequence is graphic. This is an alternative way to check whether a sequence is graphic.

7 Multigraphs and digraphs

The purpose of this chapter

The first half of this chapter introduces multigraphs: a more general version of graphs, allowing loops and duplicate (parallel) edges. By putting this content here, I've made my choice between two unsatisfying options: should multigraphs be an exception, or should they be the default?

I've decided they should be the exception, not introduced until Chapter 7, and not considered to be graphs except in a few asides in later chapters. The advantage of this option is that it simplifies some definitions, especially early on when you have enough new complexity to worry about; also, in many applications, it is all you need from graph theory.

The option I didn't take would have been to make Definition 7.1 the definition of a graph, and to refer to graphs without loops and parallel edges as **simple graphs**. (I will still use this term when I need to emphasize that I'm not considering a multigraph.) The advantage of this option is that it lets you state and prove every result in fuller generality.

In practice, the differences between simple graphs and multigraphs almost always fall somewhere on the following spectrum:

1. They apply to multigraphs only: even when you start with a simple graph, you will be forced to handle multigraphs to proceed.
2. They apply to both simple graphs and multigraphs, and there are applications where the multigraph version is useful.
3. They apply to both simple graphs and multigraphs, but not in a way that's useful: maybe the loops and parallel edges can always be ignored, or maybe their existence trivializes the problem.
4. They apply only to simple graphs.

In future chapters, I will clearly call out situations of the first two types, and try to mention the other situations in passing if it seems unclear.

I will handle directed graphs, introduced in the second half of this chapter, similarly, and refer to graphs as **undirected graphs** when it seems important to emphasize that they are not directed.

Notably, the problem of finding Euler tours discussed in Chapter 8 is important to solve for both multigraphs and for directed graphs, and I think it's the first such problem in the textbook; this is one of the reasons I've placed this chapter right before it.

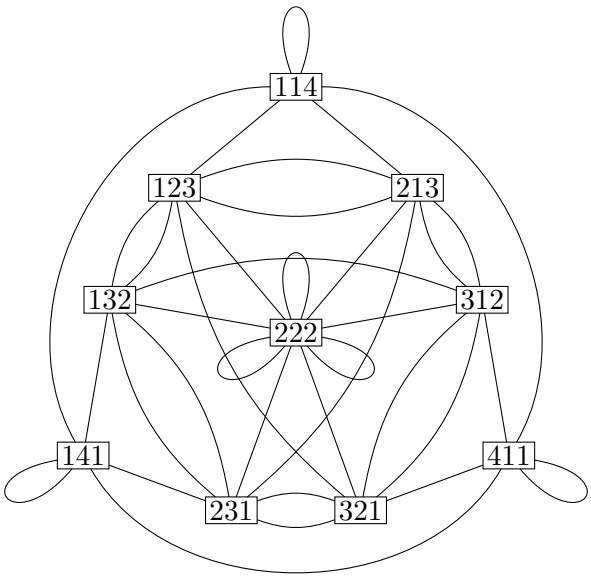


Figure 7.1: A multigraph representation of the 6-pebble game

7.1 Multigraphs

Consider a game[1] with the following rules. There are six pebbles, split between three piles; each pile must contain at least one pebble. There are two types of valid moves: you may swap two piles, or you may move a pebble from one pile to another, as long as the starting pile has at least two pebbles in it.

Figure 7.1 shows a graph of the possible states of the 6-pebble game (with vertex xyz representing a state with x, y, and z pebbles in the three piles); edges in the graph represent valid moves. Well... it doesn't quite show a graph of the game: it has some edges that a graph cannot have. These edges represent moves that do not change the state, as well as situations where two different moves accomplish the same result.

Question:	Why would we draw a loop from vertex 114 back to itself?
Answer:	Swapping the first two piles in state 114 just takes us back to state 114, since the two piles have the same number of pebbles.

Question:	Why are there two edges between vertices 123 and 213?
Answer:	The game has two ways to turn 123 into 213 (or the reverse): we can move a pebble from the 2-pebble pile to the 1-pebble pile, or we can swap the 1-pebble pile with the 2-pebble pile.

[1] Well, it's a "game" because it has rules, but there is no victory condition. Sorry.

Question:	Okay, so what's going on at vertex 222?
Answer:	The six "reasonable" edges out of vertex 222, going to vertices 123, 132, 213, 231, 312, and 321, all represent moving a pebble from one pile to another. There are also *three* moves that accomplish nothing: we can pick any two piles and swap them.

We cannot have these edges in a graph, so we define a **multigraph** to be a more general kind of structure that allows these edges to exist. The formal definition needs to be a bit more complicated, though.

Definition 7.1. *A **multigraph** G is a triple consisting of an arbitrary set of **vertices** $V(G)$, an arbitrary set of **edges** $E(G)$, and an **incidence relation** between them: we say that vertex x is **incident on** edge e, or e is incident on x, if they are related by the incidence relation.*

*For each edge e, either one or two vertices can be incident on e, and they are called its **endpoints**. As for graphs, two vertices are called **adjacent** if they are incident on the same edge.*

The edges of simple graphs were defined as pairs of vertices. This is no longer possible, because two edges can have the same pair of endpoints; to distinguish them, there must be something more to their identity. In diagrams (such as in Figure 7.1) we often don't mark the difference between such edges, but to deal with them formally, we would want to give them special names.

We have special names for the two situations that can exist in a multigraph, but not in a simple graph:

Definition 7.2. *An edge in a multigraph which has only one endpoint is called a **loop**. Two edges are called **parallel** if they have the same set of endpoints.*

For example, in Figure 7.1, there is a loop incident on vertex 114, two parallel edges joining vertices 123 and 213, and three parallel loops incident on vertex 222.

Many notions that we've introduced for graphs still make sense for multigraphs, with some modifications. There are two relatively large changes that are worth spending some time discussing: one related to walks, paths, and cycles, and another related to vertex degrees.

A walk in a multigraph is no longer just a sequence of vertices: it matters *how* we get from one vertex to another. For example, in the 6-pebble game, we consider a multigraph model if it matters which kind of move we used at each step. (If this distinction does not matter, then we do not need the added complexity of multigraphs.) Thus:

Definition 7.3. *An **$x - y$ walk of length k** in a multigraph G is a sequence*

$$x_0, e_1, x_1, e_2, x_2, \ldots, x_{k-1}, e_k, x_k$$

that alternates between vertices and edges of G, satisfying the following: $x = x_0$, $y = x_k$, and for each $i = 1, \ldots, k$, the endpoints of edge e_i are x_{i-1} and x_i.

For example, suppose that we start in state 114 of the pebble game, and we make three moves: we swap the first two piles, then move one pebble from the last pile to the first, then swap the first two piles again. To represent this as a walk, we'll in the 6-pebble multigraph, we will first need to invent notation for the different types of edges. Let's say that $s_{ij}\{x, y\}$ denotes a move between states x and y by swapping piles i and j, and $m_{ij}\{x, y\}$, denotes a move between states x and y by moving a pebble between piles i and j. (Keep in mind that edges don't have a distinguished start and end, so $s_{ij}\{x, y\}$ is the same edge as $s_{ij}\{y, x\}$.) Then our walk would be represented by the sequence

$$114,\ s_{12}\{114, 114\},\ 114,\ m_{31}\{114, 213\},\ 213,\ s_{12}\{213, 123\},\ 123.$$

This is very cumbersome, so usually we don't pull out the full notation unless we need it. Even if we're not looking this closely, though, the definition still matters. For example, when listing the different walks between two vertices, it is important to know when two walks are considered to be different: they are different if the two sequences are different.

Paths, closed walks, and cycles do not change very much, except that they are built on the new foundation of Definition 7.3. A path is still defined to be a walk in which all vertices are distinct; a closed walk is still a walk in which the first vertex is equal to the last. For cycles, you might remember that the definition in Chapter 3 was rather complicated; here, we can give a simpler definition.

Definition 7.4. *A walk*

$$x_0, e_1, x_1, e_2, x_2, \ldots, x_{k-1}, e_k, x_k$$

*in a multigraph G is a **cycle** if it is a closed walk ($x_0 = x_k$) where vertices $x_1, x_2, \ldots, x_k$ are all distinct, edges $e_1, e_2, \ldots, e_k$ are all distinct, and $k \geq 1$.*

With this definition, it is now possible to encounter cycles of length 1 and 2. A cycle of length 1 occurs whenever the multigraph contains a loop: it is a cycle x, e, x where e is a loop incident on x. A cycle of length 2 occurs whenever the muldigraph contains two parallel edges: it is a cycle x, e, y, e', x where x, y are distinct vertices and e, e' are distinct edges incident on both x and y.

7.2 Degrees in multigraphs

We would like to generalize the definition of vertex degrees to apply to multigraphs, while giving the same answer as our old definition for simple graphs when there happen to be no loops or parallel edges. It is not obvious how to do so in a natural way, so let's stop and think for a moment about it.

Definitions are not set in stone—as mathematicians, we can choose how to make them. However, we should try to make interesting and useful definitions. Pay attention to the reasoning here if you want to understand *why* definitions are the way they are!

To begin with, for simple graphs, we can count the degree of a vertex x in two equivalent ways: by counting the edges incident on x, or by counting the vertices adjacent to x. In a multigraph, these two methods can give different answers: for example, in Figure 7.1, vertex 123 is incident on 7 edges, but only has 5 neighbors!

We must immediately reject the second method of counting, because it completely ignores the multigraph structure: why bother having two edges between 123 and 132 if it's not going to affect the degree of either vertex? The first method of counting is not bad, and could have ended up the definition. However, it also has a problem.

By this rule, the degree sequence of the 6-pebble graph would be $9, 7, 7, 7, 7, 7, 7, 5, 5, 5$. By the Handshake Lemma (which we proved in Chapter 4 for simple graphs), the sum of degrees in a graph is twice the number of edges; applying that here, we would conclude that the graph has $\frac{9+6\cdot 7+3\cdot 5}{2} = 33$ edges. But this is incorrect: if you count the edges in Figure 7.1 one by one, you will get 36.

Question:	Where does the discrepancy between 33 and 36 come from?
Answer:	It comes from loops: when we naively apply the Handshake Lemma, the 6 loops in the graph only get counted as 3.

Question:	Looking at the proof of the Handshake Lemma, why does this happen?
Answer:	In the proof, we assumed that adding an edge to a graph increases the degree sum by 2. However, if the degree of a vetex were defined to be the number of incident edges, then a loop would only increase the degree of a single vertex by 1.

The Handshake Lemma is so valuable that we are willing to add a special case to our definition, just for this purpose. To rescue the lemma, we need the degree sum to increase by 2 when a loop is added to a graph. It wouldn't make any sense for a loop incident on a vertex x to increase the degree of any vertex other than x; thus, it must contribute 2 to the degree of x. This tells us what the definition of vertex degree must be:

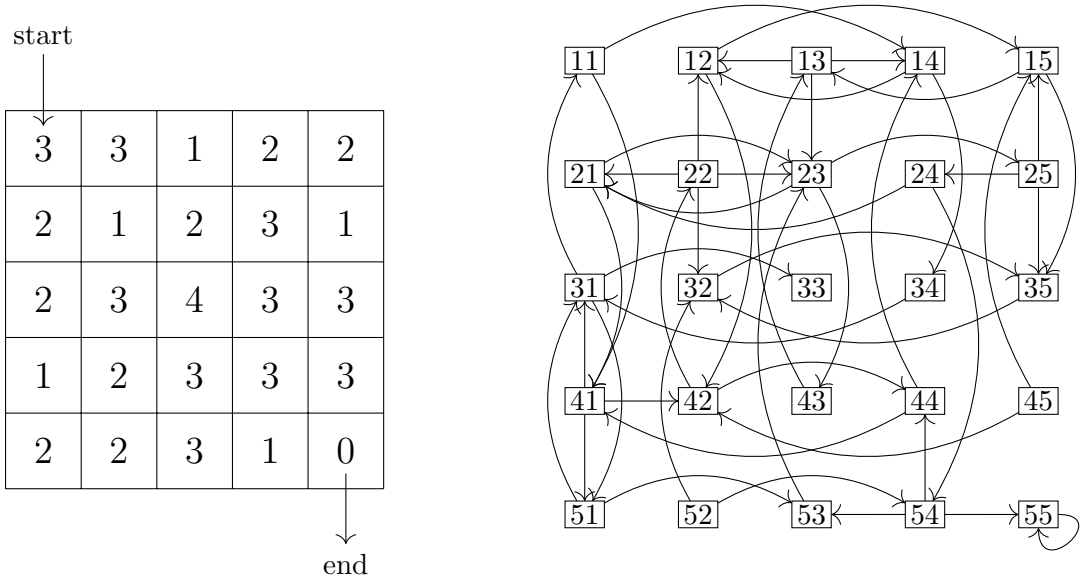
Definition 7.5. *The **degree** of a vertex x in a multigraph G is the number of edges incident on x, counting every loop incident on x twice.*

By this definition, the degree sequence of the 6-pebble graph is $12, 7, 7, 7, 7, 7, 7, 6, 6, 6$.

You might ask: is there a graphic sequence problem for multigraphs? There is, but the answer to it is much less exciting than it is for graphs. Practice problem 3 at the end of this chapter gives the condition, and asks you to discover the proof yourself.

Finally, the definition of a graph isomorphism changes when it is applied to multigraphs. There are two equivalent ways to make that change:

- We could say that an isomorphism between multigraphs G and H is still a bijection $\varphi\colon V(G) \to V(H)$, but with a slightly different condition on φ. Rather than simply preserving adjacency, it should be the case that for any vertices x and y in G (possibly equal) the number of edges between x and y should be the same as the number of edges between $\varphi(x)$ and $\varphi(y)$.



(a) A numerical maze

(b) A digraph representation of the maze

Figure 7.2: Solving a maze using directed graphs

- We could also say that an isomorphism between multigraphs G and H is a pair of bijections: a bijection $\varphi\colon V(G) \to V(H)$, and a second bijection $\varphi'\colon E(G) \to E(H)$. The condition relating these should now be that for a vertex $x \in V(G)$ and an edge $e \in E(G)$, x is incident on e if and only if $\varphi(x)$ is incident on $\varphi'(e)$: the bijections φ and φ' preserve the incidence relation.

These two options are equivalent, and therefore equally good, but the second is more in the spirit of our definition of a multigraph. The intuition you should have is that a discrete structure (like a graph or a multigraph) consists of two parts: some objects, and some relationships between them. In the case of a multigraph, the objects are the vertices and edges, and the relationships between them are given by the incidence relation. An isomorphism between two discrete structures should be a bijection between the objects that preserves the relationships. This idea will guide you to the correct definition of an isomorphism both in graph theory and outside it.

7.3 Directed graphs

Figure 7.2a shows an unusual kind of maze: a numerical maze. Like a normal maze, it has a start (the top left cell) and an end (the bottom right cell). Instead of paths and walls, however, you solve the maze by using the numbers: if you'e in a cell with the number k, then you can jump to another cell that's k steps away either horizontally or vertically.

We might try to represent an ordinary maze by a graph, so that a path in the graph from the start vertex to the end vertex will give us a solution to the maze. In the numerical maze, we encounter an unexpected difficulty: the jumps we make in the maze are often one-way. From the top left corner, which has a 3 in it, we can jump to a cell three steps to the left or to a cell

three steps down. However, neither of these numbers contains a 3, so neither of them will let us return to where we just were! As a result, a graph is unsuitable as a model for this maze: adjacency between the cells is an asymmetric relationship.

The structure we use in such cases is called a **directed graph**, or **digraph** for short. In a diagram, we draw a directed graph by making each edge an arrow, rather than a line, as in Figure 7.2b. An arrow from vertex x to vertex y represents, informally, an adjacency that exists only when going from x to y, not when going from y to x. The formal definition is:

Definition 7.6. *A **directed graph** or **digraph** D is a pair consisting of an arbitrary set of **vertices** $V(D)$ and a set of **arcs** or **directed edges** $E(D)$. Each arc is an ordered pair (x, y) where $x, y \in V(D)$.*

*We say that arc (x, y) **starts** at x and **ends** at y; it is an arc **from x to y**, or **out of x** and **into y**.*

We may combine this definition with the definition of a multigraph in the previous section, obtaining the concept of a **directed multigraph**. However, even without that added flexibility, we allow the arcs (x, y) and (y, x) to coexist in a digraph: after all, these are not the same arc! For example, in Figure 7.2b, arcs $(21, 23)$ and $(23, 21)$ both exist: we can jump in either direction between this pair of cells, becaues they are 2 steps apart, and both contain a 2. The pair (x, x) is also a valid ordered pair, so it can also be an arc in a directed graph: it represents a loop from x to x.

In directed graphs, the notion of vertex degrees becomes even more muddled than it was for multigraphs. Properly speaking, we should not describe the degree of a vertex by a single number. Instead, we count the number of arcs into a vertex and the number of arcs out of a vertex separately.

Definition 7.7. *The **indegree** of a vertex x in a digraph D, denoted $\deg_D^-(x)$, is the number of arcs into x.*

*The **outdegree** of a vertex x in a digraph D, denoted $\deg_D^+(x)$, is the number of arcs out of x.*

As with the ordinary notion of degree, we drop the subscript and write $\deg^-(x)$ and $\deg^+(x)$ if it is clear which digraph containing vertex x we mean.

An even more powerful version of the Handshake Lemma holds for indegrees and outdegrees in directed graphs. Before reading ahead to see how to prove it, see if you can figure out a proof of Lemma 7.1 lemma yourself, by imitating the proof of Lemma 4.1 in Chapter 4.

Lemma 7.1. *In any digraph D, the vertex indegrees add up to the number of arcs, as do the vertex outdegrees:*

$$\sum_{v \in V(D)} \deg_D^-(v) = |E(D)| = \sum_{v \in V(D)} \deg_D^+(v).$$

Proof. We will prove that for any directed graph with m edges, the identity in the lemma holds, and we will do it by induction on m.

The base case is $m = 0$. The lemma holds in this case because the sum of indegrees, the number of arcs, and the sum of outdegrees are all equal to 0: there are no arcs, so no vertex has a nonzero indegree or outdegree.

Assume that the lemma holds for all directed graphs with $m - 1$ arcs. Let D be a directed graph with m arcs, and let (x, y) be any arc of D. We can apply the inductive hypothesis to $D - (x, y)$ (the digraph we get by deleting arc (x, y)), an digraph with $m - 1$ arcs.

What is the relationship between the indegrees and outdegrees in D and in $D - (x, y)$? There are three cases:

- In D, the outdegree of x is 1 higher than in $D - (x, y)$, because arc (x, y) contributes 1 to that outdegree.
- In D, the indegree of y is 1 higher than in $D - (x, y)$, because arc (x, y) contributes 1 to that indegree.
- $\deg^+_{D-(x,y)}(v) = \deg^+_D(v)$ and $\deg^-_{D-(x,y)}(v) = \deg^-_D(v)$ in all other cases.

Adding up all the changes, we see that

$$\sum_{v \in V(D)} \deg^-_D(v) = 1 + \sum_{v \in V(D)} \deg^- D - (x, y)(v)$$

and

$$\sum_{v \in V(D)} \deg^+_D(v) = 1 + \sum_{v \in V(D)} \deg^+ D - (x, y)(v).$$

By our inductive hypothesis, both of the right-hand sides are equal to $1 + (m - 1) = m$, proving the lemma for D: an arbitrary m-arc directed graph. Then, by induction, the lemma holds for directed graphs with any number of arcs. □

Less commonly, in addition to the indegree and outdegree of a vertex, we define the **net degree** of x to be $\deg^\pm_D(x) = \deg^+(x) - \deg^-(x)$ and the **total degree** of x to be $\deg_D(x) = \deg^+(x) + \deg^-(x)$.

Question: How can we interpret the net degree and the total degree of x?

Answer: The net degree measures how many more arcs at x go out, compared to in; it tells us whether x is a "net exporter" or "net importer" of arcs. The total degree is what the degree of x would be if we ignored the orientations of the arcs, and treated D as an undirected graph.

Question: What do the net degrees add up to in a directed graph?

Answer: We can write the sum of net degrees as

$$\sum_{v \in V(D)} \deg^\pm_D(v) = \sum_{v \in V(D)} \deg^+(v) - \sum_{v \in V(D)} \deg^-(v),$$

which simplifies to 0 by Lemma 7.1.

Question:	What do the total degrees add up to?
Answer:	For a similar reason, they add up to $2\|E(D)\|$. This also follows from the Handshake Lemma applied to the undirected graph we'd obtain from D by ignoring the orientations of the arcs.

In directed graphs, the notion of isolated vertices (with indegree and outdegree 0) still makes sense, but it is sometimes useful to consider the indegree and outdegree separately. We call a vertex x in a digraph a **source** if $\deg^-(x) = 0$, and a **sink** of $\deg^+(x) = 0$.

Question:	Does the graph in Figure 7.2b contain any sources?
Answer:	Yes: vertices 45 and 52 are sources. (You might be tempted to call 11 a source because it's a starting point, but it's not: it's possible to return to vertex 11 after leaving it.)

Question:	Does it contain any sinks?
Answer:	Yes: vertex 33 is a sink, because the number in the center of the grid is 4, and there are no cells to hop to that are 4 spaces away in any direction. Sinks in such a graph represent "dead ends" in the maze which cannot be left.

7.4 Directed walks, paths, and cycles

Moving on, the other big change with directed graphs—and, often, the reason we study directed graphs to begin with—is the behavior of walks, paths, and cycles. The definition of a walk in a digraph superficially does not seem very different from the definition we saw in Chapter 3 for undirected graphs:

Definition 7.8. *An* ***$x - y$ walk of length k*** *in a directed graph D is a sequence*

$$x_0, x_1, x_2, \ldots, x_k$$

of vertices of D where $x = x_0$, $y = x_k$, and for each $i = 1, \ldots, k$, (x_{i-1}, x_i) is an arc of D.

The difference is that this definition requires all the arcs (x_{i-1}, x_i) to point in the same direction along the walk, and this small difference is a dramatic change. It is no longer true that an $x - y$ walk is essentially the same as an $x - y$ walk except for which way you go: in fact, it's possible that an $x - y$ walk exists, but a $y - x$ walk does not!

A path is still defined to be a walk in which all vertices are distinct; a closed walk is still a walk in which the first vertex is equal to the last. As we did for multigraphs, we will give a new definition of cycles:

Definition 7.9. *A walk $x_0, x_1, \ldots, x_k$ in a directed graph D is a* ***directed cycle*** *if it is a closed walk ($x_0 = x_k$) where vertices $x_1, x_2, \ldots, x_k$ are all distinct, and $k \geq 1$.*

As with multigraphs, we are happy to have directed cycles of length 1 or 2, because they still represent some nontrivial interesting structure in D.

To finish this introduction to directed graphs, let's put the new ideas about directed graphs together into a theorem: the directed version of Theorem 4.4 from Chapter 4.

Theorem 7.2. *Every directed graph D with no sinks contains a directed cycle.*

Proof. Let $x_0, x_1, x_2, \ldots, x_k$ be a longest directed path in D. Because D has no sinks, we know in particular that $\deg^+(x_k) > 0$, so there must exist some arc (x_k, y) that starts at x_k.

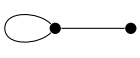
If $y \notin \{x_0, x_1, x_2, \ldots, x_{k-1}\}$, then the sequence $x_0, x_1, x_2, \ldots, x_k, y$ would be a longer directed path, contrary to our assumption. Therefore $y = x_i$ for some $i \leq k-1$, and $x_i, x_{i+1}, \ldots, x_k, x_i$ is the directed cycle we wanted. □

Question:	Does the same result hold for directed graphs with no sources?
Answer:	Yes: we can write a similar proof, but look for a vertex y with an arc into x_0, rather than out of x_k.

Question:	Does the converse hold? That is, if D has a directed cycle, is it true that it has no sources or sinks?
Answer:	No, and the digraph in Figure 7.2b is a counterexample. This digraph has two sources and a sink, and yet it still has many directed cycles: for example, the length-3 cycle $12, 15, 13, 12$.

7.5 Practice problems

1. Consider the 6-pebble multigraph in Figure 7.1.
   a) How many $114 - 411$ walks of length 1 does it have?
   b) What about length 2?
   c) What about length 3?
   d) For practice, write out one of the length-3 walks as a sequence of vertices and edges, using the $s_{ij}\{x, y\}$ and $m_{ij}\{x, y\}$ notation for edges.
2. Consider the multigraph below. How many closed walks of length 10 begin and end at the vertex on the left?



3. Here is why we consider the graphic sequence problem for simple graphs, and not for multigraphs.
   a) Let $d_1 \geq d_2 \geq \cdots \geq d_n$ be a sequence of nonnegative integers. Show that it is the degree sequence of a multigraph if and only if $d_1 + d_2 + \cdots + d_n$ is even.

b) Let $d_1 \geq d_2 \geq \cdots \geq d_n$ be a sequence of nonnegative integers. Show that it is the degree sequence of a multigraph *without loops* if and only if $d_1 + d_2 + \cdots + d_n$ is even and $d_1 \leq d_2 + d_3 + \cdots + d_n$.

4. Let G be a simple graph with n vertices and maximum degree r. We assume that rn is even, so that an r-regular graph on n vertices exists. (But G might not be r-regular; some of its vertices can have degree less than r.)

   a) Prove that there is an r-regular multigraph H, with $V(H) = V(G)$, such that G is a subgraph of H. (In other words: we can add edges to G to make it regular.)

   b) Give an example of a graph G for which the graph H in part (a) cannot possibly be a simple graph.

5. Solve the numerical maze in Figure 7.2a. What kind of graph-theoretic object represents your solution?

   (For an extra challenge, use breadth-first-search, as described in Chapter 3, and find the shortest path through the maze.)

6. A game is played with two piles of stones. On a turn, a player picks a pile which is not empty, and takes one or more stones from it. The game ends once both piles are empty.

   We can represent this game by a digraph whose vertices are the states, with arcs representing the valid moves. Let $D_{n,m}$ be the digraph we get when the game is played with a pile of size n and a pile of size m.

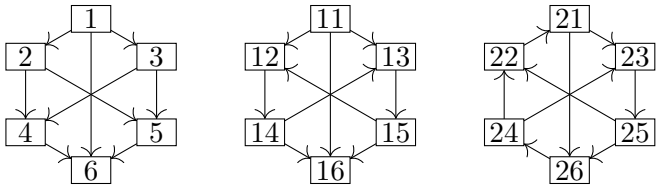
   a) Draw a diagram of $D_{3,2}$. (It should have 12 vertices.)

   b) Which vertices of $D_{n,m}$ are sources? Which are sinks?

   c) Are there any values of n and m for which $D_{n,m}$ contains a directed cycle?

7. If D and D' are directed graphs, then an isomorphism from D to D' s a bijection $\varphi\colon V(D) \to V(D')$ such that $(x, y) \in E(D)$ if and only if $(\varphi(x), \varphi(y)) \in E(D')$. The idea of isomorphism is the same for directed graphs: all properties of directed graphs which don't depend on how the vertices are named or how the graph is drawn should be the same for two isomorphic digraphs.

   Here are three directed graphs which would be isomorphic if we forgot about the directions of the arcs.



   Prove that as directed graphs, none of these are isomorphic.

8 Euler tours

The purpose of this chapter

This chapter concludes the part of this book devoted to vertex degrees. It is an application of the concept of vertex degrees, as well as some results from Chapter 4, to solve a problem that could be said to be the beginning of graph theory.

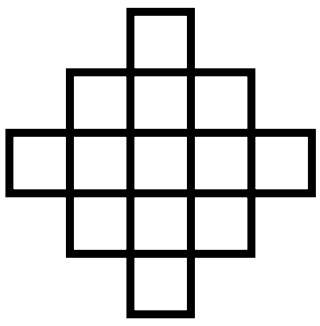
There is a striking contrast in graph theory between the theory of Euler tours and Eulerian graphs, discussed in this chapter, and the theorem of Hamilton cycles and Hamiltonian graphs, discussed in Chapter 16. Superficially, the two problems are very similar, yet we are able to pin down the exact circumstances in which an Euler tour exists; for Hamiltonian cycles, hard work is required in most cases.

We will consider the Eulerian tour problem for multigraphs; I will put the word "multigraph" in statements of theorems to remind you of this. Of course, everything we say will also be true for graphs: graphs are just multigraphs that happen not to have any loops or parallel edges.

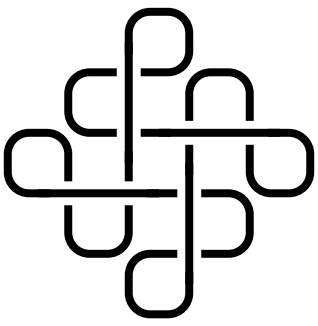
8.1 Don't lift your pencil!

You might have seen a brainteaser like this before! Look at the diagram in Figure 8.1a. Can you draw this shape on paper without lifting your pencil (or pen), and without going over the same line more than once?

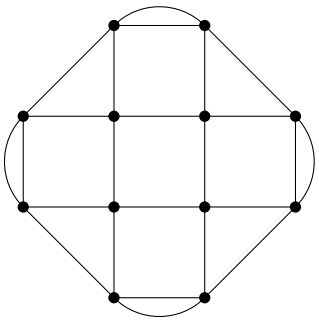
You can: Figure 8.1b shows you one particularly symmetric way to do it. (To make it clear what the solution is, the lines are bent away from each other slightly to indicate where the path does or does not cross itself.) As a bonus, the path in Figure 8.1b returns to where it started.



(a) Can you draw this figure without lifting your pencil?

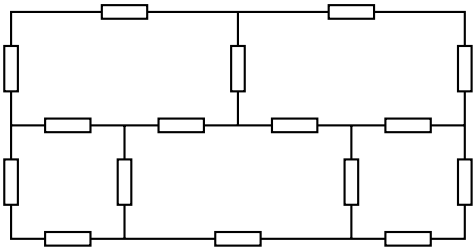


(b) Solution to the challenge in Figure 8.1a

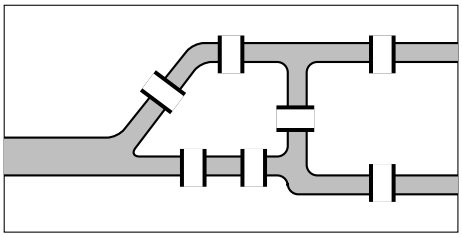


(c) Modeling the puzzle as a multigraph

Figure 8.1: Drawing without lifting your pencil



(a) The five rooms puzzle



(b) The bridges of Kőnigsberg

Figure 8.2: Two more puzzles equivalent to pencil drawing

This is a book on graph theory, though, not on pencil drawings. So what does this question have to do with graph theory? This is the same question that this book started with: how do we model this problem as a graph?

The places in the drawing that matter most are the places where several lines meet: when tracing the drawing with your pencil, those are the only places where we make a decision. So it makes sense to have a vertex for each of those points. The rest of the drawing consists of lines that go from one such point to another, and their shape does not matter for solving the puzzle: only the starting and ending point matters. So for every such line, we will add an edge between the two vertices it joins. If we apply idea to Figure 8.1a, the result is shown in Figure 8.1c.

Question:	The result is a multigraph; how did that happen?
Answer:	We get parallel edges whenever there are two or more lines that start and end at the same pair of intersection points.

Question:	What sort of graph-theoretic object are we looking for when we solve the brainteaser?
Answer:	We are looking for a walk in the graph which uses every edge exactly once.

There are two other puzzles worth mentioning that are equivalent graph-theoretically to the pencil-drawing puzzle. The first is the five rooms puzzle, shown in Figure 8.2a. Here, five rooms are connected by a total of 16 doors, and the challenge is to plot a path through the rooms that passes exactly once through every door.

Question:	What should the vertices and edges be in the multigraph that makes the five rooms puzzle equivalent to the pencil-drawing puzzle?
Answer:	There should be six vertices: the five rooms, and the outside region. Each door between two rooms should be an edge.

The final such puzzle, the problem of the seven bridges of Kőnigsberg, has historical significance. It was first studied by Leonhard Euler, in 1736, and it is notable for being the first

mathematical problem to be abstracted into a graph-theoretic model before being solved. (It predates modern graph theory, so the terminology used by Euler was quite different from what you are reading here.) The problem itself is based on a map of the city of Kőnigsberg, shown somewhat abstractly in Figure 8.2b. In Euler's time, there were seven bridges connecting the two sides of the river and the two large islands in the middle of the river. The challenge was to find a path through the city that crossed each bridge exactly once.

Question:	What should the vertices and edges be in the multigraph that models the bridges of Kőnigsberg?
Answer:	There should be a vertex for each of the four landmasses, with an edge for every bridge.

The problem we are solving in all three of these puzzles is named in Euler's honor.

Definition 8.1. *A walk that contains every edge of a graph or multigraph exactly once is called an* ***Euler walk****. If, additionally, the Euler walk is closed, it is called an* ***Euler tour****.*

All three puzzles we have looked at are about finding an Euler walk. Even though the solution in Figure 8.1b happens to be an Euler tour, we have not had a reason to consider Euler tours for their own sake so far. However, you are about to see in this chapter that Euler tours are actually the more fundamental, and more natural object. This is reflected in the following definition:

Definition 8.2. *A graph or multigraph is called* ***Eulerian*** *if it has an Euler tour.*

8.2 First steps

The key to determining whether an Euler walk or Euler tour exists turns out to be in classifying vertices as **odd** or **even**. We will use these terms just as in Chapter 4, calling a vertex even if it has even degree and odd if it has odd degree.

The following result does not completely answer our question, but is easier to prove than the main theorem of this chapter, so it is where we will begin.

Lemma 8.1. *In an Eulerian multigraph, every vertex must be even.*

Proof. Let G be an Eulerian multigraph, and let the sequence of alternating vertices and edges

$$x_0, e_1, x_1, e_2, x_2, \ldots, e_m, x_m$$

be an Euler tour of G. We prove a stronger statement: if a vertex x appears k times among $x_1, \ldots, x_m$, then $\deg_G(x) = 2k$. Intuitively, the argument for this is that if we follow the Euler tour around G, then every time the tour enters and leaves x, it uses 2 more edges incident on x.

Formally, let I be the set $\{i : 1 \leq i \leq m \text{ and } x_i = x\}$: the set of all positive positions in the Euler tour where x appears. If $|I| = k$, then the k edges $\{e_i : i \in I\}$ and the k edges $\{e_{i+1} : i \in I\}$ must be incident on x, contributing $2k$ to $\deg_G(x)$. This has two caveats:

- If $x = x_i = x_{i+1}$ for some i, then edge e_i is double-counted in the argument above. However, in this case, edge e_i is a loop with both endpoints at x, contributing 2 to $\deg_G(x)$: it *should* be double-counted.
- If $x = x_m$, then there is no edge e_{m+1} to count. However, in this case, $x = x_0$, because the Euler tour is a closed walk, so edge e_1 should be counted instead.

So in both unusual cases, the total contribution to the degree of x is still $2k$.

Every edge of G appears in the Euler tour exactly once, and so by counting the contribution of edges $e_1, \ldots, e_m$ to the degree of x, we compute the degree completely. This shows that $\deg_G(x) = 2k$, an even number; since x was arbitrary, all vertices must be even. □

Question:	Why doesn't Lemma 8.1 solve the problem completely?
Answer:	If every vertex in a multigraph is even, the lemma doesn't guarantee that the multigraph *does* have an Euler tour.

In other words, we have shown that the condition "every vertex in G is even" is a **necessary condition** for G to be Eulerian: that is, G can't be Eulerian without it. We have not determined whether it is a **sufficient condition** for G to be Eulerian: whether all graphs that satisfy the condition do in fact have Euler tours.

Question:	If $x_0, e_1, x_1, e_2, x_2, \ldots, e_m, x_m$ were an Euler walk, but not an Euler tour, what would change in the proof of Lemma 8.1?
Answer:	When $x = x_m$, it would no longer be true that $x = x_0$ as well, so the degree of x would be $2k-1$ rather than $2k$: x would be odd. Also, when $x = x_0$, the argument wouldn't "notice" the contribution of edge e_1 to x's degree, because I only contains positive integers, so x would be odd in this cases as well.

Following this reasoning carefully gives us the following claim:

Lemma 8.2. *If a multigraph G has an $x - y$ Euler walk, where $x \neq y$, then vertices x and y must be odd, while all other vertices must be even.*

Question:	Which of the puzzles in Figure 8.2 have solutions?
Answer:	Neither puzzle has a solution. If we convert them to multigraphs, then in both cases there will be four odd vertices. However, Lemma 8.2 implies that a graph with an Euler walk can have at most two odd vertices.

The condition "at most two odd vertices" is a bit more complicated than the condition "no odd vertices", but that's not the real reason that Euler tours are taken to be more fundamental than Euler walks.

Imagine for a moment that we had a method for finding Euler tours, but *not* Euler walks, and suppose that we were faced with a graph G where only two vertices, x and y, were odd. Well, we could fix this quite easily: we could simply add an edge e with endpoints x and y. In the new graph—call it H—all vertices would be even, so H would have an Euler tour.

In our imagination, we already have a method for finding Euler tours, so we could easily find an Euler tour in H. That Euler tour would have to use the new edge e at some point. Well, if we simply delete edge e from the tour, we could follow the tour starting from x and visit all the other edges, ending at y. This is an Euler walk in G.

Outside our imagination, we don't have any systematic method of finding Euler walks or Euler tours, yet. However, this argument is a **reduction** from the Euler walk problem to the Euler tour problem. It tells us that all we need to focus on is finding Euler tours in graphs where all vertices are even—as soon as we find an algorithm to do this, we will immediately have an algorithm for Euler walks as well!

8.3 Cycle decompositions

Let's take a detour[1] and consider cycle decompositions.

A **decomposition** of a graph G is a partition of the edges of G: we split up the graph into many pieces that share no edges. We can think of the pieces as sets of edges (in which case their union as sets is $E(G)$) or as subgraphs (in which case their union as graphs is G). There are many problems out there about finding special kinds of decompositions of a graph.

In particular, a **cycle decomposition** of G is a set of cycles in G such that every edge of G is in exactly one cycle. Here, we can think of these cycles in *three* ways: as walks which happen to be cycles, or as sets of edges, or as cycle subgraphs of G.

When studying the Euler tour problem in 1912, Oswald Veblen realized that a single closed walk and a collection of many cycles share a common feature: in both cases, if they visit a vertex k times, they use $2k$ edges incident on that vertex. Therefore the necessary condition for Eulerian graphs is also necessary for having a cycle decomposition. However, Veblen was able to prove [5] that for cycle decompositions, this condition is both necessary and sufficient!

This proof uses Theorem 4.4 from Chapter 4 in a key step. We proved this theorem for simple graphs, but if you go back and examine our argument, you should be able to convince yourself that it applies to multigraphs as well.

Theorem 8.3. *A multigraph has a cycle decomposition if and only if all of its vertices are even.*

Proof. Suppose first that G is a multigraph with a cycle decomposition $\mathcal{C}$, and let x be an arbitrary vertex of G.

For every cycle $x_0, e_1, x_1, \ldots, e_k, x_k$ in $\mathcal{C}$ that contains x as one of its vertices x_i, the cycle also contains two edges incident on x: edges e_{i-1} and e_i. Alternatively, if the cycle has length 1, it has the form x, e, x for an edge e; then e is a loop incident on x. In both cases, the edges of the cycle contribute 2 to the degree of x.

[1] Heh-heh.

Together, the cycles in $\mathcal{C}$ include each edge of G exactly once. So if we add up the contributions of each cycle's edges to the degree of x, we just get the total degree of x in G. This is a sum of 0's and 2's, so it is even; since vertex x was arbitrary, all vertices of G must be even.

This argument shows that the condition is necessary; now, we must prove that it is sufficient. So let G be a multigraph in which all vertices are even; our task is to find a cycle decomposition of G. We will do this by strong induction on the number of edges in G. As our base case, if G has 0 edges, then the empty set is a cycle decomposition: it includes every edge exactly once, because there are no edges to include!

Suppose G has $m > 0$ edges. Pick any component of G that has at least one edge. In that component, no vertices can have degree 0: an isolated vertex would be a component with no edges, all by itself. Also, no vertices can have degree 1: by assumption, all vertices in G are even. Therefore the component has minimum degree 2. Theorem 4.4 says that a graph with minimum degree 2 must contain a cycle; applying the theorem to this component of G, we find a cycle C.

In $G - C$ (the graph we get by deleting the edges of C, leaving the vertices as they are), all vertices are still even: the degree of every vertex that lies on C goes down by 2, and all other degrees are unchanged. Also, $G - C$ has fewer than m edges (since C contained at least one edge), so by induction, $G - C$ has a cycle decomposition. Add C to that cycle decomposition, and we get a cycle decomposition of G.

By induction, a cycle decomposition exists for any number of edges. □

This proof also suggests an algorithm for finding a cycle decomposition. Simply find any cycle, set it aside, and repeat with the remainder of the multigraph until we are out of edges. How do we find a cycle? Well, the proof of Theorem 4.4 begins by taking a longest path, but this is done only to simplify the proof. Really, we can build up a path by aimless wandering through the multigraph, taking care only not to leave a vertex by the same edge we used to enter it. As soon as we revisit the vertex, we obtain a cycle in the same way that we obtained it from the longest path.

8.4 Gluing cycles together

Let's first pause to clear the air about one thing: the condition that all vertices are even is *not* sufficient to guarantee that a multigraph is Eulerian. However, the exception is rather silly.

Question:	How can you find a multigraph where all vertices are even, but which does not have an Euler tour? (The simplest example has only 2 edges.)
Answer:	Simply make sure that your multigraph has multiple connected components with edges in them! A walk can never jump from one connected component to another, so you can never include all edges of the multigraph in a single walk.

This is a minor quibble; we will be fine if we restrict our attention to connected graphs. We can even allow graphs with multiple components, as long as all components except for one are merely isolated vertices. The cycle decomposition we get from Theorem 8.3 is the key ingredient in putting together an Euler tour.

Theorem 8.4. *A multigraph is Eulerian if and only if it is connected (except possibly for isolated vertices) and all of its vertices are even.*

Proof. We know that both parts of this condition are necessary: by Lemma 8.1, all vertices must be even, and we have just discussed why the graph cannot have two components with edges. It remains to show that together, these two conditions are sufficient.

Let G be a multigraph which has only one connected component that is not an isolated vertex, and in which every vertex of G is even. The first thing we'll do is use Theorem 8.3 to find a cycle decomposition of G.

Next, we'll build up our Eulerian tour step by step. This is a proof by algorithm. Throughout this process, we will update two objects:

- a "partial tour" PT, which is a closed walk that uses each edge at most once, but might miss some edges.
- a set $\mathcal{U}$ of "unprocessed cycles": a cycle decomposition of just the edges that are *not* part of PT.

Initially, we'll take PT to be one of the cycles in our cycle decomposition, and $\mathcal{U}$ to be the set of all the other cycles. We will stop when $\mathcal{U} = \varnothing$.

One by one, we will remove a cycle from $\mathcal{U}$, and splice it into PT. To make this possible, it's important to find a cycle in $\mathcal{U}$ that contains one of the vertices visited by PT. There must always be such a cycle: otherwise, we'd get two connected components in G! There would be no edge between two disjoint sets: the set of vertices visited by PT, and the set of vertices that are part of a cycle in $\mathcal{U}$. But both of these sets contain edges, so that would contradict our assumption about the connected components of G.

So let's suppose that we have found such a cycle C. Let PT be the closed walk

$$x_0, e_1, x_1, e_2, x_2, \ldots, e_k, x_k,$$

let C be the the closed walk

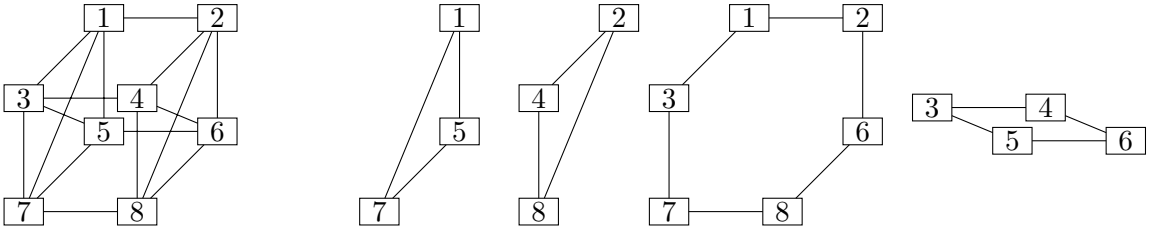
$$y_0, f_1, y_1, f_2, y_2, \ldots, f_\ell, y_\ell,$$

and suppose that they have a vertex in common: $x_i = y_j$. Then to incorporate C into PT, replace PT by the closed walk

$$x_0, e_1, x_1, \ldots, e_i, \underbrace{y_j, f_{j+1}, y_{j+1}, \ldots, f_\ell, y_\ell, f_1, y_1, \ldots, f_j, y_j}_{\text{a shifted version of } C}, e_{i+1}, x_{i+1}, \ldots, e_m, x_m.$$

This is a long sequence of symbols, but essentially, all it says is this: replace the shared vertex x_i by a version of C that's shifted to start and end at y_j (the same vertex as x_i).

After this process, PT contains all the edges which were previously traversed by C, as well as all the edges it previously contained. Therefore after we've updated PT, and removed C from



(a) A modified cube graph

(b) A cycle decomposition of the graph in Figure 8.3a

Figure 8.3: A graph and its cycle decomposition, in preparation for finding an Euler tour

$\mathcal{U}$, it's still true that $\mathcal{U}$ is a cycle decomposition of the edges not part of PT. Also, it's still true that PT uses every edge at most once: none of the new edges were previously used by PT.

Repeat this until $\mathcal{U}$ is empty. Throughout the algorithm, it's been true that all edges of G are either part of PT, or part of a cycle in $\mathcal{U}$. Now that there are no cycles in $\mathcal{U}$, all edges must be part of PT, so PT is an Euler tour. □

To truly understand the proof and the algorithm, let's go through an example, finding an Euler tour of the graph in Figure 8.3a. We will begin with a cycle decomposition. The one shown in Figure 8.3b is actually not the best one to use—life would be easier for us if we had fewer longer cycles. The decomposition with many short cycles will let us see more steps of the algorithm in action.

We can view the graph in Figure 8.3a as a simple graph, and not a multigraph. I am relying on this to simplify notation slightly: our cycles and walks will only have to record the vertices used, and not the edges.

1. Initially, our partial tour PT is just the first cycle: $1, 5, 7, 1$.
2. Next, we want to add a cycle that shares a vertex with PT: that is, we *can't* use the cycle $2, 4, 8, 2$. Let's take the fourth cycle: $3, 4, 6, 5, 3$. (We could also have used the third cycle.)

   The only common vertex between this cycle and PT is vertex 5. So we shift the cycle we're adding, putting in the form $5, 3, 4, 6, 5$. Then, we replace the 5 in PT by this shifted cycle:

   $$1, \underbrace{5, 6, 4, 3, 5}, 7, 1.$$

3. At this point, PT has lots of vertices, so we can choose any cycle to add. Let's add the second cycle: $2, 4, 8, 2$. This shares vertex 4 with PT, so we rewrite it as $4, 8, 2, 4$, and splice it into PT:

   $$1, 5, 6, \underbrace{4, 8, 2, 4}, 3, 5, 7, 1.$$

4. Finally, we add the only remaining cycle: $1, 2, 6, 8, 7, 3, 1$. This shares many vertices with PT, but in particular, it shares vertex 1. We replace *one* occurrence of vertex 1 in PT by this cycle, which we don't need to shift:

   $$\underbrace{1, 2, 6, 8, 7, 3, 1}, 5, 6, 4, 8, 2, 4, 3, 5, 7, 1.$$

The result is an Euler tour. To check this for yourself, draw all the vertices of the graph in Figure 8.3a, with no edges between them. Then, follow the walk we found, drawing each edge as you use it. As you go, check that you never draw an edge more than once; at the end, check that you have drawn all of Figure 8.3a.

8.5 Variations on Euler tours

Of course, now that we have an algorithm for Euler tours, we immediately have an algorithm for Euler walks—and a test for their existence. The only thing that changes is that a multigraph with two odd vertices can still have an Euler walk.

Question:	In a multigraph with exactly two odd vertices, is it possible for them to be in different connected components?
Answer:	It is not: if this happened, then looking at one of those connected components by itself, we'd see a subgraph with only one odd vertex. However, Corollary 4.2 requires the number of odd vertices in any graph to be even. (It is still true for multigraphs.)

Question:	Can you think of a way to modify the algorithm for Euler tours so that it finds an $x - y$ Euler walk instead?
Answer:	Begin by finding an $x - y$ path P. Deleting P will leave a graph where all vertices are even, so it will have a cycle decomposition. Now we can follow the algorithm, but taking $PT = P$ to start with, splicing in the cycles one by one, as before.

Another interesting problem is the problem of finding Euler tours in a directed graph. The first novelty in the directed setting is that we no longer check for whether a vertex is even: entering and leaving a vertex has a different effect on its indegree and outdegree.

Question:	If a directed graph D has an Euler tour, what can you say about the degree of a vertex that appears on the tour k times?
Answer:	Each arc into the vertex adds 1 to its indegree, and each arc out of the vertex adds 1 to its outdegree. Thus, the vertex will have indegree k and outdegree k.

Question:	What, then, is the corresponding necessary condition for a digraph to be Eulerian?
Answer:	Every vertex must have an indegree equal to its outdegree. (We can call such vertices **balanced**.)

To prove a necessary and sufficient condition for directed graphs, we must retrace the steps we took in the undirected setting. The only new ingredient is the result we use to find a single cycle, which we proved in the previous chapter. Can you anticipate it before it is used?

Lemma 8.5. *A digraph has a cycle decomposition if and only if all of its vertices are balanced.*

Proof. Suppose first that D is a digraph with a cycle decomposition $\mathcal{C}$, and let x be an arbitrary vertex of D. For every cycle $x_0, x_1, \ldots, x_k$ in $\mathcal{C}$ that contains x as one of its vertices x_i, the cycle contains one arc into x and one arc out of x: arcs (x_{i-1}, x_i) and (x_i, x_{i+1}). (If $x = x_0 = x_k$, these are arcs (x_{k-1}, x_k) and (x_0, x_1).) We can find the indegree and outdegree of x by adding up the contribution from each cycle in $\mathcal{C}$; since each cycle contributes an equal number to the indegree and outdegree of x, in the end x is balanced. This proves that the condition in the lemma is necessary.

Next, we show by induction on the number of arcs in D that the condition is sufficient. As the base case, if there are no arcs, then the empty set is a cycle decomposition. So now assume D is a digraph with $m > 0$ arcs in which all the vertices are balanced.

Let D' be the subgraph of D obtained by deleting all isolated vertices. In D', every vertex has a positive indegree and a positive outdegree (which are equal to each other); by Theorem 7.2, D' contains a cycle C, because it has no sinks.

Delete the edges of C from D to get a digraph $D - C$ with fewer arcs: by the inductive hypothesis, it has a cycle decomposition. Add C to that cycle decomposition to obtain a cycle decomposition of D, completing the proof. □

The only difficulty with stating the final result is that we do not exactly know what it means to have a connected component in a directed graph. I will give you the statement first, and then we will determine what it means.

Corollary 8.6. *A directed graph is Eulerian if and only if it is weakly connected (except possibly for isolated vertices) and all of its vertices are balanced.*

The proof is the same as the proof of Theorem 8.4: we start with a cycle decomposition, and splice the cycles into a partial Euler tour, one by one.

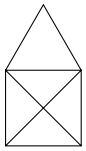
Question:	Looking at the proof of Theorem 8.4, where do we use the connectedness of the graph?
Answer:	We needed it to be impossible for the partial tour PT and the unused cycles $\mathcal{U}$ to have no vertices in common.

So the notion of connectedness we need is simply that it should be impossible to split the vertices of our digraph D into two sets, with no arcs between them in either direction—because that's what we'd get if PT and the cycles of $\mathcal{U}$ shared no vertices. This is a notion of connectedness that ignores the orientations of the arcs.

Formally, we call a digraph D **weakly connected** if replacing each arc by an undirected edge would result in a connected undirected graph. Now we know what Corollary 8.6 means!

8.6 Practice problems

1. The classic brainteaser in the "draw without lifting your pencil" genre is the picture of a stylized house shown below:

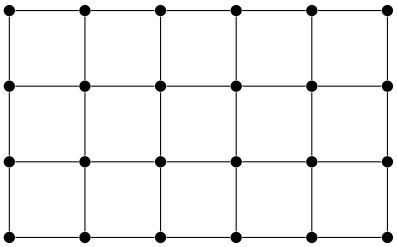


   In order to draw this picture without lifting your pencil or retracing any line, where must you start and end? (Of course, you can always swap the start and end.)

2. Find a cycle decomposition of the Harary graph $H_{10,4}$ (as defined in Chapter 5), and use it to find an Euler tour of $H_{10,4}$.

3. Find an Euler tour of the 4-dimensional hypercube graph, Q_4.

4. The graph below is not Eulerian, because some of its vertices are odd. What is the maximum length of a closed walk in this graph that does not use any edge more than once? Prove your answer.



5. In the complete graph K_7, every vertex has degree 6, so by Theorem 8.3, K_7 has a cycle decomposition.

   a) Suppose we wanted to decompose K_7 into as few cycles as possible. How many cycles would we want to use, and of what lengths? Give an example of such a decomposition.

   b) Suppose we wanted to decompose K_7 into as many cycles as possible. How many cycles would we want to use, and of what lengths? Give an example of such a decomposition.

6. Prove that every multigraph has a "double tour": a closed walk which uses every edge exactly two times, once in each direction. (That is, an edge e with endpoints x and y must appear once as $\ldots, x, e, y, \ldots$ and once as $\ldots, y, e, x, \ldots$.)

7. We will cover **perfect matchings** in detail starting in Chapter 12; briefly, a perfect matching is a spanning 1-regular subgraph of a graph. This exercise presents part of a method due to Noga Alon [1] for finding perfect matchings using Euler tours.

   a) Let G be a **bipartite** multigraph: that is, let there be a partition $V(G) = A \cup B$, with $A \cap B = \varnothing$, such that all edges of G have one endpoint in A and one endpoint in B.

   Prove that if every vertex of G is even, then G has an even number of edges.

b) Let G be a multigraph in which every vertex is even and (as in part (a)) the number of edges is even. Prove that it's possible to color half of G's edges red and the other half blue in such a way that every vertex x is the endpoint of $\frac{1}{2}\deg(x)$ red edges and $\frac{1}{2}\deg(x)$ blue edges.

c) Prove by induction on k that for all $k \geq 0$, every 2^k-regular bipartite multigraph has a perfect matching.

Bibliography

[1] Noga Alon. "A simple algorithm for edge-coloring bipartite multigraphs". In: *Information Processing Letters* 85.6 (2003), pp. 301–302. DOI: 10.1016/S0020-0190(02)00446-5.

[2] Pál Erdős and Tibor Gallai. "Gráfok előírt fokszámú pontokkal". In: *Matematikai Lapok* 11 (1960), pp. 264–274. URL: https://www.renyi.hu/~p_erdos/1961-05.pdf.

[3] S.L. Hakimi. "On Realizability of a Set of Integers as Degrees of the Vertices of a Linear Graph. I". In: *Journal of the Society for Industrial and Applied Mathematics* 10.3 (1962), pp. 496–506. DOI: 10.1137/0110037.

[4] Václav Havel. "Poznámka o existenci konečných grafů". In: *Časopis pro pěstování matematiky* 80.4 (1955), pp. 477–480. DOI: 10.21136/CPM.1955.108220.

[5] Oswald Veblen. "An Application of Modular Equations in Analysis Situs". In: *Annals of Mathematics, Second Series* 14.1 (1912), pp. 86–94. DOI: 10.2307/1967604.

[6] D.B. West. *Introduction to Graph Theory*. Introduction to Graph Theory. Prentice Hall, 1996. ISBN: 9780132278287.