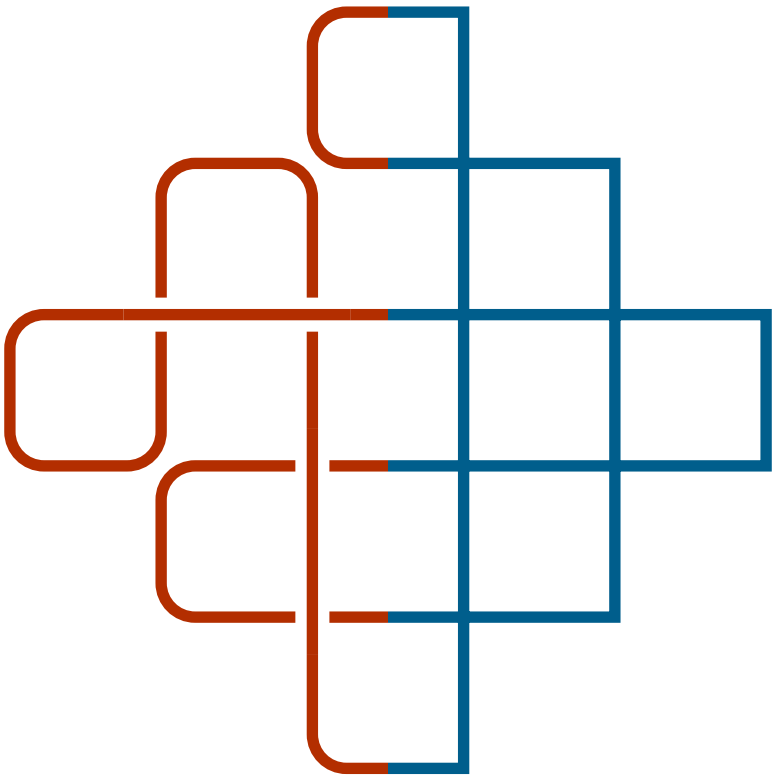
Mikhail Lavrov

Start Doing Graph Theory

Part II: Vertex Degrees



available online at https://vertex.degree/

Contents

About this document **3**
Where it's from . . . 3
What's inside . . . 3
The cover . . . 3
The license . . . 3

5 Regular graphs **4**
5.1 Degree sequences . . . 4
5.2 Regular graphs . . . 6
5.3 How many regular graphs are there? . . . 9
5.4 The Petersen graph . . . 12
5.5 The degree/diameter problem . . . 14
5.6 Practice problems . . . 17

6 Graphic sequences **19**
6.1 An unusual party . . . 19
6.2 A graphic sequence algorithm . . . 22
6.3 Two examples . . . 23
6.4 The Havel–Hakimi theorem . . . 25
6.5 More on degree sequences . . . 28
6.6 Practice problems . . . 29

7 Multigraphs and digraphs **32**
7.1 Multigraphs . . . 33
7.2 Degrees in multigraphs . . . 35
7.3 Directed graphs . . . 37
7.4 Directed walks, paths, and cycles . . . 40
7.5 Practice problems . . . 42

8 Euler tours **44**
8.1 Don't lift your pencil! . . . 44
8.2 First steps . . . 46
8.3 Cycle decompositions . . . 48
8.4 Gluing cycles together . . . 50
8.5 Variations on Euler tours . . . 52
8.6 Practice problems . . . 55

Bibliography **57**

About this document

Where it's from

If you downloaded this PDF file yourself from https://vertex.degree/contents, presumably you know what you were doing. But in case you're confused or got the PDF file somewhere else, let me explain!

This is not an entire graph theory textbook. It is one part of the textbook *Start Doing Graph Theory* by Mikhail Lavrov, in case it's convenient for you to download a few smaller files instead of one large file. You can find the entire book at https://vertex.degree/; all of it can be downloaded for free.

The complete textbook has some features that the individual parts couldn't possibly have. In this PDF, if you click on a chapter reference from a different part of the book, then you will just be taken to this page, because I can't take you to a page that's not in this PDF file. The hyperlinks in the complete book are fully functional; there is also a preface and an index.

What's inside

In this part of the book, I go into further detail about what the degrees of vertices in graphs, and what we can learn about graphs from their degree sequences. Chapter 5 is an overview of regular graphs. Chapter 6 covers the graphic sequence problem: the problem of determining whether there is a graph with a given degree sequence. Chapter 7 is an introduction to multigraphs and directed graphs. Chapter 8 is about finding Euler tours in graphs, and determining when they exist.

The cover

The cover of this PDF shows a partially-solved puzzle of the "trace this pattern without lifting your pencil" variety. These can be found by solving an Euler tour in a graph, as you will see in Chapter 8.

The license

This document is licensed under a Creative Commons Attribution-ShareAlike 4.0 International License: see https://creativecommons.org/licenses/by-sa/4.0/ for more information.

5 Regular graphs

The purpose of this chapter

Most chapters in this book are all about giving you tools to deal with any graph you encounter. This chapter has a bit of that, but it is much more about showing you examples of some particular useful graphs.

In Theorem 5.1, we use circulant graphs (which we previously defined in Chapter 2 to construct examples of regular graphs in every case where it is possible. In the next section, I attempt to convey some of the variety of possible regular graphs out there (except, of course, in the cases without much variety). Next, we meet the Petersen graph, which will appear as an example or counterexample many times throughout this book: it will be especially useful to have in our back pocket when we get to Chapter 17 and Chapter 20.

The last section discusses the degree/diameter problem; this problem is not exclusive to regular graphs, but many of the examples found when making progress on it are regular. It is also an example of a research problem in graph theory where the main form of progress on it is in fact by finding examples.

There is no general way to find examples; otherwise, problems like the degree/diameter problem would already be solved. However, being familiar with a wide variety of examples, and knowing what they're good for, is valuable when it comes to writing existence proofs. The proof of Theorem 5.1 would have been much harder if we did not already have circulant graphs at our disposal.

In this chapter, we take our first serious look at necessary and sufficient conditions. This is a theme we will return to throughout the book; if you need an introduction to necessary and sufficient conditions, you can find it in Appendix A.

5.1 Degree sequences

In Chapter 4, we looked at the degrees of individual vertices; now, let's look at all of them together.

Definition 5.1. *The **degree sequence** of a graph 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"

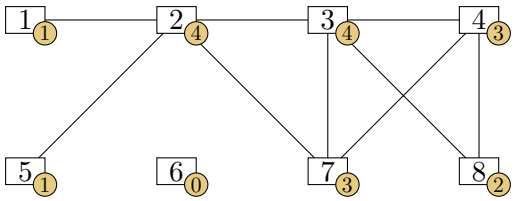


Figure 5.1: An 8-vertex graph labeled with the degrees of its vertices

of a graph: a multiset keeps track of how often its elements appear, but not their order, which is exactly what we want here.

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 dot, 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 begin determining whether two graphs are isomorphic.

Question:	If you want to know whether two graphs are isomorphic, what do their degree sequences tell you?
Answer:	If G and H have different degree sequences, they're definitely not isomorphic. If they have the same degree sequences, they might be isomorphic, but it's too early to say. In other words, G and H having the same degree sequence is a necessary condition, but not a sufficient condition for them to be isomorphic.

Though it is easy to compute the degree sequence of a given graph, it is harder to go in reverse: to take a sequence of nonnegative integers in descending order, and determine if it is the degree sequence of some graph. If it is, we call it 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?*

[1] I want to make it clear that in this book, sequences in "descending order" or "ascending order" can have ties: consecutive terms may be equal. Sometimes a sequence in descending order is called "non-increasing" to make it clear that ties are allowed, but I think this is awkward.

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, which states that a graph cannot have an odd number of vertices with an odd degree. □

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 rules always enough to tell if a sequence is graphic?
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. They are necessary conditions: if a sequence fails one of the two preliminary tests, it's definitely not graphic. This can save us a lot of work when faced with a larger problem of this type.

However, the preliminary tests are not sufficient conditions: if a sequence 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 5.2. *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.

In fact, the existence problem for regular graphs is much easier than the general case. There are no further complications, and 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.

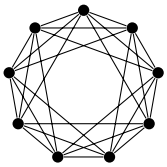
If we were solving the problem for specific values of r and n, the proof would be very short: we could simply draw a diagram and verify that the graph shown has the right number of vertices, all with the correct degree. To solve the problem in general, we need to give a family of graphs as the solution.

Not every family of graphs is sufficiently flexible for our purposes. For example, in Chapter 4, we defined the family of hypercube graphs. These are all regular graphs: for any r, if we want an r-regular graph, the r-dimensional hypercube graph Q_r will do. However, there is only one r-dimensional hypercube graph, and it has 2^r vertices. If we want a regular graph with some other number of vertices, we have to do something else.

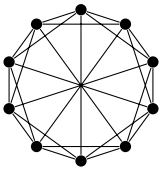
Fortunately, we do have a very flexible family of regular graphs at our fingertips: the circulant graphs introduced in Chapter 2. These are actually even more flexible than we need: for large n, we can pick from many examples of circulant graphs for each possible degree. To remind you of the definition, the circulant graph $\mathrm{Ci}_n(d_1, d_2, \ldots, d_k)$ has vertex set $\{0, 1, \ldots, n\}$, which we think of as being arranged around a circle, in that order. Two vertices x and y are adjacent if $x - y \equiv \pm d_i \pmod{n}$ for some i; if x and y are d_i steps apart around the circle.

Usually, each offset d_i gives each vertex x two neighbors: $x - d_i \bmod n$ and $x + d_i \bmod n$. As a result, if r is even, we can get an r-regular circulant graph by using $\frac{r}{2}$ offsets. To give a concrete example, $\mathrm{Ci}_n(1, 2, \ldots, \frac{r}{2})$ will be an r-regular graph for all even r between 2 and $n-1$, proving almost half of the theorem by itself. (For an example, see Figure 5.2a: this is the graph we use when $n = 9$ and $r = 6$.)

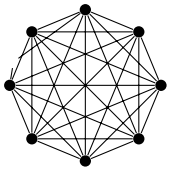
Question:	What if $r = 0$?
Answer:	I am reluctant to write $\mathrm{Ci}_n()$, because this looks weird, but it's true that if we don't give a circulant graph any offsets, then there will be no edges: the result is a 0-regular graph, for any positive integer n.



(a) $\mathrm{Ci}_9(1,2,3)$



(b) $\mathrm{Ci}_{10}(1,2,5)$



(c) $\mathrm{Ci}_8(1,2,3,4)$

Figure 5.2: Circulant graphs in the proof of Theorem 5.1

It is a bit trickier to get an r-regular circulant graph when r is odd. We know that it is impossible to find an r-regular graph on n vertices if both r and n are odd, so we may assume that n is even. But how does this help us?

Question:	In what exceptional case does an offset d_i in a circulant graph not contribute $+2$ to the degree of every vertex?
Answer:	When n is even and the offset is equal to $\frac{n}{2}$.

For any x, $x - \frac{n}{2} \bmod n$ and $x + \frac{n}{2} \bmod n$ are the same vertex: if we think of the vertices as being arranged around a circle, then this is the vertex opposite x. So including the offset $\frac{n}{2}$ lets us obtain an r-regular graph when r is odd. Again, we must give a concrete example to make the proof precise about it does, so let our example be $\mathrm{Ci}_n(1, 2, \ldots, \frac{r-1}{2}, \frac{n}{2})$. (For an example, see Figure 5.2b: this is the graph we use when $n = 10$ and $r = 5$.)

Question:	Why is this an r-regular graph?
Answer:	For each $i = 1, \ldots, \frac{r-1}{2}$, a vertex x is adjacent to $x + i \bmod n$ and $x - i \bmod n$, giving it $2 \cdot \frac{r-1}{2}$ or $r - 1$ neighbors. The last neighbor is $x + \frac{n}{2} \bmod n$.

Because $r \leq n - 1$, it is always true that $\frac{r-1}{2} < \frac{n}{2}$, so the initial progression $1, 2, \ldots, \frac{r-1}{2}$ never risks overlapping the exceptional case $r = n$. Similarly, in our previous case when r was even, the offsets $1, 2, \ldots, \frac{r}{2}$ never included the "exceptional" offset $\frac{n}{2}$, because $\frac{r}{2} < \frac{n}{2}$.

This completes the proof: for all r and n such that $0 \leq r \leq n-1$ and at least one of r or n is even, either the circulant graph $\mathrm{Ci}_n(1, 2, \ldots, \frac{r}{2})$ (for even r) or the circulant graph $\mathrm{Ci}_n(1, 2, \ldots, \frac{r-1}{2}, \frac{n}{2})$ (for odd r) is an r-regular graph on n vertices. □

Question:	What graph do we end up constructing when $r = n - 1$?
Answer:	For odd n, we get $\mathrm{Ci}_n(1, 2, \ldots, \frac{n-1}{2})$, and for even n, we get $\mathrm{Ci}_n(1, 2, \ldots, \frac{n}{2})$, but in both cases, the result is isomorphic to the complete graph K_n. This is because in both cases, we've included every possible offset. For an example of this, see Figure 5.2c.

I chose the circulant graphs I did in the proof of Theorem 5.1 not just because they're the simplest possible choice, but for another reason. The r-regular circulant graph on n vertices that we used has a special name: the *Harary graph*, denoted $H_{n,r}$. For example, the graphs in Figure 5.2 are $H_{9,6}$ (Figure 5.2a), $H_{10,5}$ (Figure 5.2b), and $H_{8,7}$ (Figure 5.2c).

We will see these graphs again in Chapter 26, where we will put them to the same use as their inventor Frank Harary did in 1962 [6]: as examples of graphs of a given connectivity with as few edges as possible.

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 found 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. We say that there are 3 different 1-regular graphs with vertex set $\{1, 2, 3, 4\}$, but that up to isomorphism, there is only one 1-regular graph with 4 vertices. This is a term we'll use often, so let me give it a formal definition:

Definition 5.3. *A description of a graph G* ***up to isomorphism*** *is a description which is not necessarily true of G, but is true of some graph isomorphic to G.*

Here are some of the ways this phrase is used, in other circumstances.

- We say that the solution to a problem is unique up to isomorphism when all solutions to the problem are isomorphic graphs.
- We say that S is the set of all graphs of some type, up to isomorphism, when every graph of that type is isomorphic to some graph in S, and no two graphs in S are isomorphic. If $|S| = k$, we might also say that there are k graphs of that type, up to isomorphism.
- We say that the result of an operation is some particular graph G, up to isomorphism, if the operation results in a graph isomorphic to G, but possibly not with the same vertex set as G.

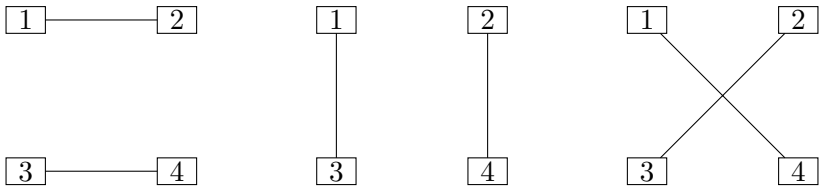
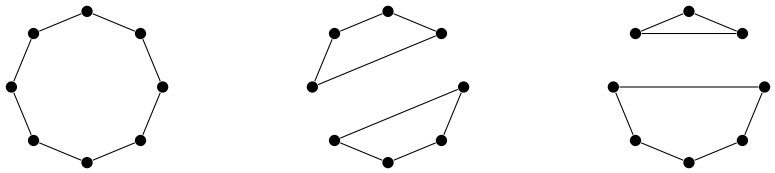


Figure 5.3: Three different(?) 1-regular graphs with vertex set $\{1, 2, 3, 4\}$



(a) C_8 (b) Two copies of C_4 (c) Copies of C_3 and C_5

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

Statements up to isomorphism are also sometimes phrased in terms of *unlabeled graphs*, which is a perspective I'll say more about in Chapter 11, but I prefer to avoid it, because it causes more confusion. For any graph G, the vertex set $V(G)$ is a set of distinguishable objects, whether or not we distinguish them in a diagram of G.

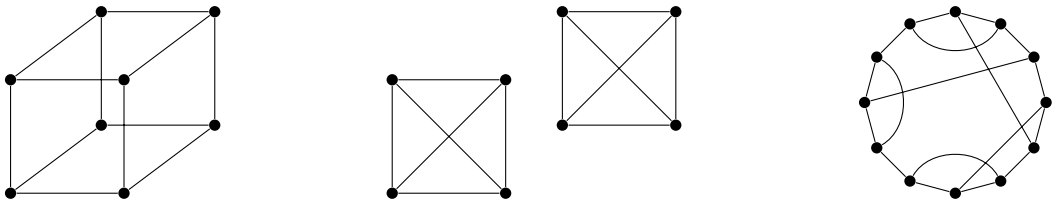
Moving on, let's consider what happens when $r = 2$. The example we constructed in Theorem 5.1 is $\mathrm{Ci}_n(1)$, but this is isomorphic to a more common graph: the cycle graph C_n. This is also unique in a sense, but a slightly different sense:

Proposition 5.2. *Up to isomorphism, C_n is the unique connected 2-regular graph with n vertices.*

Proof. First of all, we check that C_n a connected 2-regular graph. In C_n (with vertex set $\{1, 2, \ldots, n\}$) vertex x is adjacent to $x - 1$ and $x + 1$, treating $1 - 1$ as n and $n + 1$ as 1, so $\deg(x) = 2$ for all $x \in V(C_n)$. For all $x, y \in V(C_n)$, the sequence $(x, x+1, \ldots, y)$ is an $x-y$ walk if $x < y$, and the sequence $(x, x - 1, \ldots, y)$ is an $x - y$ walk if $x > y$; this makes C_n connected.

Next, let G be an arbitrary connected 2-regular graph. Then in particular G has minimum degree 2, so Theorem 4.4 applies and tells us that G contains a cycle C. For all $x \in V(C)$, we know $\deg_C(x) = 2$ because C is a cycle, and $\deg_G(x) = 2$ because G is 2-regular, so x has no neighbors in G other than the two it has in C. There are no edges with exactly one endpoint in $V(C)$, so by Lemma 3.2, either $G = C$ or else G wouldn't be the connected graph we took it to be. But $G = C$ makes G isomorphic to a cycle graph, completing our proof. □

The difference between the 1-regular and 2-regular case is that we can obtain more examples by taking graphs with multiple connected components. Each component, by Proposition 5.2, is a cycle; we can mix and match cycles however we like to reach a total of n vertices. Figure 5.4 shows three non-isomorphic possibilities for an 8-vertex 2-regular graph: it could be isomorphic to C_8 (Figure 5.4a), or to the union of two copies of C_4 (Figure 5.4b), or to the union of copies of C_3 and C_5 (Figure 5.4c).



(a) Q_3 (b) Two copies of K_4 (c) The Frucht graph

Figure 5.5: Three examples of 3-regular graphs

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.

Figure 5.5 shows several 3-regular graphs. You will recognize the graph in Figure 5.5a: it is the cube graph Q_3. The graph in Figure 5.5b should not be too much of a surprise either: it is the disjoint union of two copies of K_4, and is the smallest 3-regular graph that is not connected.

You might be forgiven for thinking, based on the examples you've seen, that regular graphs are highly symmetric—after all, every vertex has the same degree, which makes every vertex like every other, at least superficially. To dissuade you from that impression, I've included a final example in Figure 5.5c: the Frucht graph. It is named after Robert Frucht, who discovered it in 1949 [4]. The Frucht graph is notable for being completely asymmetric; in other words, it has no automorphisms, other than the identity automorphism!

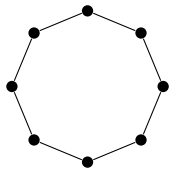
The Frucht graph is just the smallest graph with this property. When n is large, it is very rare for a 3-regular graph with n vertices to have a nontrivial automorphism. Don't be misled by graphs like circulant graphs, which have a lot of symmetry—they are just the graphs it is easiest to point to, precisely for that reason. (Think about how much effort it would be to describe the Frucht graph exactly, compared to how easy it is to define the circulant graph $\mathrm{Ci}_{12}(1, 6)$, which has the same number of vertices and edges.)

The Online Encyclopedia of Integer Sequences (OEIS) lists integer sequences with notable mathematical properties. Entry A002851 of the OEIS [12] 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. Starting from $2n = 4$, the sequence is

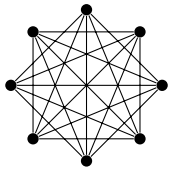
$$1, 2, 5, 19, 85, 509, 4060, 41301, 510489, 7319447, \ldots$$

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. We discover that semblance of order by taking the complement graph: toggling all the edges from present to absent and vice versa.

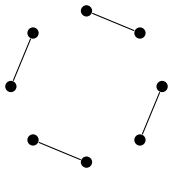
Question:	If G is an r-regular graph on n vertices, what can you say about the degrees in its complement $\overline{G}$?
Answer:	Every vertex $x \in V(G)$ has r neighbors in G, out of $n-1$ other vertices, so its neighbors in $\overline{G}$ are the $(n-1)-r$ vertices it was not previously adjacent to. Therefore $\overline{G}$ is $(n-r-1)$-regular.



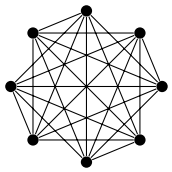
(a) C_8



(b) $\overline{C_8}$



(c) A 1-regular graph



(d) A 6-regular graph

Figure 5.6: Regular graphs and their complements

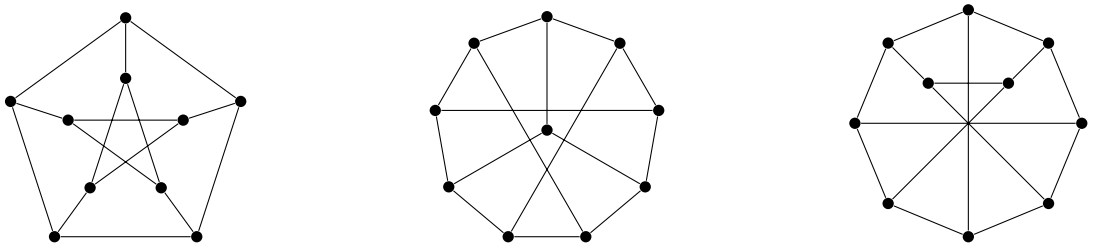


Figure 5.7: Three diagrams of the Petersen graph

Figure 5.6 shows some examples of regular graphs and their complements. For example, the cycle graph C_8 is an 8-vertex 2-regular graph we understand, so we must also be able to understand its complement, an 8-vertex 5-regular graph. We already understand the complete graph K_n quite well; it is the only n-vertex $(n-1)$-regular graph, up to isomorphism, becuase if every vertex has degree $n-1$, then it is adjacent to all $n-1$ other vertices. In fact, the empty graph we previously took as our 0-regular example is commonly written $\overline{K_n}$, taking the complete graph as a starting point.

Up to isomorphism, there is only one 1-regular graph on n vertices, and only when n is even. Taking its complement, we learn that up to isomorphism, there is only one $(n-2)$-regular graph on n vertices, and only when n is even. Figures 5.6c and 5.6d show an example of this.

Among 2-regular graphs, cycles are special; they are the only connected graphs. Among $(n-3)$-regular graphs, the complements of cycles are not as special, because (for $n \geq 5$) all $(n-3)$-regular graphs are connected. Still, by taking the complement of every graph in Figure 5.4, we could find all the 5-regular 8-vertex graphs, up to isomorphism; the same strategy works for all n.

5.4 The Petersen graph

Having seen a bit of the variety of regular graphs out there, we'll now look at another useful family of regular graphs. We'll begin with a specific example, known as the Petersen graph after its discoverer, Julius Petersen.

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. Entire books [9] have been written about this graph.

Three diagrams of the Petersen graph are shown in Figure 5.7, showing some of the symmetry it has: one with 5-fold rotational symmetry, one with 3-fold symmetry, and one which nearly has 4-fold symmetry (but not quite). In fact, the Petersen graph is much more symmetric than any of these diagrams show, which can be seen from its combinatorial definition:

Definition 5.4. *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.

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 $\{1, 2, \ldots, n\}$.

For now (in this section), I will prove only one property of the Petersen graph, which might look silly on its own. However, it will be a starting point for other proofs, and in the next section, we will see one example of how it can be useful. This lemma is also a good example of how to use symmetry in a proof.

Lemma 5.3. *The Petersen graph has no cycles of length* 3 *or* 4*.*

Proof. To check such a claim with as little effort as possible, it helps to make use of the many automorphisms of the Petersen graph. Take any two adjacent vertices of the Petersen graph: these are unordered pairs ab and cd, where a, b, c, d are four distinct elements of $\{1, 2, 3, 4, 5\}$. Then there is an automorphism of the Petersen graph that relabels a to 1, b to 2, c to 3, and d to 4 in every vertex, taking our two adjacent vertices to 12 and 34.

How do we use this? Well, suppose C is an arbitrary cycle in the Petersen graph, and ab and cd are two vertices adjacent in C. By applying the automorphism above to C, we get a new cycle C' in the Petersen graph on which 12 and 34 are two consecutive vertices. The automorphism is a bijection, so C and C' have the same number of vertices: if we prove C' has length at least 5, we conclude that C has length at least 5.

(When writing for an audience familiar with such tricks, it is common to simply say: "Let C be a cycle in the Petersen graph. By symmetry, we may assume that 12 and 34 are two consecutive vertices along C." The proof becomes much shorter.)

On a cycle of length 3, two consecutive vertices have a common neighbor: the third vertex. On a cycle of length 4, two consecutive vertices each have another neighbor, and those two neighbors must themselves be adjacent. To complete the proof, we show that neither of these scenarios is possible in the Petersen graph, when 12 and 34 are the two consecutive vertices.

The other neighbors of 12 are 35 and 45; the other neighbors of 34 are 15 and 25. From this we can see that 12 and 34 have no common neighbors, so our cycle cannot have length 3; also, none of the neighbors of 12 and 34 are adjacent (all of them include the number 5), so our cycle cannot have length 4. □

5.5 The degree/diameter problem

In Chapter 3, we defined the distance $d_G(x, y)$ between two vertices in a graph G to be the minimum length of an $x - y$ walk in G. A related concept is the *diameter* of G: the largest distance between any pair of vertices.[2] That is, a graph has diameter D if we can reach every vertex from every other vertex in at most D steps, but there are some cases in which we cannot do it in fewer than D steps.

It's worth pointing out that the diameter is defined by a maximization problem (it is the largest distance) with a minimization problem inside it (the distance is the smallest possible length of a walk). This means that proving that the diameter of a graph has a certain value takes some getting used to: it is a mix of specific examples and universal bounds, in both directions.

Question:	How might you prove that a graph G has diameter at least D?
Answer:	By proving that $d_G(x, y) \geq D$ for two particular vertices x and y: showing that there are no $x - y$ walks of any length less than D.

[2] This term in graph theory is inspired by the diameter of a circle, since that is the largest distance between any two points in or on the circle.

Question:	How might you prove that a graph G has diameter at most D?
Answer:	By proving that $d_G(x, y) \leq D$ for all pairs of vertices x, y, showing that we can get from every vertex to every other in at most D steps.

We can see both kinds of proof in action when we prove the following result, which is also the first noteworthy use of the Petersen graph in this textbook.

Proposition 5.4. *Among all graphs with maximum degree* 3 *and diameter* 2*, the Petersen graph has the most vertices.*

Proof. First, we should verify that the Petersen graph actually does have diameter 2. The lower bound is quick: a diameter of 1 would mean that all vertices are adjacent, and the Petersen graph certainly has pairs of vertices that are not adjacent (such as 12 and 13). Therefore the Petersen graph has diameter at least 2.

For the upper bound, we must show that no matter which vertex x we choose in the Petersen graph, all vertices are within distance 2 of x. Since the Petersen graph is 3-regular, x has three neighbors; call them y_1, y_2, y_3. Each y_i has two neighbors other than x. By Lemma 5.3, y_i cannot be adjacent to another y_j (or else (x, y_i, y_j, x) would represent a cycle of length 3), and two neighbors y_i and y_j of x cannot have a common neighbor z (or else (x, y_i, z, y_j, x) would represent a cycle of length 4.) Therefore there are six different vertices $z_1, \ldots, z_6$ all adjacent to a neighbor of x.

Altogether, we've named 10 vertices: x, $y_1, \ldots, y_3$, and $z_1, \ldots, z_6$. All are different, and all are within distance 2 of x. But the Petersen graph only has 10 vertices, so we conclude that all vertices are within distance 2 of x. Since x was arbitrary, the diameter of the Petersen graph is at most 2.

The last part of the proof is the interesting part, in my opinion. We must prove that a graph G with maximum degree 3 and diameter 2 can have at most 10 vertices: the number of vertices in the Petersen graph. In such a graph, if we pick an arbitrary vertex x, we can write $V(G)$ as $\{x\} \cup Y \cup Z$, where Y is the set of vertices at distance 1 from x, and Z is the set of vertices at distance 2 from x.

The graph G has maximum degree 3, so in particular, x has at most 3 neighbors; every vertex in Y must be a neighbor of x, so $|Y| \leq 3$. Meanwhile, for every vertex $z \in Z$, there is a path (x, y, z) of length 2; here, $y \in Y$, because it is adjacent to x. Each vertex has at most 3 neighbors, but one of them is x: it has at most 2 neighbors in Z. Together, the vertices in Y have at most $3 \cdot 2 = 6$ neighbors in Z, but each vertex in Z must have a neighbor in Y: so $|Z| = 6$.

Putting this together, $|V(G)| = |\{x\}| + |Y| + |Z| \leq 1 + 3 + 6 = 10$, so G can have at most 10 vertices. The Petersen graph, with 10 vertices, is optimal. □

The degree/diameter problem studies the generalization of Proposition 5.4: for a pair of positive integers (Δ, D), what is the largest number of vertices in a graph with maximum degree Δ and diameter D?

Theorem 5.5. *A graph G with maximum degree $\Delta \geq 2$ and diameter D can have at most*

$$1 + \Delta + \Delta(\Delta-1) + \Delta(\Delta-1)^2 + \cdots + \Delta(\Delta-1)^{D-1}$$

vertices, which is $2D+1$ if $\Delta = 2$ and $1 + \Delta\frac{(\Delta-1)^D-1}{\Delta-2}$ if $\Delta > 2$.

Proof. As in the proof of Proposition 5.4, we pick an arbitrary vertex x and write the vertex set of G as

$$V(G) = Y_0 \cup Y_1 \cup Y_2 \cup \cdots \cup Y_D$$

where $Y_0 = \{x\}$ and Y_i is the set of all vertices at distance i from x.

Next, by induction on i, we show that $|Y_i| \leq \Delta(\Delta-1)^{i-1}$ for $1 \leq i \leq D$. When $i = 1$, this inequality says that $|Y_1| \leq \Delta$, which is true because x has at most Δ neighbors, and every vertex in Y_1 is a neighbor of x. This proves the base case.

An important property we'll need before we continue is that for $1 \leq i \leq D$, a vertex $y \in Y_i$ must have a neighbor in Y_{i-1}. That's because there is an $x - y$ walk of length i in G. Let z be the next-to-last vertex on that walk: the walk is $(x, \ldots, z, y)$. This z will be the neighbor of y in Y_{i-1}.

Question:	Why is z a neighbor of y?
Answer:	By the definition of a walk, yz must be an edge.

Question:	Why is there an $x - z$ walk of length $i-1$?
Answer:	Remove the last vertex y from the walk $(x, \ldots, z, y)$ to get such an $x - z$ walk.

Question:	Why is there no $x - z$ walk of length $i-2$ or less?
Answer:	We could take such a walk and add y to the end, getting an $x - y$ walk of length $i-1$ or less; this contradicts $y \in Y_i$.

Now we're ready for the induction step. Suppose that for $1 \leq i \leq D-1$, we have already shown that $|Y_i| \leq \Delta(\Delta-1)^{i-1}$, and want to move on to $|Y_{i+1}|$. Each vertex in Y_i has at least one neighbor in Y_{i-1}, so it has at most $\Delta - 1$ neighbors in Y_{i+1}. Together, there are at most $\Delta(\Delta-1)^{i-1}$ vertices in Y_i, so they have at most $\Delta(\Delta-1)^i$ neighbors in Y_{i+1}. But every vertex in Y_{i+1} must be such a neighbor, so $|Y_{i+1}| \leq \Delta(\Delta-1)^i$, completing the induction step.

Once the induction is concluded, we have

$$\begin{aligned} |V(G)| &= |Y_0| + |Y_1| + |Y_2| + \cdots + |Y_D| \\ &= 1 + \underbrace{\Delta}_{|Y_1|} + \underbrace{\Delta(\Delta-1)}_{|Y_2|} + \underbrace{\Delta(\Delta-1)^2}_{|Y_3|} + \cdots + \underbrace{\Delta(\Delta-1)^{D-1}}_{|Y_D|}, \end{aligned}$$

which is exactly the bound we wanted. When $\Delta = 2$, the bound on $|Y_i|$ simplifies to 2 for every i, giving us $2D+1$ for the sum. When $\Delta > 2$, the formula $1 + \Delta\frac{(\Delta-1)^D-1}{\Delta-2}$ comes from the formula for the sum of a finite geometric series: $a + ar + ar^2 + \cdots + ar^n = a\frac{r^{n+1}-1}{r-1}$. □

The upper bound in Theorem 5.5 is called the Moore bound after Edward Moore (the same Moore that discovered the distance-finding algorithm discussed in Chapter 3). Moore also posed the problem of finding the graphs which achieve the bound exactly, and such graphs are known as Moore graphs. For example, Proposition 5.4 tells us that the Petersen graph is a Moore graph. In order for the inequalities of Theorem 5.5 to be equations, Moore graphs must all be regular graphs: the maximum degree Δ must actually be the degree of every vertex.

Moore graphs are very rare, apart from a few initial cases: they exist for all $\Delta \geq 2$ when $D = 1$, and for all $D \geq 1$ when $\Delta = 2$. I will leave it to you, in the practice problems, to understand these two constructions. Moore graphs were first studied by Alan Hoffman and Robert Singleton in 1960 [8], who found the Petersen graph for $(\Delta, D) = (3, 2)$ and a graph called the Hoffman–Singleton graph for $(\Delta, D) = (7, 2)$. What's more, they proved that when $D = 2$ or $D = 3$, and $\Delta \geq 3$, there are no more Moore graphs—with one possible exception. Hoffman and Singleton were unable to determine if there is a Moore graph with $\Delta = 57$ and $D = 2$; to this day, we do not know the answer!

Since 1960, the degree/diameter problem has advanced considerably; a 2013 survey by Mirka Miller and Jozef Širáň [11] summarizes much of the recent progress. We now know that, aside from the two graphs found by Hoffman and Singleton, and the possible $(\Delta, D) = (57, 2)$ case, there are no Moore graphs with $\Delta \geq 3$ and $D \geq 2$. In most cases, the best constructions we know are very far from the upper bound of Theorem 5.5. Many, but not all of them are regular graphs.

5.6 Practice problems

1. What are the possible values of the diameter of an 8-vertex graph? Give an example for each possible value.

2. If n and r are both odd, then an r-regular graph on n vertices does not exist, but a Harary graph $H_{n,r}$ still does. In this case, $H_{n,r}$ is a nearly-regular graph: it is a graph on n vertices where $n - 1$ of them have degree r, and one has degree $r + 1$.

   It is defined starting from the Harary graph $H_{n,r-1}$, or $\mathrm{Ci}_n(1, 2, \ldots, \frac{r-1}{2})$. To this graph, we add $r + 1$ edges that increase the degree of vertex 0 by 2, and the degree of vertices $1, 2, \ldots, n - 1$ by 1 each.

   How can we do this? Prove that your method works in general.

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 5.7, 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. Prove that for all $n \geq 5$, the complement of the path graph P_n has diameter 2.

6. a) Prove that the Kneser graph $K(n, k)$ is connected when $n > 2k$.

   b) Find and prove a similar condition that determines when $K(n, k)$ has diameter 2.

   c) Find and prove a similar condition that determines when $K(n, k)$ has no cycles of length 3 or 4.

7. a) For all $\Delta \geq 2$, there is a Moore graph with maximum degree Δ and diameter 1 (and $\Delta + 1$ vertices). What is it?

   b) For all $D \geq 1$, there is a Moore graph with maximum degree 2 and diameter D (and $2D + 1$ vertices). What is it?

8. Prove that, just like the Petersen graph, the Hoffman–Singleton graph has no cycles of length 3 or 4. (You do not know very much about the Hoffman–Singleton graph, but all you need to know for this problem is there in this chapter.)

9. Let $\Delta \geq 3$. Prove that if a graph G with maximum degree $\Delta \geq 2$ and diameter D is not regular, then it has at most

$$1 + (\Delta - 1) + (\Delta - 1)^2 + \cdots + (\Delta - 1)^{D-1} = \frac{(\Delta - 1)^D + 1}{\Delta - 2}$$

   vertices: approximately $\frac{\Delta - 1}{\Delta}$ of the Moore bound.

10. (BMO 1972) There are n persons present at a meeting. Every two persons are either friends of each other or strangers to each other. No two friends have a friend in common. Every two strangers have two and only two friends in common.

    a) Prove that each person has the same number of friends at the meeting.

    b) If that number is 5, find n.

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 as well as more boring. This leaves much less class time for discussing edge swaps and the ideas in the proof of the Havel–Hakimi theorem (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.[3] 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—

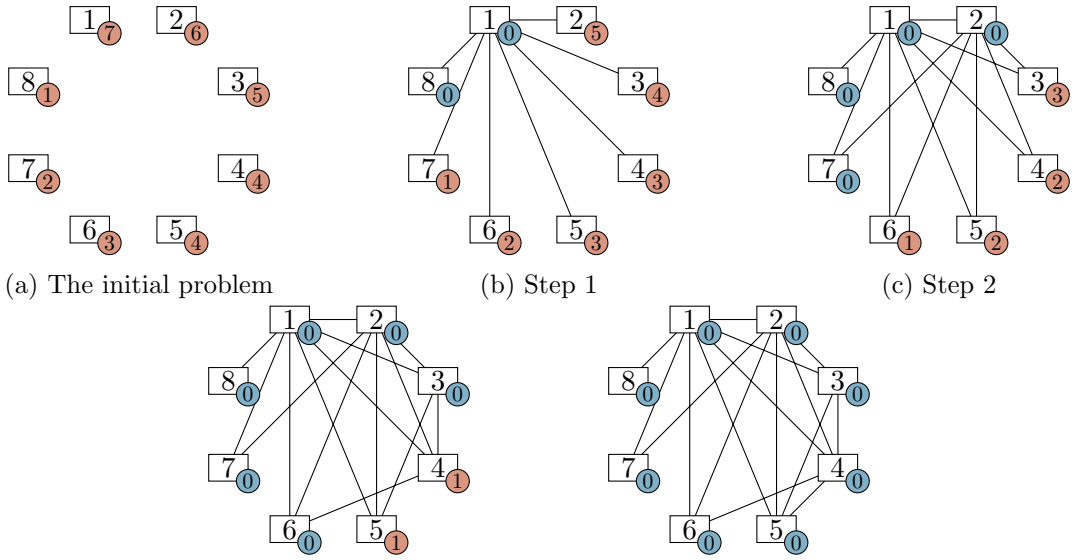
"—That can't be right!" you interrupt. "You're saying that the graph which represents who knew each other at the party had the degree sequence $99, 98, 97, \ldots, 2, 1, 0$? But that's not a graphic sequence!"

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.*

[3] This appears to be a very common story to tell. In West's *Introduction to Graph Theory* [14], a version of it appears as "The Handshake Problem", and another version was a problem in the 1985 British Mathematical Olympiad.



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

(d) Step 3 (e) Step 4

Figure 6.1: Reconstructing the 8-person party

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?

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 be odd, in violation of Corollary 4.2 to the handshake lemma. So we must duplicate the 6, the 4, or 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 way to think about it with the aid of a diagram. In Figure 6.1a, I have drawn vertices 1 through 8, each with an extra label: the degree that the vertex wants to have. As we work on filling in the graph, we will update these labels to show the remaining number of incident edges that the vertex wants, but doesn't have yet.

For one of these vertices, we can immediately assign the neighbors. Vertex 1 needs to be adjacent to every other vertex, so we can draw all of those edges, as shown in Figure 6.1b. This reduces the label on vertex 1 by 7 (we've given it 7 more incident edges), and the label on each other vertex by 1 (we've given each of them 1 incident edge). It's important to point out that vertex 1 and 8 now want 0 more edges, so they will not be used again.

We can continue. Vertex 2 needs 5 more edges, and there are only 5 other vertices that can accept edges, so it must be adjacent to all of them! We add all of those edges in Figure 6.1c. Updating the label on each vertex, we see that vertices 2 and 7 are also satisfied.

Two more steps solve the problem entirely. When we process vertex 3 in Figure 6.1c, we arrive at the diagram in Figure 6.1d. At that point, we only have two vertices left to deal with: vertices 4 and 5. 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. In Figure 6.1b, there are 6 vertices left to deal with (vertices 2 through 7), and they demand $5, 4, 3, 3, 2, 1$ additional edges respectively. This is a 6-vertex version of the same problem, which 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 must be an edge between the 2 vertices, and this already describes the graph uniquely. 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 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 which is most in danger: 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 $d_1, d_2, \ldots, d_n$. 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 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, I will give each vertex an extra label showing its demand.

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.

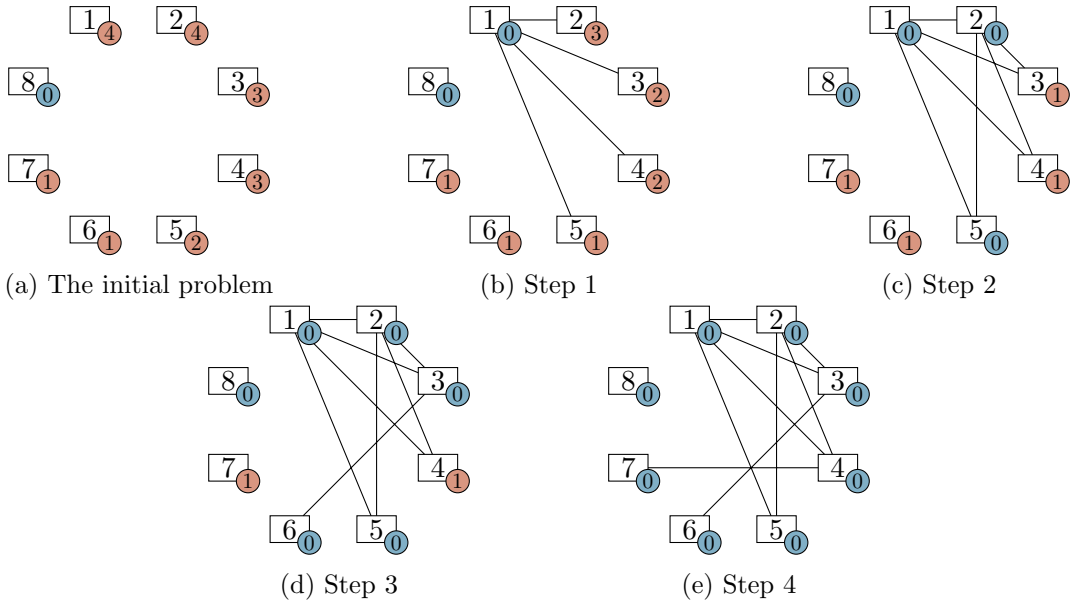


Figure 6.2: Finding a graph with degree sequence $4, 4, 3, 3, 2, 1, 1, 0$

Once no more active vertices remain, we stop.

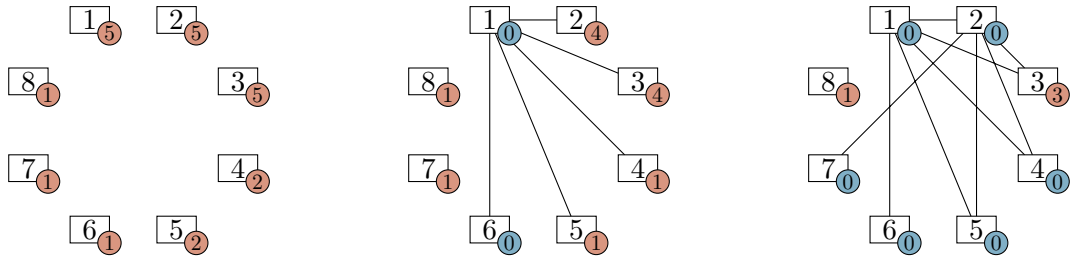
Question:	What is the maximum number of iterations we will need?
Answer:	Since at least one vertex becomes inactive at each step, we can be be certain to stop after n iterations (in an n-vertex graph), though we may stop sooner.

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 a graph with no edges in which the demand of each vertex is equal to the degree we want it to have, as shown in Figure 6.2a. (Vertex 8 starts out inactive; it does not affect the problem in any way.)

We begin one of the vertices with demand 4 (I choose 1, though it is also possible to choose 2) and add edges to the 4 other vertices with the highest demands (vertices 2, 3, 4, and 5). The result is Figure 6.2b, with one fewer active vertex.

Our next choice is unique: vertex 2, with demand 3. The vertices that will become its remaining neighbors must include vertices 3 and 4 (with demand 2), as well as one of the vertices with demand 1; we arbitrarily choose it to be vertex 5. 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. In Figure 6.2d, we add edge 36, leaving only two active vertices, and in Figure 6.2e, we add edge 47 and finish the problem. We can check our answer by checking that the label on each vertex in Figure 6.2a is equal to the degree it has in Figure 6.2e.

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 vertex 1 (we could also have picked vertex 2 or 3, since all of them have demand 5) and join it to vertices 2 through 6 (the 5 vertices with the highest demands). Vertices 1 and 6 become inactive at this step.

Next, we pick one of the vertices with demand 4: say, vertex 2. We draw 4 edges, one of which must go to 3, and the rest must 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! In Figure 6.3c, vertex 3 has demand 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. A critic might complain that 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. But 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 be too tedious. 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 edges that we guessed. 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

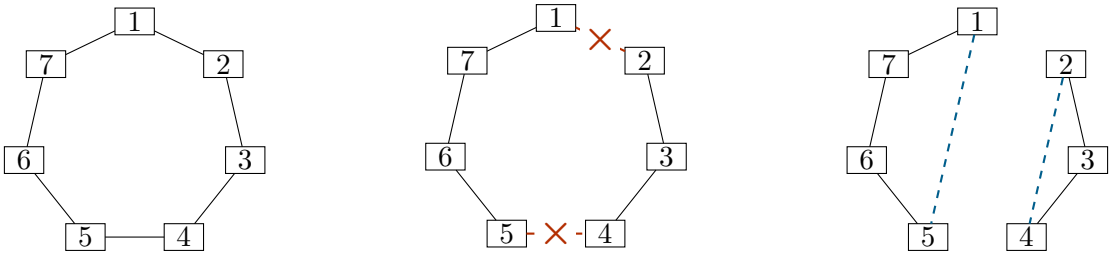
The result we need to prove is known as the Havel–Hakimi theorem: named after Václav Havel, who proved it in 1955 [7], and S. L. Hakimi, who rediscovered the corresponding algorithm for testing if a sequence is graphic in 1962 [5]. Before we prove it, we will restate it in a different way, closer to how it was presented by Havel.

Theorem 6.2 (Havel–Hakimi theorem)**.** *Let G be a graph with vertex set $V(G) = \{1, 2, \ldots, n\}$ such that $\deg(1) \geq \deg(2) \geq \cdots \geq \deg(n)$. Let $k = \deg(1)$ and let $S = \{2, 3, \ldots, k+1\}$.*

Then there is a graph H with $V(G) = V(H)$ and with $\deg_G(x) = \deg_H(x)$ for all x such that the neighbors of vertex 1 in H are precisely the vertices of S.

Question:	How does the Havel–Hakimi theorem justify the correctness of our algorithm?
Answer:	In the first step, vertex 1 (up to tiebreaks) is exactly the vertex we will process, and the edges from 1 to S are exactly the edges we add. The Havel–Hakimi theorem 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 our algorithm, there are never any edges between two active vertices. So if we only consider the subgraph induced by the active vertices (with the leftover demands), then every step is exactly like the first.



(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 the Havel–Hakimi theorem the way we did 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 edges 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*. To perform one in a graph G, we first find four vertices vw and xy such that vw and xy are edges of G, but vx and wy are not. Then, we delete the edges vw and xy that did exist, and replace them by the edges vx and wy that previously did not exist. Figure 6.4 shows an example of an edge swap. You can see how 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." (I have sneakily reduced the number of hypotheses, eliminating ones we don't really need. vertex x in Lemma 6.3 is vertex 1 in the Havel–Hakimi theorem, which has the highest degree in G, but we will not to assume this.)

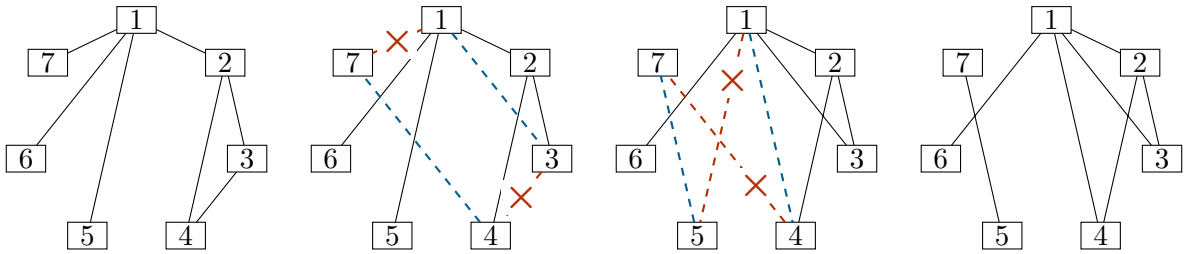
Lemma 6.3. *Let x be a vertex in a graph G and let S be a set of at most $\deg(x)$ vertices in G not equal to x with the highest degrees, breaking ties arbitrarily.*

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

Proof. Let $y \in S$ be a vertex not adjacent to x. Since $\deg(x) \geq |S|$, but x is not adjacent to every vertex in S, it must also have neighbors outside S; call one such neighbor z. To summarize, xy is not an edge of G, but we wish it were; xz is an edge of G, but we don't want it.

We want to fix this state of affairs with an edge swap, but that needs a fourth vertex.

Question:	What kind of fourth vertex w lets us perform an edge swap that delete edge xz and add edge xy?
Answer:	At the same time, we must also delete edge yw and add edge zw. So w must be adjacent to y, but not to z.



(a) x has one edge to S (b) First edge swap (c) Second edge swap (d) Final result

Figure 6.5: Two iterations of Lemma 6.3 in an example; $x = 1$ and $S = \{2, 3, 4\}$

(It does not matter whether $w \in S$ or not, and it does not matter whether w is adjacent to x or not.)

This is where our choice of S becomes important. We chose the vertices of S to have the highest degrees. Even if we broke some ties to do it, we know that the degree of every vertex in S is greater than or equal to the degree of every vertex not in S, other than x. In particular, $\deg(y) \geq \deg(z)$.

On the other hand, we already have an example of a vertex adjacent to z, but not y: vertex x. Before we look at any other vertices, z is in the lead. To reach $\deg(y) \geq \deg(z)$, y must counter that lead at some point: it must have a neighbor z does not. Such a neighbor is exactly the vertex w we wanted.

By performing the edge swap which deletes edges xz and yw and adds edges xy and zw, We achieve the result we want: x is adjacent to one more vertex of S. □

Figure 6.5 shows two examples of using the edge swap given by Lemma 6.3. Throughout, we want vertex $x = 1$ to become adjacent to all three vertices of $S = \{2, 3, 4\}$. (In fact, $\deg(x) = 4$, which is bigger than $|S|$. In the proof of the Havel–Hakimi theorem, we apply Lemma 6.3 to a case where $\deg(x) = |S|$, but only the inequality $\deg(x) \geq |S|$ is necessary.)

1. In Figure 6.5a, x has only one neighbor in S: vertex 2. We take $y = 3$ (a neighbor x wants to have in S, but doesn't) and $z = 7$ (a neighbor x has, but doesn't want). We look for a third vertex adjacent to y, but not z, and find vertex $w = 4$ (vertex 2 would also have worked). Then, we swap edges 17 and 34 for edges 13 and 47, as shown in Figure 6.5b.
2. After we do this, x only has two neighbors in S: vertices 2 and 3. It is still not adjacent to vertex 4, so we take $y = 4$; we take $z = 5$ to be the neighbor that x is going to lose. The two vertices adjacent to $y = 4$ but not $z = 5$ are 2 and 7; let's take $w = 7$. Then, we swap edges 15 and 47 for edges 14 and 57, as shown in Figure 6.5c.
3. In the final result, shown in Figure 6.5d, x is adjacent to every vertex of S.

I put Lemma 6.3 first because from here, the proof of the Havel–Hakimi theorem 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 1 in S.

Then actually, all vertices of S must be adjacent to 1. If not, we could use Lemma 6.3 (taking $x = 1$) to increase the number of neighbors of 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 the Havel–Hakimi theorem, 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.

By applying Lemma 6.3 repeatedly, 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 an arbitrary graph which has that degree sequence, then perform many edge swaps at random, getting a random graph with this degree sequence.[4] We can approximately answer questions like "are most graphs with this degree sequence connected" 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.

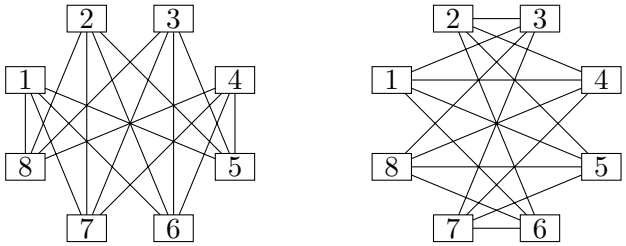
6.6 Practice problems

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.

[4] 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.

2. 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 the two graphs are isomorphic, 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 $\{3, 4, 7, 8\}$.



   Find a sequence of edge swaps to turn G into H. (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.

6. There is another proof of Theorem 6.4 that is closer to our argument for the Havel–Hakimi theorem.

   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. Under what conditions on a, b, and n is a sequence of the form

$$\underbrace{a, a, \ldots, a}_{n}, \underbrace{b, b, \ldots, b}_{n}$$

   a graphic sequence?

8. 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:

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

(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 Pál Erdős and Tibor Gallai [3] 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.

When it comes to directed graphs, introduced in the second half of the chapter, the situation is a bit different. Although some of the theory is the same as for undirected graphs, by default, results don't carry over. Accordingly, I will discuss directed graphs in a few big chunks, outside this chapter: all of Chapter 12 and Chapter 28 deal with directed graphs, and one section is devoted to them in Chapter 8 and in Chapter 20.

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.

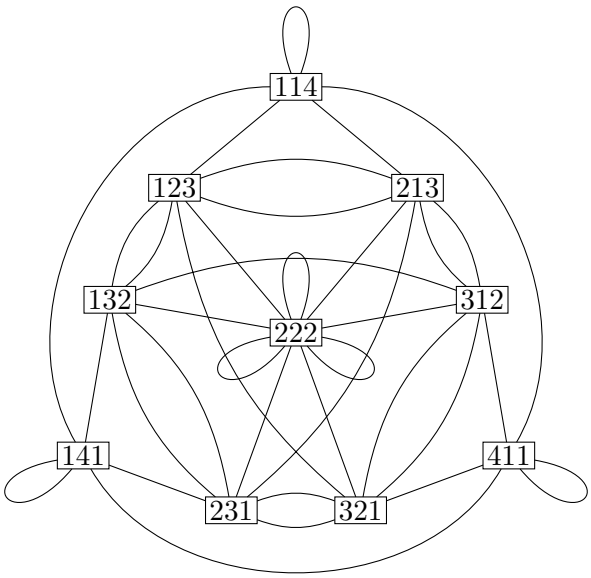


Figure 7.1: A multigraph representation of the 6-pebble game

7.1 Multigraphs

Consider a game[5] 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.

[5] 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 a symmetric **incidence relation** between them: we say that vertex x is **incident** to edge e, or e is incident to x, if they are related by the incidence relation.*

*In a multigraph, two vertices are called **adjacent** if there is an edge incident to both.*

Previously, we defined edges 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 need to give them individual labels.

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 to vertex 114, two parallel edges joining vertices 123 and 213, and three parallel loops incident to vertex 222.

There are relationships between graphs and multigraphs in both directions:

Definition 7.3. *A **simple graph** is a graph. The **simplification** of a multigraph G is the simple graph with the same vertex set, and an edge $\{x, y\}$ (for $x, y \in V(G)$ with $x \neq y$) whenever G has at least one edge incident to both x and y. Effectively, this removes all loops, and replaces parallel edges with a single edge.*

When a multigraph has no loops or parallel edges, we will treat it as the same object as its simplification; even though on a low level they are defined differently, reasoning about them in practice yields identical outcomes.

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 7.4. *An* $x - y$ *walk of length* l *in a multigraph* G *is a sequence*

$$(x_0, e_1, x_1, e_2, x_2, \ldots, x_{l-1}, e_l, x_l)$$

that alternates between vertices and edges of G*, satisfying the following:* $x = x_0$*,* $y = x_l$*, and for each* $i = 1, \ldots, l$*, 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 and cycles have the same definition, but with new implications. Previously, we were not able to define cycle graphs C_1 or C_2; this is possible in the world of multigraphs. The multigraph C_1 is a single vertex with a loop; the multigraph C_2 has two vertices and two edges between them.

In all cases, to pick out a path and cycle in a multigraph, we need to specify not only the vertices and the order in which they occur, but also the specific edges we use between those vertices. The upshot is that paths and cycles can still be represented by walks (closed walks, in the case of a cycle), but they must be the multigraph walks defined in Definition 7.4.

If there's any bright side to this, it is that it's easier to tell when a closed walk represents a cycle. This is true of a closed walk

$$(x_0, e_1, x_1, e_2, x_2, \ldots, x_{l-1}, e_l, x_0)$$

as long as $e_1, e_2 \ldots, e_l$ are distinct, vertices $x_0, x_1, \ldots, x_{l-1}$ are distinct, and $l \geq 1$.

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 to 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 to 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? In other words, counting vertices adjacent to x yields the degree in the simplification of the multigraph.

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 (Lemma 4.1), 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 vertex were defined to be the number of incident edges, then a loop would only increase the degree of a single vertex by 1.

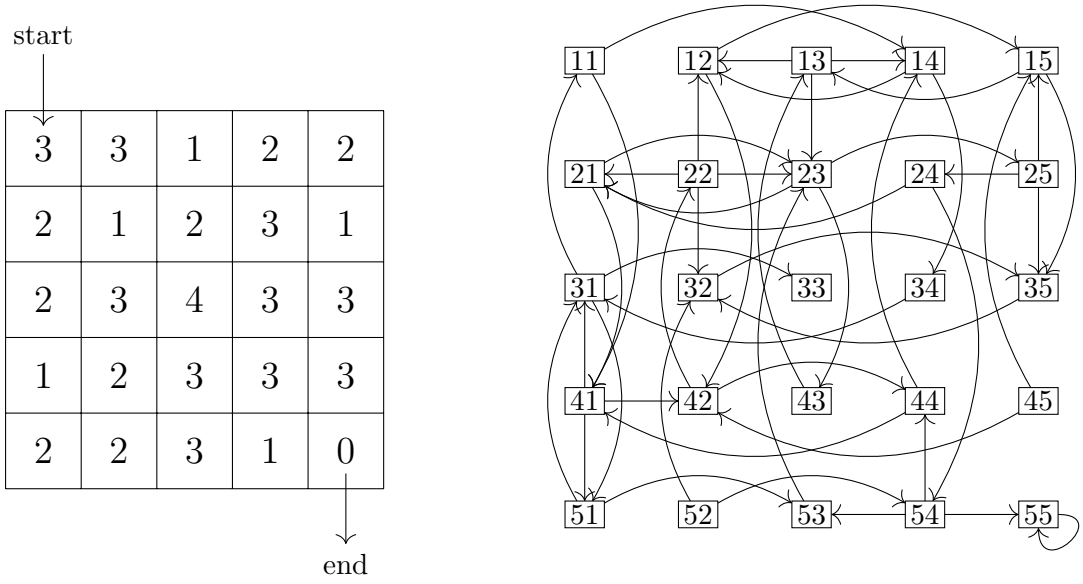
The handshake lemma is so valuable that we are willing to add a special case to our definition, just for this purpose. To rescue the lemma, we need the degree sum to increase by 2 when a loop is added to a graph. It wouldn't make any sense for a loop incident to a vertex x to increase the degree of any 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 to x, counting every loop incident to 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:



(a) A numerical maze

(b) A digraph representation of the maze

Figure 7.2: Solving a maze using directed graphs

- 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 $\varphi(y)$.
- 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 to e if and only if $\varphi(x)$ is incident to $\varphi'(e)$: 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're 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 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**.*

In principle, we could combine this definition with the definition of a multigraph in the previous section, obtaining the concept of a directed multigraph. I will not do so in this book; juggling all these definitions is complicated enough as it is! 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, because 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.

It is sometimes convenient to forget about the directions of the arcs, and go back to an undirected object.

Definition 7.7. *The **underlying graph** of a digraph D is the multigraph G with vertex set $V(G) = V(D)$ and an edge incident to x and y for each arc $(x, y) \in E(D)$. The digraph D is called an orientation of G.*

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 7.8. *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 induction hypothesis to $D - (x, y)$, a 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 the induction 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 the degree of x in the underlying graph.

Question:	What do the net degrees add up to in a directed graph?
Answer:	We can write the sum of net degrees as $$\sum_{v \in V(D)} \deg_D^{\pm}(v) = \sum_{v \in V(D)} \deg^+(v) - \sum_{v \in V(D)} \deg^-(v),$$ 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 underlying graph of D.

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 7.9. *An* $x - y$ ***walk of length*** l *in a digraph* D *is a sequence*

$$(x_0, x_1, x_2, \ldots, x_l)$$

of vertices of D *where* $x = x_0$*,* $y = x_l$*, and for each* $i = 1, \ldots, l$*,* (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 directed walk, and this small difference is a dramatic change. While an $x - y$ walk in an undirected graph is essentially the same as an $x - y$ walk except for which way you go, a directed graph might have an $x - y$ walk but no $y - x$ walk.

By analogy with P_n and C_n, there are two families of directed graphs: the *directed path graph* $\overrightarrow{P_n}$ has vertex set $\{1, 2, \ldots, n\}$ with an arc $(i, i+1)$ for $i = 1, 2, \ldots, n-1$, and the *directed cycle graph* $\overrightarrow{C_n}$ is obtained from $\overrightarrow{P_n}$ by adding the arc $(n, 1)$. Both of these are defined for all $n \geq 1$.

When dealing with directed graphs, the terms *path* and *cycle* refer to copies of the directed path graph and the directed cycle graph, respectively; I may use the terms "directed path", "directed cycle", or "directed walk" for emphasis. A directed path or cycle is represented by a walk $(x_0, x_1, x_2, \ldots, x_l)$ if it has vertices $\{x_0, x_1, \ldots, x_l\}$ and arcs $\{(x_0, x_1), \ldots, (x_{l-1}, x_l)\}$. In the case of a cycle, it follows that the walk representing it is closed, with $x_l = x_0$.

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 cycle.*

Proof. Let the walk $(x_0, x_1, x_2, \ldots, x_l)$ represent a path in D of the largest possible length. Because D has no sinks, we know in particular that $\deg^+(x_l) > 0$, so there must exist some arc (x_l, y) that starts at x_l.

If $y \notin \{x_0, x_1, x_2, \ldots, x_l\}$, then the walk $(x_0, x_1, x_2, \ldots, x_l, y)$ would represent a longer path, contrary to our assumption. Therefore $y = x_i$ for some i, and the closed walk $(x_i, x_{i+1}, \ldots, x_l, x_i)$ represents the 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_l.

Question:	Does the converse hold? That is, if D has a 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 cycles: for example, the length-3 cycle represented by $(12, 15, 13, 12)$.

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 with no 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 cycle?

7. The game RPS-7 [10], invented by David C. Lovelace, is an extension of Rock-Paper-Scissors to 7 gestures: Rock, Paper, Scissors, Air, Fire, Sponge, and Water. The gestures have the following relationships:

   - Rock pounds out Fire, crushes Scissors, and crushes Sponge.
   - Fire melts Scissors, burns Paper, and burns Sponge.
   - Scissors swish through Air, cut Paper, and cut Sponge.
   - Sponge soaks paper, uses Air pockets, and absorbs Water.
   - Paper fans Air, covers Rock, and floats on Water.
   - Air blows out Fire, erodes Rock, and evaporates Water.
   - Water erodes Rock, puts out Fire, and rusts Scissors.

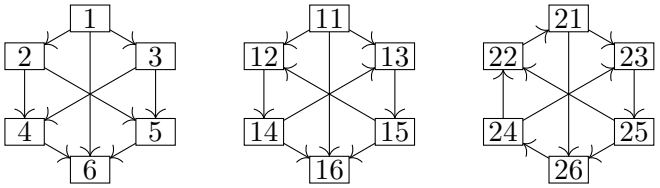
   Let RPS_7 be the graph whose vertices are the 7 gestures, with an arc (x, y) whenever gesture x beats gesture y.

   a) Find a cycle of length k in RPS_7 for each $k = 3, 4, 5, 6, 7$.

   b) For which n can an RPS-n game be constructed so that each object beats the same number of other objects?

8. a) Write a reasonable definition of an isomorphism between two directed graphs.

   b) Here are three directed graphs with isomorphic underlying graphs.



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

8 Euler tours

The purpose of this chapter

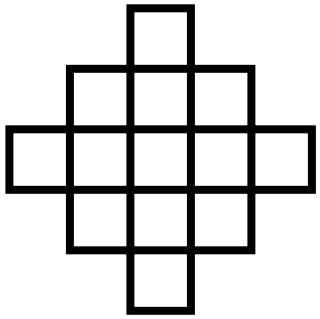
This chapter concludes the part of this book devoted to vertex degrees. It is an application of the concept of vertex degrees, as well as some results from Chapter 4, to solve a problem that could be said to be the beginning of graph theory.

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 17. 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.

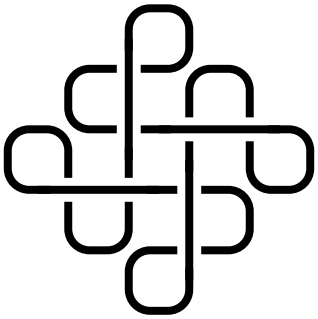
Everything we say about the Euler tour problem for graphs remains true for multigraphs, with no substantial change. For directed graphs, the situation is a bit more complicated, and it is discussed at the end of the chapter. However, many of the important applications of Euler tours, such as the one in Chapter 20, are applications to directed graphs, so this is an important case of the problem to understand.

8.1 Don't lift your pencil!

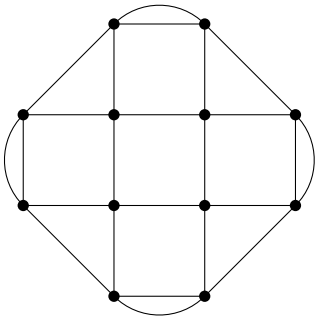
You might have seen a brainteaser like this before! Look at the diagram in Figure 8.1a. Can you draw this shape on paper without lifting your pencil (or pen), and without going over the same line more than once?



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

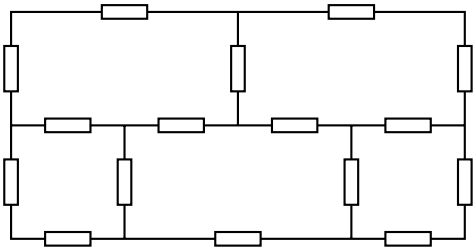


(b) Solution to the challenge in Figure 8.1a

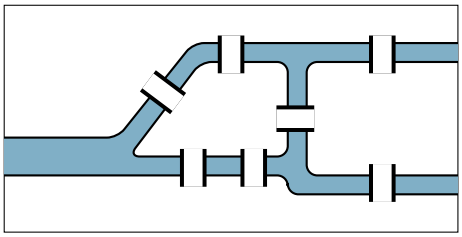


(c) Modeling the puzzle as a multigraph

Figure 8.1: Drawing without lifting your pencil



(a) The five rooms puzzle



(b) The bridges of Königsberg

Figure 8.2: Two more puzzles equivalent to pencil drawing

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.

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 well-known puzzles that are an instance of the same type of problem. One 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: one for each room, where we consider the outside region to be a room as well. Each door between two rooms should be an edge incident to both rooms.

Another such puzzle, the problem of the seven bridges of Königsberg, has historical significance. It was first studied by Leonhard Euler, in 1736 [2], and it may be one of the first mathematical problems 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 Königsberg, 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 Königsberg?
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 depend on the degree of the vertices. To simplify terminology just for this chapter, we will call a vertex *odd* or *even* if it has odd degree or even degree, respectively.

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. I will give the proof in the language of multigraphs, so that we can be sure it works even then.

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

Proof. Let G be an Eulerian multigraph, and let the walk

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

with $x_m = x_0$ 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 to 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 $\{e_{i+1} : i \in I\}$ are all the edges incident to 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, $x = x_0$ as well, 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 really 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[6] 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 lets us reduce 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[7] and consider cycle decompositions.

In general, a decomposition of G is a set of subgraphs $\{G_1, G_2, \ldots, G_k\}$ such that every edge of G is an edge of exactly one subgraph G_i. We have already seen one kind of decomposition: the decomposition of a graph into its connected components. We will encounter two other examples of decompositions later: 1-factorizations in Chapter 16 and ear decompositions in Chapter 25.

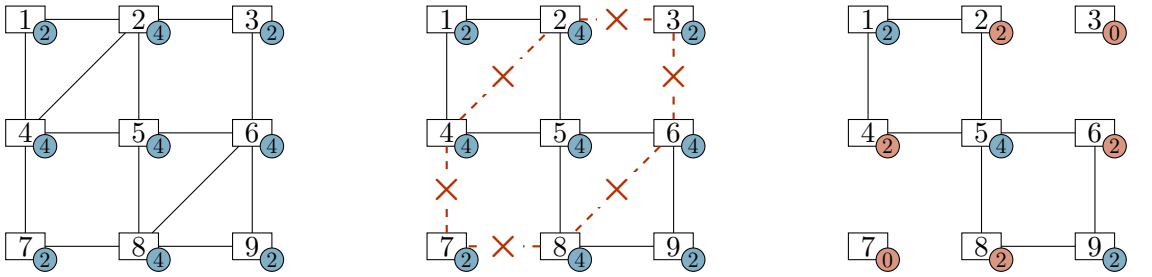
In particular, a *cycle decomposition* of G is a set of cycles in G such that every edge of G is an edge of exactly one of the cycles. This uses the terminology in a straightforward way, but one thing is awkward: the notation. We cannot write a cycle decomposition $\{C_1, C_2, \ldots, C_k\}$, because the notation C_n already refers to the n-vertex cycle graph. When we must, we resort to a standard option all mathematicians take when not allowed to use subscripts: use superscripts instead, writing the cycle decomposition as $\{C^{(1)}, C^{(2)}, \ldots, C^{(k)}\}$.

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 to that vertex. Therefore the necessary condition for Eulerian graphs is also necessary for having a cycle decomposition. However, Veblen was able to prove [13] that for cycle decompositions, this condition is both necessary and sufficient!

A key step of this proof uses Theorem 4.4: if a graph has minimum degree at least 2, then it contains a cycle. 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 can be carried out in multigraphs as well.

[6] Or multigraph: even if G is a simple graph, it might already have an edge xy, so we need to switch to a multigraph model to add edge e.

[7] Heh-heh.



(a) The initial graph G (b) Deleting a cycle C (c) Vertex degrees in $G - C$

Figure 8.3: Finding a cycle decomposition, one cycle at a time

Question:	Alternatively, there is a way to generalize the theorem to multigraphs without looking at the proof of Theorem 4.4 at all. How?
Answer:	If a multigraph has a loop, it has a cycle of length 1, and if it has two parallel edges, it has a cycle of length 2. In both cases, we already get the conclusion we want. If the multigraph has neither of these, then we can think of it as a simple graph.

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

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

Cycles are 2-regular graphs, so for every cycle $C \in \mathcal{D}$ with $x \in V(C)$, $\deg_C(x) = 2$. Together, the cycles in $\mathcal{D}$ include each edge of G exactly once, so we can split up the degree $\deg_G(x)$ as a sum over the degree of x on cycles. Let $\mathcal{D}_x$ be the set of all cycles in $\mathcal{D}$ containing x: then

$$\deg_G(x) = \sum_{C \in \mathcal{D}_x} \deg_C(x) = \sum_{C \in \mathcal{D}_x} 2 = 2|\mathcal{D}_x|.$$

In particular, $\deg_G(x)$ must be even.

This argument shows that the condition is necessary; now, we must prove that it is sufficient. So let G be a graph 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 connected 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 at least 2. By Theorem 4.4, this component contains a cycle C.

In $G - E(C)$ (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. (An example of such a deletion is illustrated in Figure 8.3.) Also, $G - E(C)$ has fewer than m edges (since C has at least one edge), so our induction hypothesis

applies to it: $G - E(C)$ 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 gives us an algorithm for finding a cycle decomposition. Simply find any cycle, set it aside, and repeat with the remainder of the graph 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 graph, 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

Before we go on, let me clear the air about one thing: the condition that all vertices are even is not sufficient to guarantee that a graph is Eulerian. However, the exception is rather silly.

Question:	How can you find a graph where all vertices are even, but which does not have an Euler tour? (Small examples include a multigraph with 2 edges and a simple graph with 6 edges.)
Answer:	Just make sure that your graph has multiple connected components with edges in them! A walk can never jump from one component to another, so you can never include all edges of the graph 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 graph 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 graph which has only one connected component aside from isolated vertices, and in which every vertex of G is even. For simplicity, delete all the isolated vertices (which will not affect our argument) so that we can simply assume G is connected. (If G has only isolated vertices, then a walk of length 0 is an Euler tour.) The first thing we'll do is use Theorem 8.3 to find a cycle decomposition $\mathcal{D}$ of G.

Next, we'll build an Euler tour step by step. This is a proof by algorithm. Throughout this process, we will update the following objects:

- a "partial tour" PT, which is a closed walk that uses each edge at most once, but might miss some edges.
- A subgraph H whose edges are exactly the edges missed by PT.
- A set $\mathcal{U}$ of "unprocessed cycles": a cycle decomposition of H.

Initially, we'll take PT to be a walk representing one cycle $C \in \mathcal{D}$. Let $H = G - E(C)$, and let $\mathcal{U} = \mathcal{D} - \{C\}$. 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! Assuming otherwise, let S be the set of vertices visited by PT: S is not empty, but also does not contain any vertices of the cycles in $\mathcal{U}$. Every edge of G is either used by PT (and joins two vertices in S) or is an edge of a cycle in $\mathcal{U}$ (and joins two vertices not in S). No edge of G has exactly one endpoint in S, and by Lemma 3.2, G is not connected, which is a contradiction.

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

$$(x_0, x_1, x_2, \ldots, x_{l-1}, x_0)$$

and let $C \in \mathcal{U}$ be a cycle containing a vertex x_i. Choose a closed walk

$$(y_0, y_1, y_2, \ldots, y_{k-1}, y_0)$$

to represent C with the property that $y_0 = x_i$. Then to incorporate C into PT, replace PT by the closed walk

$$(x_0, x_1, x_2, \ldots, x_{i-1}, \underbrace{y_0, y_1, y_2, \ldots, y_{k-1}, y_0}, x_{i+1}, \ldots, x_{l-1}, x_0).$$

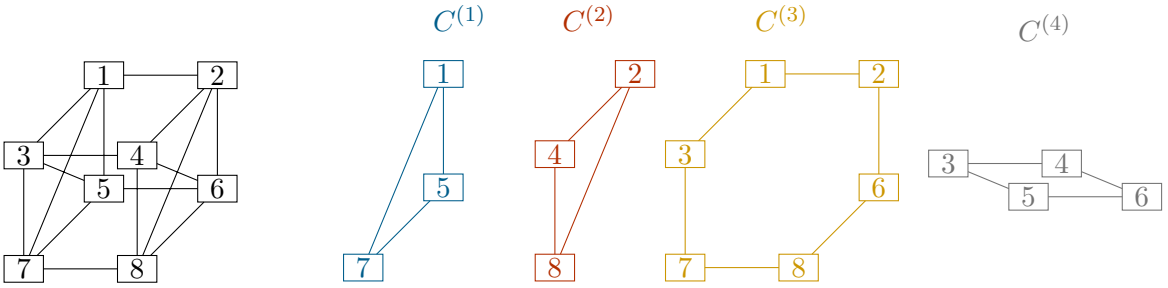
That is, the vertex x_i is replaced by the entire closed walk representing C. Replace H by $H - E(C)$, and $\mathcal{U}$ by $\mathcal{U} - \{C\}$.

After this process, every edge used by the old PT is still used by the new PT, and so is every edge in $E(C)$: after all, $E(C) = \{y_0y_1, y_1y_2, \ldots, y_{k-1}y - 0\}$, and all of these appear as pairs of consecutive vertices in the new PT. Therefore H is still the subgraph whose edges are exactly the edges missed by PT, and $\mathcal{U}$ is still a cycle decomposition of H. Also, it's still true that the new PT uses every edge at most once: all of the new edges came from H, so they did not appear in the old 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 8.4a. We will begin with a cycle decomposition. The one shown in Figure 8.4b 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.

1. Initially, our partial tour PT is a walk $(1, 5, 7, 1)$ representing $C^{(1)}$.



(a) A modified cube graph

(b) A cycle decomposition $\{C^{(1)}, C^{(2)}, C^{(3)}, C^{(4)}\}$

Figure 8.4: A graph and its cycle decomposition, in preparation for finding an Euler tour

2. Next, we want to add a cycle that shares a vertex with PT: that is, we can't use $C^{(2)}$. Let's take $C^{(4)}$ instead; $C^{(3)}$ would also work.

   The only vertex of $C^{(4)}$ appearing on PT is vertex 5. So we choose to represent $C^{(4)}$ by the closed walk $(5, 3, 4, 6, 5)$, and replace the 5 in PT by this closed walk: now,

   $$PT = (1, \underbrace{5, 3, 4, 6, 5}, 7, 1).$$

3. At this point, PT has lots of vertices, so we can add either remaining cycle. Let's add $C^{(2)}$. It shares vertex 4 with PT, so we represent $C^{(2)}$ by $(4, 2, 8, 4)$ and splice it into PT: now,

   $$PT = (1, 5, 3, \underbrace{4, 2, 8, 4}, 6, 5, 7, 1).$$

4. Finally, only $C^{(3)}$ is left to add; it shares many vertices with PT, but in particular, it shares vertex 1, so we choose to represent it by $(1, 2, 6, 8, 7, 3, 1)$. We replace the first occurrence of 1 in PT by this closed walk, getting the final result:

   $$PT = (\underbrace{1, 2, 6, 8, 7, 3, 1}, 5, 3, 4, 2, 8, 4, 6, 5, 7, 1).$$

To check that PT is an Euler tour for yourself, draw all the vertices of the graph in Figure 8.4a, with no edges between them. Then, follow PT, 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 8.4a.

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, as well as a test for their existence. The only thing that changes is that a graph with two odd vertices can still have an Euler walk.

Question:	In a graph 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 components by itself, we'd see a subgraph with only one odd vertex. However, by Corollary 4.2 to the handshake lemma, the number of odd vertices in any graph is even.

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:	Instead of a partial tour PT, begin with a partial walk PW representing an $x - y$ path P, and let $H = G - E(P)$. In H, all vertices are even, so we can take $\mathcal{U}$ to be a cycle decomposition of H. From here, the algorithm can be continued 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; we proved it in Chapter 7. 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 $\mathcal{D}$, and let x be an arbitrary vertex of D.

In the directed cycle graph $\overrightarrow{C_n}$, every vertex has indegree 1 and outdegree 1. Together, the cycles in $\mathcal{D}$ include each arc of D exactly once, so we can split up the indegree and outdegree

of x as a sum over cycles. Let $\mathcal{D}_x$ be the set of all cycles in $\mathcal{D}$ containing x: then

$$\deg_D^+(x) = \sum_{C \in \mathcal{D}_x} \deg_C^+(x) = \sum_{C \in \mathcal{D}_x} 1 = |\mathcal{D}_x|, \text{ and}$$
$$\deg_D^-(x) = \sum_{C \in \mathcal{D}_x} \deg_C^-(x) = \sum_{C \in \mathcal{D}_x} 1 = |\mathcal{D}_x|.$$

In particular, $\deg_D^+(x) = \deg_D^-(x)$: x is balanced.

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

Suppose D has $m > 0$ arcs; the same is true if we pass to D' by deleting all isolated vertices of D. If a balanced vertex has outdegree 0, it also has indegree 0, so it is an isolated vertex; therefore for all $x \in V(D')$, $\deg^+(x) \geq 1$. By Theorem 7.2, D' contains a cycle C.

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

Starting from a cycle decomposition, the algorithm for finding an Euler tour is the same as it was for undirected graphs. However, there is a further difficulty with stating the final result. We do not know exactly which graphs it applies to, because do not yet know what it means for a directed graph to be connected.

I will give you the statement first, and then we will determine what it means. This is the proper way to go about it: we figure out what we should prove by figuring out the assumptions we need to use.

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.

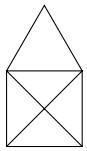
Question:	Looking at the proof of Theorem 8.4, where did we need the graph to be connected?
Answer:	We needed it to be impossible for the partial tour PT and the unused cycles $\mathcal{U}$ to have no vertices in common.

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: 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 the underlying graph is connected. 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:

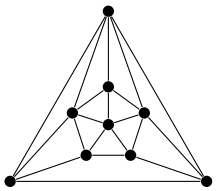


   In order to draw this picture without lifting your pencil or retracing any line, where must you start and end? (Of course, you can always swap the start and end.)

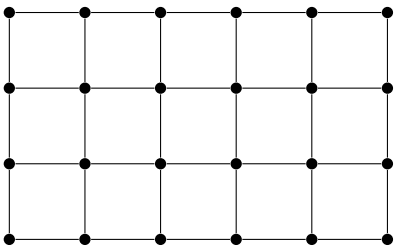
2. Find a cycle decomposition of the circulant graph $\mathrm{Ci}_{10}(1,2)$, and use it to find an Euler tour of $\mathrm{Ci}_{10}(1,2)$.

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

4. Let G be the graph below. Find a vertex x such that $G - x$ is Eulerian.



5. 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.



6. 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.

7. Suppose that the complete graph K_n, where $95 < n < 105$, has a cycle decomposition in which every cycle has length 5. What must the value of n be?

8. Prove that every graph has a "double tour": a closed walk which uses every edge exactly two times, once in each direction. (That is, for every edge xy, the walk contains one segment $\ldots, x, y, \ldots$ and one segment $\ldots, y, x, \ldots$.)

9. This exercise presents part of a method due to Noga Alon [1] for finding perfect matchings (which we will cover in detail starting in Chapter 13) using Euler tours.

   a) Let G be a *bipartite graph*: that is, $V(G)$ is the union of two sets A and 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 graph in which every vertex is even and (as in part (a)) the number of edges is even. Prove that G has a spanning subgraph H such that for every vertex x, $\deg_H(x) = \frac{1}{2}\deg_G(x)$.

   c) Prove by induction on k that for all $k \geq 0$, every 2^k-regular bipartite graph has a 1-regular spanning subgraph.

10. (BMO 1977) Prove that for each integer $n > 1$ it is possible to construct a necklace having $2n^2$ beads in all, these being of $2n$ different colors, in such a way that for each pair of different colors there is at least one pair of adjacent beads of these two colors. Is it possible to do the same using $2n^2 - 1$ beads in all? Give a reason for your answer.

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] W. W. Rouse Ball. *Mathematical Recreations and Essays*. Fourth. New York, NY: Macmillan and Company, 1905. URL: https://www.gutenberg.org/ebooks/26839.

[3] 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.

[4] Robert Frucht. “Graphs of Degree Three with a Given Abstract Group”. In: *Canadian Journal of Mathematics* 1 (4 1949), pp. 365–378. DOI: 10.4153/CJM-1949-033-6.

[5] 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.

[6] Frank Harary. “The maximum connectivity of a graph”. In: *Proceedings of the National Academy of Sciences of the United States of America* 48 (1962), pp. 1142–1146. DOI: 10.1073/pnas.48.7.1142.

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

[8] Alan Hoffman and Robert Singleton. “On Moore graphs with diameters 2 and 3”. In: *IBM Journal of Research and Development* 4 (5 1960), pp. 497–504. DOI: 10.1147/rd.45.0497.

[9] D.A. Holton and J. Sheehan. *The Petersen Graph*. Australian Mathematical Society Lecture Series. Cambridge University Press, 1993. ISBN: 9780521435949.

[10] David C. Lovelace. *RPS-7*. 2003. URL: https://umop.com/rps7.htm (visited on 12/04/2025).

[11] Mirka Miller and Jozef Širáň. “Moore graphs and beyond: a survey of the degree/diameter problem”. In: *Electronic Journal of Combinatorics* (DS14 2013). DOI: 10.37236/35.

[12] N.J.A. Sloane. *Sequence A002851 in the On-Line Encyclopedia of Integer Sequences*. URL: https://oeis.org/A002851 (visited on 12/04/2025).

[13] 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.

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