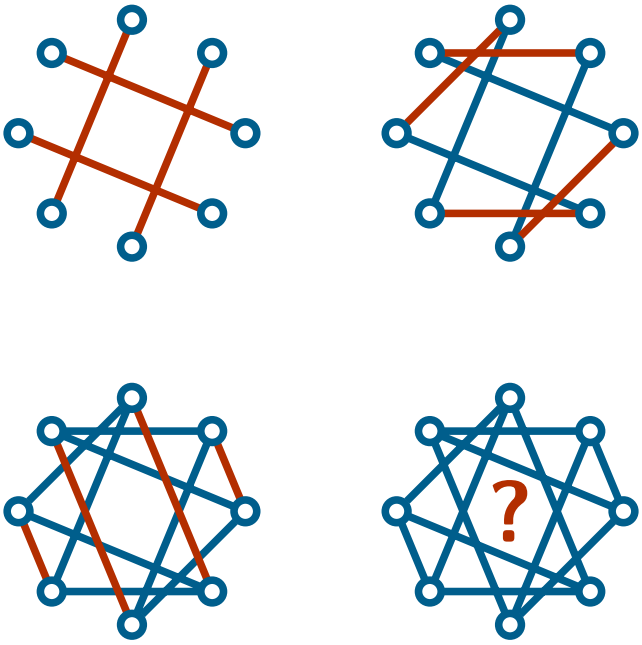
Mikhail Lavrov

Start Doing Graph Theory

Part IV: Matchings



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

13 Bipartite matching 4
- 13.1 Two chess puzzles . . . 4
- 13.2 Bipartite graphs . . . 6
- 13.3 Odd cycles . . . 9
- 13.4 Matching problems . . . 11
- 13.5 Maximum and maximal . . . 12
- 13.6 Vertex covers . . . 13
- 13.7 Practice problems . . . 16

14 Kőnig's theorem 18
- 14.1 Kőnig's theorem . . . 18
- 14.2 Augmenting paths . . . 19
- 14.3 The augmenting path algorithm . . . 22
- 14.4 Analyzing the algorithm . . . 24
- 14.5 Using the algorithm . . . 27
- 14.6 Practice problems . . . 29

15 Hall's theorem 31
- 15.1 Hall's theorem . . . 31
- 15.2 Regular bipartite graphs . . . 33
- 15.3 A three-card magic trick . . . 35
- 15.4 Generalized tic-tac-toe . . . 38
- 15.5 Incomparable sets . . . 40
- 15.6 Practice problems . . . 43

16 Matchings in general graphs 45
- 16.1 Tutte sets . . . 45
- 16.2 Saturated graphs . . . 48
- 16.3 Proof of Tutte's theorem . . . 49
- 16.4 1-factorizations . . . 51
- 16.5 Increasing walks . . . 55
- 16.6 Practice problems . . . 56

Bibliography 59

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

This part of the book covers the matching problem in graphs, especially in bipartite graphs. Chapter 13 introduces bipartite graphs and the matching problem; Chapter 14 and Chapter 15 present two views of the solution to the bipartite matching problem. Chapter 16 is about the matching problem in general graphs, and some of is applications.

The cover

The cover of this PDF shows the first few steps of finding a 1-factorization of the complete graph K_8, writing it as a union of 7 perfect matchings. The first three matchings are shown; finding the other four is left up to you!

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.

13 Bipartite matching

The purpose of this chapter

In this chapter, we will not yet see any of the big theorems about matchings. Here, we will only lay the foundations and begin to study bipartite graphs, matchings, and vertex covers.

I have gone to some trouble in this textbook to postpone the definition of bipartite graphs until this chapter. This is done for two reasons. First, I do not want the first few chapters to be overloaded with definitions and nothing but definitions—as much as I can help it, at least. Second, I do not want to give a definition when we have no use for it. This is because, to the extent I can, I want the definitions we make and the questions we ask about them to seem like reasonable definitions we make and reasonable questions to ask.

As a result, it is only now that I define bipartite graphs, because now we can ask the bipartite matching problem. Here, it is reasonable to ask whether (and in which way) a graph is bipartite, and so I hope that the definitions and initial theorems do not feel pointless.

13.1 Two chess puzzles

The title of this section may be a slight exaggeration; you will not need to know the rules of chess. These are, instead, two puzzles about placing pieces on a chessboard.

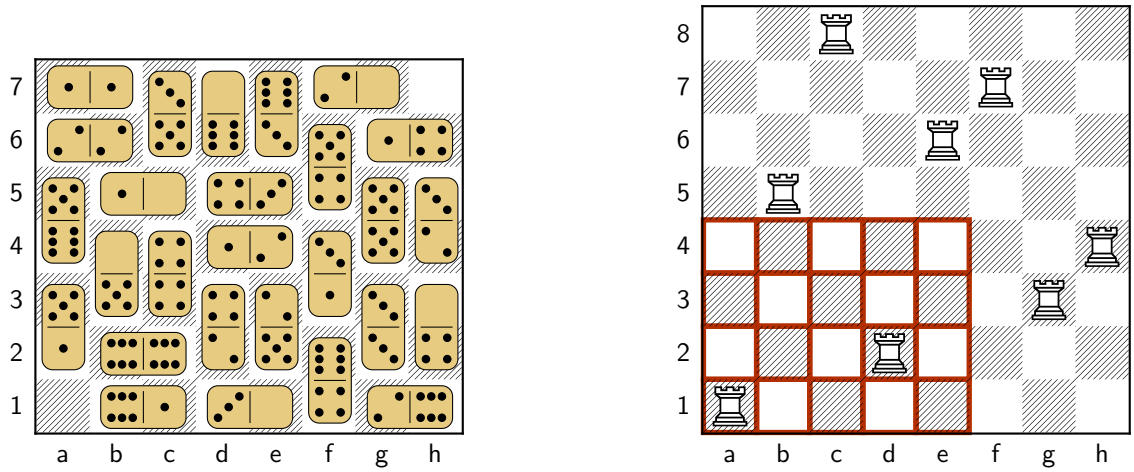
The first is a classic puzzle about placing dominoes on a chessboard. A complete set of dominoes has 28 dominoes; enough to completely cover an 8×7 grid. If I take away one domino, then it's possible to cover every square of the grid except for two opposite corners: Figure 13.1a shows one of many solutions.

But the standard chessboard is 8×8. So suppose you have 31 dominoes: a complete set with three extras. Can you cover every square of the 8×8 chessboard except for two opposite corners?

A second puzzle is shown in Figure 13.1b. Here, we place 8 rooks on a chessboard. A rook can move any number of spaces horizontally or vertically; we don't want the rooks to get in the way of each other, so we want to place them so that no two rooks are in the same row or column.[1] I'll add a condition to make this more difficult in a moment, but first...

Question:	How many ways are there to place 8 rooks on an 8×8 chessboard so that no two rooks are in the same row or column?
Answer:	There are $8! = 40\,320$ ways.

[1] In the same "rank" or "file", if you're a chess purist.



(a) Dominoes on a 8×7 chessboard

(b) Rooks on a chessboard

Figure 13.1: Placing pieces on a chessboard

If we pick a column for the rook in row 1, then the rook in row 2, and so on, then we will have 8 options for the first rook, 7 for the second rook, 6 for the third rook, and so forth. By the product principle, we can multiply these numbers together, getting $8 \cdot 7 \cdot 6 \cdot 5 \cdot 4 \cdot 3 \cdot 2 \cdot 1$ or $8!$ arrangements.

Now I can ask you the real puzzle. Suppose we exclude the 4×5 area highlighted in Figure 13.1b. Is it still possible to place 8 rooks on the chessboard, still in different rows and columns, so that this area remains empty?

We will be able to answer both questions with the tools in this chapter. But first, let's take a moment to ask the question we've been asking since Chapter 1: how do we model each puzzle with a graph? In both cases, there are actually two reasonable approaches!

Let's start with the domino puzzle. We've already seen a tiling puzzle in Chapter 1, and we can model this puzzle in the same way. Create a graph with a vertex for each valid way to place a single domino: on an 8×8 chessboard with two opposite vertices removed, there will be 108 vertices. Then, add an edge between two vertices if they represent incompatible placements: dominoes that share a square of the board. To place 31 dominoes, we are looking for a 31-vertex *independent set*: a set of 31 vertices that do not have any edges between them.

As we will see in Chapter 18, finding independent sets is a hard problem. So let's instead take a different graph to start with: the 8×8 grid graph $G(8, 8)$, with vertices $(1, 1)$ and $(8, 8)$ removed.

Question:	If we model the chessboard with this graph, how do we interpret placing dominoes?
Answer:	Each placed domino corresponds to an edge of the graph. So we are looking for a way to select 31 edges that do not share any vertices.

Such a set of edges is called a matching. This part of the textbook is all about finding matchings! Please be patient, though; I will introduce matchings formally a bit later on in this chapter.

Now, let's move on to the rook puzzle. It, too, has two models: in one, we're looking for an independent set, and in the other, we're looking for a matching. We'll want to use the second model to solve the problem!

Question:	How do we model the rook puzzle with a graph so that a valid placement of rooks is an independent set?
Answer:	For this, let every square of the chessboard be a vertex (excluding the forbidden squares highlighted in Figure 13.1b, if we want to avoid them). Make two vertices adjacent if they share a row or column.

Question:	How do we model the rook puzzle with a graph so that a valid placement of rooks is a matching?
Answer:	For this, take a 16-vertex graph, with a vertex for each row or column. Make a row vertex adjacent to a column vertex if we allow a rook to be placed at the intersection of that row and column.

13.2 Bipartite graphs

The graph we just defined for the rook placement problem has an unusual feature. Its vertices come in two types: the vertices representing rows, and the vertices representing columns. There is a special name for such graphs.

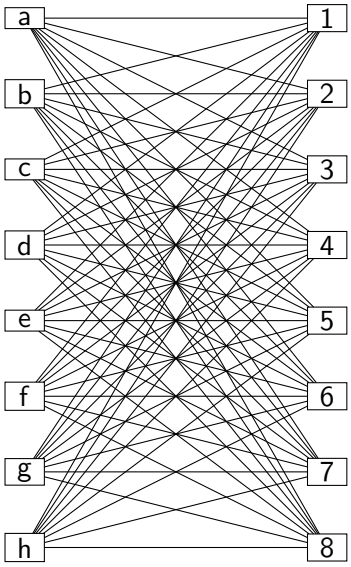
Definition 13.1. *A **bipartite graph** G is a graph for which we can choose disjoint sets A and B, with $A \cup B = V(G)$, such that every edge of G has one endpoint in A and one endpoint in B. The pair (A, B) is called a **bipartition** of G, and the sets A and B are called the two **sides**[2] of the bipartition.*

In the rook placement problem, we can take the bipartition (A, B) with $A = \{\mathsf{a}, \mathsf{b}, \ldots, \mathsf{h}\}$ (the columns of the chessboard) and $B = \{1, 2, \ldots, 8\}$ (the rows). Figure 13.2 shows the two graphs we get in this way: Figure 13.2a shows the graph for placing rooks with no restrictions, and Figure 13.2b shows the subgraph when we forbid a 4×5 rectangle on the board.

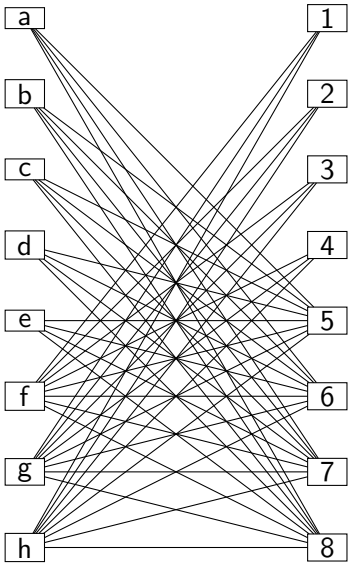
As always, we can draw a graph in any way we like; however, when we want to make it visually clear that a graph is bipartite it is traditional to arrange the two sides A and B in two columns (as in Figure 13.2) or in two rows.

The graph in Figure 13.2a representing the entire chessboard is a special case: it is isomorphic to a complete bipartite graph. (Just as the complete graph has all the edges a graph could possibly have, a complete bipartite graph has all the edges a bipartite graph could possibly have.) Here is one general definition:

[2] In graph theory literature, these have no consistent name; they can also be called "parts", or "blocks", or even "partite sets".



(a) Graph for the entire chessboard



(b) Subgraph excluding the forbidden squares

Figure 13.2: Modeling rook placements with bipartite graphs

Definition 13.2. *For any $m \geq 1$ and $n \geq 1$, the* ***complete bipartite graph*** $\boldsymbol{K_{m,n}}$ *is the graph with $m+n$ vertices $\{1, 2, \ldots, m+n\}$ and all mn possible edges xy where $1 \leq x \leq m$ and $m+1 \leq y \leq m+n$.*

We have already seen one special case: the star graph S_n is isomorphic to the complete bipartite graph $K_{1,n-1}$.

In the case of the graphs for the rook placement problem, the bipartition (A, B) is a natural part of the definition of the graph. The two vertices come in two types, representing the rows and the columns of the chessboard, and the rule defining adjacency is a relationship between the two types of vertex.

It turns out that the grid graph representing the domino problem is also a bipartite graph. Here, though, the bipartition (A, B) is not given to us. In order to see that the graph is bipartite, we have to find a pair (A, B) satisfying the definition of a bipartition.

Question:	Can you think of a way to classify the squares of any chessboard into two types, so that every domino we place covers one square of each type?
Answer:	The classification is by color: let A be the set of light squares and let B be the set of dark squares.

The chessboard-style coloring proves that every grid graph $G(m, n)$ (and every subgraph of a grid graph) is bipartite. (If we remove opposite corners of the chessboard, we delete two vertices of the grid graph, but it remains bipartite when we do so.) Now, let's talk about how to find the pair (A, B) in general.

A problem that involves labeling the vertices of a graph (such as with two sides A and B) might be difficult because we might have to guess, make mistakes, and backtrack. In the case of a bipartition, we are actually quite lucky: there are frequent forced deductions. In a bipartite graph, if there is an edge xy where $x \in A$, then we know that $y \in B$ without guessing. The following algorithm relies only on making such forced deductions.

Given a graph G, we intend to give each vertex a label, A or B, to indicate which side of the bipartition it is on. Initially, all vertices start unlabeled.

1. Pick an arbitrary unlabeled vertex and label it with A.
2. Go through all vertices newly labeled with A, and label all their neighbors with B.
3. Go through all vertices newly labeled with B, and label all their neighbors with A.
4. Repeat steps 2–3 until a vertex has been labeled with both A and B, or until no more unlabeled vertices are being labeled.
5. If the graph is not connected, repeat steps 1–4 for each connected component.

If the graph we are working with is bipartite, then at the end, we get a bipartition (A, B), where A is the set of all vertices labeled with A, and B is the set of all vertices labeled with B.

Question:	What happens if the graph is not bipartite?
Answer:	We will give a vertex both labels, and stop. Such a vertex cannot exist in a bipartite graph, so if we are forced to have such a vertex, then the graph cannot be bipartite.

Question:	Why are we free to label a vertex with A in step 1: at the beginning, and when we consider each new connected component?
Answer:	In each connected component, we can swap the roles of A and B. So for every solution where this vertex is in B, there is another solution where it is in A.

The bipartition algorithm is quite similar to the distance-finding algorithm we studied in Chapter 3: in both algorithms, we visit all the vertices by a breadth-first search. The only difference is in what we do when we visit the vertices. In the distance-finding algorithm, we gave a vertex labels $0, 1, 2, 3, \ldots$ at successive stages. In this algorithm, we alternate the labels $A, B, A, B, \ldots$ at successive stages, instead.

Question:	Let G be a connected graph, and let x be the vertex initially placed in A. How can we define the sets A and B in terms of distances from x?
Answer:	In the end, A will be the set of all vertices at an even distance from x, and B will be the set of all vertices at an odd distance from x.

13.3 Odd cycles

Although the bipartition algorithm is a convenient way to test a specific graph for being bipartite, it is not as useful theoretically: when proving a general result, we would like some testable criteria. One way to prove a graph is bipartite is to find a bipartition (A, B) and show that it satisfies the definition, but we'd also like to be able to prove that a graph is not bipartite. For this, we have the following theorem:

Theorem 13.1. *A graph G is bipartite if and only if it contains no cycles of odd length (no odd cycles, for short).*

We can interpret this theorem as saying that the odd cycle graphs $C_3, C_5, C_7, C_9, \ldots$ are the simplest non-bipartite graphs: every other graph that isn't bipartite contains one of them.

Proof of Theorem 13.1. First, we prove that in a bipartite graph, all cycles must have even length. In fact, we will prove that all closed walks have even length! The length of a cycle is equal to the length of a walk representing it, so this will be sufficient.

Let G be bipartite with bipartition (A, B), and let $(x_0, x_1, \ldots, x_{l-1}, x_0)$ be a closed walk. Without loss of generality, suppose that $x_0 \in A$. Because edges x_0x_1, x_1x_2, and so on must have one endpoint on each side of the bipartition, we must have $x_1 \in B$, $x_2 \in A$, and so on. In general, we can prove by induction that $x_i \in A$ whenever i is even, and $x_i \in B$ whenever i is odd: the base case is x_0, and then induction step simply uses the fact that the endpoints of edge x_ix_{i+1} must be on opposite sides.

Since the endpoints of edge $x_{l-1}x_0$ must be on opposite sides, and $x_0 \in A$, we must have $x_{l-1} \in B$, which means $l - 1$ is odd. Therefore l, the length of the walk, must be even. The first half of the proof is complete!

It is easy at this point in the proof to get confused and prove the same thing we just did a second time—for example, this would happen if we continued by proving that if G contains an odd cycle, it is not bipartite. To continue correctly, we will assume that G is not bipartite, and prove that it contains an odd cycle.

The condition "G is not bipartite" is very difficult to work with: it says that a bipartition (A, B) does not exist, or that any candidate pair (A, B) somehow fails to be a bipartition. To make use of this, we define a pair (A, B) that ought to be a bipartition in any bipartite graph. Inspired by our bipartition algorithm, we begin by choosing a vertex from each connected component of G, and then defining:

- A to be the set of all vertices at an even distance from a chosen vertex;
- B to be the complement of A: the set of all vertices at an odd distance from a chosen vertex.

Since G is not bipartite, this is not a bipartition. So there must be some edge xy between two vertices both in A, or both in B. Since x and y are necessarily in the same connected component, they must both be at an even distance, or both at an odd distance, from the chosen vertex z in that component.

Consider the walk that begins at x, follows some shortest $x-z$ walk, then follows some shortest $z-y$ walk, then takes the edge xy. The first two segments of this walk both have odd length or both have even length, so combined, their length is even. The final edge xy increases the length by 1. We've found a closed walk of odd length.

To complete the proof, let $(x_0, x_1, \ldots, x_{l-1}, x_0)$ be a shortest closed walk of odd length: we will prove that it represents an odd cycle. Suppose not: then there are some positions i and j with $0 \leq i < j < l$ and $x_i = x_j$.

Question:	The other way the closed walk can fail to represent a cycle is if $l < 3$. Why can we rule this out?
Answer:	Since l is odd, the only such case is $l = 1$. But a closed walk of length 1 cannot exist, because a vertex cannot be adjacent to itself. (In a multigraph, this would be possible with a loop, but in a multigraph, the loop would also be an odd cycle.)

The closed walk $(x_i, x_{i+1}, \ldots, x_j)$ has length $j - i$; the closed walk

$$(x_0, x_1, \ldots, x_i, x_{j+1}, x_{j+2}, \ldots, x_{l-1}, x_0)$$

has length $l - (j - i)$. Since the sum of these two lengths is the odd number l, at least one of these lengths must be odd; since $i < j < l$, both lengths are smaller than l. Therefore we've found an even shorter closed walk of odd length, contradicting our initial choice of a walk. We conclude that the pair i, j cannot exist: the closed walk we found does represent an odd cycle! This completes the proof. □

Question:	In the second half of the proof, why do we need to laboriously define A and B; can we just skip directly to defining $(x_0, x_1, \ldots, x_{l-1}, x_0)$ to be a shortest closed walk of odd length?
Answer:	Before we can define a shortest closed walk of odd length, we need to know that the graph contains such closed walks in the first place: in an empty set of walks, there is no shortest walk!

To conclude this section, let's consider two examples of bipartite graphs, and how we prove that they are bipartite. In one case, we will use the definition directly; in the other, we will use Theorem 13.1.

Proposition 13.2. *For all $n \geq 1$, the hypercube graph Q_n is bipartite.*

Proof. The vertices of Q_n are bit strings $b_1 b_2 \ldots b_n$ where each b_i is either 0 or 1. To define a bipartition (A, B) of $V(Q_n)$, we place a vertex $b_1 b_2 \ldots b_n$ in A if $b_1 + b_2 + \cdots + b_n$ is even, and in B if $b_1 + b_2 + \cdots + b_n$ is odd.

Each edge of Q_n joins two vertices x and y that differ only in one position; $x_i \neq y_i$ for some i, but $x_j = y_j$ for all $j \neq i$. As a result,

$$(x_1 + x_2 + \cdots + x_n) - (y_1 + y_2 + \cdots + y_n) = x_i - y_i = \pm 1 :$$

every difference $x_j - y_j$ with $j \neq i$ cancels. Since the sums differ by ± 1, one of them is even and one is odd: the edge xy has one endpoint in A and one in B. Therefore (A, B) is a bipartition of Q_n, completing the proof. □

Question:	What is the relationship between this bipartition and distances in Q_n?
Answer:	This definition places a vertex in A whenever it is at an even distance from vertex $00\ldots0$.

Proposition 13.3. *All trees are bipartite.*

Proof. We know from Theorem 10.2 that trees have no cycles at all—in particular, they have no odd cycles, so they are bipartite by Theorem 13.1. □

In Chapter 10, we already proved an equivalent statement to Proposition 13.3: it was Proposition 10.7. Our proof back then was much longer, because we did not have Theorem 13.1 at our disposal!

13.4 Matching problems

To discuss problems such as the domino puzzle and the rook puzzle, we make the following definition.

Definition 13.3. *A **matching** M in a graph G is a spanning subgraph of G in which every vertex has degree 0 or 1: no two edges in $E(M)$ share an endpoint.*

A vertex $x \in V(G)$ with $\deg_M(x) = 1$ is **covered** by M, and **uncovered** otherwise (when $\deg_M(x) = 0$). The vertices covered by M are exactly the endpoints of the edges in $E(M)$.

When it comes to a matching, we are almost exclusively interested in the edge set $E(M)$: it tells us where to place rooks or dominoes. It is common to simply define a matching to be a set of edges, no two sharing an endpoint. However, the spanning subgraph M is occasionally convenient to use; it is easy to go from M to $E(M)$, but not so easy to go from $E(M)$ back to M, so I have chosen the subgraph and not the set of edges as the fundamental object.

We can think about matchings in all kinds of graphs. But we will start by thinking about matchings in bipartite graphs for two reasons. First, many applications will naturally give us a bipartite graph in which to look for a matching. Second, the theory turns out to be simpler for bipartite graphs.

The empty subgraph M with no edges is also a matching by our definition—it's just not a very good one. It is more difficult, but also more interesting, to select more edges. The larger a matching is, the more vertices it covers, and so the absolute limit is to cover every vertex of G: this is what we're looking for in both examples we've seen so far.

Definition 13.4. *A **perfect matching** M in a graph G is a matching that covers every vertex of G.*

The bipartite matching problem can be considered from two points of view. The first is the optimistic point of view, in which we ask whether a perfect matching exists. This is very common in theoretical applications, including applications to other areas of math.

There is also what I call the "realistic" point of view. In many practical applications, while a perfect matching is still the best option, a very big matching is still good if it is not perfect. For example, one practical bipartite matching problem occurs in every math department when instructors are assigned to courses. It might not be possible in a given semester to offer every course, but does the math department give up and cancel classes for the semester? No! Instead, the department simply tries to schedule as many courses as possible.

Question:	What bipartite graph can we use to model assigning instructors to courses?
Answer:	The vertices of the graph are the course sections and the instructors that can teach them; the edges represent which courses an instructor is qualified to teach.

(Further complications on top of the matching problem arise when we consider giving an instructor multiple sections, which must not be scheduled at the same time.)

In general, instead of asking whether a perfect matching exists, the second point of view is to ask how big the largest matchings are. It's this point of view that we will start with in this textbook. Of course, a complete answer to this question will also tell us whether a perfect matching exists!

13.5 Maximum and maximal

In combinatorial optimization problems like the matching problem, we are trying to find an object X that's as large as possible, subject to some condition. It's common to say that X is:

- *maximum* if it really is as large as you can get; there are no better solutions than X.
- *maximal* if you cannot add anything to X without violating the restriction; there might be better solutions, but you cannot get to them without removing something from X first.

Every maximum solution is also maximal, but the reverse might not hold.

In the same way, the words *minimum* and *minimal* get used when we're trying to pick a set that's as small as possible, subject to some condition.

In this textbook, I will also use this terminology, but I will avoid conveying important information solely by the difference between "maximum" and "maximal". Usually, we are interested in maximum and minimum objects, because we want to solve the optimization problems we pose, and I will let this situation pass by without further comment. If we really need a maximal object that is not necessarily maximum, or a minimal object that is not necessarily minimum, I will specifically point this out for you.

However, no matter which terminology you use, it's important to realize that there is a difference, and in particular there is a difference in the matching problem. For example, consider the

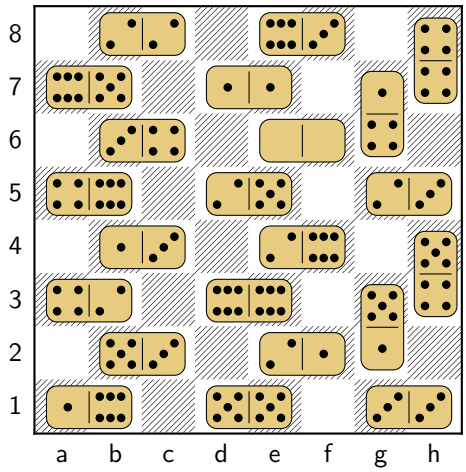


Figure 13.3: A clumsy packing of dominoes

domino placement in Figure 13.3. It is not a maximum matching in the 8×8 grid graph: with no squares removed, it is easy to get a perfect matching by, for example, placing 32 horizontal dominoes. However, it is a maximal matching: there is no empty space to add another domino.

In fact, in the case of dominoes, a much harder problem is the clumsy packing problem: what is the minimum number of edges in a maximal matching of the $n \times n$ grid graph? The diagram in Figure 13.3 shows a construction found by Gyárfás, Lehel, and Tuza in 1988 [2].

Looking at this example also tells us that the bipartite matching problem cannot be solved by a greedy algorithm that selects the first edges it sees. We should expect to have to work hard before we find a general solution.

13.6 Vertex covers

Here is a new, more abstract example of a bipartite matching problem. Let G be the bipartite graph with vertices $\{10, 11, \ldots, 19\}$ on one side, $\{20, 21, \ldots, 29\}$ on the other side, and edges defined by the following rule: vertices x and y are adjacent when they begin with different digits and the product $x \cdot y$ is divisible by 6. A diagram of the multiples-of-six graph is shown in Figure 13.4, though it is a little busy in places. What is the largest matching in this graph?

Question:	Is there a perfect matching in this graph?
Answer:	No: vertices 11 and 13 on the top are only adjacent to vertex 24, so they cannot both be covered by a matching.

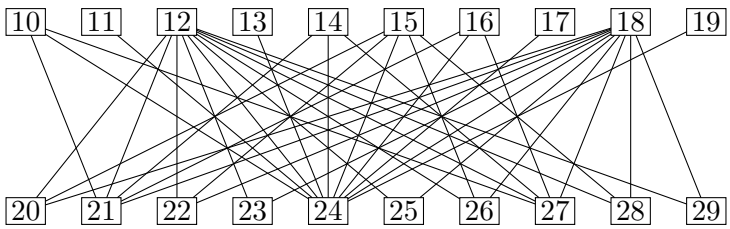


Figure 13.4: The multiples-of-six graph

Question:	Look for a matching greedily: for each of the vertices $10, 11, \ldots, 19$, match it to the first unused vertex on the other side, if you can. What matching do you get?
Answer:	The matching whose edges are $\{10, 21\}$, $\{11, 24\}$, $\{12, 20\}$, $\{14, 27\}$, $\{15, 22\}$, and $\{18, 23\}$.

We've made a good start, but we have a gap: we've found a matching with 6 edges, but we can only prove that the matching can't have more than 9 edges. (That is, we've ruled out a perfect matching with 10 edges.) We either need to find a better solution, or we need to prove a better upper bound. Is there a bottleneck—some kind of limited resource that we exhaust if we try to find a larger matching?

Yes! The limited resource is the vertices divisible by 3. To get a product divisible by 6, we need both a multiple of 2 and a multiple of 3, but the multiples of 3 are much harder to come by: they are $12, 15, 18, 21, 24$, and 27. Every edge in a matching needs to use one of these vertices, so there can be at most 6 edges in a matching.

This argument generalizes.

Definition 13.5. *A **vertex cover** in a graph G is a set of vertices $U \subseteq V(G)$ such that every edge of G has one or both endpoints in U.*

For example, in the multiples-of-six graph, the set $\{12, 15, 18, 21, 24, 27\}$ is a vertex cover. It is not the only one. For example, the set of all vertices in a graph is guaranteed to be a vertex cover. The hard task is not to find a vertex cover, but to find a vertex cover that is as small as possible.

Vertex covers are important because, just as in the example of the multiples-of-six graph, we can use them to prove upper bounds on the size of a matching:

Proposition 13.4. *If M is a matching and U is a vertex cover in the same graph G, then $|E(M)| \leq |U|$.*

Proof. Consider the sum

$$\sum_{x \in U} \deg_M(u).$$

Each term $\deg_M(x)$ of this sum is at most 1, by the definition of a matching, so the value of the sum is at most $|U|$: the number of vertices in U.

However, $\deg_M(u)$ counts the number of edges of M incident to x, so we can also think of the sum as counting the number of times any edge of M is incident to any vertex of U. Each edge of M must be incident to at least one vertex of U, by the definition of a vertex cover, so each edge of M contributes at least 1 to the sum. Therefore the sum is at least $|E(M)|$ the number of edges in M.

Putting these inequalities together, we conclude that $|E(M)| \leq |U|$. □

Question:	Does Proposition 13.4 only apply to bipartite graphs?
Answer:	No: it is true for any graph.

In a bipartite graph G, however, there are two particularly easy examples: if (A, B) is a bipartition of G, then A and B are both vertex covers. By Proposition 13.4, there cannot be a matching with more than $\min\{|A|, |B|\}$ edges; in particular, there can only be a perfect matching if $|A| = |B|$.

Question:	What does this tell us about the domino problem on an 8×8 chessboard with two opposite squares removed?
Answer:	A complete 8×8 chessboard has 32 squares of each color, but two opposite squares have the same color; if they are removed, only 30 squares of that color are left. So at most 30 dominoes can be placed: not enough to cover the remaining 62 squares.

Question:	Can you use vertex covers to show that the 4×5 rectangle in Figure 13.1b cannot be avoided?
Answer:	Take $U = \{\mathsf{5}, \mathsf{6}, \mathsf{7}, \mathsf{8}, \mathsf{f}, \mathsf{g}, \mathsf{h}\}$. These four rows and three columns cover the entire chessboard except for the highlighted rectangle, showing that at most 7 rooks can be placed outside the rectangle.

Proposition 13.4 can be used with any matching M and any vertex cover U. However, if we want to get upper bounds on the size of a matching, then the smaller U is, the better the bounds we get. To get the best upper bound, we should try to find the smallest vertex cover possible. However, even for the smallest vertex cover in a graph, this bound might not tell us the full truth about matchings.

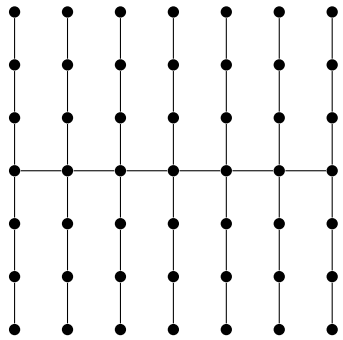
Question:	In the complete graph K_{100}, how many edges are in the largest matching?
Answer:	There are 50, because a 50-edge matching covers all 100 vertices: it is perfect.

Question:	In the complete graph K_{100}, how many vertices are in the smallest vertex cover?
Answer:	There are 99. We can leave out any one vertex from the vertex cover, but a set U missing two vertices x and y does not contain either endpoint of the edge xy.

It will turn out that in bipartite graphs, the problems of finding the largest matching and the smallest vertex cover do "meet in the middle". This is a result known as Kőnig's theorem (Theorem 14.2), which we will prove in the next chapter. Kőnig's theorem is the reason we are studying matchings in bipartite graphs first.

13.7 Practice problems

1. In the graph shown below, find a matching M and a vertex cover U with $|M| = |U|$.



2. Mathematicians that study arrangements of non-attacking rooks on chessboards of various shapes often define an object called the rook polynomial. This is a polynomial $p(x)$ in which the coefficient of x^k is the number of ways to place k non-attacking rooks onto the chessboard.

   a) Find the rook polynomial of a 5×5 chessboard.

   b) Find the rook polynomial of a 5×5 chessboard with one square missing.

3. The "cop-and-highway-robber game"[3] is played on a bipartite graph. One player (the highway robber) chooses an edge of the graph and occupies it for highway robbery. Simultaneously and without seeing the edge chosen, the other player (the cop) chooses a vertex of that edge to guard. If the cop guards an endpoint of the edge chosen by the highway robber, the highway robber gets caught and loses; otherwise, the highway robber gets away and wins.

   This is a game, like rock-paper-scissors, where both players need to choose their strategy at random; if either player plays predictably, then the other player can exploit this.

   a) Suppose that the graph has a matching M. Find a strategy for the highway robber to win with probability $1 - \frac{1}{|M|}$, no matter what the cop does.

[3] Whose name I made up, inspired by the study of cops-and-robbers games on graphs, which are something fancier.

b) Suppose that the graph has a vertex cover U. Find a strategy for the cop to win with probability $\frac{1}{|U|}$, no matter what the highway robber does.

(Why is this another proof of Proposition 13.4?)

4. a) Prove that when n is a multiple of 3, it is possible to place $\frac{1}{3}n^2$ dominoes on an $n \times n$ chessboard so that there is no more room to place another domino.

   b) Prove that whenever dominoes are placed on an $n \times n$ chessboard so that there is no more room to place another domino, at least $\frac{1}{3}(n^2 - n)$ dominoes must have been used.

5. Prove that a bipartite graph with minimum degree at least 2 contains a cycle of length at least 4.

6. Let G be a bipartite graph with bipartition (A, B). Prove the "bipartite handshake lemma":

$$\sum_{x \in A} \deg(x) = \sum_{x \in B} \deg(x) = |E(G)|.$$

7. Coins are placed on several of the squares on an 8×8 chessboard (at most one coin per square). Every row of the chessboard has the same number of coins on it, but every column of the chessboard has a different number of coins (possibly 0). How many coins are on the chessboard?

8. To explore the difference between maximum and maximal matchings, consider the following graph (which can be made arbitrarily large):

   a) Find a perfect matching in this graph.

   b) Find the smallest matching in this graph above which is maximal (there is no larger matching that contains all of its edges).

   c) Let G be a graph (not necessarily bipartite) and let M be a maximal matching in G. Prove that G has a vertex cover U with $|U| = 2|M|$.

   d) Prove that the example in this problem is essentially as bad as it gets: if M is a maximal matching, then there is no matching with more than $2|M|$ edges.

9. (BMO 2005) The equilateral triangle ABC has sides of integer length n. The triangle is completely divided (by drawing lines parallel to the sides of the triangle) into equilateral triangular cells of side length 1.

   A continuous route is chosen, starting inside the cell with vertex A and always crossing from one cell to another through an edge shared by the two cells. No cell is visited more than once. Find, with proof, the greatest number of cells which can be visited.

14 Kőnig's theorem

The purpose of this chapter

This chapter presents the algorithmic view of matchings in bipartite graphs. The augmenting path algorithm is the most serious algorithm seen in this textbook so far; accordingly, I've given it the most serious treatment, with a detailed proof that the algorithm works correctly, and an example of how the algorithm is used. I do not intend to treat every algorithm similarly; this is not a computer science textbook. This is, however, an introductory graph theory textbook, so I hope that this chapter introduces you to the algorithmic side of graph theory.

In my opinion, Kőnig's theorem (Theorem 14.2) is the right setting to consider the algorithmic questions, because it is a result comparing two optimization problems, which is how we usually encounter matchings in algorithmic applications. Of course, Kőnig's theorem has its uses in pure mathematics, and Hall's theorem (Theorem 15.1 in the next chapter) can also be proved by giving an algorithm. However, most uses of Hall's theorem are about abstractly proving the existence of a perfect matching in a general, incompletely specified family of graphs—and that is the sort of problem we will consider in the next chapter.

Kőnig's theorem will also become important in Chapters 25 through 28. We will use it to prove Menger's theorem (Theorem 27.1). In the other direction, algorithms for the maximum flow problem can be used to solve the bipartite matching problem.

14.1 Kőnig's theorem

As we've already seen in Chapter 4 with the notation for minimum degree ($\delta(G)$) and maximum degree ($\Delta(G)$), graph theorists like to use Greek letters for numerical invariants of graphs. We have recently learned two new such invariants, so let's learn the notation for them.

Definition 14.1. *The **matching number** $\alpha'(G)$ of a graph G is the number of edges in a maximum (largest) matching of G.*

Definition 14.2. *The **vertex cover number** $\beta(G)$ of a graph G is the number of vertices in a minimum (smallest) vertex cover of G.*

Unfortunately, the letters α ("alpha") and β ("beta") here have no particular meaning to help you remember them. There is a meaning to the apostrophe present in $\alpha'(G)$ but not in $\beta(G)$: it indicates that the invariant is an "edge version" of another invariant. This means that you can already begin to guess what $\alpha(G)$ (the "vertex version" of the matching number) might mean: later on, in Chapter 18, you will learn if your guess was right.

Using the new notation, we can rewrite Proposition 13.4, which we stated as an inequality between $|E(M)|$ (the number of edges in a matching M) and $|U|$ (the number of vertices in a vertex cover U). The new version reads:

Proposition 14.1. *In any graph G, $\alpha'(G) \leq \beta(G)$.*

Actually, it might not be immediately obvious that Proposition 13.4 and Proposition 14.1 are the same: the first is talking about an arbitrary matching and vertex cover, and the second is only talking about maximum matchings and minimum vertex covers.

However, if Proposition 13.4 holds for all M and U, then in particular it holds when M is a maximum matching and U is a minimum vertex cover, which proves Proposition 14.1. Conversely, if Proposition 14.1 holds, then for any matching M and vertex cover U, we have $|E(M)| \leq \alpha'(G) \leq \beta(G) \leq |U|$, proving Proposition 13.4. (The inequalities $|E(M)| \leq \alpha'(G)$ and $\beta(G) \leq |U|$ come from the meanings of the words "maximum" and "minimum".)

The main theorem of this chapter was proved in 1931 by Dénes Kőnig [5]. Actually, it is ambiguous which result you mean if you say "Kőnig's theorem", because Kőnig proved many results in graph theory: he was, in fact, the author of the first graph theory textbook [6], written in 1936.[4] In this textbook, the theorem we'll call Kőnig's theorem is the following:

Theorem 14.2 (Kőnig's theorem)**.** *For any bipartite graph G, $\alpha'(G) = \beta(G)$.*

We will prove this theorem by finding an algorithm. Given a matching M, the algorithm will either construct a larger matching M', or find a vertex cover U with $|E(M)| = |U|$.

Question:	If we find such a U, what can we conclude?
Answer:	Since every matching has at most $|U|$ edges, and M has exactly $|U|$ edges, we know that M is a maximum matching by Proposition 13.4. By similar reasoning, U is a minimum vertex cover.

However, we know that we cannot always improve a sub-optimal matching simply by adding edges. So what does it take to improve a sub-optimal matching?

14.2 Augmenting paths

We will need a set-theoretic operation that is not as well-known as union and intersection, but is the natural operation to consider when tracking changes. (This remains true whether we're tracking changes made to a matching in a graph, or tracking changes made to a Wikipedia entry, for example.) At its most basic, it's defined on sets: if A and B are two sets, then their **symmetric difference $\boldsymbol{A \oplus B}$** is the set $(A \cup B) - (A \cap B)$: the set containing all elements in A, or in B, but not in both A and B.

[4] By a happy coincidence, this is 200 years after Euler first modeled a problem—the problem of the seven bridges of Königsberg—using what we'd now call a graph.

IMO 2024	**IMO 2025**	**2024 $\oplus$ 2025**
United States of America	People's Republic of China	Belarus
People's Republic of China	United States of America	United Kingdom
Republic of Korea	Republic of Korea	Hungary
India	Japan	Japan
Belarus	Poland	Israel
Singapore	Israel	Vietnam
United Kingdom	India	
Hungary	Singapore	
Poland	Vietnam	
Türkiye	Türkiye	

Figure 14.1: Top ten IMO teams in 2024 and in 2025

We can think of $A \oplus B$ as a summary of the changes that need to be made to A to get B. If we start with A, and "toggle" all the elements of $A \oplus B$ (removing them from A if they are in A, and adding them to A if not), then the result is B.

Question: If we know A and $A \oplus B$, what operation on them lets us find B?

Answer: It's the symmetric difference again: B is $A \oplus (A \oplus B)$. The elements in $A \oplus B$ but not A are the elements only in B, which we need to add to A; the elements in both A and $A \oplus B$ are exactly the ones not in B, which we need to remove.

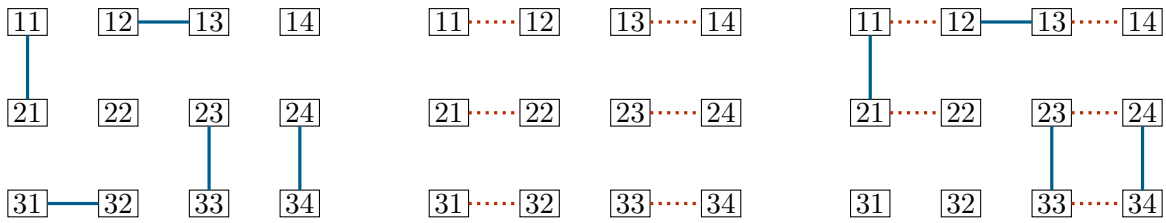
Question: Which elements does a triple symmetric difference $(A \oplus B) \oplus C$ contain?

Answer: The elements contained in an odd number (one, or all three) of the sets A, B, C. This rule generalizes, and can also be used to prove that $\oplus$ is associative: $(A \oplus B) \oplus C = A \oplus (B \oplus C)$, because the rule applies to both sides.

For example, Figure 14.1 shows a table listing the top ten countries in the International Mathematical Olympiad in 2024 and in 2025. If we think of these lists as sets, we can take the symmetric difference, shown in the third column. This symmetric difference shows the six countries whose status changed from 2024 to 2025: Belarus, the United Kingdom, and Hungary left the top ten list, while Japan, Israel, and Vietnam entered it.

Rather than continue thinking about competition performance, let's switch to thinking about graphs. Given two graphs G and H, the **symmetric difference $G \oplus H$** is the graph with vertex set $V(G \oplus H) = V(G) \cup V(H)$ and edge set $E(G \oplus H) = E(G) \oplus E(H)$. We're taking the symmetric difference of the edge sets, which are the star of the show; we take the union of the vertex sets to be sure that we've included the endpoints of every edge we need.

We want to use this idea to see what changes need to be made to get from one matching to another. We are looking for small, incremental changes. So we will compare two matchings M and N such that M only has one more edge than N.



(a) Matching M (b) Matching N (c) $M \oplus N$

Figure 14.2: Two matchings in a 3×4 grid

Figure 14.2 shows one such example. To make the drawing more clear, I have not included the edges of the original graph (which you may assume to be a 3×4 grid graph). Matching M (in Figure 14.2a) has only 5 edges, while N (in Figure 14.2b) is a perfect matching with 6 edges.

As a graph, $M \oplus N$ (shown in Figure 14.2c) consists of four connected components. Two of them are just isolated vertices, and don't have a lot to tell us. Another, with the vertices $\{23, 24, 33, 34\}$, is a cycle, and represents only that the matchings M and N match those four vertices in two different ways.

The last component is the most interesting one: the path represented by $(22, 21, 11, 12, 13, 14)$. It's this component that tells us how to improve matching M: we must swap the two edges $\{11, 21\}$ and $\{12, 13\}$ for the three edges $\{11, 12\}$, $\{13, 14\}$, and $\{21, 22\}$.

We will call such a subgraph an M-augmenting path (or just an augmenting path, if it is clear what matching M it is meant to augment). In general, if M is a matching in a graph G, then an ***M*-augmenting path** is a path in G with the following properties:

- The path begins and ends at two vertices that are not covered by M.
- The edges used by the path alternate between edges not in M and edges in M.

In a symmetric difference of two matchings, we can also see connected components (paths or cycles) that satisfy only the second property, but not the first: we call such paths and cycles *M-alternating*.

The length of an M-augmenting path must be odd, or else it will not end at an uncovered vertex. If it has length $2k + 1$, then the $1^{\text{st}}, 3^{\text{rd}}, \ldots, (2k + 1)^{\text{th}}$ edges are not part of M, while the $2^{\text{nd}}, 4^{\text{th}}, \ldots, (2k)^{\text{th}}$ edges are.

If we find a path P in $M \oplus N$ that helps us improve M to be more like N, then we should apply that difference to M by taking the symmetric difference $M \oplus P$. This is generalized by the following lemma:

Lemma 14.3. *Let M be a matching in a graph G and let P be an M-augmenting path. Then $M \oplus P$ is a matching with one more edge than M.*

Proof. We verify that $M \oplus P$ is a matching by checking the degree of every vertex in $M \oplus P$.

Let P be an $x - y$ path; then $\deg_P(x) = \deg_P(y) = 1$. Because P is an M-augmenting path, $\deg_M(x) = 0 = \deg_M(y) = 0$. Therefore x and y have degree 1 in $M \oplus P$ as well; they each get an incident edge from P, and do not gain or lose anything from M.

For every vertex $z \in V(P)$ other than x and y, the path P has two edges incident to z, one of which is in M and the other is not (because P is M-alternating). In $M \oplus P$, the edge in M is lost, but the edge not in M is kept, so $\deg_{M \oplus P}(z) = 1$.

Finally, all vertices not on the path are unaffected by the change from M to $M \oplus P$: they still have the same degree they had in M, which is at most 1. Therefore $M \oplus P$ is a matching.

Looking above, we can see that vertices x and y have degree 1 in $M \oplus P$, but degree 0 in M; for every other vertex, its degree in M and $M \oplus P$ is the same. Therefore $M \oplus P$ is a matching that covers two more vertices than M, which means it has one more edge than M. $\square$

In general, given two matchings M and N, the symmetric difference $M \oplus N$ is a graph with maximum degree 2: each vertex has at most one incident edge in each matching. The connected components of such a graph are M-alternating paths and M-alternating cycles.

Suppose now that, as in our example, N is a slight improvement over M: $|E(N)| = |E(M)| + 1$. Then, after canceling out the edges common between M and N, the symmetric difference $M \oplus N$ still has one more edge from N than from M. So there must also be at least one connected component of $M \oplus N$ where this is true!

Question:	If a connected component of $M \oplus N$ is a cycle, can it contain more edges from N than from M?
Answer:	No: to be M-alternating, exactly half of its edges must be from M, and exactly half must be from N.

Question:	If a connected component of $M \oplus N$ is a path, can it contain more edges from N than from M?
Answer:	Only if it is an M-alternating path of odd length where the first and last edge both come from N. This is an M-augmenting path!

In particular, this argument proves that there exists an M-augmenting path, but not usefully: to find it, we needed to know the bigger matching N. In our proof of Kőnig's theorem, we will see how to find an M-augmenting path in a more helpful fashion, but only in bipartite graphs.

14.3 The augmenting path algorithm

Let G be a bipartite graph with bipartition (A, B), and let M be a matching in G. The main tool we need to prove Kőnig's theorem is an algorithm for finding an M-augmenting path.

Our algorithm will be yet another breadth-first search algorithm. However, we will not explore the graph along all possible trajectories, but only along ones that can form an M-augmenting path.

We are searching for a path between two uncovered vertices, so let's introduce some notation to help us. Write A as $A_0 \cup A_1$, where the vertices in A_0 are uncovered by M and the vertices

in A_1 are covered by M. (That is, $x \in A_k$ if x has degree k in M.) Split B into B_0 and B_1 in the same way.

Since an augmenting path has odd length, and every path in G alternates between A and B, an augmenting path must have one end in A_0 and one end in B_0. We will arbitrarily choose to start our exploration in A_0, with the goal of eventually reaching a vertex in B_0.

To keep track of our progress, let A_{exp} and B_{exp} be the set of **exp**lored vertices in A and in B, respectively. These will change over the course of the algorithm. Since we start our exploration in A_0, initially $A_{\text{exp}} = A_0$ and $B_{\text{exp}} = \varnothing$.

Finally, we will also want to know how we reached the explored vertices: when we add a new vertex x to $A_{\text{exp}} \cup B_{\text{exp}}$, we make note of $f(x)$, the vertex **f**rom which we explored x. This will help us find an augmenting path, if one exists.

The augmenting path algorithm repeats the following steps, which I've given short names to help us refer to them.

1. **Explore:** For each vertex $a \in A_{\text{exp}}$, explore all its neighbors: if a neighbor b of a is not already in B_{exp}, add it, and set $f(b) = a$. (After the first iteration, this only needs to be done for the vertices newly added to B_{exp}.)
2. **Check:** At this step, there are two stopping conditions to check, which conclude the algorithm with different results.
   - **Path?** If there is a vertex $b \in B_{\text{exp}} \cap B_0$ (an explored vertex uncovered by M), stop and return the path represented by the walk
   $$(b, f(b), f(f(b)), \dots)$$
   that traces back the vertices from which we reached b, ending at a vertex in A_0. We will show that this is an M-augmenting path.
   - **Cover?** If no new vertices were added to B_{exp} in the most recent **Explore** step, stop and return the set U defined[5] to be $(A - A_{\text{exp}}) \cup B_{\text{exp}}$. We will show that U is a vertex cover and $|U| = |E(M)|$.
3. **Match:** Otherwise, $B_{\text{exp}} \subseteq B_1$: all the vertices in B_{exp} are covered by M. For each vertex b added to B_{exp} in the most recent **Explore** step, find the edge $ab \in E(M)$ incident to b. Explore vertex a (the vertex that b is matched to): add it to A_{exp}, and set $f(a) = b$.

To prove the correctness of this algorithm, we must show that when the **Path** or **Cover** stopping conditions are met, the path really is an M-augmenting path and the cover really is a vertex cover of the same size as M.

Question:	We must also show that the algorithm cannot loop forever. Why must it halt?
Answer:	At every iteration in which we don't stop by the **Cover** condition, we add a new vertex to B_{exp}, and $\lvert B_{\text{exp}}\rvert$ is limited in size by $\lvert B\rvert$.

[5] When learning about this algorithm for the first time, this definition may feel like it came out of nowhere! In the next section, we'll discuss where this formula for the vertex cover comes from.

In practice, this algorithm is applied multiple times. Starting from an initial matching, which does not need to be very good, we repeatedly use the algorithm to grow the matching by one edge. Eventually, we either find a perfect matching, or find a vertex cover that proves we cannot proceed any further. This self-certifying feature, in addition to being theoretically useful in the proof of Kőnig's theorem, is also practically convenient: it means that we can verify the output of the algorithm independently if we're not certain that an implementation of it is bug-free.

Which initial matching do we start with? One option is to begin with the empty matching: $E(M) = \varnothing$.

Question:	What does the algorithm do if $E(M) = \varnothing$?
Answer:	We start with all vertices of A already in A_{exp}, and all vertices of B in B_0. If even a single edge exists, we will explore a vertex of B_0 and halt in the first step, returning a path of length 1.

Question:	When a path of length 1 is an M-augmenting path, what does this mean?
Answer:	It means that the edge on that path can be added to M without removing any edges.

Starting from the empty matching, not just the first time we augment the matching but the first few times will typically end equally quickly: they will find an edge between two uncovered vertices. In effect, this is no different from a greedy algorithm. Only once the greedy algorithm would give up, because it cannot add any more edges, does the augmenting path algorithm start finding longer augmenting paths.

14.4 Analyzing the algorithm

A common technique when analyzing an algorithm is to prove an invariant of the algorithm. Just like a graph invariant is a property that does not change when the graph is relabeled, an invariant of an algorithm is also a property that does not change—it does not change over the course of the algorithm. Appendix A discusses this proof technique, but this is the first time we really need it in the main body of the textbook.

There are two invariants we track for the augmenting path algorithm: one invariant to help us find an M-augmenting path, and one invariant that

For the first invariant, we define a path $P(x)$ for each vertex $x \in A_{\text{exp}} \cup B_{\text{exp}}$. It is the path represented by

$$(x, f(x), f(f(x)), \dots)$$

that starts at x and traces back the vertices from which we reached x, ending at a vertex in A_0. Recall that in the **Path** stopping condition, this is the path we return, for a vertex $x \in B_{\text{exp}} \cap B_0$.

Lemma 14.4. *For any vertex $x \in A_{\text{exp}} \cup B_{\text{exp}}$, the path $P(x)$ is an M-alternating path.*

Proof. More precisely, we prove the following: each edge $yf(y)$ is in $E(M)$ if $y \in A_{\text{exp}}$, and not in $E(M)$ if $y \in B_{\text{exp}}$. The path $P(x)$ must alternate between A_{exp} and B_{exp}: like any other path in G, it alternates between A and B, and it only passes through explored vertices. So this claim will prove that $P(x)$ is an M-alternating path.

Let $a \in A_{\text{exp}}$ be a vertex on $P(x)$. Then either $a \in A_0$ (in which case, it is the end of the path) or else a was explored from a vertex $f(a) \in B_{\text{exp}}$. We explore from vertices in B_{exp} in a **Match** step; therefore $af(a) \in E(M)$.

Let $b \in B_{\text{exp}}$ be a vertex on $P(x)$. Then b was explored from a vertex $f(b) \in A_{\text{exp}}$; we must prove that $bf(b) \notin E(M)$. If $f(b) \in A_0$, this is automatic, because then M has no edges incident to $f(b)$. Otherwise, $f(b)$ was itself explored from some vertex $f(f(b))$, and as we've proved above, $f(b)f(f(b)) \in E(M)$. Therefore $bf(b) \notin E(M)$, because $f(b)$ cannot be incident to two edges of M. This completes the proof. □

Next, we turn our attention to the set $U = (A - A_{\text{exp}}) \cup B_{\text{exp}}$, which we return if the **Cover** stopping condition is satisfied. First of all, let's understand where this set comes from.

Question:	If a vertex cover U satisfies $\lvert U\rvert = \lvert E(M)\rvert$, can U ever use both endpoints of an edge in M?
Answer:	No: if we select just one endpoint of each edge in M, we've already selected $\lvert E(M)\rvert$ different vertices. Choosing both endpoints of an edge in M is wasteful and we cannot afford it.

Question:	If a vertex cover U satisfies $\lvert U\rvert = \lvert E(M)\rvert$, how must it treat A_0 and B_0?
Answer:	U cannot contain any vertices in A_0 and B_0. Again, we've already reached $\lvert E(M)\rvert$ vertices just by selecting a vertex from each edge in M. We can't select any more without going over our limit, so we can't select any vertices not covered by M.

These two guiding questions tell us what to do to the vertices we explore. Our initial set A_{exp} consists just of A_0: these vertices cannot be part of U. This means that to cover the edges with an endpoint in A_0, we must add all their other endpoints to U. These neighbors are exactly what we put in B_{exp} in the **Explore** step. Next, if a vertex b we've added to U is an endpoint of an edge $ab \in E(M)$, we can't put the other endpoint of ab in U as well, so $a \notin U$. So the vertices matched to B_{exp} cannot be part of U, and these are exactly the vertices we put in A_{exp} in the **Match** step.

This logic continues: at each step, vertices we put in A_{exp} are vertices we know cannot be part of U, and vertices we put in B_{exp} are vertices we know must be part of U. Since we don't know anything about vertices outside $A_{\text{exp}} \cup B_{\text{exp}}$ a reasonable first try is to define $U = (A - A_{\text{exp}}) \cup B_{\text{exp}}$, and we we will see that this turns out to work.

We can define $U = (A - A_{\text{exp}}) \cup B_{\text{exp}}$ at any point over the course of the algorithm, not just at the end. It is not a vertex cover before the end, but it satisfies an invariant of its own:

Lemma 14.5. *Immediately before every* ***Explore*** *step,* $|U| = |E(M)|$.

Proof. At the beginning of the algorithm, $A_{\text{exp}} = A_0$ and $B_{\text{exp}} = \varnothing$; therefore $U = A - A_0 = A_1$. Every edge of M has one endpoint in A_1 and one endpoint in B_1; therefore $|U| = |A_1| = |E(M)|$.

In the **Explore** step, some vertices are added to B_{exp}. Then, in the **Match** step, for every vertex b that we added to B_{exp}, we add a vertex a to A_{exp}: the vertex a such that $ab \in E(M)$. Such a vertex a cannot already have been in A_{exp}: it is not in A_0 (because $ab \in E(M)$), and so it can only be explored in the **Match** step, and only from b.

Therefore, if we add k vertices to B_{exp} in the **Explore** step, we also add k vertices to A_{exp} in the **Match** step (provided we do not stop before then). As a result, $|A - A_{\text{exp}}|$ decreases by k and $|B_{\text{exp}}|$ increases by k, so $|U| = |A - A_{\text{exp}}| + |B_{\text{exp}}|$ remains unchanged

Since $|U|$ starts equal to $|E(M)|$, and a single **Explore**–**Check**–**Match** iteration of the algorithm does not change $|U|$, we must also have $|U| = |E(M)|$ before every **Explore** step. □

Many uses of invariants to study algorithms are proofs by induction in disguise, and you can see that in the proof of Lemma 14.5. Here, we begin with a base case: we show that $|U| = |E(M)|$ at the beginning of the algorithm. Next, we show that whenever this property holds, it still holds after one more iteration.

Together, Lemma 14.4 and Lemma 14.5 are not enough to draw any conclusions about what happens at the end of the algorithm: we need to see how the stopping conditions contribute.

Proposition 14.6. *If the* ***Path*** *stopping condition holds, then for a vertex* $b \in B_{\text{exp}} \cap B_0$*, the path* $P(b)$ *is an* M*-augmenting path.*

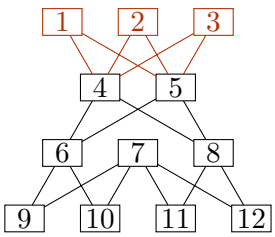
Proof. We already know from Lemma 14.4 that $P(b)$ is an M-alternating path. Moreover, its endpoints are uncovered by M: its start is b, which is in B_0, and its end can only be a vertex in A_0, because all other explored vertices were explored from somewhere. Therefore $P(b)$ is an M-augmenting path. □

Proposition 14.7. *If the* ***Cover*** *stopping condition holds, then* $U = (A - A_{\text{exp}}) \cup (B_{\text{exp}})$ *is a vertex cover with* $|E(M)| = |U|$.

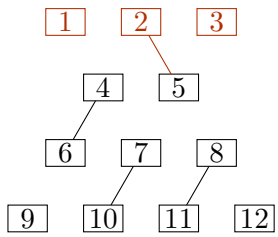
Proof. To show that U is a vertex cover, take any edge $ab \in E(G)$, where $a \in A$ and $b \in B$. If $a \in A_{\text{exp}}$, then in the most recent explore step, we would have made sure that $b \in B_{\text{exp}}$ (if it was not already there for some other reason); in this case, $b \in U$. But if $a \notin A_{\text{exp}}$, then $a \in U$. In either case, one of the endpoints of ab is in U.

From Lemma 14.5, we know that $|E(M)| = |U|$ before any **Explore** step. If the **Cover** stopping condition holds, then no new vertices were added to A_{exp} in the explore step, so $|U|$ did not change; therefore $|E(M)| = |U|$ at the end of the algorithm. □

This completes the proof that the algorithm is correct. It also gives us all the tools we need to prove Kőnig's theorem.



(a) The volcano graph



(b) A 4-edge matching M

Figure 14.3: The volcano graph and an initial matching

Proof of Theorem 14.2. Let M be a maximum matching in the bipartite graph G; starting with the matching M, run the augmenting path algorithm.

The algorithm cannot find an M-augmenting path, because by Lemma 14.3, we could use it to obtain a bigger matching, which contradicts our choice of M. So it must stop by satisfying the **Cover** stopping condition. By Proposition 14.7, the algorithm outputs a vertex cover U with $|E(M)| = |U|$.

Since $\alpha'(G) = |E(M)|$ (by our choice of M) and $\beta(G) \leq |U|$ (the minimum vertex cover is at least as small as U), we conclude that $\alpha'(G) \geq \beta(G)$. From Proposition 14.1, we know $\alpha'(G) \leq \beta(G)$; therefore the two are equal. □

14.5 Using the algorithm

Algorithms like the augmenting path algorithm are not really intended to be performed by hand. Very rarely, if a graph is neither so small that we can find a matching without any algorithms, nor too big to deal with by hand, we can use the algorithm to check a matching we suspect to be optimal. Most of the time, though, you should use a computer. So the purpose of the example here is simply as an illustration, so that you can better understand the algorithm and its proof by seeing it in action. (You will gain even better understanding if you try doing this yourself, and I've included a practice problem about this. After you understand the algorithm perfectly, there's no point.)

The graph we will use as an example is a graph I've called *volcano graph*, shown in Figure 14.3a. It does not really have any special properties, but I defined it and named it and I have a hat with the volcano graph on it, so it is my favorite. One of my whims about the definition is that I insist that the top of the volcano graph must always be drawn in red.

Question:	Why is the volcano graph bipartite?
Answer:	All edges have one endpoint in $\{1, 2, 3, 6, 7, 8\}$ and one endpoint in $\{4, 5, 9, 10, 11, 12\}$.

I have mentioned that the first few rounds of finding augmenting paths are essentially equivalent to a greedy algorithm, so I will skip them to avoid wasting time on cases that don't help us understand anything. We will begin with the matching M shown in Figure 14.3b, with $E(M) = \{\{2, 5\}, \{4, 6\}, \{7, 10\}, \{8, 11\}\}$.

We must make an arbitrary choice between the two sides: let's say that side A (the side that we explore from) is $\{1, 2, 3, 6, 7, 8\}$. For our initial matching, we start with $A_0 = \{1, 3\}$ and $B_0 = \{9, 12\}$.

The algorithm proceeds as follows:

1. **Explore** vertices 4 and 5, adding them to B_{exp}, with $f(4) = 1$ and $f(5) = 3$.
2. **Check** the stopping conditions:
   - Is there a **Path?** No: $B_{\text{exp}} \cap B_0 = \varnothing$.
   - Is there a **Cover?** No: we added 2 new vertices to B_{exp}.
3. **Match** vertex 4 to vertex 6 and vertex 5 to vertex 2. Add 2 and 6 to A_{exp}, with $f(2) = 5$ and $f(6) = 4$.
4. **Explore** vertices 9 and 10, with $f(9) = f(10) = 6$. As before, add them to B_{exp}.
5. **Check** the stopping conditions:
   - Is there a **Path?** Yes: $B_{\text{exp}} \cap B_0 = \{9\}$.

   We stop, and output the augmenting path represented by

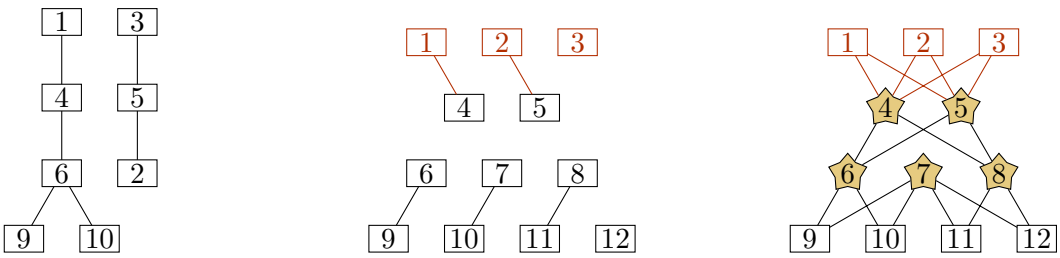
$$(9, f(9), f(f(9)), f(f(f(9))))) = (9, 6, 4, 1).$$

Of the edges on the augmenting path, edge $\{4, 6\}$ is part of M, while edges $\{1, 4\}$ and $\{6, 9\}$ are not. Therefore we improve our matching M by removing $\{4, 6\}$, adding $\{1, 4\}$ and $\{6, 9\}$ instead. The resulting matching (call it M') is shown in Figure 14.4b.

It is common to represent the exploration process by a forest, with the vertices arranged in levels. At the top level are the vertices in A_0; from there, the levels alternate between vertices added to B_{exp} in an **Explore** step and vertices added to A_{exp} in a **Match** step. The forest we get in this example is shown in Figure 14.4a.

Question:	How can we tell which edges of this forest are edges of M?
Answer:	They are the edges whose top vertex is in a B_{exp} level and whose bottom vertex is in an A_{exp} level. In Figure 14.4a, these are just the edges down from $\{4, 5\}$, but if we had kept going, any edges down from $\{9, 10\}$ would also have been in M.

Question:	The forest in Figure 14.4a looks like it has other augmenting paths, such as the one represented by $(10, 6, 4, 1)$. Could we have used this path instead?
Answer:	No: this path only looks like an augmenting path because we stopped before the next **Match** step, but in reality, vertex 10 is covered by M. If we had kept going, we would have found a longer augmenting path.



(a) The exploration forest (b) A 5-edge matching M' (c) The vertex cover U

Figure 14.4: An example of the augmenting path algorithm

We have improved the matching, but we're still not done, so we continue with another round of the algorithm. The edges of the new matching are $\{\{1,4\},\{2,5\},\{6,9\},\{7,10\},\{8,11\}\}$, so the uncovered vertices are given by $A_0 = \{3\}$ and $B_0 = \{12\}$. Now we:

1. **Explore** vertices 4 and 5, adding them to B_{exp}, with $f(4) = f(5) = 3$.
2. **Check** the stopping conditions:
   - Is there a **Path?** No: $B_{\text{exp}} \cap B_0 = \varnothing$.
   - Is there a **Cover?** No: we added 2 new vertices to B_{exp}.
3. **Match** vertex 4 to vertex 1 and vertex 5 to vertex 2. Add 1 and 2 to A_{exp}, with $f(1) = 4$ and $f(2) = 5$.
4. **Explore**, but fail to find any vertices: vertices 1 and 2 are only adjacent to vertices 4 and 5, which are already in B_{exp}.
5. **Check** the stopping conditions:
   - Is there a **Path?** No: $B_{\text{exp}} \cap B_0 = \varnothing$.
   - Is there a **Cover?** Yes: we did not add any new vertices to B_{exp}.

   We stop, and output the vertex cover

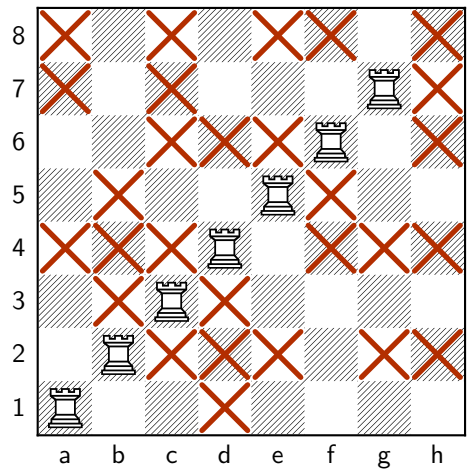
$$U = (A - A_{\text{exp}}) \cup B_{\text{exp}} = \{6,7,8\} \cup \{4,5\}.$$

Indeed, the vertex $U = \{4,5,6,7,8\}$ shown in Figure 14.4c really is a vertex cover: it consists of the middle two layers of the volcano graph. Every edge between the first two layers is incident to 4 or 5, every edge between the last two layers is incident to 6, 7, or 8, and every edge between the second and third layer has both endpoints in U. Since $|U| = |E(M')| = 5$, we conclude that M' is a maximum matching.

14.6 Practice problems

1. Use the augmenting path algorithm to improve the matching in Figure 14.2a. Does this result in the perfect matching in Figure 14.2b, or a different one?
2. Viewing the domino tiling in Figure 13.3 as a matching, find a set of 10 augmenting paths (with no vertices in common) that will transform it into a perfect matching.

3. On the chessboard below, how many rooks can be placed so that no two rooks occupy the same row or column, and all the crossed-out squares are empty?



A 7-rook solution is shown. Use the augmenting path algorithm to find either an 8-rook solution, or a vertex cover proving that the 7-rook solution is optimal.

4. Let G be a bipartite graph with bipartition (A, B) and minimum degree $\delta(G)$; let U be a vertex cover that does not contain either A or B.

   a) Prove that $|U| \geq 2\delta(G)$. When does it follow that $\alpha'(G) \geq 2\delta(G)$, and why?

   b) In the case $|A| = |B| = 10$ and $\delta(G) = 3$, give an example of a bipartite graph in which this lower bound is true: $\alpha'(G) = 6$.

5. If G is not bipartite, then Proposition 14.1 still guarantees that $\alpha'(G) \leq \beta(G)$, but the two quantities may be different.

   a) Find $\alpha'(G)$ and $\beta(G)$ when $G = C_{2k+1}$, a cycle graph with an odd number of vertices, in terms of k. Verify that they are, in fact, different.

   b) Find a graph G which satisfies $\alpha'(G) = \beta(G)$, but is not bipartite.

6. Let G be a bipartite graph G with bipartition (A, B) which is r-regular: every vertex has degree r. Let $n = |A|$.

   a) Prove that $|B| = n$ as well, and find $|E(G)|$ in terms of n and r.

   b) Prove that a set of k vertices in G can cover at most kr of the edges. What does this say about the size of a vertex cover?

   c) Use Kőnig's theorem to prove that G must have a perfect matching.

7. Prove if the augmenting path algorithm is applied to a maximum matching, then A_{exp} is the set of all vertices on side A which are left uncovered by some maximum matching in the graph.

   (You must prove two things: that if $a \in A_{\text{exp}}$, then there is some maximum matching which does not cover a, and that if $a \in A - A_{\text{exp}}$, then all maximum matchings cover a.)

15 Hall's theorem

The purpose of this chapter

Applications of bipartite matchings outside of graph theory are usually all-or-nothing: either we find a perfect matching, or else we have made no progress. Additionally, the graphs we face are not arbitrary: they obey some mathematical laws defining their edges.

In such a scenario, Hall's theorem is usually the way to proceed. Hall's condition is a necessary and sufficient condition for the matching we want, and the abstract definition of our graph means it's often easy to verify. In this chapter, we will deduce Hall's theorem from Kőnig's theorem and then—because this is not enough to make a complete chapter—we will see some applications, ranging from recreational mathematics to serious combinatorics.

Hall's theorem is also often referred to as Hall's marriage theorem, and Hall's condition is then called the marriage condition. I'm not philosophically opposed to this (it can sometimes be handled poorly when teaching, but it can also be handled well), but I don't like the terminology, so I've left it out of this chapter.

15.1 Hall's theorem

Chapter 14 was devoted to proving Kőnig's theorem (Theorem 14.2), which is the result you want to use for finding a maximum matching in a bipartite graph. In many applications, that is absolutely the wrong question to ask, because only a perfect matching counts as a solution; if no perfect matching exists, we don't care if the largest matching leaves 2 vertices or 992 vertices uncovered.

Of course, Kőnig's theorem can also tell us when a perfect matching exists, in terms of the number of vertices in a minimum vertex cover, but this feels like the wrong way of thinking. In a graph G with bipartition (A, B), where $|A| = |B| = n$, we want to distinguish between $\beta(G) = n$ and $\beta(G) \leq n - 1$ to determine whether a perfect matching exists, and this doesn't feel like a dramatic change. What's more, a vertex cover containing $n - 1$ vertices isn't a very intuitive explanation of why there is no perfect matching.

However, there are some situations in which we can tell a clear story about why a perfect matching fails to exist.

Question:	What if the graph contains an isolated vertex, x?
Answer:	Then there is no perfect matching, because there is no matching that covers x.

Question:	What if vertices x and y have degree 1 and share the same neighbor z?
Answer:	Then a matching can contain edge xz and cover x, or contain edge yz and cover y, but not both: so there is still no perfect matching.

Generalizing these concepts, we can describe a minimum requirement for a graph to have a perfect matching. First, we need a new definition.

Definition 15.1. *The* ***neighborhood*** $N_G(S)$ *of a set* $S \subseteq V(G)$ *in a graph* G *is the set of all vertices of* G *adjacent to at least one vertex in* S*. If the graph* G *is clear from context, we simply write* $N(S)$*.*

For example, a vertex x is isolated if $N(\{x\}) = \varnothing$. Two leaf vertices x and y share their only neighbor z if $N(\{x, y\}) = \{z\}$. What do these situations have in common? A set of vertices has a neighborhood which is too small: $N(S)$ is smaller than S itself.

Question:	Let S be an arbitrary subset of A. Can there be a perfect matching if $\lvert N(S)\rvert < \lvert S\rvert$?
Answer:	No: a perfect matching must pair each vertex of S with a different vertex in $N(S)$, which is impossible if there are not $\lvert S\rvert$ different vertices in $N(S)$ to use.

This explains why this condition is necessary for a perfect matching to exist. Hall's theorem (proved by Philip Hall in 1935 [3]) states that the condition is also sufficient:

Theorem 15.1 (Hall's theorem)**.** *A bipartite graph* G *with the bipartition* (A, B) *has a matching that covers all vertices in* A *if and only if it satisfies Hall's condition: for all subsets* $S \subseteq A$*,* $\lvert N(S)\rvert \geq \lvert S\rvert$*.*

In particular, when $\lvert A\rvert = \lvert B\rvert$*,* G *has a perfect matching if and only if it satisfies Hall's condition.*

Question:	What if $\lvert A\rvert > \lvert B\rvert$?
Answer:	Then Hall's condition will not be satisfied if we take $S = A$, because $N(A) \subseteq B$ and therefore $\lvert N(A)\rvert < \lvert A\rvert$.

Question:	What if $\lvert A\rvert < \lvert B\rvert$?
Answer:	Then there is no perfect matching. In this case, Hall's theorem promises us a matching that covers every vertex in A, but leaves some vertices in B uncovered.

Applying Hall's theorem when $|A| < |B|$ can still be very useful if the vertices in A are different in type from the vertices in B. For example, in the magic trick, we really want a matching that covers every vertex $\{a, b, c\}$ corresponding to a set of 3 cards: then the matching tells my assistant what to do when handed such a set. It would be fine if the matching did not cover some ordered pairs (a, b): that would mean that my assistant will never hand me those two cards.

By proving Kőnig's theorem, we have done all the hard work of proving Hall's theorem, and can now deduce it as a corollary. (It would also have been possible to go the other way, and prove Kőnig's theorem as a corollary of Hall's theorem.)

Proof of Theorem 15.1. We already know that Hall's condition is necessary for a matching to cover A to exist: if Hall's condition is violated for some subset $S \subseteq A$, then there is not even a matching that covers every vertex in S. It remains to prove that Hall's condition is sufficient. We will do this by assuming that there is no matching that covers A, and finding a set S for which Hall's condition is violated.

Every edge in a matching has one endpoint in A and one endpoint in B. So if there is no matching that covers A, then there is no matching with $|A|$ edges: $\alpha'(G) < |A|$. By Kőnig's theorem, $\beta(G) < |A|$: there is a vertex cover U such that $|U| < |A|$.

This set U is a small set of vertices with a lot of neighbors, since every edge of G has at least one endpoint in U. To find a set S for which Hall's condition is violated, we want the opposite: a large set of vertices with few neighbors. So we define S to be $A - U$: the set of all vertices in A that are not part of the vertex cover.

For all vertices $x \in S$, if y is adjacent to x, then U contains one endpoint of xy, but U does not contain x, so U must contain y. Therefore all of $N(S)$ is contained in U. In addition to the vertices of $N(S)$ (which are all on side B), U contains all of $A - S$ (on side A); therefore

$$|U| \geq |N(S)| + |A - S| = |N(S)| + |A| - |S|.$$

However, we also know $|U| < |A|$, so $|N(S)| + |A| - |S| < |A|$. This can be rearranged to get $|N(S)| < |S|$, proving that S violates Hall's condition. □

Question:	Suppose I give you a set S which violates Hall's condition. Can you use it to find a vertex cover with fewer than $|A|$ vertices?
Answer:	Yes: the set $(A - S) \cup N(S)$ is a vertex cover. For every edge xy, where $x \in A$ and $y \in B$, we have two cases: if $x \in S$, then y is in the vertex cover, and if $x \notin S$, then x is in the vertex cover.

15.2 Regular bipartite graphs

It is important to be clear about how Hall's theorem is meant to be used. It is not meant to be wielded like a hammer, going through the subsets of A one by one and searching for one that violates Hall's condition. Such an algorithm would take an exponentially long time! No: if you have a concrete graph in front of you, and it has no short mathematical description, use the augmenting path algorithm in Chapter 14 to find a maximum matching and a minimum vertex

cover. (There's one exception: if you can quickly spot a small set that violates Hall's condition, of course you should use that.)

Instead, we use Hall's theorem when dealing with a graph or a family of graphs in which the edges follow a regular mathematical pattern, and we can prove that Hall's condition is satisfied.

Our first example predates both Kőnig's theorem and Hall's theorem: it originally appeared in 1894, in a thesis by Ernst Steinitz about point-line configurations [11], though it was not stated in the language of graph theory.

Theorem 15.2. *For all $r \geq 1$, if G is an r-regular bipartite graph, then it has a perfect matching.*

I admit that I am so fond of this theorem that I've put two versions of it in a practice problem already, once in Chapter 8 and once in Chapter 14. But we have not had the opportunity to use Hall's theorem to prove it, so let's do that.

Proof of Theorem 15.2. Here's a starting observation that proves it's permissible to at least hope for a perfect matching. If G has a bipartition (A, B), then we can count the edges of G in two ways. First, we could sum the degrees of all vertices in A and get $r|A|$: this counts every edge once, because every edge has an endpoint in A. We could also sum the degrees of all vertices in B and get $r|B|$. The two methods must give the same answer, so $r|A| = r|B|$, and $|A| = |B|$.

To prove that a perfect matching exists, we show that Hall's condition holds for an arbitrary subset $S \subseteq A$. As in the previous paragraph, there are $r|S|$ edges with an endpoint in S. Similarly, there are $r|N(S)|$ edges with an endpoint in $N(S)$. By definition, every edge with an endpoint in S has the other endpoint in $N(S)$; in other words, the $r|N(S)|$ edges with an endpoint in $|N(S)|$ include among them all $r|S|$ edges with an endpoint in S. This gives the inequality $r|N(S)| \geq r|S|$, or $|N(S)| \geq |S|$. □

Theorem 15.2 is one of the few results in matching theory that is useful for multigraphs, not just simple graphs. You would not expect this to happen! After all, loops and parallel edges can never help us find a larger matching: a matching has maximum degree 1, which means it is always a simple graph. In fact, bipartite graphs cannot contain loops in the first place, though they can have parallel edges: a loop can never have one endpoint in A and the other in B.

Question:	So why would it matter that Theorem 15.2 is true for multigraphs?
Answer:	It is possible that a bipartite multigraph is only r-regular due to the presence of parallel edges, and if we eliminate those, then checking Hall's condition becomes much harder.

I should emphasize that when we go from an r-regular multigraph to a simple graph, Hall's condition will still be true: it will just no longer follow automatically from Theorem 15.2.

To confirm for ourselves that Theorem 15.2 still does work for multigraphs, we could go through all our proofs and verify that nothing changes in the presence of parallel edges. (We would need

to go through many proofs because of the dependencies between them.) However, there is a shortcut.

Let G be a bipartite r-regular multigraph, and let G' be the simplification of G. We can quickly check that the two-paragraph proof of Theorem 15.2 still works for multigraphs, and therefore G satisfies Hall's condition. Therefore G' also satisfies Hall's condition: adjacent vertices in G are still adjacent in G', so neighborhoods don't change at all. Therefore G' has a perfect matching, which corresponds to a perfect matching in G.

It is possible to generalize the result of Theorem 15.2. Suppose that some vertices in A can have degree more than r, and some vertices in B can have degree less than r. Then $r|S|$ is a lower bound on the number of edges with an endpoint in S, while $r|N(S)|$ is an upper bound on the number of edges with an endpoint in $N(S)$, but this only extends the inequalities we have; we are still able to obtain $r|N(S)| \geq r|S|$ and verify Hall's condition.

Question:	Does this give us a perfect matching?
Answer:	No, because it's now possible that $|A| < |B|$.

However, we can still obtain the following corollary:

Corollary 15.3. *For all r, if G is a bipartite graph with bipartition (A, B) such that every vertex in A has degree at least r, while every vertex in B has degree at most r, then G has a matching that covers A.*

Armed with Hall's theorem and several of its variants, we are now ready to tackle some problems outside graph theory.

15.3 A three-card magic trick

Here is a mathematical card trick I know. A version of it using the ordinary 52-card deck was invented by Fitch Cheney [8], but I will present a simplified version with a deck of 8 cards, numbered 1 through 8.

You choose 3 cards from the deck, however you like, and give them to my assistant, while I look away or even temporarily leave the room. The assistant then hands me two of the cards, one at a time, and hands the third card back to you, for reference.

I have not seen the third card you're holding, at any point. Nevertheless, I can now perform some quick mental arithmetic and name the number on that card!

How is this possible? The answer doesn't involve any hidden communication: my assistant doesn't try to communicate the third card via eyebrow gestures and split-second timing. The answer involves only mathematics. Before describing a particular implementation of this trick, let's understand how it is even possible to make the trick work.

Consider the following bipartite graph. On one side of the bipartition, the vertices are sets of three cards: $\{1, 2, 3\}$ through $\{6, 7, 8\}$. In the magic trick, you choose three cards, which we can represent by a choice of one of these vertices. On the other side, the vertices are ordered

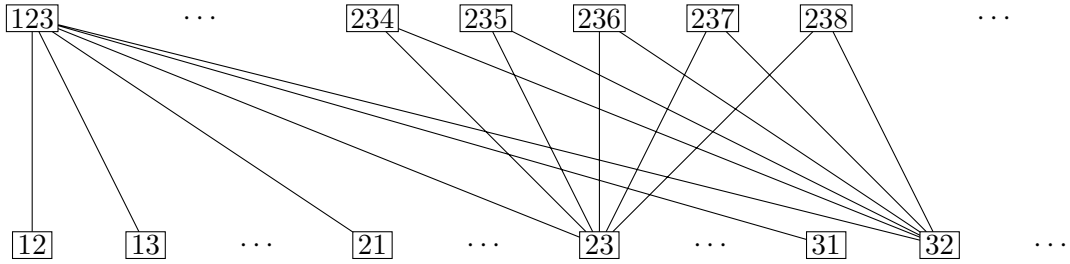


Figure 15.1: The bipartite graph for the three-card magic trick

pairs of two cards: $(1,2)$, $(1,3)$, and so on through $(8,7)$. In the magic trick, this represents the information I get: the order in which the assistant hands me the cards makes the pair an ordered pair.

For the edges, join each set of three cards to the 6 ordered pairs that can be formed from it: these represent the 6 choices my assistant has, in deciding which two cards to give me and in which order. The graph has 112 vertices, so it is too big to draw in full, but Figure 15.1 shows a portion of it: enough to see all the neighbors of vertices $\{1,2,3\}$ on the first side and of $(2,3)$ and $(3,2)$ on the second side.

My assistant's strategy, whatever it might be, picks a designated neighbor of each vertex on the first side of the graph; for example, if $(1,2)$ is the designated neighbor of $\{1,2,3\}$, then if you choose the cards numbered 1, 2, and 3, my assistant will hand me card 1 and then card 2. Let M (for "magic") be the spanning subgraph containing only the edges between a vertex on the first side and its designated neighbor; by construction, every vertex on the first side has degree 1 in M.

I presumably know M ahead of time, so if I am handed card 1 and then card 2, I should think about the edges of M incident to vertex $(1,2)$. If the edge between $\{1,2,3\}$ and $(1,2)$ is the only such edge in M, then I know that the third card must have been 3. If, however, there are multiple edges of M incident to $(1,2)$, then I have many theories, and cannot identify the third card. In other words, the magic trick only works if every vertex on the second side has degree at most 1 in M: if M is a matching!

Does such a matching exist? A first step is to check the number of vertices. There are $\binom{8}{3} = \frac{8\cdot 7\cdot 6}{3!} = 56$ vertices on the first side: the number of ways to choose a set of 3 objects out of 8. There are $8 \cdot 7 = 56$ vertices on the second side: the number of ways to choose a pair of 2 distinct objects out of 8. So it's conceivable to have a matching M that covers all 56 vertices on the first side, as long as it also covers all 56 vertices on the second side: it must be a perfect matching. We haven't ruled out the possibility that the magic trick works.

Question:	Why do we divide by 3! in $\frac{8\cdot 7\cdot 6}{3!}$, but don't divide by anything in $8 \cdot 7$?
Answer:	In the first case, we're counting sets, which are unordered, so we divide by the 3! ways to order the elements. In the second case, we're counting ordered pairs: my assistant hands me a first card and a second card.

To see that the graph for the 8-card magic trick has a perfect matching, all we have to do is observe that it is 6-regular and apply Theorem 15.2.

Question:	Why is the graph 6-regular?
Answer:	For every pair of cards I see, there are 6 possibilities for the missing third card. For every three cards my assistant sees, there are 6 possible actions: 3 ways to choose the first card to pass me, multiplied by 2 ways to choose the second card.

We can generalize to a k-card magic trick: you choose k cards from a (potentially very large) deck, my assistant hands me $k-1$ of them in some order, and I must guess the last card.

Proposition 15.4. *The k-card magic trick can be performed if the deck contains at most $k!+k-1$ cards.*

Proof. Suppose the deck contains n cards; construct the bipartite graph with bipartition (A, B) where A is the set of all unordered k-card hands, B is the set of all ordered sequences of $k-1$ cards, and there is an edge whenever a k-card hand contains all cards in the $(k-1)$-card sequence.

Then every vertex in A has degree $k \cdot (k-1)! = k!$: when my assistant takes a k-card hand and hands me $k-1$ cards in some order, there are k ways to choose which card I don't get to see, and $(k-1)!$ ways to sort the cards I do see. Every vertex in B has degree $n-k+1$: when I am handed $k-1$ cards, the number of k-card hands they could have come from is equal to the number of possibilities for the kth card, which is $n-(k-1)$ because that card can be any of the cards except for the $k-1$ I see.

If we let $r = k!$, then every vertex in A has degree at least (actually, exactly) r. Provided $n-k+1 \leq r$, every vertex in B has degree at most r; this inequality is equivalent to $n \leq k!+k-1$. When this happens, by Corollary 15.3, the graph has a matching M that covers all vertices in A.

At least in principle, if my assistant and I agree on a matching M, then we have a strategy for the k-card trick. When handed k cards, my finds that k-card hand as a vertex in M, and looks at the only neighbor of that vertex to see which $k-1$ cards to give me, and in which order. When I am handed that sequence of $k-1$ cards, I can do the reverse: find that sequence as a vertex in M, and look at the only neighbor of that vertex to see which k cards were chosen. (If $n < k!+k-1$, not all sequence of $k-1$ cards are covered by M, but every sequence my assistant actually hands is guaranteed to be covered!) This tells me, in particular, which card is the hidden card. □

Question:	What's the problem with implementing this strategy?
Answer:	For an arbitrary matching, too much memorization is necessary! Essentially, my assistant and I have to memorize what to do in every possible situation.

In 2002, Michael Kleber [4] not only proved Proposition 15.4, but also described a strategy for the k-card trick that can be implemented in practice, and I will finish our discussion of magic tricks by describing it in the three-card case.

My assistant's job is to add up the numbers on the three cards, and find the remainder when the sum is divided by 3. The card my assistant hands back to you is the smallest of the three cards if the remainder is 0, the middle card if the remainder is 1, and the largest card if the remainder is 2. My assistant hands me the other two cards in increasing order if the missing card is one of the three smallest cards I won't see, and in decreasing order otherwise.

Question:	What will my assistant do if you choose the cards $1, 3, 6$?
Answer:	$1+3+6 = 10$, which leaves a remainder of 1 when divided by 3, so my assistant will hand you the middle card: 3. Of the cards I won't see, the three smallest ones are $2, 3, 4$, of which 3 is one, so my assistant hands me 1, then 6.

My job is to label the unseen cards, in order from largest to smallest, as <2, <1, <0, >2, >1, and >0. I then guess the one where the inequality sign ($<$ or $>$) matches the relationship between the first and second card I am given, and the number (0, 1, or 2) matches the remainder when the sum of my two cards is divided by 3.

Question:	What will I do if I am handed the cards 1 and 6 in that order?
Answer:	Since $1 < 6$ and $1 + 6$ leaves a remainder of 1 when divided by 3, I want the card labeled <1: the second-smallest unseen card. So I guess card 3.

Though this strategy is workable, and generalizes well, I wonder if there's a simpler procedure specifically for the three-card magic trick. Maybe you will find one!

15.4 Generalized tic-tac-toe

The ordinary game of tic-tac-toe is played on a 3×3 grid. Players take turns placing their mark to claim an empty space in the grid: a $\times$ for the first player and a $\bigcirc$ for the second player. A player wins by claiming all three spaces in a horizontal, vertical, or diagonal line; if this does not happen when the board is filled, then the game is a draw.

The ordinary game of tic-tac-toe is not very interesting once you've learned how to play: every game between two skilled players ends in a draw. However, there are many ways to generalize it. The game gomoku is played on a 15×15 board, and the winning condition is to get 5 consecutive pieces in a row. Mathematicians have also tried playing tic-tac-toe in three (or more?) dimensions. A $3 \times 3 \times 3$ game is not very interesting, because the first player has an insurmountable advantage when claiming the center space, but $4 \times 4 \times 4$ tic-tac-toe is worth playing. You just have to learn to spot the winning lines (one example is shown in Figure 15.2).

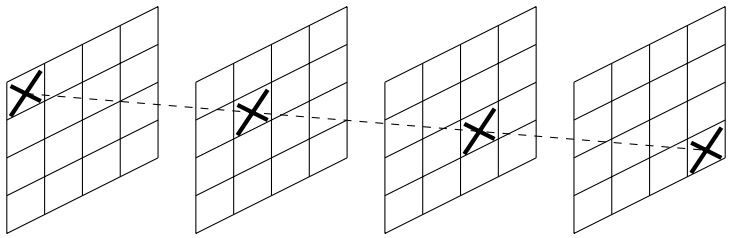


Figure 15.2: A winning line in $4 \times 4 \times 4$ tic-tac-toe

We will consider tic-tac-toe in even further abstraction: there is a set of points $\mathcal{P}$, which the players take turns claiming, and a set of winning lines $\mathcal{L}$. Each element of $\mathcal{L}$ is a subset of $\mathcal{P}$, and a player that claims all the elements of a winning line wins.

Graph theory cannot solve all possible tic-tac-toe games at once, but an application of Hall's theorem can let us stop thinking about games that are too boring: either player can easily guarantee a draw by a strategy with very little interaction.

Proposition 15.5. *In a generalized tic-tac-toe game where every set of k winning lines contain at least $2k$ points in total, either player can guarantee themselves at least a draw.*

Proof. We will construct a somewhat unusual bipartite graph to represent the game. On one side of the bipartition, we will have $\mathcal{P}$: the set of points. On the other side of the bipartition, we will not put $\mathcal{L}$, but rather two disjoint sets called $\mathcal{L}^+$ and $\mathcal{L}^-$. For every winning line $\ell \in \mathcal{L}$, we put a "positive vertex" ℓ^+ into $\mathcal{L}^+$ and a "negative vertex" ℓ^- into $\mathcal{L}^-$. Both ℓ^+ and ℓ^- will be adjacent to the same set of points in $\mathcal{P}$: all the points contained in ℓ.

Next, we set out to check Hall's condition for a matching in this graph to cover $\mathcal{L}^+ \cup \mathcal{L}^-$. Let $S \subseteq \mathcal{L}^+ \cup \mathcal{L}^-$ be an arbitrary set. Then either $|S \cap \mathcal{L}^+|$ or $|S \cap \mathcal{L}^-|$ is at least $\frac{1}{2}|S|$: either at least half the vertices in S are positive, or at least half the vertices are negative. The two situations are symmetric, so we assume $|S \cap \mathcal{L}^+| \geq \frac{1}{2}|S|$.

If $k = |S \cap \mathcal{L}^+|$, then there are k winning lines corresponding to vertices in $S \cap \mathcal{L}^+$, and by the condition we've assumed, there are at least $2k$ points on these lines. All these points are in $N(S \cap \mathcal{L}^+)$, and therefore in particular they are in $N(S)$. Therefore $|N(S)| \geq 2k \geq |S|$, and Hall's condition holds.

Hall's theorem gives us a matching in our unusually constructed graph, but what do we do with this matching? Well, for every line ℓ, the matching gives us two points on ℓ: the point matched to ℓ^+ and the point matched to ℓ^-. In other words, from every winning line we've selected two points, such that each point has been selected from at most one of the winning lines through it.

Using these selected points, you can obtain a draw whether you are the first player or the second. Suppose your opponent has played on one of the points selected from winning line ℓ. Then respond by claiming the other point selected from ℓ, if you have not claimed it already. In all other cases—if your opponent's move is not the point selected from any line, or if you've already claimed the other point, or if it's the first move of the game—play arbitrarily. (A strategy of this type is called a pairing strategy in game theory.)

Question:	What if your opponent plays on that point with the goal of winning along some line other than ℓ?
Answer:	You don't care, because that other line has two selected points of its own.

If you use the pairing strategy, you will never lose, because your opponent can never claim all the points on any line. To do so, eventually your opponent would need to claim one of the selected points from that line, but then you will just claim the other point. You will probably not win, either, because the pairing strategy makes absolutely no effort to do so; it's possible that you will win accidentally. However, you are guaranteed at least a draw. □

The condition of Proposition 15.5 seems hard to check, but for many tic-tac-toe games, it turns out k winning lines will contain far more than $2k$ in most cases, leaving only a few cases to check. For example, consider a 5×5 board. A single line is guaranteed to have 5 points; a second line is guaranteed to add 4 new ones; a third line is guaranteed to add 3 new ones, for a total of 12.

Question:	How can we argue rigorously that 3 lines contain at least 12 points?
Answer:	They have $5 + 5 + 5 = 15$ points if we double-count the intersections, and 3 lines have at most 3 intersection points.

Since 12 points is enough for up to 6 lines, we can skip ahead to considering a set of 7 lines. These cover most of the board, and with a little bit more thinking, we can conclude that the worst case is a set of all $k = 12$ lines, which cover all 25 points. Therefore Proposition 15.5 applies to tic-tac-toe on a 5×5 board.

15.5 Incomparable sets

Our setting in this section will be the world of subsets of an n-element sets; we might as well assume they are subsets of $\{1, 2, \ldots, n\}$, because the exact elements won't matter.

Two subsets X and Y are called *comparable* if $X \subseteq Y$ or $Y \subseteq X$, and *incomparable* otherwise. For example, the set $\{2, 3\}$ is comparable to (and a subset of) the set $\{2, 3, 4\}$; it is incomparable to $\{1, 3, 4\}$. You can imagine that if the elements represent objects you want to own, then $\{2, 3\}$ is clearly not as good as $\{2, 3, 4\}$, but it is not certain which of $\{2, 3\}$ or $\{1, 3, 4\}$ is better. For example, what if elements 1, 3, and 4 are different kinds of cookies, while element 2 is an expensive car?

Suppose we want to pick a family[6] of sets in which no two sets are comparable. An example is the family $\big\{\{1, 2, 3\}, \{2, 4\}, \{1, 3, 4\}\big\}$.

[6] A "family" of sets is just a set of sets, but we give it a different word to make it easier to distinguish it from its elements.

Question:	What is largest family of subsets of $\{1, 2, 3, 4\}$ in which no two are comparable?
Answer:	$\big\{\{1, 2\}, \{1, 3\}, \{1, 4\}, \{2, 3\}, \{2, 4\}, \{3, 4\}\big\}$, the family of all 2-element subsets.

In general, if we take the family of all k-element subsets, for any k, then any two subsets in the family will be incomparable. The number of sets in this family is given by the binomial coefficient $\binom{n}{k} = \frac{n!}{k!(n-k)!}$. This is maximized when $k = n/2$: for example, when $n = 4$ we have

$$\binom{4}{0} = 1, \ \binom{4}{1} = 4, \ \binom{4}{2} = 6, \ \binom{4}{3} = 4, \ \binom{4}{4} = 1.$$

So we want the family of all $(n/2)$-element subsets. We will call this the *middle layer* family.

Question:	What if n is odd?
Answer:	When n is odd, there are two equally good middle layers: the family of $\frac{n-1}{2}$-element subsets, and the family of $\frac{n+1}{2}$-element subsets. (For example, if $n = 5$, then $\binom{5}{2} = \binom{5}{3} = 10$.

By convention, let's round $n/2$ down if n is odd. The notation for rounding down is $\lfloor n/2 \rfloor$, so the formula $\binom{n}{\lfloor n/2 \rfloor}$ tells us the size of the middle layer family.

In 1928, Emanuel Sperner proved [10] that this is the best we can do, for any n. He did not use Hall's theorem to do this, but we will.

Theorem 15.6. *If $\mathcal{F}$ is a family of subsets of $\{1, 2, \ldots, n\}$ such that no two different sets $X, Y \in \mathcal{F}$ are comparable, then $|\mathcal{F}| \leq \binom{n}{\lfloor n/2 \rfloor}$.*

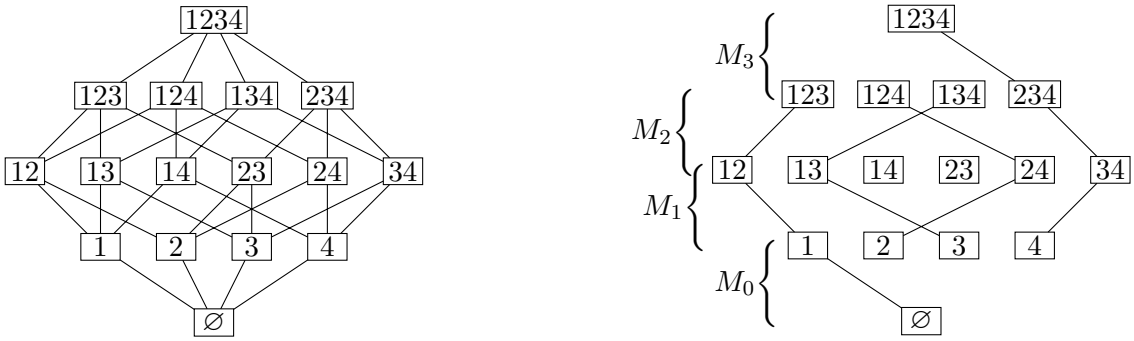
Proof. In Chapter 4, I mentioned that the hypercube graph Q_n can be useful when reasoning about set families, and that's what we will do here. We defined the vertex set of Q_n to be the set of n-bit strings: sequences $b_1 b_2 \ldots b_n$ where each b_i is either 0 or 1. To relate this to set families, a vertex $b_1 b_2 \ldots b_n$ can be identified with the subset of $\{1, 2, \ldots, n\}$ containing an element i whenever $b_i = 1$. The edges, which join n-bit strings that differ in only one position, now tell us which two sets differ only by the presence or absence of one element. For the rest of the proof, I will switch to using set-related terminology. Figure 15.3a shows, for example, Q_4 with the vertices interpreted as subsets (and $\{x, y, z\}$ written as xyz to simplify the diagram).

We proved in Proposition 13.2 that Q_n as a whole is bipartite, but a perfect matching in Q_n will not help us. Instead, we will write Q_n as a union of bipartite graphs:

$$Q_n = G_{n,0} \cup G_{n,1} \cup \cdots \cup G_{n,n-1}$$

where $G_{n,k}$ is the subgraph of Q_n induced by the sets of size k or $k + 1$. In other words:

- $G_{n,k}$ has the bipartition (A, B) where A is the set of k-element subsets of $\{1, 2, \ldots, n\}$, and B is the set of $(k + 1)$-element subsets;



(a) Q_4, interpreting vertices as subsets

(b) $H = M_0 \cup M_1 \cup M_2 \cup M_3$

Figure 15.3: Diagrams accompanying the proof of Theorem 15.6

- An edge xy with $x \in A$ and $y \in B$ exists whenever we can add one more element to x to make y. (It will be important later that when this happens, x is a subset of y.)

In $G_{n,k}$, every vertex $x \in A$ has $n-k$ neighbors in B: there are $n-k$ elements of $\{1, 2, \ldots, n\}$ not already in x which we can add. Every vertex $y \in B$ has $k+1$ neighbors in A: it has $k+1$ elements which we can remove. Provided $n-k \geq k+1$, or $k \leq \frac{n-1}{2}$, Corollary 15.3 applies, giving us a matching in $G_{n,k}$ that covers A.

Question:	What kind of matching can we get if $k \geq \frac{n-1}{2}$?
Answer:	In this case, Corollary 15.3 would apply if we switched the roles of A and B, so there is a matching that covers B.

In either case, call this matching M_k. As a general rule, M_k covers the side of $G_{n,k}$ with fewer elements, which is the side further from the middle layer(s).

Now the magic happens! Consider the union $M_0 \cup M_1 \cup M_2 \cup \cdots \cup M_{n-1}$, and call it H. Figure 15.3b shows one possible union H in the case $n = 4$.

Consider a vertex of Q_n corresponding to a subset of size k, for any k. This vertex has degree 1 or 2 in H: if $k \leq \frac{n-1}{2}$, it is incident to one edge in M_k and maybe also one edge in M_{k-1}, while if $k \geq \frac{n-1}{2}$, then it is guaranteed one edge in M_{k-1} and possibly also an edge in M_k. We can follow these edges "up" the graph (increasing the number of elements) and "down" the graph (decreasing the number of elements) until we hit a dead end—a vertex of degree 1. So the connected component containing the vertex we chose is a path, which means that every component of H is a path.

How many connected components are there? Well, from each vertex, we can always follow edges of H to get closer to the middle layer(s): those are the guaranteed edges. Therefore each component of H contains one vertex in the middle layer (or in each middle layer, for odd n). There are $\binom{n}{\lfloor n/2 \rfloor}$ vertices in the middle layer, and therefore there are $\binom{n}{\lfloor n/2 \rfloor}$ components.

So far, we have only looked at the structure of subsets of $\{1, 2, \ldots, n\}$, but now we are ready to consider the set family $\mathcal{F}$ that Theorem 15.6 is all about. You see, $\mathcal{F}$ can never contain two sets in the same connected component of H. If it did, then we could start at the set with fewer elements, and follow it up the path to get to the one with more elements. At each step, we're

adding an element, so the first set will be a subset of the second—but we've assumed that $\mathcal{F}$ does not contain two such sets.

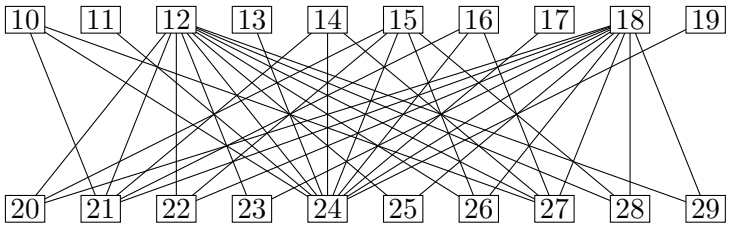
Therefore $\mathcal{F}$ contains at most one set from each connected component of H; since there are $\binom{n}{\lfloor n/2 \rfloor}$ components, we know that $\mathcal{F}$ contains at most $\binom{n}{\lfloor n/2 \rfloor}$ elements. $\square$

15.6 Practice problems

1. Is it possible to circle a letter in each of the words below so that all 9 circled letters are different? Either find a way to do it, or prove that it's impossible using Hall's condition.

AREA	APART	ERRATA
GRAPH	PAPER	RETREAT
THEORY	TREE	YOGA

2. The multiples-of-six graph from Chapter 13 is shown again below:



   Prove that it does not have a perfect matching, but this time, using Hall's theorem.

3. Let G be a bipartite graph with n vertices on each side of the bipartition whose minimum degree $\delta(G)$ is greater than $n/2$.

   Prove that G has a perfect matching.

4. Prove that when $n > k! + k - 1$, it is impossible to perform the k-card magic trick with an n-card deck and guarantee that I guess the kth card. (This is not a problem about graph theory; it is a matter of counting.)

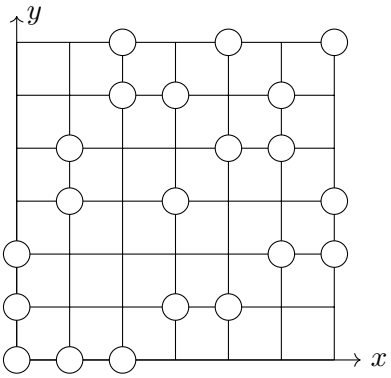
5. Find a pairing strategy for tic-tac-toe on a 5×5 board, where the goal is to win by claiming all 5 spaces along a horizontal, vertical, or diagonal line.

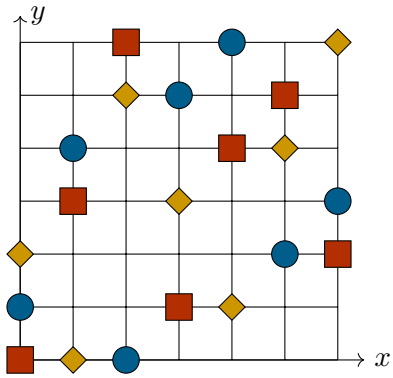
6. Let G be a bipartite graph, with bipartition (A, B), that has the following properties:

   - Every vertex on side A has degree 3 or 5;
   - Every vertex on side B has degree 2 or 4;
   - There are no edges between vertices of degree 3 and vertices of degree 4.

   Prove that G has a matching that covers all vertices in A.

7. Suppose that we mark several points at integer coordinates in the xy-plane in such a way that for every marked point (a, b), the lines $x = a$ and $y = b$ each contain two other marked points. This can be done by making a 3×3 grid, or in other, more complicated ways, such as in the first diagram below.





Prove that it is guaranteed to be possible to give the marked points 3 different colors, as in the second diagram above, so that no horizontal or vertical line passes through two marked points of the same color.

8. (Putnam 2012) A round-robin tournament of $2n$ teams lasted for $2n-1$ days, as follows. On each day, every team played one game against another team, with one team winning and one team losing in each of the n games. Over the course of the tournament, each team played every other team exactly once. Can one necessarily choose one winning team from each day without choosing any team more than once?

9. An $n \times n$ Latin square is an $n \times n$ grid filled with the integers 1 through n in such a way that every row and every column contains each integer exactly once. For example, the first 5×5 grid below is a Latin square.

1	2	3	4	5
3	1	4	5	2
5	3	1	2	4
2	4	5	1	3
4	5	2	3	1

1	2	3	4	5
5	3	4	1	2

Suppose that, as in the second grid above, the first r rows in the $n \times n$ grid are filled with integers 1 through n in a way that does not cause any contradictions: each of the r rows use each integer exactly once, and no column contains any duplicates. Prove that we can fill in the last $n-r$ rows to get a Latin square.

16 Matchings in general graphs

The purpose of this chapter

In the previous three chapters, we discussed matchings in bipartite graphs; here, we are going to take another chapter to consider matchings in general graphs. This division is not because the generalization is easy, but because it is hard. We will have to do quite a bit of work just to prove Tutte's theorem (Theorem 16.1): the equivalent of Hall's theorem in cases where the graph is not necessarily bipartite.

There is a lot more that we're going to leave undone. We proved Kőnig's theorem which deals with the size of a maximum matching; we will not prove the Tutte–Berge formula, which generalizes Tutte's theorem to a Kőnig-type result. We used an algorithm to find maximum matchings in bipartite graphs. There is a more general algorithm, known as the blossom algorithm, which can handle non-bipartite graphs, but we will not discuss it; it is considerably more complicated. If you would like to learn more about matchings, I recommend Lovász and Plummer's book *Matching Theory* [9].

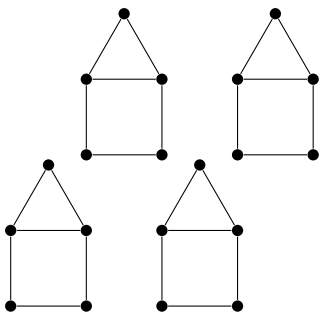
Compared to the material on Tutte's theorem, the second half of the chapter on 1-factorizations is less intense, and gives some graph-theoretical solutions to problems that come up in real life. A more relaxed path through this chapter is to learn about Tutte sets and the statement of Tutte's theorem, then dive into 1-factorizations and the geometric proof of Theorem 16.3.

There are a lot of other problems about factorizations I could have included instead of the section on increasing walks. In the end, though, I felt that many people could write a chapter about decomposing K_n into triangles or Hamilton cycles, but if I did not mention this beautiful problem in my textbook, it is not likely that someone else would write about it in theirs.

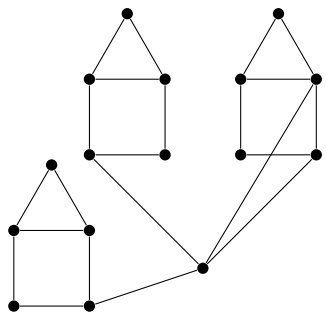
16.1 Tutte sets

In the case of bipartite graphs, we have a complete list of all the reasons that a perfect matching might fail to exist: the violations of Hall's condition. We can summarize these violations as problems of insufficiency: some vertices do not have enough neighbors for us to get a perfect matching.

If our graphs don't have to be bipartite, a second reason that we might not have a perfect matching appears, and that is parity. The complete graph K_{99}, or indeed K_{2n+1} for any n, does not have a perfect matching, even though it has all the edges we could wish for, simply because the number of vertices is odd. Each edge in a matching covers 2 vertices, and so the number of vertices covered by a matching is always even.



(a) Four houses



(b) Three houses and one more vertex

Figure 16.1: Obstacles to the existence of a perfect matching

Question:	Why didn't we need to worry about parity when proving theorems about bipartite graphs?
Answer:	In a graph with bipartition (A, B), we already know that a perfect matching only exists if $\lvert A\rvert = \lvert B\rvert$. When this happens, the total number of vertices is guaranteed to be even.

If we only had to worry about whether the total number of vertices is odd or even, that wouldn't be so bad. But there are examples showing that our life can be even worse:

Question:	In Figure 16.1a, the total number of vertices is even, and yet there's still a parity issue stopping us from having a perfect matching. How?
Answer:	Each house is a connected component with an odd number of vertices, so it has no perfect matching, and different houses cannot help each other.

We will use this idea throughout this chapter, and so we will say that a connected component with an odd number of vertices is an *odd component*, for short. However, odd components all by themselves are not the only problem.

Question:	In Figure 16.1b, there is a parity issue even though the graph has no odd components. How?
Answer:	Each of the three houses has no perfect matching on its own, and the extra vertex can be matched to only one of the houses.

We can imagine that the three houses in Figure 16.1b are on fire, and the vertex in place of the fourth house is a superhero that can put the fires out. However, the superhero can only be in one place, and cannot put out all the fires. This problem is a combination of parity and insufficiency: we have three parity problems in the graph, but only one vertex that can help us fix them.

We can think of Hall's theorem (Theorem 15.1) as saying that if a bipartite graph does not have a perfect matching, then we can summarize why, by giving an example where Hall's condition

fails. Similarly, we would like to have somewhere to point the finger of blame when a general graph does not have a perfect matching.

In both examples in Figure 16.1, we were able to tell a story about what went wrong, but can we generalize? The generalization was first found in 1947 by William Thomas Tutte [13].

We say that a set of vertices U in a graph G is a *Tutte set* if the graph $G - U$ has more than $|U|$ odd components.

Question:	Is there a Tutte set in Figure 16.1b?
Answer:	Yes: the "superhero" vertex in the bottom right.

Question:	Is there a Tutte set in Figure 16.1a?
Answer:	Yes, and the simplest is the empty set! (This is allowed, and sometimes it is the only option.)

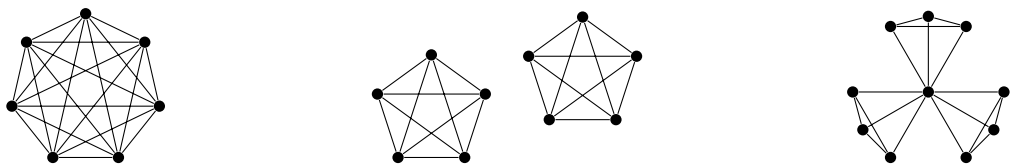
Tutte not only described what we now call Tutte sets, but proved the following theorem showing that they are the only kind of obstacle to be found.

Theorem 16.1 (Tutte's theorem)**.** *Every graph G has a perfect matching if and only if it has no Tutte set.*

As with Hall's theorem, one direction of Tutte's theorem is much shorter than the other, and the proof of that direction is what motivated our definitions: we defined a Tutte set specifically because we had an argument in mind for why a graph with a Tutte set could not have a perfect matching.

Proof of the "only if" direction of Theorem 16.1. Suppose that U is a Tutte set in a graph G, so that $G - U$ has $k > |U|$ odd components. Let M be a maximum matching in G, and let M' be the largest subgraph of M that is also a matching in $G - U$. Then M' must leave at least k vertices of $G - U$ uncovered: at least one from each odd component. However, M cannot do much better! An edge of M is missing from M' only if it has an endpoint in U, and there are at most $|U|$ such edges; these can cover at most $|U|$ of the vertices of $G - U$ that M' left uncovered. Since $k > |U|$, we know that there are still some vertices of $G - U$ that are uncovered by M; therefore M is not a perfect matching. □

To prove the other direction of Tutte's theorem, we will not use Tutte's original argument, which relied on linear algebra. Instead, we will see an argument from Lovász and Plummer's *Matching Theory*. We will arrive at it indirectly, by first describing a special class of graphs called "saturated graphs" for which the proof will be easier.



(a) K_7 (b) Two copies of K_5 (c) A nonempty Tutte set

Figure 16.2: Several examples of saturated graphs

16.2 Saturated graphs

Call a graph G *saturated*[7] if G has no perfect matching, but if e is any edge not already present in G, then adding e to G would create a perfect matching. (This includes the case of odd complete graphs K_{2n+1}: they are saturated because they do not have a perfect matching, and there is no edge e that can be added.)

Starting from any graph that does not have a perfect matching, we can arrive at a saturated graph by adding edges, one at a time, for as long as this is possible without creating a perfect matching.

Question:	How could we do that in Figure 16.1b?
Answer:	We could add edges to each house to make it a copy of K_5, and then add edges from the extra vertex to every vertex of each house.

Figure 16.2 shows several more examples of saturated graphs. The idea of Tutte sets helps us come up with more: if we have a Tutte set, we can add any edges that don't change its structure.

Question:	If we start with the four houses in Figure 16.1a, and turn each house into a copy of K_5, is the result saturated?
Answer:	No: if we add an edge between two houses, that increases the size of the matching, but it's still not a perfect matching. We can go all the way up to a graph with a copy of K_5 and a copy of K_{15} before it's saturated.

Using Tutte's theorem, we can describe what saturated graphs look like without too much trouble. (Proposition 16.2 does not describe saturated graphs completely, but it will be enough for us.)

Proposition 16.2. *If a graph G is saturated, then for some set of vertices U, G contains every edge with at least one endpoint in U, and every edge between two vertices in the same connected component of $G - U$.*

[7] The word "saturated" is commonly used in graph theory for this type of definition, but it usually needs to be qualified with some other adjectives to say what kind of problem G is saturated for. In this book, we will not need this term outside this chapter, so I will just say "saturated" for brevity, even though it's not as precise.

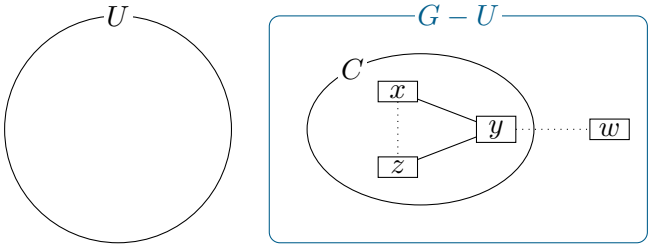


Figure 16.3: Vertices x, y, z, w in the proof of Proposition 16.2; edges xz and yw are not present

Proof of Proposition 16.2 using Tutte's theorem. If a graph G is saturated, then it has no perfect matching, so by Tutte's theorem, it must have a Tutte set U. If an edge e can be added to G that would not change U's status as a Tutte set, then the resulting graph $G + e$ still wouldn't have a perfect matching (again, by Tutte's theorem), which would violate the definition of a saturated graph. Therefore every such edge must already exist in G. Which edges are these?

Well, every edge with at least one endpoint in U is such an edge, because adding it wouldn't affect $G - U$, so U would still be a Tutte set. Also, every edge within a connected component of $G - U$ is such an edge, because adding it wouldn't affect the number of vertices in that component of $G - U$.

Therefore all such edges are already present in G, proving the proposition. □

I made a point to say that this proof used Tutte's theorem, because it means that the proof is not good enough for our purposes. What are those purposes? We are going to reverse the logic, using Proposition 16.2 to prove Tutte's theorem!

16.3 Proof of Tutte's theorem

Let's begin with a fresh proof of Proposition 16.2, so that we can use this proposition in a proof of Tutte's theorem without making the argument circular.

Proof of Proposition 16.2 without using Tutte's theorem. Start by taking U to be the set of all vertices that are adjacent to every other vertex.

Suppose that some connected component C of $G - U$ does not contain every edge it possibly could. Choose $x \in V(C)$ such that not every vertex of C is adjacent to x; let z be a vertex in C at distance 2 from x, and let y be their common neighbor. Because $x, y, z \notin U$, vertex y must not be adjacent to every vertex; there must be a fourth vertex w (potentially outside C) not adjacent to y. This is illustrated in Figure 16.3.

We have two ways to make G bigger: we could add edge xz, or we could add edge wy. Because G is saturated, each of these bigger graphs must have a perfect matching; because G has no perfect matching, those two matchings must each use the added edge. So if we remove the added edge, we see that G has two matchings that are nearly perfect: a matching M_{xz} that covers all vertices except x and z, and a matching M_{wy} that covers all vertices except w and y.

In Chapter 14, to compare two matchings M and N, we looked at their symmetric difference $M \oplus N$. We will do the same here. We know in general that $M_{xz} \oplus M_{wy}$ consists of cycles and

alternating paths: paths that alternate between edges of M_{xz} and edges of M_{wy}. In this case, we can say even more.

Question:	Where can the alternating paths in $M_{xz} \oplus M_{wy}$ start and end?
Answer:	Only at the vertices x, y, z, and w. All other vertices have degree 1 in both M_{xz} and M_{wy}, so they have degree 2 or 0 in $M_{xz} \oplus M_{wy}$.

So $M_{xz} \oplus M_{wy}$ contains an M_{wy}-alternating path that starts at vertex w and ends at one of the three vertices x, y, or z. All three options are good for us:

- If it ends at y, then it's an M_{wy}-augmenting path, since it starts and ends at an uncovered vertex.
- If it ends at x or z, then we can follow up with edge xy or zy to go to y; again, we get an M_{wy}-augmenting path.

Question:	If we follow up a $w - x$ alternating path by going from x to y, why is that path still alternating?
Answer:	We must have arrived to x by an edge of M_{wy}, because M_{xz} leaves x uncovered. Meanwhile, xy is not an edge of M_{wy}, because M_{wy} leaves y uncovered.

By Lemma 14.3, we can use the M_{wy}-augmenting path to improve M_{wy} to a perfect matching in G. But we assumed G was saturated, and didn't have a perfect matching! This is the contradiction that finishes the proof. □

Let me explain the reasons behind what's happened so far. It's common that when proving a theorem about graphs that don't have a property (such as a perfect matching), it is enough to prove the theorem about graphs that don't have the property, but have as many edges as possible: in our case, about saturated graphs. We're about to see in a moment that this is true here: proving Tutte's theorem will be much easier for saturated graphs than for graphs with no special properties.

So we want to learn about saturated graphs. Since we suspect that Tutte's theorem is true even before we have a proof, we started by using it to understand what saturated graphs ought to look like. Then, we went back and proved the same result in legitimate ways, so that it's no longer circular reasoning to use it to prove Tutte's theorem.

Now let's see if our efforts have paid off!

Proof of the "if" direction of Theorem 16.1. Let G be a graph that does not have a perfect matching.

Question:	How can we finish the proof if G has an odd number of vertices?
Answer:	In this case, $U = \varnothing$ is a Tutte set in G, so Tutte's theorem holds.

So we assume that G has an even number of vertices. Then the reason that G doesn't have a perfect matching is simply because it's missing some edges.

Greedily add edges to G, one by one, for as long as this does not create a perfect matching, until we cannot add edges any longer. The result is a saturated graph H that contains G as a spanning subgraph.

By Proposition 16.2 applied to H, there is some set of vertices U such that H contains every edge with at least one endpoint in U, and every edge with both endpoints in the same connected component of $H - U$. We will first prove that U is a Tutte set in H, and then that it is a Tutte set in G.

Let M be a maximum matching in $H - U$. In every connected component C of $H - U$, all edges are present, so the only possible reason why M might not cover every vertex of C is that C is an odd component. What's more, if there are $|U|$ or fewer odd components, then we can extend M to a perfect matching of H: match vertices in U to vertices outside U not covered by M, and if any vertices in U are left, match them to each other. So there must be more than $|U|$ odd components in $H - U$, which exactly means that U is a Tutte set in H.

Question:	Where did we need the assumption that G and H have an even number of vertices?
Answer:	Otherwise, "if any vertices in U are left, match them to each other" might not work: we might end with one vertex uncovered.

Since G is a subgraph of H, every connected component of $H - U$ breaks down into one or more components of $G - U$. However, an odd number cannot be the sum of many even numbers; therefore every odd component of $H - U$ must contain at least one odd component of $G - U$. We conclude that $G - U$ has at least $|U|$ odd components, which means that U is also a Tutte set in G. □

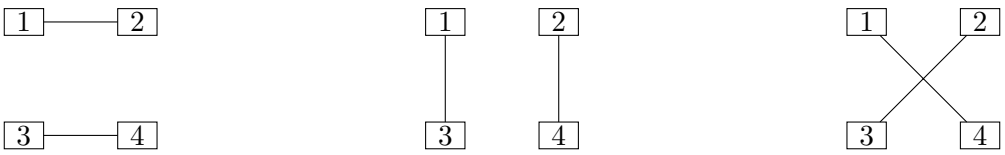
16.4 1-factorizations

A practical application of matchings in graphs that we have yet to consider is tournament design. (We will only explore the basics of the overlap between this field and graph theory.)

Suppose that we would like to organize a round-robin chess tournament[8] between n players. A single chess game can take a while, so we want to schedule as many games at the same time as possible. When n is even, we can always begin by dividing n people into $n/2$ arbitrary pairs and scheduling a game between each pair. This is a perfect matching in K_n, which is not a very difficult matching problem.

However, this only tells us what to do in the first round; future rounds are more difficult, because we don't want to repeat any games. If the first round is a perfect matching M_1 in K_n, then we would like the second round to be a perfect matching M_2 in the complement $\overline{M_1}$, the

[8] "Round-robin", in case you're unfamiliar with the terminology, means that every participant plays a game against every other participant.



(a) The 1-factor $(x_1x_2)(x_3x_4)$ (b) The 1-factor $(x_1x_3)(x_2x_4)$ (c) The 1-factor $(x_1x_4)(x_2x_3)$

Figure 16.4: A 1-factorization of K_4: three disjoint 1-factors whose product is $x_1^3x_2^3x_3^3x_4^3$

third round to be a perfect matching in $\overline{M_1 \cup M_2}$, and so on. These matching problems can easily become impossible if we schedule rounds of the tournament with no foresight.

For example, with 6 players, we hope to finish in 5 rounds, because each player has 5 opponents to face. However, if our first three matchings are poorly chosen, then they might leave us with a graph that has no perfect matching of its own. We would have to ask two people to sit out of the fourth round, and then we would have to schedule 6 rounds total.

Question:	Can this problem really occur? How do we run a 6-player tournament badly?
Answer:	Taking $V(K_6)$ to be $\{1, 2, 3, 4, 5, 6\}$, we might mistakenly begin with three matchings that all match even numbers to odd numbers: for example, $\{12, 34, 56\}$, then $\{14, 25, 36\}$, then $\{16, 23, 45\}$. Now, the remaining graph has two odd components and no perfect matching.

Instead of solving this problem one matching at a time, we will have to solve it all at once: we want to find a decomposition of G, writing at as a union of perfect matchings that share no vertices. There is a special term for this kind of decomposition.

Definition 16.1. *A* **1-*factorization*** *of a graph G is a decomposition of G into perfect matchings: a representation*

$$G = M_1 \cup M_2 \cup \cdots \cup M_k$$

where each M_i is a perfect matching, and each edge of G appears in exactly one M_i.

The term "1-factorization" goes back to the early days of graph theory, where the idea of a graph was more algebraic: an edge xy was really thought of as the product of two variables x and y, and a graph was just the product of such edges. For example, a cycle with vertices $\{x, y, z\}$ would be the expression $(xy)(xz)(yz)$. Some other modern terms have their origin in those days. For example, if you simplify the product that we use to represent our cycle, you get $x^2y^2z^2$; the degree of each variable (in the algebraic sense) is exactly the degree of the corresponding vertex (in the graph-theoretic sense)!

Viewed from this point of view, a 1-factorization really is a factorization of the graph: a way to write it as a product of factors in which every variable has degree 1. (For this reason, perfect matchings are also sometimes called 1-factors.) For example, the graph K_4, viewed as the product $(x_1x_2)(x_1x_3)(x_1x_4)(x_2x_3)(x_2x_4)(x_3x_4)$, has the 1-factorization

$$\underbrace{(x_1x_2)(x_3x_4)}_{\text{1-factor}} \cdot \underbrace{(x_1x_3)(x_2x_4)}_{\text{1-factor}} \cdot \underbrace{(x_1x_4)(x_2x_3)}_{\text{1-factor}}.$$

Figure 16.4 shows this factorization in a more modern way.

Of course, what's important is finding the 1-factorization, not representing it.

Theorem 16.3. *The complete graph K_n has a 1-factorization whenever n is even.*

Proof. It is much easier to draw a diagram of the 1-factorization than it is to give a diagram-free proof. Begin with a diagram of K_n that places $n-1$ of the vertices at evenly spaced points around a circle, and the last vertex in the middle of the circle. For the ith matching M_i, take the edge between the middle vertex to the ith vertex around the circle, as well as all edges perpendicular to this edge in the diagram.

(Figure 16.5 shows an example of this construction in the case $n = 8$.)

In a way, the diagram also leads to a geometric proof. Name the $n-1$ matchings after the $n-1$ radial edges. For each edge xy between two outer vertices, construct the diameter perpendicular to xy, and one half of that diameter will be an edge from the middle vertex. This tells us which matching contains edge xy; in particular, it tells us that xy is in exactly one matching. To see that it's a matching, we first check that xy does not share a vertex with the radial edge perpendicular to it (the lines intersect at the midpoint of edge xy, not at a vertex). In all other cases, two edges xy and $x'y'$ in the same matching are parallel: they do not share a vertex because, as lines, they do not intersect.

For a diagram-free proof, we rely on modular arithmetic instead. Number the vertices 0 through $n-1$. For each $i = 0, 1, \ldots, n-2$, define the matching M_i to contain the edge $\{i, n-1\}$ as well as all the edges $\{i-k \bmod n-1, i+k \bmod n-1\}$ for $k = 1, \ldots, n/2-1$. Since the $n-1$ values

$$i-(n/2-1), i-(n/2-2), \ldots, i-1, i, i+1, \ldots, i+(n/2-1)$$

are distinct modulo $n-1$, no vertices are repeated, and therefore M_i really is a matching.

Next, we show that no edge is contained in multiple matchings. This is true for edges incident to $n-1$, since each matching is defined to contain a different one of these edges. Otherwise, take an edge $xy \in E(M_i) \cap E(M_j)$. Since $xy \in E(M_i)$, we can write it as

$$\{x, y\} = \{i-k \bmod n-1, i+k \bmod n-1\}$$

for some k, so $x+y \equiv (i-k)+(i+k) = 2i \pmod{n-1}$. Similarly, since $xy \in E(M_j)$, then $x+y = 2j \pmod{n-1}$. But n is even and $0 \le i, j \le n-2$, so $2i \equiv 2j \pmod{n-1}$ can only occur if $i = j$.

Question:	What goes wrong at this step if n is odd?
Answer:	Then $2i \equiv 2j \pmod{n-1}$ really is possible for two values i and j. For example, if $n = 7$, we'd be working modulo 6, and $2 \cdot 1 \equiv 2 \cdot 4 \pmod 6$.

From the previous paragraph, it also follows that every edge is contained in some matching. If there are $n-1$ matchings, and each contains $n/2$ edges, then together they contain $\frac{n(n-1)}{2}$ edges total, and we've shown that there's no overlap. But there are only $\frac{n(n-1)}{2}$ edges in K_n, so we've included them all. □

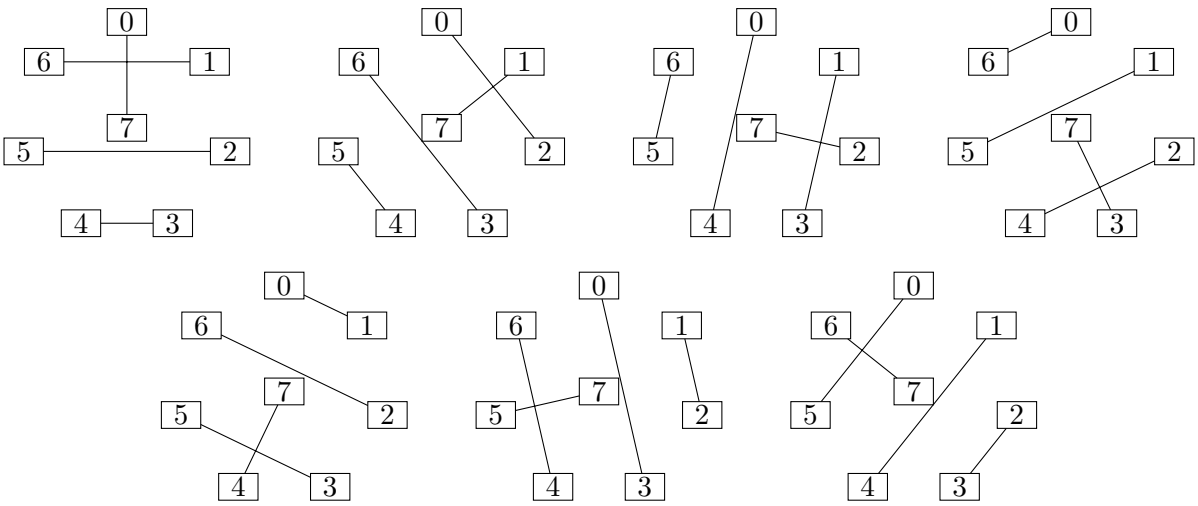


Figure 16.5: The $n = 8$ case of Theorem 16.3

In my opinion, the geometric proof is the "real" reason that Theorem 16.3 is true, but I have included the number-theoretic proof for completeness. It requires some background in number theory to follow, but in reality, working modulo $n - 1$ is just a diagram-free way to put $n - 1$ evenly spaced points around a circle.

Now we know how to schedule a round-robin tournament between n people in just $n-1$ rounds, if n is even!

Question:	What do tournament organizers do if n is odd?
Answer:	In each round, one player gets a "bye" and sits out. This is equivalent to scheduling an $(n + 1)$-player tournament with one player named "bye" who doesn't really exist.

Since $n+1$ is even whenever n is odd, adding a fictional player named "bye" reduces the problem to a case where Theorem 16.3 applies. Of course, with $n + 1$ players, we require n rounds, even if one of the players is fictional. This proves the following corollary:

Corollary 16.4. *When n is odd, the complete graph K_n has a decomposition into n matchings which are each nearly perfect (covering $n - 1$ of the n vertices).*

Many other graphs can be shown to have 1-factorizations. Let's briefly return to bipartite graphs to prove one more result, originally also due to Kőnig [7]:

Theorem 16.5. *Every regular bipartite graph has a 1-factorization.*

Proof. By Theorem 15.2, every regular bipartite graph G has a perfect matching M. If G is k-regular, then $G - E(M)$ is $(k - 1)$-regular: every vertex of G is incident to one edge of M, so its degree goes down by 1 in $G - E(M)$.

Repeat this argument, removing perfect matchings from G until it is 0-regular, and there are no more edges. (Formally, this proof should be rephrased as an induction on k; can you see how?) The perfect matchings removed at each step form a 1-factorization of G. □

16.5 Increasing walks

The problem of finding a 1-factorization of K_n was historically first studied by tournament organizers, not graph theorists. However, that does not mean it is not useful in graph theory. I encountered the following problem as a graduate student.

Take the complete graph K_n, and make it into a weighted graph by giving each edge e a weight $c(e)$; in this problem, we will insist that all the weights should be different. A walk $(x_0, x_1, x_2, \ldots, x_l)$ is called an *increasing walk* if the weights go up along the walk: if

$$c(x_0x_1) < c(x_1x_2) < \cdots < c(x_{l-1}x_l).$$

What is the longest possible increasing walk? Well, it depends on the labels. We can have very long increasing walks if the weights cooperate. Suppose that before choosing edge weights, we pick a walk $(x_0, x_1, x_2, \ldots, x_l)$ that we'd like to be increasing. Provided the walk does not repeat any edges, we can make it so! Simply set $c(x_{i-1}x_i) = i$ for $i = 1, 2, \ldots, l$.

Question:	What is the longest walk in K_n that does not repeat any edges?
Answer:	If n is odd, then all degrees are even, so K_n has an Euler tour: a walk of length $\binom{n}{2}$. If n is even, then we can delete any matching and be left with an Eulerian graph, with an Euler tour of length $\binom{n}{2} - n/2$.

This is a best-case analysis, and the short solution to it shows us why worst-case analyses are much more interesting. Instead, let's ask the question: what is the longest possible length of an increasing walk that we can guarantee, no matter what the weights of the edges are?

This problem was first studied by Ron Graham and Daniel Kleitman, who proved in 1973 [1] that it was possible to find weighted graphs in which the longest increasing walk is much shorter.

Proposition 16.6. *For all even n, there is a weighted complete graph on n vertices in which no increasing walk has length more than $n - 1$.*

Proof. Use Theorem 16.3 to decompose K_n into $n-1$ perfect matchings M_1 through M_{n-1}. For each i, and each $e \in E(M_i)$, set $c(e) = i$. Okay, that doesn't quite work, because the weights all have to be different, but it will have the same effect if $c(e)$ is any number in the interval $[i, i + \frac{1}{2}]$: we will never end up comparing two edges in a matching anyway.

An increasing walk in this weighted graph cannot use more than one edge from any matching M_i. After taking its first edge from that matching, it cannot immediately take a second edge, because no two edges in M_i share an endpoint. So the walk must follow up by going to a different matching: since the walk is increasing, it must select an edge from M_j, for some $j > i$. But this edge has a greater weight than any edge of M_i, so the increasing walk can never return to M_i again.

Since there are only $n-1$ matchings, the increasing walk cannot use more than $n-1$ edges. □

Graham and Kleitman proved more than this. They generalized Proposition 16.6 to work for odd n as well, excluding the special case $n = 3$ and $n = 5$. Moreover, they proved that this upper bound is always achievable! I will present their solution in a different style, as described by Peter Winkler [14] and attributed to Ehud Friedman.

Proposition 16.7. *In every weighted complete graph on n vertices, there is an increasing walk of length at least $n - 1$.*

Proof. Imagine that we put n different people on the n vertices of the complete graph. Then, we call out the edges of the graph one by one, in increasing order of weight. Whenever we call out an edge xy, the two people standing on vertices x and y trade places, walking along edge xy in opposite directions.

At the end, we will ask each person to describe the walk they took. All these walks must be increasing, because all edges were announced in increasing order. Let $l_1, l_2, \ldots, l_n$ be the lengths of the n walks.

Question:	How can we express the sum $l_1 + l_2 + \cdots + l_n$ in another way?
Answer:	The sum counts each edge of the graph twice, since two people walk each edge, so it is equal to $2\lvert E(K_n)\rvert$, which is $n(n-1)$.

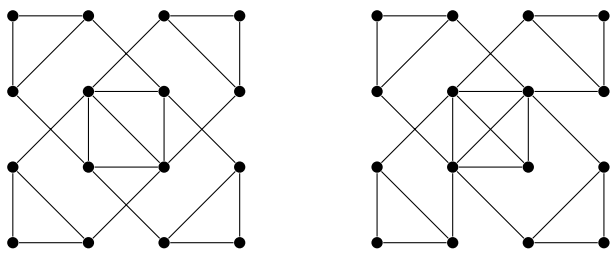
If the lengths $l_1, l_2, \ldots, l_n$ add up to $n(n-1)$, then their average is

$$\frac{l_1 + l_2 + \cdots + l_n}{n} = \frac{n(n-1)}{n} = n - 1,$$

and it is impossible for all n lengths to be below average. Therefore at least one walk must have length at least $n - 1$. □

16.6 Practice problems

1. In one of the graphs below, find a perfect matching. In the other, prove that there is no perfect matching, by finding a Tutte set.



2. Theorem 15.2 from the previous chapter implies that every 3-regular bipartite graph has a perfect matching.

   Prove that the word "bipartite" cannot be left out: give an example of a 3-regular graph which is not bipartite, and does not have a perfect matching.

3. (USAMO 1989) The 20 members of a local tennis club have scheduled exactly 14 two-person games among themselves, with each member playing in at least one game. Prove that within this schedule there must be a set of 6 games with 12 distinct players.

4. Use Tutte's theorem to prove Hall's theorem.

5. Let $o(G)$ denote the number of odd components in a graph G. Prove that if G is a graph and U is a subset of $V(G)$, then

$$\alpha'(G) \leq \frac{1}{2}\big(|V(G)| - o(G-U) + |U|\big).$$

(More is true. The Tutte–Berge formula is a generalization of Tutte's theorem saying that the two sides of this inequality are equal for some set U.)

6. Prove that if G is a graph with an even number of vertices and no perfect matching, then it has a Tutte set S with $o(G-S) \geq |S| + 2$.

7. Suppose that $2n$ people from two different n-person teams participate in a tournament. Describe how to schedule n rounds of simultaneous games between players on opposing teams, so that each person plays once against everyone on the other team.

(In other words, find a 1-factorization of the complete bipartite graph $K_{n,n}$. Theorem 16.5 assures us that one exists, but doesn't say what it is.)

8. Let G be a copy of K_8 with vertices $\{000, 001, \ldots, 111\}$ (just like the vertices of the cube graph Q_3).

   a) For each nonempty subset $S \subseteq \{1, 2, 3\}$, let M_S be the matching consisting of all edges in G whose endpoints differ in the positions numbered by S. For example,

   $$E(M_{\{1,2\}}) = \Big\{\{000, 110\}, \{001, 111\}, \{010, 100\}, \{011, 111\}\Big\}.$$

   Prove that the seven matchings $M_{\{1\}}, M_{\{2\}}, M_{\{3\}}, M_{\{1,2\}}, M_{\{1,3\}}, M_{\{2,3\}}, M_{\{1,2,3\}}$ are a 1-factorization of G.

   b) Prove that this 1-factorization is not isomorphic to the one found in Theorem 16.3. That is, prove that it is not possible to replace the labels $\{000, 001, \ldots, 111\}$ by $\{0, 1, \ldots, 7\}$ in any way to turn this 1-factorization into the one shown in Figure 16.5.

9. Define an increasing path to be a path such that one of the walks representing it is an increasing walk.

We can prove a version of Proposition 16.7 for increasing paths by changing the rules slightly in that proof. Whenever edge xy is called out, if the person on x has already visited y, or the person on y has already visited x, then both people stay put. This ensures that at the end, each person's walk represents an increasing path.

   a) Suppose that each person walks a path of length at most L. Prove that each person is responsible for "rejecting" at most $\frac{L(L-1)}{2}$ edges.

   b) At the end, each of the $\binom{n}{2}$ edges was either walked by two people or rejected by at least one person. Use this to prove the inequality $n \cdot \frac{L}{2} + n \cdot \frac{L(L-1)}{2} \geq \binom{n}{2}$.

   c) Prove $L \geq \sqrt{n-1}$.

Graham and Kleitman proved a similar result, but unlike the problem of increasing walks, the exact worst-case answer is still not known in the case of increasing paths.

10. A graph G is called claw-free if it does not have a copy of the star graph S_4 as an induced subgraph. In other words, no vertex $x \in V(G)$ can have three neighbors y_1, y_2, y_3 with no edges between them.

    Sumner's theorem [12] states that every connected claw-free graph with an even number of vertices has a perfect matching. Prove this using Tutte's theorem.

    (Here is a hint for one possible solution: choose a careful ordering $x_1, x_2, \ldots, x_k$ of the vertices in S, then prove by induction on i that the number of connected components of $G - S$ with a neighbor in $\{x_1, x_2, \ldots, x_i\}$ cannot exceed $i + 1$.)

Bibliography

[1] Ronald Graham and Daniel Kleitman. "Increasing paths in edge ordered graphs". In: *Period. Math. Hungar.* 3 (1973). Collection of articles dedicated to the memory of Alfréd Rényi, II, pp. 141–148. ISSN: 0031-5303. DOI: 10.1007/BF02018469.

[2] András Gyárfás, Jeno X. Lehel, and Zsolt Czap Tuza. "Clumsy packing of dominoes". In: *Discrete Mathematics* 71.1 (1988), pp. 33–46. DOI: 10.1016/0012-365X(88)90028-3.

[3] Philip Hall. "On Representatives of Subsets". In: *Journal of the London Mathematical Society* 10.1 (1931), pp. 26–30. DOI: 10.1112/jlms/s1-10.37.26.

[4] Michael Kleber and Ravi Vakil. "The best card trick". In: *The Mathematical Intelligencer* 24 (2002), pp. 9–11. DOI: 10.1007/BF03025305.

[5] Dénes Kőnig. "Gráfok és mátrixok". In: *Matematikai és Fizikai Lapok* 38 (1931), pp. 116–119.

[6] Dénes Kőnig. *Theorie der endlichen und unendlichen Graphen.* Vol. 72. American Mathematical Society, 2001.

[7] Dénes Kőnig. "Über Graphen und ihre Anwendung auf Determinantentheorie und Mengenlehre". In: *Mathematische Annalen* 77 (1916), pp. 453–465. DOI: 10.1007/BF01456961.

[8] William Wallace Lee. *Math miracles.* Durham, N.C.: Seeman Printery, 1950.

[9] László Lovász and Michael D. Plummer. *Matching Theory.* North Holland, 1986. ISBN: 9780821847596.

[10] Emanuel Sperner. "Ein Satz über Untermengen einer endlichen Menge". In: *Mathematische Zeitschrift* 27.1 (1928), pp. 544–548. DOI: 10.1007/BF01171114.

[11] Ernst Steinitz. *Über die Konstruction der Configurationen* n_3. Druck v. Dr. R. Galle, 1894.

[12] David Sumner. "Graphs with 1-Factors". In: *Proceedings of the American Mathematical Society* 42 (1 1974), pp. 8–12. DOI: 10.2307/2039666.

[13] William Thomas Tutte. "The Factorization of Linear Graphs". In: *Journal of the London Mathematical Society* 1.2 (1947), pp. 107–111. DOI: 10.1112/jlms/s1-22.2.107.

[14] Peter Winkler. "Puzzled: Solutions and sources". In: *Communications of the ACM* 51.9 (2008), pp. 103–103. DOI: 10.1145/1378727.1389570.