DISCRETE MATHEMATICS

for Computer Science

Alexander Golovnev, Alexander S. Kulikov, Vladimir V. Podolskii, Alexander Shen

Welcome!

Thank you for downloading this book! It supplements the Introduction to Discrete Mathematics for Computer Science specialization at Coursera and contains many interactive puzzles, autograded quizzes, and code snippets. They are intended to help you to discover important ideas in discrete mathematics on your own, and to show you corresponding applications of these ideas in computer science.

This book contains material corresponding to the first course in the associated specialization at Coursera, Mathematical Thinking in Computer Science. Future editions will cover the additional four courses, Combinatorics and Probability, Graph Theory, Number Theory and Cryptography, and Delivery Problem.

There are 180 problems and 90 code snippets in the book. Most of the problems come with solutions and more than 50 are graded automatically (allowing you to check your solution immediately). We're constantly working on extending and improving this book. Please ask questions, report typos, and suggest improvements through this form. Check https://leanpub.com/discrete-math for updates of the book.

Contents

0 About the Book . . . 5

0.1 Active Learning 5
0.2 Problem-based Learning 5
0.3 Python Programming Language 6
0.4 Acknowledgments 7

I Mathematical Thinking in Computer Science

1 Proofs: Convincing Arguments . . . 11

1.1 Warm Up 11
1.2 Existence Proofs 18

2 Finding an Example . . . 29

2.1 How to Find an Example 29
2.2 Optimality 37
2.3 Computer Search 47

3 Recursion and Induction . . . 55

3.1 Recursion 55
3.2 Induction 73

4 Logic . . . 89

4.1 Examples and Counterexamples 89
4.2 Logic 93
4.3 Reductio ad Absurdum 99

5 Invariants 105

5.1 Double Counting 105

5.2 Invariants 107

5.3 Termination 108

5.4 Even and Odd Numbers 112

6 Project: 15-Puzzle 117

6.1 The Puzzle 117

6.2 Permutations and Transpositions 118

6.3 Why 15-puzzle Has No Solution 129

6.4 When 15-puzzle Has a Solution 131

6.5 Implementation 138

7 Appendix 143

7.1 Cutting a Figure 143

7.2 Using SAT-solvers 143

7.3 Using ILP-solvers 147

7.4 Visualizing Football Fans 149

0. About the Book

0.1 Active Learning

This book covers ideas and concepts in discrete mathematics which are needed in various branches of computer science. To make the learning process more efficient and enjoyable, we use the following *active learning components* implemented through our Introduction to Discrete Mathematics for Computer Science specialization at Coursera.

Interactive puzzles provide you with a fun way to "invent" the key ideas on your own. The puzzles are mobile-friendly, so you can play with them anywhere. The goal of every puzzle is to give you a clean and easy way to state problems where nothing distracts you from inventing a method for solving it. In turn, the corresponding method usually has a wide range of applications to various problems in computer science.

Autograded quizzes allow you to immediately check your understanding after learning a new concept or idea.

Code snippets are helpful in two ways: 1) they show you how ideas from discrete mathematics are used in programming, and 2) they serve as interactive examples and challenges: tweak the given piece of code, run it, and see what happens.

Programming challenges will help you to solidify your understanding. As Donald Knuth said, "I find that I don't understand things unless I try to program them."

0.2 Problem-based Learning

Throughout the book (and the associated specialization at Coursera) we follow a "try this before we explain everything" approach: we always ask you to solve a problem first, and then we explain how to solve it and introduce important ideas needed to solve it. We believe, this way you will get a deeper understanding and also develop a better appreciation for the beauty of the underlying ideas (not to mention the self-confidence that you get if you invent these ideas on your own!). Don't be discouraged if you can't solve all the problems. Just having attempted them is often enough to engage your mind, and make you more curious about the solution.

We use the following two basic types of questions in the book.

Stop and think questions invite you to slow down and contemplate the current material before continuing to the next topic. We *always* provide an answer to the corresponding question right after it. We strongly encourage you (as the name suggests) to stop and do your best to answer the question.

Problems usually require more effort to solve. We use some of them to warm you up and to develop your curiosity. Such problems are followed by detailed solutions. Some other problems are left for you as exercises.

Many questions in the book are *graded automatically* through Coursera. They are marked with:

Try it online!

Note that such a link is *clickable*: clicking on it will open a browser page where you can submit your answer and get instant feedback. This requires an active subscription to our Coursera specialization. At the same time, the book is self-contained: if you are unable to watch the videos and access the interactive puzzles at Coursera, just read the book and solve the problems on a piece of paper.

0.3 Python Programming Language

0.3.1 Why Programming?

Why on earth do we start the book with discussing a programming language? After all, this is a math (rather than programming) book!

That's true. But we believe that many pieces of code shown in this book will help you in many ways:

- They will show you a rich variety of applications of discrete math ideas in various branches of computer science.
- Code snippets can serve as interactive examples: you may want to tweak the given piece of code, run it, and see what happens.
- By trying to implement a particular idea, you are forced to understand every single detail of it.
- It is often easier to reason in terms of specific objects in programming rather than abstract mathematical concepts.

We have set up everything in a way that will allow you to run the code snippets used in this book even if you have never tried to write a program before. You don't even need to install or set up anything: everything can be run in the cloud, through your Internet browser. At the same time, we also provide instructions for those who would like to learn the basics of `Python` while learning discrete math.

0.3.2 Why Python?

OK, let's do some programming while learning discrete math. But why `Python` *instead of any other popular programming language?*

Let us convince you that `Python` is an excellent choice for our purposes.

High-level language. It is particularly easy to start using `Python` (even if you haven't programmed before). The syntax is reader friendly (and close to a natural language). The code is compact: most of the pieces of code in this book are less than ten lines long!

Interactive mode. It can be used in an interactive mode (also known as REPL, for read-eval-print loop). This allows you to talk to your computer using `Python` as a language: the computer then *reads* your input, *evaluates* it, and *prints* the result. This way, you work stuff out and get instant feedback from the machine.

"Batteries included". The `Python` standard library offers a wide range of facilities, and many external libraries are available as well. In particular, this will allow us to generate a random sequence, plot a function, and draw a graph in just one line of code!

This (partly) explains why `Python` is often used for software prototyping, and in such areas as machine learning, data science, and web development.

Of course, advantages always come at the cost of some disadvantages. The high-levelness of `Python` makes it less flexible in performance tuning. This is OK for us, as we will only be using simple snippets of code where optimizing is not an issue.

0.3.3 How to Catch Up with Python?

OK, let's try! Where do I start?

Locally. To install Python on your machine, go to the Get Started section of python.org and follow the instructions. If you are new to Python, we encourage you to install PyCharm to start working with Python: this (free of charge) professional IDE will make the process of writing and running your code smoother and more efficient.

In the cloud. Alternatively, you may run all our code snippets from your Internet browser, without installing or configuring anything on your machine. For this, visit the page github.com/alexanderskulikov/discrete-math-python-scripts and click the badge "Open in Colab". This will show you a list of notebooks that can be run in an interactive mode right in your browser (together with links to a tutorial on notebooks).

0.4 Acknowledgments

This book was greatly improved by the efforts of a large number of individuals whom we owe a debt of gratitude.

We thank the students of the Coursera specialization as well as the students of the Modern Software Engineering B.Sc. program at St. Petersburg State University for their continuous and valuable feedback.

We also thank Huck Bennett, Marie Brodsky, Anuj Kumar Karmakar, and Terence Minerbrook for carefully reading an earlier draft of this book.

I Mathematical Thinking in Computer Science

1 Proofs: Convincing Arguments 11
1.1 Warm Up
1.2 Existence Proofs

2 Finding an Example 29
2.1 How to Find an Example
2.2 Optimality
2.3 Computer Search

3 Recursion and Induction 55
3.1 Recursion
3.2 Induction

4 Logic 89
4.1 Examples and Counterexamples
4.2 Logic
4.3 Reductio ad Absurdum

5 Invariants 105
5.1 Double Counting
5.2 Invariants
5.3 Termination
5.4 Even and Odd Numbers

6 Project: 15-Puzzle 117
6.1 The Puzzle
6.2 Permutations and Transpositions
6.3 Why 15-puzzle Has No Solution
6.4 When 15-puzzle Has a Solution
6.5 Implementation

7 Appendix 143
7.1 Cutting a Figure
7.2 Using SAT-solvers
7.3 Using ILP-solvers
7.4 Visualizing Football Fans

1. Proofs: Convincing Arguments

Why are some arguments convincing while others are not? What makes an argument convincing? How can you establish your argument in such a way that no room for doubt is left? How can mathematical thinking help us deal with this? In this section, we will start by digging into these questions. Our goal here is to learn by examples how to understand proofs, how to discover them on your own, how to explain them, and — last but not least — how to enjoy them: we will see how a small remark or a simple observation can turn a seemingly non-trivial question into one with an obvious answer.

1.1 Warm Up

1.1.1 Why Proofs?

Proofs are absolutely necessary in mathematics, computer science, programming, and many other areas. Once you have an algorithm for a problem, you need to prove that it is correct. "The program works for me, what else do I need?" is not an approach that would scale well. A library function may run billions of times while being used by thousands of diverse programs. If it returns a single wrong result, say, once in every million calls, it could be disastrous. Think of an air traffic controller. If in their entire career they receive information with just one wrong value, just one time, hundreds of people could perish.[1] This is why mathematical proofs must be rigorously demonstrated to provide correct conclusions.

That is why throughout the whole book we will be focusing on formal proofs. Here, "formal" does not mean "long" or "unclear"! We will encounter many short and elegant proofs that are formal and convincing. Using our carefully designed interactive puzzles, we will try to push you softly to discover some of the proofs on your own. Nothing compares to the sense of happiness and self-satisfaction of an "Aha!" moment when you find a solution to a mathematical problem!

1.1.2 Tiling a Chessboard

Problem 1 Can a chessboard be tiled by domino tiles? Here, a chessboard is an 8×8 square divided into 64 squares 1×1 (see Figure 1.1), a domino tile is a 1×2 (or 2×1) rectangle, and by saying "tiled" we mean that there are no overlaps or empty spaces. Try it online (level 0)!

[1]For some specific examples, Google for "Bugs in the Space Program", "Therac-25", and "Toyota unintended acceleration".

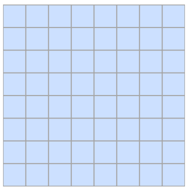

Figure 1.1: An 8×8 chessboard. Can it be tiled with domino tiles?

Yes, there are many such tilings. Two of them are shown in Figure 1.2.

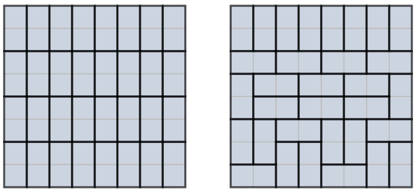

Figure 1.2: Two examples of tiling a chessboard.

> **Stop and Think!** Do we need both these examples to solve Problem 1, or one is enough?

In fact, one example is enough: any such example shows that it is possible to tile a chessboard.

> **Problem 2** Now consider the chessboard without one of the corners, see Figure 1.3. Can we tile it with domino tiles? Try it online (level 1)!

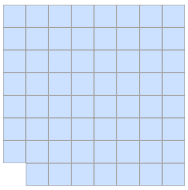

Figure 1.3: A chessboard without a corner. Can it be tiled with domino tiles?

Let's try. For example, let us use horizontal tiles, starting from the top row. Everything goes OK until we come to the bottom row, see Figure 1.4. This last row is problematic. We can put three tiles, but one cell is left uncovered.

> **Stop and Think!** Can we say now that Problem 2 is solved and we proved that the required tiling does not exist?

No, we cannot: one attempt to tile the board was unsuccessful. But this *does not* mean that the task is impossible. We are not limited to using horizontal tiles only; we can try doing something more sophisticated. Let us try a spiral, see Figure 1.5.

> **Stop and Think!** Can we say now that Problem 2 is solved and we have proven that the required tiling does not exist?

Of course not: we tried twice, but there are many more ways to try. What if some of them are successful? For a smaller board we may try all possibilities. Say we we invent a systematic way to enumerate them, making sure that no possible tiling has been missed. However, here the space of possibilities is rather large.

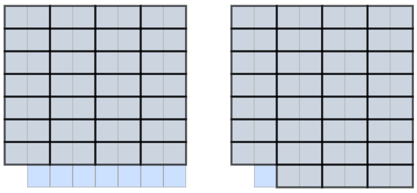

Figure 1.4: An unsuccessful attempt to tile a chessboard without a corner.

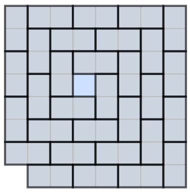

Figure 1.5: An unsuccessful attempt at a spiral tiling of a chessboard without a corner.

Stop and Think! Can you find a tiling or some general reason why we will always fail?

More specifically:

Stop and Think! Imagine there is a tiling of an 8×8 chessboard without a corner, by 1×2 tiles. How many tiles are in this tiling?

The full board contains $8 \times 8 = 64$ cells, so without a corner we have $64 - 1 = 63$ cells. Each domino tile consists of two cells, so the answer is $63/2 = 31.5$ tiles.

Stop and Think! The answer 31.5 is absurd: the number of tiles should be an integer. How is this even possible?

Recall the assumption we started with: "Imagine there is a tiling...". *If* there was a tiling of 63-cell board with domino tiles, it *would* use $63/2 = 31.5$ tiles. This, of course, is impossible — therefore, such a tiling does not exist. Thus, we get the proof we were looking for.

Problem 3 Consider an 8×8 chessboard without two adjacent corners, see Figure 1.6. Can it be tiled by domino tiles? Try it online (level 2)!

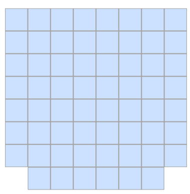

Figure 1.6: A chessboard without two adjacent corners. Can you tile it with domino tiles?

We already know that one should find the number of tiles needed: we have $8 \times 8 - 2 = 62$ cells, so we need $62/2 = 31$ tiles. We got an integer number, so we do not run into a problem and the tiling exists.

Stop and Think! Do you agree with this reasoning?

If you do, you are too fast. The argument shows only that *if* a tiling existed, it *would* consist of

31 tiles. But it does not show that a tiling exists. Informally speaking, we see only that some specific obstacle (non-integer number of tiles) does not prevent the existence of a tiling, but there may be other obstacles.

> **Stop and Think!** Give a correct proof of the existence of a tiling for the board without two adjacent corners.

Here it is, see Figure 1.7.

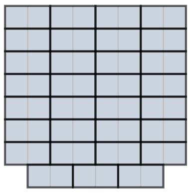

Figure 1.7: A tiling of a chessboard without two corners.

After training with these simple examples, we can tackle a more difficult question.

> **Problem 4** Consider an 8×8 chessboard without two opposite corners, see Figure 1.8. Can it be tiled by domino tiles? Try it online (level 3)!

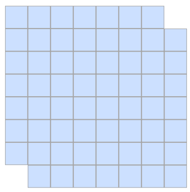

Figure 1.8: A chessboard without two opposite corners.

Again, the board contains $8 \times 8 - 2 = 62$ cells, so $62/2 = 31$ tiles would be needed. This is an integer number. But we already know that it does *not* mean that a tiling exists. Mathematicians would say that the even number of cells is a *necessary* condition for the existence of a tiling, but we do not know whether it is a *sufficient* condition.

> **Stop and Think!** Can you complete the existence proof by constructing some tiling?

Let's try. One such attempt is shown in Figure 1.9. As you see, this attempt was not successful: two cells are not covered.

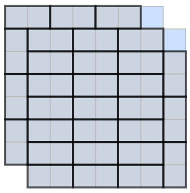

Figure 1.9: An unsuccessful attempt to tile a chessboard without two opposite corners.

The situation does not look hopeless: two cells are not covered, and the only problem is that they are not neighbors, so we cannot cover both by one tile. But maybe one can move some tiles to make the non-covered cells neighboring? Or maybe we can just start anew and have better luck? Let us try, see Figure 1.10. Now the empty cells are in the same column, but they are not neighbors.

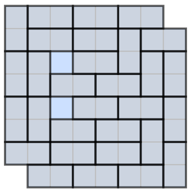

Figure 1.10: Another unsuccessful attempt to tile a board without two opposite corners.

If you play a bit more with this puzzle (level 3), you will see that this is more than just bad luck: every time at least two uncovered cells remain. But why?

Stop and Think! How can we prove that such a tiling is not possible?

Here a new tool is needed. If you are a chess player or have seen a real chessboard, you may have noticed that our drawings ignore one important feature of a chessboard. It has cells of two colors, usually black and white. In our color scheme, we will distinguish light and dark cells, see Figure 1.11.

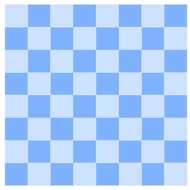

Figure 1.11: Chessboard coloring.

Stop and Think! How many dark and how many light cells are there on the chessboard?

Each row contains 4 light and 4 dark cells, so in total we have $8 \times 4 = 32$ light and 32 dark cells.

Thus, we have the same number of light and dark cells. Could we see this without counting them? One could show that the number of chairs in a room is equal to the number of people in it by asking everyone to sit down: if each person is seated and no chairs are empty, these two numbers are equal. (Unless somebody sits on two chairs or some chair is shared.)

Stop and Think! Can you prove in a similar way that the number of dark cells is the same as the number of light cells, by pairing them into light-dark pairs?

Any tiling of a chessboard will work: looking at the chessboard, we see that neighboring cells are always of different colors, hence every tile is a dark-light pair (Figure 1.12).

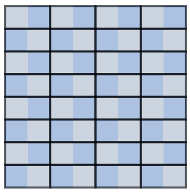

Figure 1.12: Pairing proves that the number of light cells is the same as the number of dark cells.

After this digression let us return to our board without opposite corners (Figure 1.13).

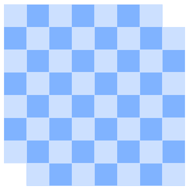

Figure 1.13: Looking again at the board without opposite corners.

Stop and Think! How many dark and how many light cells are on this board?

We do not need to count them again: two corner cells that are deleted are both dark. Thus, we have 30 dark cells and 32 light cells.

Stop and Think! Do you see why this board cannot be tiled by dominos?

Imagine that a tiling existed. It would use $62/2 = 31$ tiles. Each tile covers one dark cell and one light cell. So in total all tiles would cover 31 dark cells and 31 light cells. And — *aha!* — our board has 30 dark and 32 light cells. This mismatch shows that tiling is impossible.

Let us repeat the argument in a slightly different way. If a board can be tiled, then the numbers of dark and light cells are the same (=the number of tiles), because each tile covers one dark and one light cell. Hence, if these two numbers (dark and light cells) are different, no tiling is possible.

In a more concise exposition, this argument could be compressed into one paragraph.

Theorem 1.1.1 An 8×8 chessboard without two opposite corners cannot be tiled by 1×2 dominoes.

Proof. Consider the standard coloring of the chessboard with two colors (dark and light) where neighboring cells have opposite colors. It is easy to see that

- each tile contains one dark cell and one light cell;
- the board has 32 cells of one color and 30 cells of the other color (two deleted corners have the same color).

The first observation implies that any tileable region has the same number of dark and light cells, and then the second observation shows that our board is not tileable. ∎

Stop and Think! We have seen a partial tiling of the board without two opposite corners where two cells remain uncovered. What are the colors of these cells? (Try to give an answer without looking at the picture of the tiling.)

We do not need to know the specific tiling: if we have 32 light and 30 dark cells, and the tiling pairs every light and dark cell with two remaining unpaired, they must be light colored cells.

We have seen that a chessboard with one deleted cell is not tileable for trivial reasons (the number of cells is not even). For two deleted cells we had two examples. The first one, without two neighboring corners, was tileable; the other one, without opposite corners, was not.

Stop and Think! Why can't the argument that we use to prove the non-tileability in the second case be applied to the first case? What is the difference?

Let us formulate this question in a more specific way. Consider a chessboard without two cells. We know that sometimes the rest is tileable (example: two adjacent corners) and sometimes it is not (example: two opposite corners). Try it online (levels 4 and 5)!

Stop and Think! Can you state a general rule that distinguishes between tileable and untileable boards without two cells?

The following problem provides an answer to this question.

Problem 5 Prove that a chessboard without two cells can be tiled by domino tiles if and only if the deleted cells are of opposite colors.

The statement includes a strange expression "if and only if" (also known as "iff"). This mathematical jargon means that we have to prove two things:

- if we delete two cells of the opposite colors, then the rest is tileable ("if" part);
- if the board without two cells is tileable, then the deleted cells are of opposite colors ("only if" part).

Stop and Think! We have shown that any tileable region has the same number of dark and light cells, so the board without two cells of the same color is not tileable. What did we prove: the "if" part or the "only if" part?

It remains to prove that the board without one dark and one light cell is always tileable. To keep the suspense, we will not give the solution of this problem. Here is a diagram to think about, Figure 1.14. If you are old enough (some of the authors are), you may remember the computer game where a growing snake moves along itself eating food items placed in certain cells and increasing the length. In terms of this game, this picture shows a snake that has reached maximal possible length (includes all cells) so its head can be glued to its tail.

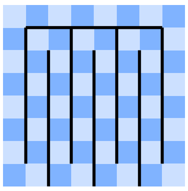

Figure 1.14: This "circular snake" helps us to prove that the board without two cells of opposite colors is tileable. Do you see how? For starters, tile the entire board by cutting the snake into domino tiles. What happens if you delete two cells from the snake?

Problem 6 We want to cover the figure shown in Figure 1.15 by 1×2 domino tiles. Is it possible (a) if we cover the highlighted cell by a horizontal tile? (b) if we cover the highlighted cell by a vertical tile? Try it online (questions 1 and 2)!

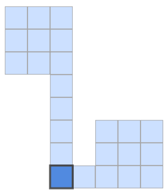

Figure 1.15: Board for Problem 6.

Now consider a 5×5 board divided into 25 cells 1×1 (Figure 1.16). It is not possible to tile it

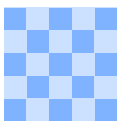

Figure 1.16: A 5×5 board (Problem 7).

by 1×2 tiles. (Why? Since the number of tiles needed for that, i.e., $25/2 = 12.5$, is not an integer.)

However, if we delete one cell, there may be a chance to cover the remaining 24 cells by 12 tiles. For example, if you delete the left upper corner, you can tile the rest using vertical tiles in the first column and horizontal tiles elsewhere. Let us call a cell *good* if the rest (5×5 board without this cell) can be tiled by dominoes.

Problem 7 Which of the cells are good? How many good cells are there? Try it online (question 3)!

Problem 8 Can we tile an 8×8 board by 1×3 tiles? (They can be placed both horizontally and vertically.) Can we tile 8×8 board without one corner by 1×3 tiles?

Problem 9 Can we tile a 10×10 board by 2×2 tiles? Can we tile this board by 1×4 tiles (that can be placed horizontally or vertically)?

Dark and light cells strike back. We have a full characterization of tileability of a chessboard without two cells: if they are of the same color, then there is no tiling; otherwise, the tiling exists. But what if the board is missing more than two cells? Specifically, do you see a way to implement a program that is given a subset of the cells of the board (i.e., for each cell it is indicated whether it is present or not) and quickly checks whether this region is tileable? We'll learn how to do this later in the book, when we study matchings in graphs! Interestingly, the solution will be based on dark and light cells again. (Technically, one should look for a maximal matching between dark and light cells in the bipartite neighborhood graph; we will explain what all these words mean.)

1.2 Existence Proofs

In this section, we study *existence proofs*, i.e., proofs of *existential statements*. For example,

> There exists an object which satisfies a particular property.

To prove this statement, we can provide an example of such an object: this is called a *constructive proof*.

There is another way to prove an existential statement: we can prove that such an object exists without providing an example of the object! Sounds counterintuitive, doesn't it? These are known as *non-constructive proofs* and we will see them in Section 1.2.4. For now, we will work with constructive proofs.

1.2.1 One Example Is Enough

We have seen constructive proofs of existential statements in the previous section.

Stop and Think! In the previous section we proved statements of two types: (a) some region can be tiled by dominos, and (b) some region cannot be tiled. Which of them are existential statements: (a) or (b)?

The existence of a tiling is (by definition) an existential statement. We proved it by showing an example of the required tiling. As we have said, one example is enough. This already constitutes a formal proof of existence. One does not need to explain how this example has been found, though this may be an interesting and non-trivial question (discussed later in Section 2.1).

Problem 10 Is it possible to cut down the figure shown in Figure 1.17 into two congruent pieces, i.e., into two pieces of the same shape and size? (In other words, does there *exist* a way to cut the figure into two congruent pieces?)

For this problem, a solution is not difficult to find, see Figure 1.18.

Stop and Think! What if we want to cut down the same figure into three congruent pieces?

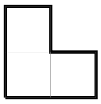

Figure 1.17: Can you cut this figure into two congruent pieces?

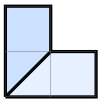

Figure 1.18: A solution to Problem 10.

This is even easier, since the figure consists of three squares. To make things more challenging, what if we pose the same question, but with four pieces?

Problem 11 Prove that the same figure can be cut down into four congruent pieces. (Equivalently, prove that there *exists* a way to cut down the figure into four congruent pieces.)

The hint is that the small pieces could be of the same shape as the figure itself. The construction is given in Figure 1.19. Note that just this example is enough to solve the problem. It provides a complete proof, and we do not need to add anything else.

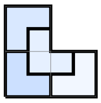

Figure 1.19: A solution to Problem 11.

Let's consider a somewhat more advanced problem of a similar flavor.

Problem 12 Prove that the octagon shown in Figure 1.20 can be cut down into two congruent pieces (an octagon is a polygon with 8 sides). Try it online (level 2)!

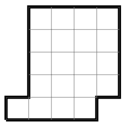

Figure 1.20: Can you cut this octagon into two congruent pieces?

This problem might look difficult at first, but there is again a simple solution. We can actually cut this figure along grid lines, see Figure 1.21. (To see that two pieces are the same, just move the left piece two cells to the right.)

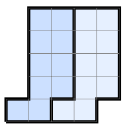

Figure 1.21: A solution to Problem 12.

How curious! There is another solution to this puzzle, see Figure 1.22.

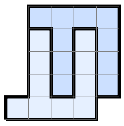

Figure 1.22: Another solution to Problem 12.

Problem 13 Prove that this figure can be cut down into *three* congruent pieces.

In this problem we do *not* require the cuts to go along the grid lines. This allows us to solve the problem by extending the previous solution a bit, see Figure 1.23.

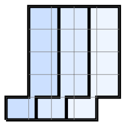

Figure 1.23: A solution to Problem 13.

If we required the cuts in this solution to go along the grid lines, our task would be impossible.

Stop and Think! Do you see why?

The reason is that the figure consists of 20 cells, and 20 is not divisible by 3.

We conclude with a more challenging problem of the same type.

Problem 14 Prove that it is possible to cut the figure from Figure 1.24 into two congruent pieces.

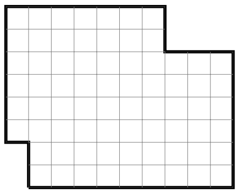

Figure 1.24: Can you cut this figure into two congruent pieces?

Finding a solution to this problem is difficult, so don't spend too much time on it. Knowing that a solution is possible and not knowing it is intolerable. For this reason, we provide a solution in Section 7.1.

Let us illustrate the "one example is enough" principle physically rather than mathematically. Imagine that you have three sticks and a length of string. Tie the string around the three sticks so that they form a free-standing structure in which the sticks do not touch. Could you make a free-standing structure with these restrictions? Put it another way: Does such a structure *exist*? It may surprise you but the answer is yes! Again, to prove this existential statement we only have to provide a single working example. See the video in our course or this wikipedia page.

1.2.2 Existential Statements in Number Theory

Recall that an integer a is *divisible* by a positive integer b if $k = a/b$ is an integer. In other terms, a is divisible by b if there *exists* an integer k such that $a = kb$.

In `Python`, to check whether a is divisible by b, one checks whether the *remainder* of a when divided by b is equal to zero. The remainder is found using the *modulo operator* — %. The following snippet shows that 237 is divisible by 3 and is not divisible by 7.

```
print(237 % 3)
print(237 % 7)
```

```
0
6
```

Stop and Think! Is 123 123 123 divisible by 123?

Yes, it is: $123\,123\,123 = 123 \cdot 1\,001\,001$, so we may take $k = 1\,001\,001$ in the definition above. The ratio k can be easily found by a computer program (or just a calculator). You may also get the answer without any tools (even without pencil or paper) if you think about the long division procedure. (Be careful: a common error is to get $k = 111$.)

Problem 15 Is there a positive integer that is divisible by 13 and ends with 15?

To prove that such a number exists, it is enough to give a single example. One such example is 715: it ends with 15 and it is divisible by 13 ($715 = 13 \cdot 55$). This already proves the existence, and we don't even need to explain how we have found this integer. Still, the following three lines of code help to find all such integers in the range $[0, 9\,999]$.

```
for n in range(10 ** 4):
    if n % 13 == 0 and n % 100 == 15:
        print(n)
```

```
715
2015
3315
4615
5915
7215
8515
9815
```

This program checks all numbers in `range(10 ** 4)`. Here, `10 ** 4` stands for $10^4 = 10\,000$. In `Python`, `range(N)` where `N` is some non-negative number is a *list*[2] (sequence) of `N` numbers $0, 1, 2, \ldots, \mathtt{N} - 1$. The `for`-loop goes over all of them in this order; the `if` operator checks whether they have the required properties. The last two digits of an integer `n` can be computed as `n % 100`. In general, `n % m` denotes the *remainder* when dividing `n` by `m`.[3]

Problem 16 Is there an integer that is divisible by 15 and ends with 13?

In this case, a similar program will produce no output. This does indicate that there is no such integer since we only checked the positive integers below 10^4 in the program. But this is indeed the case: such an integer must be divisible by 5 (since it is divisible by 15), but all integers divisible by 5 end with either 0 or 5.

Problem 17 Find a two-digit (positive) integer that becomes 7 times smaller when its first (=leftmost) digit is removed.

Let's try. Consider all two-digit integers that are divisible by 7:

$$14, 21, 28, 35, 42, 49, 56, 63, 70, 77, 84, 91, 98.$$

We know that dividing the required integer by 7 should result in a single digit integer. This allows us to rule out all numbers starting from 70 from the list. We can then check manually that out of the remaining numbers the only one satisfying the required property is 35.

[2] Technically speaking, in version 3 of `Python` `range(N)` is no longer a list, but a so-called *generator*, but for us the difference is not so important.

[3] Imagine we have n identical books on the table and pack them into boxes that contain m books each. Then $n \,\%\, m$ books *remain* unpacked.

The argument above is simple, but still some reasoning is needed. One could use a brute force search instead.

```python
for n in range(10, 100):
    if n == 7 * int(str(n)[1:]):
        print(n)
```

```
35
```

This code goes through all integers in the range $[10, 99]$. In general, `range(a, b)` where $\mathtt{a} \le \mathtt{b}$ are integers, denotes a list that contains `a`, `a+1`, $\ldots$, `b-1` (empty if $\mathtt{a} = \mathtt{b}$). To remove the first digit of a number, we convert it to a string (by calling the `str()` function), then use slicing (`[1:]`) to remove the first symbol of the resulting string, and finally convert the resulting string back to an integer.

Problem 18 Find an integer that becomes 57 times smaller when its first digit is removed.

Solving this problem on a piece of paper is already not that easy, but it is possible to adjust our code above (try it!) and get 7 125. Indeed, $7\,125 = 57 \cdot 125$ (check that this is correct!). Note that you don't need to include the program in the solution: an example is enough. The questions on why and how this number was picked are irrelevant, since this example already answers the question. On the other hand, checking that $7\,125 = 57 \cdot 125$ *is* a part of the solution (even if this part is left to the reader, as we did above).

Stop and Think! Do you see a way to find this example by hand?

Here is one possible line of reasoning. Let c be the first digit of the unknown number x, let z be the number x without the first digit, and let k be the number of digits in z.

$$x = \boxed{c}\underbrace{\boxed{\qquad z \qquad}}_{k}$$

Then,

$$x = c\underbrace{00\ldots00}_{k \text{ zeros}} + z = c \cdot 10^k + z\,.$$

Since x should become 57 times smaller when its first digit is removed,

$$c \cdot 10^k + z = 57 \cdot z\,.$$

By moving z to the right hand side, we get

$$c \cdot 10^k = 56 \cdot z\,. \tag{1.1}$$

Stop and Think! Can you find the value of c by looking at this equation? Recall that c is the first digit, i.e., an integer between 1 and 9.

Note that the right hand side of (1.1) is divisible by 7. And the only case[4] when the left hand side is divisible by 7, too, is $c = 7$. Plugging this into (1.1), we simplify it to

$$10^k = 8z\,.$$

Again, this means that 10^k should be divisible by 8. Hence k is certainly greater than 2 as both $10 = 10^1$ and $100 = 10^2$ are not divisible by 8. However, 8 divides 10^3: in this case, $z = 125$. This leads us to the final answer: $x = 7\,125$.

[4] Here we are cheating a bit: this is not that easy to see. To prove this, one needs some tools from basic number theory that we will discuss later. The left hand side of (1.1) can be factored as $c \cdot 2^k \cdot 5^k$, and this product is divisible by a prime number 7, so one of the factors should be divisible by 7. The only possibility is $c = 7$. But even if we know nothing about prime numbers and factorization, it would be a natural idea to try $c = 7$; recall that we do not need to explain how the example is found.

Stop and Think! Are there other examples of a number that becomes 57 times smaller when its first digit is removed?

Note that 10^k is divisible by 8 for *any* $k \geq 3$: indeed, if $k \geq 3$, then 10^k is divisible by 10^3. E.g., for $k = 4$, we get $z = 1\,250$ and $x = 71\,250 = 57 \cdot 1\,250$. This way, we get an infinite series of solutions (for all integer $k \geq 3$). All these solutions are obtained simply by padding our first solution $7\,125$ by zeros.

Problem 19 Are there other examples besides the solutions we have found for all $k \geq 3$?

Problem 20 Is there an integer that becomes 37 times smaller when its first digit is removed?

Problem 21 Is there an integer that becomes 58 times smaller when its first digit is removed?

Problem 22 Are there positive integers a, b, c such that $a^3 + b^3 = c^3$? Are there positive integers a, b, c, d such that $a^4 + b^4 + c^4 = d^4$?

We will discuss the answer to the last problem later in the book!

In some cases, nobody knows whether a simple existential statement is true or false. For example, nobody knows if there is a positive odd perfect integer N or not. Here "odd" means that N is not divisible by 2, and "perfect" means that N is equal to the sum of all its positive divisors, including 1 but excluding N itself (e.g., $N = 28$ is perfect since $1 + 2 + 4 + 7 + 14 = 28$, but 28 is not odd).

1.2.3 Proofs of Non-existence

A witness can certify that you made a payment to Mr. X or visited Moscow. But it would be much more difficult to prove that you did *not* pay Mr. X or that you have never been to Moscow. The situation with an existential statement is similar: to prove it, one example of an object with the required property is enough. But you cannot disprove it by providing an example or several examples of objects that do not have the required property; some general reasoning is needed to show that objects with the required property do not exist. We have already seen this kind of reasoning when proving that some regions are not tileable. In this section, we'll give more examples of this type. (We will give rigorous mathematical notation for existential and universal statements, and for their negations, in Section 4.2.)

Imagine that you go for a hike with a friend and want to split the weight evenly, i.e., to split items you take into two groups of the same weight. Try it online!

Stop and Think! Suppose we have three items with weights 1, 2, and 3. Can we split them into two groups of the same weight?

This question is almost trivial:

$$1 + 2 = 3$$

gives a required split (1 and 2 go in one group, 3 goes in the other one).

Instead of splitting, we could use signs of positive or negative value to generate a zero sum:

$$\pm 1 \pm 2 \pm 3 = 0\,.$$

The weights with plus and minus signs form two groups of equal weights. Our example can be represented now as

$$+1 + 2 - 3 = 0 \text{ or } -1 - 2 + 3 = 0\,.$$

Now, let's consider a more interesting example.

Stop and Think! Can we split the weights 1, 2, 3, 4, 5, and 7 into two groups of equal weights? (Reformulation: choose signs in $\pm 1 \pm 2 \pm 3 \pm 4 \pm 5 \pm 7$ to get 0.)

Stop and Think! If we split these weights evenly, what is the weight in each group?

This is easy: the total weight is

$$1+2+3+4+5+7=22,$$

and if we have two equal parts, each one is 11. Hence, we can reformulate the problem as follows: *form a group of weight* 11. Note that it is enough to find one such group: the rest will automatically have the same weight 11.

Stop and Think! Do you see such a group?

Indeed it is easy to find one:

$$4+7=11\,.$$

As we observed, all the remaining weights sum up to the same number:

$$1+2+3+5=11\,.$$

Thus, the problem is solved.

There is no guarantee that an even splitting is always possible. Let us, for example, substitute 7 by 6 (one item is now lighter):

Stop and Think! Can we split the weights 1, 2, 3, 4, 5, and 6 into two groups of equal weights?

Let us try the same approach and compute the total weight. Now it is one unit smaller: 21, and each part should have weight $21/2 = 11.5$. This cannot happen since all weights are integers. This means that a required splitting does not exist (the corresponding existential statement is provably false, as a mathematician would say).

Stop and Think! Can we split the weights 2, 4, 6, 8, 10, and 12 into two groups of equal weights?

This is also impossible. Do you see why? It is essentially the same problem as the previous one but we have multiplied all the weight units by 2.

Alternatively, we can say that the total weight is 42, so if we split the weight into two equal parts, the weight of each part is 21. But all weights are even, so it is impossible to obtain the odd sum of weights (not divisible by 2).

Stop and Think! Can we split the weights 1, 2, 3, 4, 5, and 17 into two groups of equal weights?

The sum of weights is now

$$1+2+3+4+5+17=32\,,$$

so each part (if a splitting exists) has weight $32/2 = 16$, and this is an integer. Still the splitting is impossible.

Stop and Think! Do you see why?

The obstacle is easy to see: each part should have weight 16, and one of the items is too heavy (weight 17) for that. We see that the splitting is also impossible, but the obstacle is of a different type.

It would be nice if our list of obstacles was complete. If so, we would be able to check whether the splitting exists by checking whether it is prevented by one of the known obstacles. However, this problem is not nearly so simple. For this problem, no one knows the complete list of obstacles and no one knows a quick way to tell (for a given set of weights) whether the splitting exists.

This problem belongs to the class of so called NP-*complete problems*. Roughly speaking, this means that the problem is (at least currently) infeasible. More precisely, the situation is as follows. We do not know a way to solve this problem (and other NP-complete problems) efficiently, but we also do not know a proof that this is impossible. Technically, it is called "the P vs. NP problem" and it is one of the most famous and important unsolved mathematical problems (with a $1M prize from Clay Mathematics Institute).

You see that our simple weight splitting problem is very close to the current boundary of Computer Science knowledge. Or, it might be better to say that the boundary is very close to it. No current approach works for ten thousand weights that are 64-bit integers (which is usually not much for computers to deal with). The exhaustive search in the space of all splittings is not feasible, and no significantly better approaches are known.

1.2.4 Non-constructive Proofs

Can we prove the existence of an object with some property without giving an example? This sounds counterintuitive, but sometimes it is possible.

Stop and Think! There exists an integer n such that

$$2^n + 3^n \leq 10^{1000} \quad \text{and} \quad 2^{n+1} + 3^{n+1} > 10^{1000}$$

Do you see why?

For small n (say, for $n = 1$) the first inequality is true. On the other hand, for large n the second one is true. Consider, for example, $n = 4000$. For this n, the first term alone is large enough: $2^{n+1} > 2^{4000} = 16^{1000} > 10^{1000}$. But we need to find n such that *both* inequalities are true at the same time. We need to find n such that $2^n + 3^n \leq 10^{1000}$, but increasing n by 1, we reverse the inequality.

Stop and Think! Do you see why there exists n with this property?

Intuitively, if we were inside and now are outside, we had to cross the boundary at some point. Let us increase n (going from 1 to 4000) and just stop before $2^n + 3^n$ becomes greater than 10^{1000}. This will give us the required value of n. This finished the proof, but we can also find the value of n as follows.

```
for n in range(4000):
    if 2 ** n + 3 ** n > 10 ** 1000:
        print(n - 1)
        break
```

```
2095
```

Another classical example deals with irrational numbers. Recall that a real number x is called *rational* if it can be represented as $\frac{a}{b}$ where a and b are integers. Otherwise, it is called *irrational*. E.g., $\sqrt{2}$ is irrational. Indeed, assume that $\sqrt{2} = \frac{a}{b}$ for some integers a, b. Then, $2 = \frac{a^2}{b^2}$, and hence $2b^2 = a^2$. We may assume that at least one of a and b is odd (if both are even, keep canceling the factor 2 until one becomes odd). If a is odd, then a^2 is odd and cannot be equal to $2b^2$. Therefore, a is even and b is odd (due to our assumption). But this is also not possible: if $a = 2k$, then $2b^2 = a^2 = 4k^2$, so $b^2 = 2a^2$, but b^2 should be odd for odd b. The contradiction shows that $\sqrt{2}$ is irrational.

Problem 23 Prove that there exist two irrational numbers x and y such that x^y is rational.

To prove this, consider $x = y = \sqrt{2}$. If x^y is rational, then we are done. Otherwise, $x^y = (\sqrt{2})^{\sqrt{2}}$

Figure 1.25: There are ten pigeons and nine holes, therefore there must be a hole having more than one pigeon. (Source: Wikipedia.)

is irrational. But then, raising it to the $\sqrt{2}$ power gives a rational number:

$$\left((\sqrt{2})^{\sqrt{2}}\right)^{\sqrt{2}} = (\sqrt{2})^{\sqrt{2}\cdot\sqrt{2}} = (\sqrt{2})^2 = 2\,.$$

Note that we again proved that there exist x and y satisfying the requirements of Problem 23 without explicitly specifying the values of x and y!

Problem 24 Prove that either $\sin^2(10^{1000}) \geq 1/2$ or $\cos^2(10^{1000}) \geq 1/2$. [Hint: check out basic trigonometric identities or the Pythagorean theorem.]

It is not so easy to find out which of these two inequalities holds. Indeed, the first step of computing these two values would be to reduce 10^{1000} radians to the interval $[0, 2\pi]$. For this, one needs to know the value of π with very high precision (about a thousand decimal digits). Still, we know for sure that one of these values is guaranteed to be at least $1/2$, even though we do not know which one.

The previous two examples are based on the so-called *law of excluded middle* that says that, for any statement, either it is true or its negation is true. For example, for any number x, either $x = 5$ or $x \neq 5$. We do not need to know the value of x to be sure that one of these two statements is true, and at the same time without knowing the value of x we do not know which of these two statements is actually true.

Another useful method for proving existence non-constructively is *the pigeonhole principle*: if there are more pigeons than holes, and each pigeon occupies some hole, then some two pigeons share the same hole (see Figure 1.25). As simple as it is, it has many non-trivial applications.

Problem 25 Prove that there exists a positive integer divisible by 57 that uses only digits 0 and 1.

Consider the numbers $1, 11, 111, 1111, 11111, \ldots$ and divide each of these numbers by 57. We get a sequence of remainders: $1 \bmod 57 = 1$, $11 \bmod 57 = 11$, $111 \bmod 57 = 54$, $1111 \bmod 57 = 28$, $11111 \bmod 57 = 53$ and so on ($a \bmod 57$ is the remainder of a when divided by 57). Since this sequence is infinite and there are only 57 possible remainders modulo 57, at some point we should see repetitions, i.e.,

$$\underbrace{111\ldots111}_{m\text{ digits}} \bmod 57 = \underbrace{111\ldots111}_{n\text{ digits}} \bmod 57$$

for $m \neq n$.

Stop and Think! Do you see how this helps?

If two numbers have the same remainder when divided by 57, then their difference is a multiple of 57. But the difference has decimal representation $11\ldots1100\ldots00$, so it is the desired example. (Assuming $m > n$, we have n zeros at the end and $m - n$ ones before them.)

In fact, one can find a number of the form $111\ldots111$ that is divisible by 57, the trailing zeros are not needed. We will see later in the book that trailing zeros do not help either, since 10 is coprime with 57 (i.e., the only positive number that divides 10 and 57 is 1).

Problem 26 Using a Python program (if needed), find a number of the form $111\dots111$ that is divisible by 57.

Problem 27 As of December 2020, there are 66 758 learners enrolled into our Mathematical Thinking in Computer Science course. Prove that there are 183 of them that share a birthday.

Often a non-constructive proof can be replaced by a constructive one. Sometimes a brute force search helps; sometimes another argument is needed. For Problem 23, one may note that

$$(\sqrt{2})^{2\log_2 3} = 2^{\log_2 3} = 3$$

and both $\sqrt{2}$ and $2\log_2 3$ are irrational (and 3 is rational).

Problem 28 We discussed why $\sqrt{2}$ is irrational above. Show that $2\log_2 3$ is also irrational. [Hint: unique factoring is useful here.]

One may ask what actually happens with $\sqrt{2}^{\sqrt{2}}$. The Gelfond – Schneider theorem guarantees that this number is irrational (and even *transcendental* which means that it is not a root of a non-zero polynomial with integer coefficients).

We end this section with two examples where finding a constructive proof is difficult. Let us generate 30 seven-digit numbers.

```python
from random import randint, seed

seed(10)

for i in range(30):
    print(randint(10 ** 6, 10 ** 7 - 1), end=' ')
    if i % 6 == 5:
        print()
```

```
1546686 8195564 9096041 1248847 4457754 8760813
9242586 5656024 3688205 1577196 9735382 9222271
6499115 2276567 5194248 7059236 1747532 8053077
3326819 6957282 7402286 8067637 5758324 5401017
8664354 3930486 6085259 7083207 3231003 8665986
```

The function `randint(a,b)` returns a random integer in the range `a...b`. We call it 30 times, using some additional tricks to have a nice printout. The second argument `end=' '` tells the `print` function that it should print a space character (instead of the newline character as it does by default). Then a call `print()` is used to get a newline character after groups of six numbers. (The call `seed()` initializes the pseudorandom generator used. This is done for reproducibility: it ensures that every time this program is called, the same 30 integers are generated. To get a different 30 numbers, just change the argument of the `seed` function.)

We claim that *there exist two disjoint subsets of this set of* 30 *integers that have equal sum*. In other words, we can color some of the elements blue, and color some other elements red so that the sum of the red numbers is equal to the sum of the blue numbers.

Stop and Think! Would you believe this claim?

It looks counterintuitive: the numbers are rather long, and having two equal sums looks at first as a strange coincidence. Still the following (non-constructive) argument proves our claim. For every subset A of this 30-element set consider an integer $S(A)$, the sum of all elements in A. All $S(A)$ are less than $30 \cdot 10^7$ (about third of a billion). On the other hand, we have 2^{30} subsets (each can be described by a 30-bit string saying which of the numbers are included and which are not, and there are 2^{30} binary strings of length 30). Now the crucial point (pigeonhole principle):

since $2^{30} = 1024^3 > 10^9 > 30 \cdot 10^7$*, there exist different* A *and* B *such that* $S(A) = S(B)$.

These A and B may include some common numbers, but these numbers can be deleted and still $S(A) = S(B)$ (since we delete the same numbers). This is exactly what we claimed. (Note that A and B are both non-empty: since $A \neq B$, at least one of them is non-empty and has positive sum, and the other has the same sum.) Still, this non-constructive argument does not give any information about these two subsets.

Our last example is *factoring*: one can prove using Fermat's little theorem[5] that the number

4120234369866595438555313653325759481798
1169984432798284545562643387644556524842
6198098870423161841879261420247188869492
5609317763750334211309823974851509449091
0691026986103186270411488086697056490290
3653658867433731720813104105190864254793
282601391257624033946373269391

(it does not fit in a single line!) is *composite*, which means that it is a product of two smaller integers. It is an existential statement (there *exists* a factoring), but nobody has been able to find such a factoring. (There even was a \$70 000 reward for doing this.)

Another common (and powerful!) way of proving something non-constructively is called the *probabilistic method*. Roughly, its main idea is the following: to prove that an object satisfying a particular property exists, take a *random* object and show that there is a *non-zero* chance that it satisfies the required property. From this we conclude that an object satisfying the property *exists*. We will see applications of this method later in the book.

Summary

- A chessboard without two cells can be tiled by domino tiles if and only if the deleted cells are of opposite colors.
- Structure of a proof reflects the structure of the claim.
- One example suffices to prove an existential statement.
- A general argument is required to prove that an existential statement is false.
- Non-constructive proofs show that a certain object exists without giving an example.

[5]Fermat's little theorem states that $(a^p - a)$ is divisible by p for any integer a and any prime p. We will discuss this theorem later.