Discrete Mathematics

John W. Snow

Contents

1 The Division Algorithm and Modular Congruence

The *integers* are the set $\mathbb{Z} = \{\ldots, -2, -1, 0, 1, 2, 3, \ldots\}$. This is the set of numbers we are most familiar with doing arithmetic with. We can also list the integers with one ellipsis as $\mathbb{Z} = \{0, -1, 1, -2, 2, -3, 3, \ldots\}$.

Theorem 1.1. The Division Algorithm: Suppose that n is a positive integer and that a is any integer. There are unique integers q and r so that $a = nq + r$ and $0 \leq r < n$.

Note. The Division Algorithm is not actually an algorithm. It is a theorem that merely states we can divide any integer a by an integer n and get a unique quotient q and nonnegative remainder r. Note that the remainder must be less than what we are dividing by. The Division Algorithm does not actually tell us *how* to do this division (then it would be an algorithm). It just says we can. Each number in the Division Algorithm has a name.

Definition 1.2. Suppose that $a = nq + r$ as in the division algorithm. Then n is called the *divisor*, a is the *dividend*, q is the *quotient*, and r is the *remainder*. We denote the quotient q as $a \,\mathbf{div}\, n$ and the remainder r as $a \,\mathbf{mod}\, n$.

Example 1.3. Find $39 \,\mathbf{div}\, 7$ and $39 \,\mathbf{mod}\, 7$.

Solution: We find the largest multiple of 7 which is less than or equal to 39. This is $35 = 7 \cdot 5$. Then we note that $39 = 7 \cdot 5 + 4$. Therefore, $39 \,\mathbf{div}\, 7 = 5$ and $39 \,\mathbf{mod}\, 7 = 4$. Alternatively, if we ask a calculator to divide, we will get something like $39 \div 7 = 5.57142\ldots$. Dropping the decimal gives $39 \,\mathbf{div}\, 7 = 5$. Then subtracting gives $39 \,\mathbf{mod}\, 7 = 39 - 7 \cdot 5 = 4$.

Example 1.4. Find $567 \,\mathbf{div}\, 10$ and $567 \,\mathbf{mod}\, 10$.

Solution: Quotients and remainders are easy when dividing by 10. The remainder is the ones digit. The quotient consists of all of the other digits. Since $567 = 10 \times 56 + 7$, $567 \,\mathbf{div}\, 10 = 56$ and $567 \,\mathbf{mod}\, 10 = 7$.

Example 1.5. Find $-567 \,\mathbf{div}\, 10$ and $-567 \,\mathbf{mod}\, 10$.

Solution: It may be tempting here to just negate the answers from the last example since $-567 = 10(-56) + (-7)$. However, the Division Algorithm requires that our remainders be nonnegative. The largest multiple of 10 *less than or equal to* -567 is $-570 = 10 \cdot (-57)$ and $-567 = 10 \cdot (-57) + 3$. Therefore, $-567 \,\mathbf{div}\, 10 = -57$ and $-567 \,\mathbf{mod}\, 10 = 3$.

Note. Most programming languages use the notation $a\%n$ for $a \,\mathbf{mod}\, n$. Most programming languages also calculate mods incorrectly for negative numbers. When a is negative, $a\%n$ will return a negative remainder like the tempting incorrect answer to the previous example.

Example 1.6. What are the possible remainders when dividing by 7?

Solution: The Division Algorithm for $n = 7$ requires that r be an integer so that $0 \leq r < 7$. The only integers satisfying this are $0, 1, 2, 3, 4, 5$, and 6.

Example 1.7. What is $a \,\mathbf{mod}\, 2$ when a is even? What is $a \,\mathbf{mod}\, 2$ when a is odd?

Solution: If a is even, then a is a multiple of 2 so there is an integer q with $a = 2q + 0$. Then $a \,\mathbf{mod}\, 2 = 0$. If a is odd, then when we divide a by 2 we have a nonzero remainder. The only possible remainders when dividing by 2 are 0 and 1, so there is an integer q with $a = 2q + 1$. This means that $a \,\mathbf{mod}\, 2 = 1$.

Definition 1.8. Suppose that n is a positive integer and that a and b are any integers. We say that a and b are *congruent modulo* n if $(a - b)$ is a multiple of n. That is, a and b are congruent modulo n if there is an integer q with $a - b = nq$. In this case we write $a \equiv b \pmod{n}$. If a and b are not equivalent modulo n, then we write $a \not\equiv b \pmod{n}$.

Example 1.9. Is $87 \equiv 62 \pmod{5}$?

Solution: We subtract $87 - 62 = 15 = 5 \cdot 3$. Since this difference is a multiple of 5, $87 \equiv 62 \pmod 5$.

Example 1.10. Is $94 \equiv 81 \pmod 3$?

Solution: We subtract $94 - 81 = 13$. Since 13 is not a multiple of 3, $94 \not\equiv 81 \pmod 3$.

Modular congruence shares some nice properties with equality that will help us in doing arithmetic and algebra involving modular congruence.

Theorem 1.11. *Suppose that n is a positive integer.*

1. *Equivalence modulo n is reflexive: for every integer a, $a \equiv a \pmod n$.*
2. *Equivalence modulo n is symmetric: for all integers a and b if $a \equiv b \pmod n$, then $b \equiv a \pmod n$.*
3. *Equivalence modulo n is transitive: for all integers a, b, and c, if $a \equiv b \pmod n$ and $b \equiv c \pmod n$, then $a \equiv c \pmod n$.*

Note. The properties listed in Theorem 1.11 declare that modular congruence is something called an *equivalence relation*. We will talk about equivalence relations in depth later. If $a = nq + r$ as in the Division Algorithm, then $a - r = nq$, so $a \equiv r \pmod n$. This observation along with Theorem 1.11 can be used to prove that checking for modular congruence reduces to comparing remainders.

Theorem 1.12. *Suppose that n is a positive integer and that a and b are any integers. Then $a \equiv b \pmod n$ if and only if $a \,\mathbf{mod}\, n = b \,\mathbf{mod}\, n$.*

Example 1.13. Is $57 \equiv 17 \pmod{10}$?

Solution: $57 \,\mathbf{mod}\, 10 = 7$ and $17 \,\mathbf{mod}\, 10 = 7$, so $57 \equiv 17 \pmod{10}$.

Example 1.14. Is $57 \equiv 13 \pmod{10}$?

Solution: $57 \,\mathbf{mod}\, 10 = 7$ and $13 \,\mathbf{mod}\, 10 = 3$, so $57 \not\equiv 13 \pmod{10}$.

Example 1.15. Is $57 \equiv -13 \pmod{10}$?

Solution: Since this question involves a negative integer we go back to the definition.

$$57 - (-13) = 57 + 13 = 70 = 10 \cdot 7$$

so $57 \equiv -13 \pmod{10}$.

Exercises 1.16. Complete these exercises.

1. Why does the equality $23 = 5 \cdot 3 + 8$ not satisfy the conditions in the Division Algorithm with $a = 23$ and $n = 5$?
2. Why does the equality $-23 = 7 \cdot (-3) + (-2)$ not satisfy the conditions in the Division Algorithm with $a = -23$ and $n = 7$?
3. What are the possible remainders when dividing by 5?
4. Calculate each of the following:
   (a) $187 \,\mathbf{div}\, 7$ and $187 \,\mathbf{mod}\, 7$
   (b) $100 \,\mathbf{div}\, 16$ and $100 \,\mathbf{mod}\, 16$
   (c) $89 \,\mathbf{div}\, 5$ and $89 \,\mathbf{mod}\, 5$
   (d) $-73 \,\mathbf{div}\, 5$ and $-73 \,\mathbf{mod}\, 5$
   (e) $-188 \,\mathbf{div}\, 9$ and $-188 \,\mathbf{mod}\, 9$

5. Which of these congruences are true?

   (a) $83 \equiv 13 \pmod 5$

   (b) $83 \equiv -13 \pmod 5$

   (c) $184 \equiv 337 \pmod{17}$

   (d) $13 \equiv -13 \pmod 2$

   (e) $-84 \equiv -48 \pmod 9$

6. Find a number x so that $x \equiv -x \pmod 5$

7. For each number x below, calculate the sum of the digits in x. Call this sum y. Calculate $x - y$. For which n is $x \equiv y \pmod n$? Do you notice any patterns?

   (a) $x = 13$

   (b) $x = 245$

   (c) $x = 1983$

   (d) $x = 11111$

8. Books are identified by ISBN numbers. In the ISBN-13 system, the ISBN is a 13 digit code

$$x_1x_2x_3x_4x_5x_6x_7x_8x_9x_{10}x_{11}x_{12}x_{13}.$$

   The first 12 digits encode the book's national origin along with publisher and language information. The 13^{th} digit is a check digit selected so that this equation is true:

$$x_1 + 3x_2 + x_3 + 3x_4 + x_5 + 3x_6 + x_7 + 3x_8 + x_9 + 3x_{10} + x_{11} + 3x_{12} + x_{13} \equiv 0 \pmod{10}$$

   The first 12 digits of an ISBN are 978037571457. Find the check digit.

2 Modular Arithmetic

Modular congruence interacts nicely with arithmetic.

Theorem 2.1. *Suppose that n is a positive integer and that a and b are any integers. Then*

- $(a+b) \bmod n = ((a \bmod n) + (b \bmod n)) \bmod n$,
- $(a-b) \bmod n = ((a \bmod n) - (b \bmod n)) \bmod n$, *and*
- $(a \cdot b) \bmod n = ((a \bmod n) \cdot (b \bmod n)) \bmod n$.

Note. What this theorem states is that we can interchange arithmetic and mod-ing by n at will, and we will always end up at the same result – as long as the last thing we do is mod by n.

Example 2.2. Calculate $((5+8\cdot 6)^2 + 9\cdot(7+8)) \bmod 10$.

Solution: We will do this multiple times to see how we can use Theorem 2.1 to make the arithmetic convenient. First, we perform all of the arithmetic and then mod. Recall that $a \bmod 10$ is just the last digit in a. This makes finding the mod easy for the example. We highlight where we perform arithmetic on each line.

$$\begin{aligned}((5+8\cdot 6)^2 + 9\cdot(7+8)) \bmod 10 &= ((5+48)^2 + 9\cdot(7+8)) \bmod 10\\ &= ((53)^2 + 9\cdot(7+8)) \bmod 10\\ &= (2809 + 9\cdot(7+8)) \bmod 10\\ &= (2809 + 9\cdot 15) \bmod 10\\ &= (2809 + 135) \bmod 10\\ &= (2944) \bmod 10\\ &= 4.\end{aligned}$$

Next, we perform the same arithmetic, but each time we perform an operation we mod by 10. This has the effect of keeping our numbers small. Here, we use the modular congruence notation since we are only calculating mods on part of the arithmetic expression.

$$\begin{aligned}((5+8\cdot 6)^2 + 9\cdot(7+8)) &\equiv ((5+8)^2 + 9\cdot(7+8)) \pmod{10}\\ &\equiv ((3)^2 + 9\cdot(7+8)) \pmod{10}\\ &\equiv (9 + 9\cdot(7+8)) \pmod{10}\\ &\equiv (9 + 9\cdot 5) \pmod{10}\\ &\equiv (9+5) \pmod{10}\\ &\equiv (4) \pmod{10}\end{aligned}$$

Performing arithmetic modulo n allows us to do arithmetic with only the numbers $0, 1, \ldots n-1$. In a sense this arithmetic is the usual arithmetic with integers, but once a calculation reaches n or higher, the portion above $n-1$ is lost as overflow. This arithmetic largely works as our usual integer arithmetic. However, there are some odd things. For example, in the computation above note that $9\cdot 5 \equiv 5 \pmod{10}$. We define an environment in which to do arithmetic modulo n.

Definition 2.3. For any positive integer n, the *integers modulo* n are the set $\mathbb{Z}_n = \{0, 1, 2, \ldots n-1\}$. When we do arithmetic in $\mathbb{Z}_n$, it is understood that all arithmetic is modulo n. Therefore, we can add, multiply, and subtract numbers in $\mathbb{Z}_n$ and get numbers in $\mathbb{Z}_n$, as long as the last operation we do is mod by n.

Note. When we perform arithmetic in $\mathbb{Z}_n$, we should use modular congruence notation, such as $7\cdot 8 \equiv 16 \pmod{20}$. However, mathematicians tend to be lazy and tend to create or abuse notation to make life simpler. Some folks would write this as $7\cdot 8 \equiv_{20} 16$ or $7\cdot 8 =_{20} 16$. We might even write, "In $\mathbb{Z}_{20}$, $7\cdot 8 = 16$." If we know that we are working in $\mathbb{Z}_{20}$, then we might just write $7\cdot 8 = 16$. If we perform the usual arithmetic and then mod, we should use the modular congruence notation but we can write just one mod at the end, such as $7\cdot 8 = 56 \equiv 16 \pmod{20}$.

Definition 2.4. A number b is a *multiplicative inverse* of a number a if $a \cdot b = 1$. Two non-zero numbers a and b are *zero divisors* if $a \cdot b = 0$.

Zero divisors and multiplicative inverses happen to be important in solving equations in the integers.

Example 2.5. The number 3 happens to have a multiplicative inverse in $\mathbb{Z}_{10}$. Find it.

Solution: We will find an inverse for 3 by brute force. We will multiply 3 by each element of $\mathbb{Z}_{10}$ and see which one gives a product of 1. In $\mathbb{Z}_{10}$:

$$3 \cdot 0 = 0, \quad 3 \cdot 1 = 3, \quad 3 \cdot 2 = 6, \quad 3 \cdot 3 = 9, \quad 3 \cdot 4 = 2,$$
$$3 \cdot 5 = 5, \quad 3 \cdot 6 = 8, \quad 3 \cdot 7 = 1, \quad 3 \cdot 8 = 4, \quad 3 \cdot 9 = 7.$$

Since $3 \cdot 7 = 1$ in $\mathbb{Z}_{10}$, 3 and 7 are multiplicative inverses in $\mathbb{Z}_{10}$.

Example 2.6. There are zero divisors in $\mathbb{Z}_{12}$ find them.

Solution: We will address this problem by brute force also. We will multiply every number in $\mathbb{Z}_{12}$ by every other number and see when we get 0. We organize the products into an array – which is actually a multiplication table for $\mathbb{Z}_{12}$.

$\times$	0	1	2	3	4	5	6	7	8	9	10	11
0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6	7	8	9	10	11
2	0	2	4	6	8	10	0	2	4	6	8	10
3	0	3	6	9	0	3	6	9	0	3	6	9
4	0	4	8	0	4	8	0	4	8	0	4	8
5	0	5	10	3	8	1	6	11	4	9	2	7
6	0	6	0	6	0	6	0	6	0	6	0	6
7	0	7	2	9	4	11	6	1	8	3	10	5
8	0	8	4	0	8	4	0	8	4	0	8	4
9	0	9	6	3	0	9	6	3	0	9	6	3
10	0	10	8	6	4	2	0	10	8	6	4	2
11	0	11	10	9	8	7	6	5	4	3	2	1

The zero divisors in $\mathbb{Z}_{12}$ are those non-zero numbers which have a 0 in their row other than the first column. These are 2, 3, 4, 6, 8, 9, and 10.

Note. The multiplication table in the last example also lets us find multiplicative inverses easily. Since $5 \cdot 7 = 1$, 5 and 7 are multiplicative inverses. Since $11 \cdot 11 = 1$, 11 is its own inverse. Also, since $1 \cdot 1 = 1$, 1 is its own inverse too. Notice that here every non-zero number is either a zero divisor or has a multiplicative inverse.

Example 2.7. We will see later that calculating exponents using certain large moduli is an essential component of many modern cryptographic algorithms. Calculate $3^{100} \textbf{ mod } 403$.

Solution: First of all, 3^{100} is too large to compute this number, divide by 403, and find the remainder. We could begin calculating powers of 3 and then mod when a power exceeds 403. This is valid, and it would require 100 multiplications. We will find our power using fewer multiplications. We are going to calculate 3^0, 3^1, 3^2, 3^4, 3^8, 3^{16}, 3^{32}, and 3^{64}. The exponents here are powers of 2. Since $3^4 = (3^2)^2$, to calculate 3^4, we need only square 3^2. Since $3^8 = (3^4)^2$, to calculate 3^8, we need only square 3^4. If we continue this way, we can calculate these exponents by squaring an mod-ing where appropriate. Performing arithmetic modulo 403:

$$3^0 = 1$$
$$3^1 = 3$$
$$3^2 = 9$$
$$3^4 = (3^2)^2 = 9^2 = 81$$
$$3^8 = (3^4)^2 = (81)^2 = 6561 = 113$$
$$3^{16} = (3^8)^2 = (113)^2 = 12769 = 276$$
$$3^{32} = (3^{16})^2 = (276)^2 = 76176 = 9$$
$$3^{64} = (3^{32})^2 = 9^2 = 81$$

Now, we can use these powers to find 3^{100} **mod** 403. The key is to note that $100 = 64 + 32 + 4$. Therefore, modulo 403 we have:

$$\begin{aligned} 3^{100} &= 3^{64+32+4} \\ &= 3^{64} \cdot 3^{32} \cdot 3^4 \\ &= 81 \cdot 9 \cdot 81 \\ &= 59049 \\ &= 211 \pmod{403}. \end{aligned}$$

The method used here is related to base 2 representations of numbers, which we will address later.

Hash Functions. Suppose that we want to store records for employees or customers or citizens efficiently in a way that they can be retrieved quickly. Assume that for each record there is a key or identification number (such as a social security number). There may be so many potential identification numbers that allotting a memory location for each number is not feasible from a memory standpoint. Storing all of the data in a list could be just as impractical because searching the list for a single record may take too much time. Instead, we can use a *hash function* to map each identification number to a code which is used to locate a place for the record in memory. An extremely simple example of this would be to mod the identification number by a modulus such as 1000 to obtain a code. The code is then linked to a memory location. It could be that many identification numbers have the same hash code. This is called collision. One way to account for collision is to have each memory location corresponding to a hash value contain a pointer to a list of records which have that same hash code. If identification codes are uniformly distributed, then a solution such as this which mods by 1000 would create a collection of lists which each would be about 1/1000 of the length of the entire list of identification numbers. This speeds up searches by a factor of 1000. Instead of having each memory location corresponding to a hash value contain a pointer to a list of records that have the same hash code, a hash function might mod by a large number (1000 would be too small for this approach) and then begin at that location in memory searching for the first open slot in memory to store the record.

Random Number Generators. When writing programs that involve simulation or cryptography, some form of randomness is often needed. Modular arithmetic can be used to build a random number generator – or a *pseudo-random* number generator, since nothing that comes out of a program is truly random. A *linear congruential random number generator* involves a modulus n, a multiplier m, and an increment b. At any point in time, the generator has a seed x. When the generator is asked for a number, it returns $(mx+b)$**modn** as the pseudo-random number and then replaces x with this number – the calculated random number becomes the new seed. For example, if we use a modulus of $n = 31$, a multiplier of $m = 17$, an increment of 5, and an initial seed of 2, the linear congruential generator will give this sequence of numbers:

seed (x)	$mx+b$	$(mx+b)$ **mod** n	seed (x)	$mx+b$	$(mx+b)$ **mod** n
2	39	8	3	56	25
8	141	17	25	430	27
17	294	15	27	464	30
15	260	12	30	515	19
12	209	23	19	328	18
23	396	24	18	311	1
24	413	10	1	22	22
10	175	20	22	379	7
20	345	4	7	124	0
4	73	11	0	5	5
11	192	6	5	90	28
6	107	14	28	481	16
14	243	26	16	277	29
26	447	13	29	498	2
13	226	9	2	39	8
9	158	3	8	141	17

Notice how the last two lines here are the same as the first two. The generator has begun to repeat. If n, m, and b are chosen appropriately, then the process can take several billion steps before it repeats. Most computers use some form of linear congruential generator for random numbers. Java, POSIX, and glibc use a modulus of $n = 2^{48}$, a multiplier of $m = 25214903917$, and an increment of $b = 11$. To make things less predictable, they do not use all of the bits of the seed in the output.

Affine Cipher. *Cryptography* is the science of encrypting messages so that they cannot be read. Cryptology is the study of cryptography. One of the oldest cryptographic methods is the Caesar cipher, attributed to Julius Caesar. This cipher replaces every letter with the one three letters away (wrapping back to the beginning of the alphabet when you reach the end). In the English alphabet, A becomes D, B becomes E, C becomes F, and so on. We can describe this cipher (and more) with modular arithmetic. Identify the letters $A, B, C, D, \ldots, Z$ with $\mathbb{Z}_{26}$, so $A = 0, B = 1, C = 2, \ldots, Z = 25$. Then Caesar's cipher encrypts a letter x as $(x + 3) \bmod 26$. Here are the steps to encrypt DAY with the Caesar cipher:

original text	D A Y
converted to numbers	3 0 24
add 3 and mod by 26	6 3 1
convert back to numbers	G D B

To decrypt and find the original message, you would subtract 3 and mod by 26. The Caesar cipher is a special case of an affine cipher. An affine cipher uses two numbers m and b and encrypts a letter x as $(mx + b) \bmod 26$. In such a cipher, m must be chosen so that a multiplicative inverse m^{-1} so that $(m \cdot m^{-1}) \bmod 26 = 1$ so that encryption is reversible. To decrypt $y = (mx + b) \bmod 26$, one would calculate $x = (m^{-1}(y - b)) \bmod 26$.

Exercises 2.8. Answer the following questions.

1. Perform these modular calculations:
   (a) $(23 \cdot (47 - 19)) \bmod 25$
   (b) $(12 - 19 \cdot 34) \bmod 64$
   (c) $(1 + 2 \cdot (3 - 4 \cdot (5 + 6 \cdot (7 + 8)))) \bmod 11$
   (d) $(1 + 2 \cdot (3 - 4 \cdot (5 + 6 \cdot (7 + 8)))) \bmod 2$
2. Calculate $7^{276} \bmod 100$ using the process of repeated squaring outlined in the section.
3. Calculate $3^{1001} \bmod 5$.
4. By trial and error, find all zero divisors in each of these.
   (a) In $\mathbb{Z}_8$
   (b) In $\mathbb{Z}_{10}$
   (c) In $\mathbb{Z}_{12}$
5. By trial and error, find all pairs of multiplicative inverses in each of these.
   (a) In $\mathbb{Z}_8$
   (b) In $\mathbb{Z}_{10}$
   (c) In $\mathbb{Z}_{12}$
6. Compare your responses to the last two questions and make a conjecture about when a number is a zero divisor or has a multiplicative inverse modulo n.
7. Find all numbers m in $\mathbb{Z}_{26}$ with multiplicative inverses.
8. Consider the linear congruential random number generator with $n = 16$, $m = 13$, and $b = 1$. Begin with a seed of 5 and calculate the pseudo-random numbers until you arrive at a repetition.

3 Base Notation

We typically use *base ten* notation to represent numbers. Ten digits – 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 – are put into places that are each assigned a value, and the place values are all powers of ten. For example:

$$23456 = 2 \cdot 10^4 + 3 \cdot 10^3 + 4 \cdot 10^2 + 5 \cdot 10^1 + 6 \cdot 10^0.$$

This number represents 6 ones plus 5 tens plus 4 hundreds plus 3 thousands plus 2 ten-thousands. There are two major benefits of such a system: We can represent arbitrarily large numbers with just a few (ten) symbols, and we have algorithms to extend arithmetic operations with single digits (those ten symbols) to larger numbers. There is nothing special about the base ten (other than most people have ten fingers to count on). Over the history of the world, cultures have used base ten, base four, base five, base twenty, base forty, and base sixty. In most cases there is a biological or cultural reason for the choice of the base. Computer scientists frequently use base two – a.k.a. *binary* – to represent two values such as on/off or true/false. In a base two system, place values represent powers of two: $2^0, 2^1, 2^2, 2^3, \ldots$, and we have only two symbols 0 and 1, which we call *bits*. In this section, we will be working with numbers in different bases. When necessary, we will use subscripts to indicate which base we are in. A number in base ten may be written something like 123_{10}. A number in base two would look like 11011_2. Base 7 would look like 53462_7.

Example 3.1. Convert 101011_2 to base ten.

Solution: We simply multiply each place value by 0 or 1, depending on what number is in that place.

$$\begin{aligned} 101011_2 &= 1 \cdot 2^5 + 0 \cdot 2^4 + 1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 \\ &= 2^5 + 2^3 + 2^1 + 2^0 \\ &= 32 + 8 + 2 + 1 \\ &= 43_{10}. \end{aligned}$$

Typically, we jump immediately to the second line of this computation. We also typically work from right to left since it is easier to read the place value off that way. This computation might look like.

$$\begin{aligned} 101011_2 &= 2^0 + 2^1 + 2^3 + 2^5 \\ &= 1 + 2 + 8 + 32 \\ &= 43_{10}. \end{aligned}$$

Example 3.2. Convert 11101_2 to base ten.

Solution: Working from right to left, and including only the place values where we have a one gives:

$$\begin{aligned} 11101_2 &= 2^0 + 2^2 + 2^3 + 2^4 \\ &= 1 + 4 + 8 + 16 \\ &= 29_{10}. \end{aligned}$$

Example 3.3. Convert 357_{10} to base 2.

Solution: This direction requires a bit more thought than the reverse; however, base 2 is easier to work with than other bases. The secret is to write 357 as a sum of powers of 2, so it may be helpful to have some of those readily available: $2^0 = 1$, $2^1 = 2$, $2^3 = 8$, $2^4 = 16$, $2^5 = 32$, $2^6 = 64$, $2^7 = 128$, $2^8 = 256$, $2^9 = 512$, and $2^{10} = 1024$. First, we find the highest power of 2 less than or equal to 357. This is 256. Then we subtract to find $357 - 256 = 101$, so that $357 = 256 + 101$. Then we repeat that process with 101. The highest power of 2 less than or equal to 101 is 64, and $101 = 64 + 37$. Then we have $357 = 256 + 64 + 37$. Next, $37 = 32 + 5$, so $357 = 256 + 64 + 32 + 5$. Finally, $5 = 4 + 1$, so

$$357 = 256 + 64 + 32 + 4 + 1 = 2^8 + 2^6 + 2^5 + 2^2 + 2^0.$$

Now to make the base 2 expansion of 357_{10}, we just need 1s in the 2^8, 2^6, 2^5, 2^2, and 2^0 places and 0s everywhere else.

$$357_{10} = 2^8 + 2^6 + 2^5 + 2^2 + 2^0 = 101100101_2.$$

The 1s record which powers of 2 are included in the sum.

Example 3.4. Convert 357_{10} to base 5.

Solution: The process for converting to bases other than 2 repeatedly uses the Division Algorithm 1.1. First, we divide 357 by 5 to get a quotient and remainder:

$$357 = 5 \cdot 71 + 2.$$

Then we divide the quotient 71 by 5 to get another quotient and remainder:

$$71 = 5 \cdot 14 + 1.$$

We then divide the new quotient 14 by 5 to get a quotient and remainder:

$$14 = 5 \cdot 2 + 4.$$

One more repetition gives

$$2 = 5 \cdot 0 + 2.$$

Once we have a quotient of 0, then our number converted to base 5 consists of the remainders in reverse order:

$$357_{10} = 2412_5.$$

Note. This process of repeated division could also be used to convert to base 2 from base ten. However, the approach of finding the highest power of 2 less than or equal to a number is faster.

Example 3.5. Convert 1234_5 to base ten.

Solution: We just use place values.

$$\begin{aligned} 1234_5 &= 1 \cdot 5^3 + 2 \cdot 5^2 + 3 \cdot 5^1 + 4 \cdot 5^0 \\ &= 1 \cdot 125 + 2 \cdot 25 + 3 \cdot 5 + 4 \cdot 1 \\ &= 125 + 50 + 15 + 4 \\ &= 194_{10}. \end{aligned}$$

Example 3.6. Add $11110_2 + 1100_2$.

Solution: To add in base 2, we stack the numbers and align them by place value. Then, we add within each place value, and we *carry*' any overflow from one place value to the next place. Here, *overflow* amounts to a quantity equal to or greater than 2. First, we stack and align:

$$\begin{array}{cccccc} & 1 & 1 & 1 & 1 & 0 \\ + & & 1 & 1 & 0 & 0 \\ \hline \end{array}$$

In the 2^0 place, $0 + 0 = 0$, and in the 2^1place, $1 + 0 = 1$:

$$\begin{array}{cccccc} & 1 & 1 & 1 & 1 & 0 \\ + & & 1 & 1 & 0 & 0 \\ \hline & & & & 1 & 0 \end{array}$$

Now, in the 2^2 place, $1 + 1 = 2_{10}$, but $2_{10} = 10_2$, so the 0 goes in the 2^2 place, and the 1 is overflow which is carried over to the 2^3 place.

$$\begin{array}{cccccc} & & 1 & & & \\ & 1 & 1 & 1 & 1 & 0 \\ + & & 1 & 1 & 0 & 0 \\ \hline & & & 0 & 1 & 0 \end{array}$$

Now in the 2^3 place, we have $1+1+1 = 3_{10} = 11_2$, so we have a 1 in the 2^3 place and a 1 as a carry.

$$\begin{array}{cccccc} & 1 & 1 & & & \\ & 1 & 1 & 1 & 1 & 0 \\ + & & 1 & 1 & 0 & 0 \\ \hline & & 1 & 0 & 1 & 0 \end{array}$$

Finall, adding in the 2^4 place gives $1+1 = 2_{10} = 10_2$. The 0 goes in the 2^4 place, and the 1 that is carried goes in the 2^5 place.

$$\begin{array}{cccccc} & 1 & 1 & & & \\ & 1 & 1 & 1 & 1 & 0 \\ + & & 1 & 1 & 0 & 0 \\ \hline 1 & 0 & 1 & 0 & 1 & 0 \end{array}$$

Note. Notice that to add two numbers in base 2, we need to be able to add up to three bits to account for the carry. This will be important later when we build circuits to add.

Example 3.7. Multiply $1101_2 \times 1011_2$. To multiply in base 2, we mimic the process we follow in base ten. We first stack the numbers, and then we multiply the top number by the number in each place of the bottom, staggering as we go. First we stack:

$$\begin{array}{cccccc} & & 1 & 1 & 0 & 1 \\ \times & & 1 & 0 & 1 & 1 \\ \hline \end{array}$$

Then we multiply by the 1 in the 2^0 place of the bottom number. Multiplying by 1 is easy:

$$\begin{array}{cccccc} & & 1 & 1 & 0 & 1 \\ \times & & 1 & 0 & 1 & 1 \\ \hline & & 1 & 1 & 0 & 1 \end{array}$$

Before we multiply by the 1 in the 2^1 place, we add a zero to stagger the next line.

$$\begin{array}{cccccc} & & 1 & 1 & 0 & 1 \\ \times & & 1 & 0 & 1 & 1 \\ \hline & & 1 & 1 & 0 & 1 \\ & & & & & 0 \end{array}$$

Then we multiply by the 1 from the 2^1 place.

$$\begin{array}{cccccc} & & 1 & 1 & 0 & 1 \\ \times & & & 1 & 0 & 1 & 1 \\ \hline & & & 1 & 1 & 0 & 1 \\ & & 1 & 1 & 0 & 1 & 0 \end{array}$$

For the next line, we stagger twice and then multiply by the 0 in the 2^2 place:

$$\begin{array}{cccccc} & & & 1 & 1 & 0 & 1 \\ \times & & & 1 & 0 & 1 & 1 \\ \hline & & & 1 & 1 & 0 & 1 \\ & & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}$$

Next, we stagger three times and multiply by the 1 in the 2^3 place:

$$\begin{array}{cccccc} & & & 1 & 1 & 0 & 1 \\ \times & & & 1 & 0 & 1 & 1 \\ \hline & & & 1 & 1 & 0 & 1 \\ & & 1 & 1 & 0 & 1 & 0 \\ & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 0 \end{array}$$

Finally, we add.

$$\begin{array}{cccccccc} & & & & 1 & 1 & 0 & 1 \\ \times & & & & 1 & 0 & 1 & 1 \\ \hline & & & & 1 & 1 & 0 & 1 \\ & & & 1 & 1 & 0 & 1 & 0 \\ & & 0 & 0 & 0 & 0 & 0 & 0 \\ + & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ \hline 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{array}$$

Base One Thousand. We often use commas to group base ten numbers to make them more readable. For example, the number 1234567890 is usually written as $1,234,567,890$, which is 1 billion, 234 million, 567 thousand, 890. It happens to be that one million is 1000^2 and that one billion is 1000^3. Then

$$1,234,567,890 = 1 \cdot 1000^3 + 234 \cdot 1000^2 + 567 \cdot 1000^1 + 890 \cdot 1000^0.$$

The commas effectively turn base ten into base one thousand. That is, grouping digits into groups of 3 converts base 10 to base 10^3.

Base Eight. If we begin with a binary number such as 111110101001_2 and separate it into groups of 3, we get base 2^3 or base 8.

$$111110101001_2 = 111\,110\,101\,001_2 = 7651_8.$$

Here, we used the base ten names for each of the sets of 3 bits to express the base 8 number. Base 8 is usually called *octal*.

Base Sixteen. It is more common to group binary numbers into sets of four bits. When we do so, we are converting the number to base 2^4 or sixteen. Writing a number in base sixteen is slightly more complex than base 8. For base 8, we can take advantage of the names of the digits $0, 1, 2, 3, 4, 5, 6, 7$ in base ten to use for each place value. In base 16, we need 16 names of place values. We use:

$$\begin{array}{cccc} 0 & 1 & 2 & 3 \\ 4 & 5 & 6 & 7 \\ 8 & 9 & A = 10_{10} & B = 11_{10} \\ C = 12_{10} & D = 13_{10} & E = 14_{10}, & F = 15_{10} \end{array}$$

Then

$$111110101001_2 = 1111\,1010\,1001_2 = FA9_{16}.$$

Note here that $1111_2 = 15_{10} = F_{16}$ and $1010_2 = 10_{10} = A_{16}$ and $1001_2 = 9_{10} = 9_{16}$. Base sixteen is called *hexadecimal*.

Exercises 3.8. Answer the following questions.

1. Convert each of these base ten numbers to base 2.

   (a) 23 (b) 31 (c) 145 (d) 986

2. Convert each of these base ten numbers to base 5.

   (a) 23 (b) 31 (c) 145 (d) 986

3. Convert each of these base 2 numbers to base ten.

(a) 111 (b) 10101 (c) 110011 (d) 111000

4. Convert each of these base 5 numbers to base ten.

   (a) 23 (b) 31 (c) 144 (d) 432

5. Convert each of these base 3 numbers to base 2.

   (a) 11 (b) 22110 (c) 21012 (d) 2222

6. Convert each of these base 2 numbers to octal.

   (a) 111001 (b) 110011101111 (c) 11101 (d) 1010101

7. Convert each of these base 2 numbers to hexadecimal.

   (a) 11010110 (b) 111100001111 (c) 11101 (d) 1010101

8. Convert each of these hexadecimal numbers to base 2 and to base ten.

   (a) ABCD (b) FACE (c) 1111 (d) 10A01

9. Add these base 2 numbers.

   (a) $1010 + 111$ (b) $1001 + 1100$ (c) $1111 + 101$ (d) $11 + 11$

10. Multiply these base 2 numbers.

    (a) $1010 \cdot 111$ (b) $1001 \cdot 1100$ (c) $1111 \cdot 101$ (d) $11 \cdot 11$

11. In an algebra class $\log(x)$ means $\log_{10}(x)$. This is the exponent to which we must raise 10 to get x. For example, $\log_{10}(100) = 2$ because $10^2 = 100$. We will use $\log(x)$ to represent $\log_2(x)$. This is the number to which we must raise 2 to get x. For example, $log(8) = 3$ because $2^3 = 8$. Calculate each of the following.

    (a) log(16) (b) log(64) (c) log(1024) (d) log(2048)

4 Propositional Logic

A *statement* is a declarative sentence which must be either true or false but not both. Statements are also called *propositions*.

Example 4.1. The following are statements.

- The grass is green.
- Clint Eastwood is the American president.
- The number 2 is less than the number 1.
- 1+1=2
- The sun will rise tomorrow.

Example 4.2. The following are not statements.

- Go to bed. (This is an imperative or command.)
- The house on the hill (This is not even a sentence.)
- Is this a statement? (This is an interrogative or question.)
- Paul is tall. (Since "tall" is relative, this might seem true to some people and false to others. It is not strictly true or false.)

Assumptions. There are two underlying assumptions in our definition of a statement. First, every statement must be either true or false. Second, no statement is both true and false. That any statement must be either true or false but not both is called the *law of the excluded middle*.

Truth Values. Every statement has what we call a *truth value* of true (often written T) or false (often written F). We will sometimes use 1 for true and 0 for false.

Symbols. We let variables represent statements. For example, we could let the letter P be the statement "It is raining." A single letter representing a statement is called an *atomic statement* or a *Boolean variable*.

Compound Statements. We can join atomic statements together with the words "and," "or," "not," and "implies." We call these words *logical operators* or *logical connectives*. The more complex statements which are formed using logical operators are called *compound statements* or *Boolean expressions*. We use symbols for each of the logical operators. These are defined below.

Conjunction. The symbol $\wedge$ means "and." If P and Q are two statements, then $P \wedge Q$ is the new statement "P and Q." For example, if P is "It is raining," and Q is "The grass is green," then $P \wedge Q$ is "It is raining, and the grass is green." The statement $P \wedge Q$ is called the *conjunction* of the statements P and Q. The statement $P \wedge Q$ is true when both of the statements P and Q are true. Otherwise, it is false. We can sum this up in this *truth table*:

P	Q	$P \wedge Q$
T	T	T
T	F	F
F	T	F
F	F	F

The first two columns of the table list all of the possible combinations of truth values for P and Q. The third column gives the corresponding truth value for $P \wedge Q$.

Restating Conjunctions. In the English language, there are many ways of expressing $P \wedge Q$. Any statement which communicates that both P and Q are true expresses $P \wedge Q$. If P is "It is raining," and Q is "The grass is green," then each of the following communicate $P \wedge Q$.

- It is raining, and the grass is green.
- It is raining, but the grass is green.
- It is raining; however, the grass is green.
- Even though it is raining, the grass is green.
- While it is raining, the grass is green.
- The grass is green, and it is raining.

Disjunction. The symbol $\vee$ means "or." If P and Q are two statements, then the statement $P \vee Q$ is "P or Q." This is called the *disjunction* of the statements P and Q. The statement $P \vee Q$ will be true when P is true, Q is true, or both are true. This can be expressed in a truth table:

P	Q	$P \vee Q$
T	T	T
T	F	T
F	T	T
F	F	F

Again, the first two columns of the truth table list the all possible combinations of truth values for P and Q (note that these are the same as the first two columns for $\wedge$ above). The last column gives the corresponding truth value for $P \vee Q$.

Exclusive Or. Our disjunction is an *inclusive or* – the times at which $P \vee Q$ are true include when P and Q are both true. It is sometimes convenient to use an *exclusive or*. The exclusive or of P and Q is written as $P \oplus Q$. The statement $P \oplus Q$ means, "P is true, or Q is true, but it is not the case that both P and Q are true." The truth table for $P \oplus Q$ is:

P	Q	$P \oplus Q$
T	T	F
T	F	T
F	T	T
F	F	F

Negation. The symbol $\neg$ means "It is not the case that..." If P is any statement then $\neg P$ means "It is not the case that P." For example, if P is "It is raining," then $\neg P$ is "It is not the case that it is raining." In English, a better way of saying this may be "It is not raining." For simplicity, we will most often read $\neg P$ as "Not P." The truth table for $\neg$ is:

P	$\neg P$
T	F
F	T

The statement $\neg P$ is called the *negation of* P.

Implication (Conditional). The symbol $\rightarrow$ means "implies." If P and Q are statements, then $P \rightarrow Q$ is "P implies Q." We will often read this as "If P, then Q." For example, if P is "I left my hat at home," and Q is "It will rain," then $P \rightarrow Q$ could be read as "If I left my hat at home, then it will rain." To determine the truth values for this new logical operator, it is useful to think of $P \rightarrow Q$ as a promise. Let P be the statement "You win," and let Q be the statement "We will go out to eat." Then $P \rightarrow Q$ is "If you win, then we will go out to eat." Think of this as a promise. The statement will be true when the promise is kept and false if it is broken. There is only one way in which the promise may be broken - if you win and we do not

go out to eat. This is the case where P is true and Q is false. Thus if P is true and Q is false, then $P \to Q$ is false. Otherwise, the promise is not broken, so the statement should be true. Here is the truth table:

P	Q	$P \to Q$
T	T	T
T	F	F
F	T	T
F	F	T

As with conjunction, there are many ways of expressing implication. Here are a few common ways of expressing $P \to Q$:

- If P, then Q.
- P implies Q.
- Q, if P.
- P only if Q.
- Q follows from P.
- Whenever P, Q.
- Q, whenever P.
- Not P unless Q.
- P is sufficient for Q.
- Q is necessary for P.

The part of an implication which comes before the arrow is called the *antecedent* or *hypothesis*. That which comes after the arrow is the *consequent* or *conclusion*. Thus in $B \to K$, B is the antecedent, and K is the consequent. Notice in the last two statements on our list that the antecedent is the sufficient part and the consequent is the necessary part.

Bi-Implication. The symbol $\leftrightarrow$ means "if and only if." The statement $P \leftrightarrow Q$ is "P if and only if Q." This means that P implies Q and Q implies P. We could write this as $(P \to Q) \wedge (Q \to P)$ This is called the *bi-implication* or the *biconditional*. Here is the truth table:

P	Q	$P \leftrightarrow Q$
T	T	T
T	F	F
F	T	F
F	F	T

Notice that the truth value for $P \leftrightarrow Q$ is exactly the negation of the truth value for $P \oplus Q$.

Order of Operations. For the most part, we will always use parenthesis to indicate order of operations in compound statements. The one exception we will make to avoid too many parenthesis is to let $\neg$ take precedence over all other operations. This means that unless a set of parenthesis is in the way, we apply all negations first. For example, rather than writing $((\neg P) \wedge (\neg Q)) \to (\neg(R \wedge S))$, we can write $(\neg P \wedge \neg Q) \to \neg(R \wedge S)$.

Example 4.3. Let L be "The lights are on." Let O be "The oven is on," and let D be "The door is open." Here are some translations using these symbols.

Symbols	Words
$L \to (O \vee D)$	If the lights are on then either the oven is on or the door is open.
$\neg(L \to D)$	It is not the case that if the lights are on then the door is open.
$(L \wedge D) \vee (L \wedge O)$	Either the lights are on and the door is open or the lights are on and the oven is on.
$\neg D \vee (L \to O)$	The door is closed, or if the lights are on, then the oven is on.

Example 4.4. Let S be "The sun will rise in the morning." Let C "Candace leaves a candle in her window," and let D be "Doug passes his math test." Translate this statement into symbols:

"If Candace leaves a candle in her window or Doug passes his math test, then the sun will rise in the morning."

Solution: We identify the atomic statements C, D and S in the statement:

$$\text{"If } \overbrace{\text{Candace leaves a candle in her window}}^{C} \text{ or } \overbrace{\text{Doug passes his math class}}^{D} \text{ then } \overbrace{\text{the sun will rise in the morning.}}^{S}$$

We notice the "or" and the "if…then…" in the statement and can label them (notice we place the $\to$ over the "then"):

$$\text{If } \overbrace{\text{Candace leaves a candle in her window}}^{C} \overset{\vee}{\text{or}} \overbrace{\text{Doug passes his math class}}^{D} \overset{\to}{\text{then}} \overbrace{\text{the sun will rise in the morning.}}^{S}$$

Finally, we can use the structure of the sentence and punctuation to determine placement of parenthesis. This gives:

$$\text{If } \overbrace{\text{Candace leaves a candle in her window}}^{(C} \overset{\vee}{\text{or}} \overbrace{\text{Doug passes his math class}}^{D)} \overset{\to}{\text{then}} \overbrace{\text{the sun will rise in the morning.}}^{S}$$

Thus our statement is $(C \vee D) \to S$. The process is not always this straightforward since there are many ways of expressing the logical operators in words.

Example 4.5. Using the symbols from above, translate this statement into symbols:

"If Doug fails his math test, then in order for the sun to rise tomorrow, it is sufficient that Candace leaves a candle in her window."

Solution: We notice the occurrence of $\neg D$, S, and C:

$$\text{If } \overbrace{\text{Doug fails his math test,}}^{\neg D} \text{ then in order for } \overbrace{\text{the sun to rise tomorrow,}}^{S} \text{ it is sufficient that } \overbrace{\text{Candace leaves a candle in her window}}^{C}$$

We notice the "if…then…" and place an $\to$ over the "then" and group it by itself. The rest of the sentence seems to be a single unit, so we place it in parenthesis:

$$\text{If } \overbrace{\text{Doug fails his math test,}}^{\neg D} \overset{\rightarrow}{\text{then}} \text{ in order for } \overbrace{\text{the sun to rise tomorrow,}}^{(S} \text{ it is sufficient that } \overbrace{\text{Candace leaves a candle in her window}}^{C)}$$

What we have in parenthesis - "In order for S, it is sufficient that C" - is an implication. The sufficient part is C, and we recall that the sufficient part of an implication is the antecedent - what comes before the arrow. Thus, what is in parenthesis is $C \to S$. We draw the arrow backwards here ($\leftarrow$) to maintain the sentence structure.

$$\text{If } \overbrace{\text{Doug fails his math test,}}^{\neg D} \overset{\rightarrow}{\text{then}} \text{ in order for } \overbrace{\text{the sun to rise tomorrow,}}^{(S} \text{ it is } \overset{\leftarrow}{\text{sufficient}} \text{ that } \overbrace{\text{Candace leaves a candle in her window}}^{C)}$$

Our statement is $\neg D \to (C \to S)$.

Exercises 4.6. Answer the following questions.

1. Which of the following are statements?
   (a) The brown and white dog ran down the long winding road.
   (b) True is spelled t-r-u.
   (c) This sentence is true.
   (d) This sentence is neither true nor false.
   (e) The old white house on the lonesome hill outside of town.
   (f) Feed the lazy dog on the porch once every day.
   (g) The clock is slow.
   (h) The car is slow.
2. The sentence "This sentence is false" is not a statement. Explain why.
3. Let M be "The moon is full." Let A be "The alarm is set for 4:00 AM," and let F be "Fred is going fishing in the morning." Below are compound statement using A, F, M. Translate each into words. (Try to be creative).
   (a) $F \vee \neg A$
   (b) $\neg F \wedge \neg A$
   (c) $\neg(F \vee A)$
   (d) $F \wedge (M \vee A)$
   (e) $A \wedge M \wedge F$
   (f) $(F \wedge M) \vee (F \wedge \neg M)$
   (g) $(M \wedge A) \to F$
   (h) $\neg A \vee (M \to F)$
4. Let F be the statement "The fox is more clever than the rabbit." Let R be "The rabbit is quicker than the fox," and let C be "The fox will catch the rabbit." Write each of the following compound statements using symbols:
   (a) The rabbit is quicker than the fox; however, the fox is more clever than the rabbit and will catch the rabbit.
   (b) The fox is not more clever than the rabbit, and the rabbit is quicker than the fox.
   (c) The rabbit is quicker than the fox, but the fox will catch the rabbit anyway.

(d) Although the fox is more clever than the rabbit, the fox will not catch the rabbit.

(e) If the fox is more clever than the rabbit or the rabbit is not quicker than the fox, then the fox will catch the rabbit.

(f) In order for the fox to catch the rabbit, it is sufficient that the rabbit is not quicker than the fox.

(g) For the fox to catch the rabbit, it is necessary that the rabbit is not quicker than the fox.

(h) While the fox is more clever than the rabbit, the rabbit is quicker than the fox; hence, the fox will not catch the rabbit.

5 Truth Values

We now turn to determining if a compound statement is true or false. Our first method will be to draw truth tables for compound statements like we did for our logical operators. We illustrate the method with an example.

Example 5.1. Draw a truth table for the statement $(P \vee Q) \wedge (\neg P \vee Q)$.

Solution: The first columns of the table will be labeled P and Q just as above. The next columns in the table will be labeled by the compound statements in our statement which are slightly more complicated than simply P or Q. These may be $\neg P$ or $P \vee Q$. The next column will be the next more complicated statement - $\neg P \vee Q$, and the next would be the next more complicate (which in this case would be the whole statement) and so on. Thus our table should have columns labeled by P, Q, $\neg P$, $P \vee Q$, $\neg P \vee Q$, and $(P \vee Q) \wedge (\neg P \vee Q)$.

P	Q	$\neg P$	$P \vee Q$	$\neg P \vee Q$	$(P \vee Q) \wedge (\neg P \vee Q)$

The atomic statements are single letters on the extreme left. They get more complicated as we move toward the right until we reach the entire statement. Now, the first two columns will list all possible combinations of truth values for P and Q as above (notice we use the same pattern):

P	Q	$\neg P$	$P \vee Q$	$\neg P \vee Q$	$(P \vee Q) \wedge (\neg P \vee Q)$
T	T				
T	F				
F	T				
F	F				

We next fill in the column for $\neg P$. The first line of the truth table for $\neg$ in section 1.10 could be read as $\neg T = F$, so anywhere we see a T under P, we should have a F under $\neg P$. The second row of $\neg$ could be read as $\neg F = T$, so when P is false, we place a T under $\neg P$. The column for $P \vee Q$ is filled out similarly. The rows of the truth table for $\vee$ could be read as $T \vee T = T$, $T \vee F = T$, and so on. Filling in these two columns gives

P	Q	$\neg P$	$P \vee Q$	$\neg P \vee Q$	$(P \vee Q) \wedge (\neg P \vee Q)$
T	T	F	T		
T	F	F	T		
F	T	T	T		
F	F	T	F		

We fill in the column for $\neg P \vee Q$ the same way - applying $\vee$ to the columns for $\neg P$ and Q

P	Q	$\neg P$	$P \vee Q$	$\neg P \vee Q$	$(P \vee Q) \wedge (\neg P \vee Q)$
T	T	F	T	T	
T	F	F	T	F	
F	T	T	T	T	
F	F	T	F	T	

Finally, we apply $\wedge$ to the last two columns to get the truth values for the entire statement. The truth table for $\wedge$ tells us that $T \wedge T = T$, and everything else is false.

P	Q	$\neg P$	$P \vee Q$	$\neg P \vee Q$	$(P \vee Q) \wedge (\neg P \vee Q)$
T	T	F	T	T	T
T	F	F	T	F	F
F	T	T	T	T	T
F	F	T	F	T	F

Example 5.2. Draw a truth table for the statement $(A \wedge B) \rightarrow \neg C$.

Solution: This statement has three letters, so we need a column for each letter. We will also need columns for $A \wedge B$, $\neg C$, and the whole statement. To get every possible combination of truth values for A, B, and C, we need eight rows (it is best to memorize this pattern to make sure you get it right). We then fill in the other columns as we did above to get

A	B	C	$A \wedge B$	$\neg C$	$(A \wedge B) \to \neg C$
T	T	T	T	F	F
T	T	F	T	T	T
T	F	T	F	F	T
T	F	F	F	T	T
F	T	T	F	F	T
F	T	F	F	T	T
F	F	T	F	F	T
F	F	F	F	T	T

Note. We can now use this truth table to determine truth values of our statement. For example, the third row says that if A and C are true but B is false, then the whole statement is true.

Calculating Truth Values. If we know the truth values of the letters in a compound statement and want to know the truth value of the whole statement, we could draw a truth table for the statement and read off the appropriate row. This is tedious - especially if there are more than two letters. A quicker way is to do arithmetic with the Ts and Fs.

Example 5.3. Suppose that A, C, and D are true while B is false. What is the truth value of the statement

$$(A \wedge \neg B) \to ((C \wedge B) \vee (\neg A \wedge D))$$

Solution: First, replace all of the As, Cs, and Ds by T, and replace the Bs by F

$$(T \wedge \neg F) \to ((T \wedge F) \vee (\neg T \wedge T))$$

We can now do arithmetic with the Ts and Fs using the truth tables as guidlines. We follow the order of operations dictated by the parenthesis and negations (the operations are indicated in red print):

$$\begin{aligned}
(T \wedge \neg F) \to ((T \wedge F) \vee (\neg T \wedge T)) &= (T \wedge \mathbf{T}) \to ((T \wedge F) \vee (\color{red}{F} \wedge T)) && (\text{since } \neg F = T \text{ and } \neg T = F) \\
&= \color{red}{T} \to ((T \wedge F) \vee (F \wedge T)) && \text{since } T \wedge T = T \\
&= T \to (\color{red}{F} \vee (F \wedge T)) && (\text{since } T \wedge F = F) \\
&= T \to (F \vee \color{red}{F}) && (\text{since } F \wedge T = F) \\
&= T \to \color{red}{F} && (\text{since } F \wedge F = F) \\
&= F && (\text{since } T \to F = F)
\end{aligned}$$

Example 5.4. Find a statement which has this truth table.

P	Q	$\cdots$	?
T	T		F
T	F		T
F	T		T
F	F		F

Solution: To construct the statement, locate the rows where we want truth. In this case, these are the middle two rows. The two rows give two conditions. The first row would require P to be true and Q to be false. A statement which would give truth here is $P \wedge \neg Q$. The second true row requires P to be false and Q to be true. A statement which has truth in this instance is $\neg P \wedge Q$. To construct the statement we need, we simply join these two statements with an $\vee$: $(P \wedge \neg Q) \vee (\neg P \wedge Q)$. This statement will have the desired truth values.

Note. The strategy is this: For each true row in the truth table form a conjunction. Each letter involved should appear once in the conjunction. If in that row the letter is false, it will be negated in the conjunction. If in that row the letter is true, the letter appears not negated in the conjunction. Form a conjunction like this for each true row. Then join these together with $\vee$.

Example 5.5. Find a statement with these this truth table.

P	Q	R	$\cdots$	?
T	T	T		T
T	T	F		F
T	F	T		T
T	F	F		T
F	T	T		F
F	T	F		T
F	F	T		T
F	F	F		F

Solution: The table has five true rows, so we must first make five conjunctions. The first true row is the row where all three statements are true. The corresponding conjunction is $P \wedge Q \wedge R$. The next true row is where P and R are true and Q is false. The corresponding statement is $P \wedge \neg Q \wedge R$ (the Q is negated because Q is false on this row). The next has only P being true, so the corresponding statement is $P \wedge \neg Q \wedge \neg R$. The fourth true row has only Q being true. It gives $\neg P \wedge Q \wedge \neg R$. The last true row has only R being true, giving $\neg P \wedge \neg Q \wedge R$. To make our statement, we now take these five conjunctions and join them with $\vee$ to get

$$(P \wedge Q \wedge R) \vee (P \wedge \neg Q \wedge R) \vee (P \wedge \neg Q \wedge \neg R) \vee (\neg P \wedge Q \wedge \neg R) \vee (\neg P \wedge \neg Q \wedge R)$$

Exercises 5.6. Answer the questions below.

1. Draw truth tables for each of these:
   (a) $\neg(P \wedge Q)$
   (b) $\neg P \vee Q$
   (c) $\neg B \to \neg A$
   (d) $A \wedge (B \vee C)$
   (e) $(A \wedge B) \vee (A \wedge C)$
2. Find the truth values of the following statements:
   (a) $(P \vee Q) \wedge (P \vee R)$ if P is false, Q is true, and R is true.
   (b) $\neg(\neg P \wedge (Q \vee \neg R))$ if P is false, Q is false, and R is true.
   (c) $(\neg P \wedge \neg Q) \vee (P \vee Q)$ if P is false and Q is true.
   (d) $\neg(P \vee Q) \wedge (P \vee \neg Q)$ if P is true and Q is false.
   (e) If the sun rises in the east, then it sets in the west.
   (f) In order for the sun to set in the west, it is necessary for it to rise in the east.
   (g) In order for the sun to set in the north, it is sufficient for it to rise in the west.
   (h) If I clap three times, the sun will rise tomorrow.
   (i) The sun rises in the east only if it sets in the north.
   (j) The sun rises in the south if and only if the sun rises in the north.
3. Suppose P and Q are statements (it does not matter what they are). Write a compound statement using P and Q which is always **true**. (You do not have to use both P and Q if you do not need to.)
4. Suppose P and Q are statements (it does not matter what they are). Write a compound statement using P and Q which is always **false**. (You do not have to use both P and Q if you do not need to.)

5. Find a statement which has this truth table

P	Q	R	$\cdots$	?
T	T	T		F
T	T	F		F
T	F	T		F
T	F	F		T
F	T	T		T
F	T	F		F
F	F	T		F
F	F	F		T

6. Find a statement which has this truth table

P	Q	R	$\cdots$	?
T	T	T		F
T	T	F		T
T	F	T		F
T	F	F		F
F	T	T		F
F	T	F		F
F	F	T		F
F	F	F		T

7. Find a statement which has this truth table

P	Q	R	$\cdots$	?
T	T	T		T
T	T	F		T
T	F	T		F
T	F	F		F
F	T	T		F
F	T	F		F
F	F	T		T
F	F	F		T

8. Find a compound statement using four atomic statements which is true when three or more of the atomic statements is true.

9. Find a compound statement using four atomic statements which is true when exactly two of the atomic statements is true.

10. Suppose that a truth table involves two letters P and Q.

    (a) There are sixteen possible final columns of the truth table. Find them.

    (b) Find a statement for each of the sixteen possible final columns.

6 Logical Equivalence

Definition 6.1. A compound statement which is always true regardless of the truth values of the atomic statements involved is called a *tautology*. The standard example of a tautology is $P \vee \neg P$. Any statement P is either true or false. This means that one of P and $\neg P$ must always be true. Hence, $P \vee \neg P$ must be true. We can draw a truth table to verify this:

P	$\neg P$	$P \vee \neg P$
T	F	T
F	T	T

We see that the last column consists of only Ts. This is the tell-tale sign of a tautology.

Example 6.2. Show that $(\neg A \vee B) \to (A \to B)$ is a tautology.

Solution: To show this statement is a tautology, we can simply draw a truth table and see that the final column contains only Ts.

A	B	$\neg A$	$A \to B$	$\neg A \vee B$	$(\neg A \vee B) \to (A \to B)$
T	T	F	T	T	T
T	F	F	F	F	T
F	T	T	T	T	T
F	F	T	T	T	T

Definition 6.3. A compound statement which is always false regardless of the truth values of the atomic statements involved is called a *contradiction*. The standard example of a contradiction is $P \wedge \neg P$. Since P and $\neg P$ will always have opposite truth values, they can never both be true, so $P \wedge \neg P$ must be false. Here is the truth table

P	$\neg P$	$P \wedge \neg P$
T	F	F
F	T	F

To show that any other statement is a contradiction, you may draw a truth table for the statement and see that the final column is all Fs.

Definition 6.4. When two compound statements always have the same truth values regardless of the truth values of the atomic statements involved, the two statements are **logically equivalent**. This means that two statements are equivalent if, when you draw their truth tables, the final columns in the two tables are identical.

Example 6.5. Show that $P \to Q$ and $\neg P \vee Q$ are logically equivalent.

Solution: We draw truth tables.

P	Q	$P \to Q$
T	T	T
T	F	F
F	T	T
F	F	T

and

P	Q	$\neg P$	$\neg P \vee Q$
T	T	F	T
T	F	F	F
F	T	T	T
F	F	T	T

We begin the two tables with P and Q (the same order in both tables), and the final columns are identical. Therefore, these two statements are equivalent.

Notation. We will denote logical equivalence using the symbol $\equiv$. For example, $P \wedge Q \equiv Q \wedge P$ or $P \to Q \equiv \neg P \vee Q$.

Basic Equivalences. There are only eight types of equivalences you need to remember. All other logical equivalences can be derived from these. We list them below.

Commutative Laws:. The english words "and" and "or" do not care about order. Saying "The grass is green, or the sky is blue" communicates the same thing as "The sky is blue, or the grass is green." The same is true with "and." Thus our first pair of equivalences should make sense:

$$P \wedge Q \equiv Q \wedge P \text{ and } P \vee Q \equiv Q \vee P$$

Notice that these resemble the commutative laws for multiplication and addition.

Associative Laws:. Our next pair of equivalences resembles the associative laws for multiplication and addition. The two statements "Jill and Jane passed math, and Janet passed math" and "Jill passed math, and Jane and Janet passed math" communicate the same thing - all three women passed. The word "and" does not care how statements are grouped together, and neither does "or." Thus

$$(P \wedge Q) \wedge R \equiv P \wedge (Q \wedge R) \text{ and } (P \vee Q) \vee R \equiv P \vee (Q \vee R)$$

Because of this equivalence, we will usually just write $P \wedge Q \wedge R$ or $P \vee Q \vee R$ and dispense with the parenthesis.

Idempotent Laws:. Our next set of equivalences again reflect the English language. All three of these statements

"It is not the case that it is not raining."
"It is raining, and it is raining."
"Either it is raining, or it is raining."

communicate the same thing - "It is raining." Thus we have three equivalences called the idempotent laws

$$\neg(\neg P) \equiv P \text{ and } P \wedge P \equiv P \text{ and } P \vee P \equiv P$$

Absorption Laws:. The next pair of equivalences are perhaps the least intuitive and least reflect a situation in English. For now, we will justify them by truth tables. The equivalences are

$$P \wedge (P \vee Q) \equiv P \text{ and } P \vee (P \wedge Q) \equiv P$$

Here is a truth table for $P \wedge (P \vee Q)$

P	Q	$P \vee Q$	$P \wedge (P \vee Q)$
T	T	T	T
T	F	T	T
F	T	T	F
F	F	F	F

Notice that when P is true this statement is true. When P is false, this statement is false. Hence they are equivalent.

Distributive Laws:. Consider the statement "It is raining, and either Hal forgot his hat or he forgot his coat." If this is true, what do we know? We know it is raining. We know that Hal forgot either his hat or his coat. In the first case, it is raining and he forgot his hat. In the second, it is raining and he forgot his coat. The statement seems to say "It is raining and Hal forgot his coat, or it is raining and Hal forgot his hat." This reflects our next pair of equivalences:

$$P \wedge (Q \vee R) \equiv (P \wedge Q) \vee (P \wedge R)$$

and

$$P \vee (Q \wedge R) \equiv (P \vee Q) \wedge (P \vee R)$$

These resemble the way in which we distribute multiplication over addition.

DeMorgan's Laws:. Consider the sentence "It is not true that Sam passed math and English." What does this mean? Let P be "Sam passed math," and let Q be "Sam passed English." The statement we are looking at is $\neg(P \wedge Q)$. In order for this to be true, $P \wedge Q$ needs to be false. This happens if at least one of P and Q is false. Thus our sentence appears to be $\neg P \vee \neg Q$ - "Either Sam did not pass math, or sam did not pass English." This is an example of one of DeMorgan's Laws:

$$\neg(P \wedge Q) \equiv \neg P \vee \neg Q \text{ and } \neg(P \vee Q) \equiv \neg P \wedge \neg Q$$

You can think of DeMorgan's Laws as distributing negation over $\wedge$ and $\vee$ - except that the negation applies to everything, even the $\wedge$ and the $\vee$.

Disjunctive Implication:. We already saw this equivalence as an example earlier. Here it is

$$P \to Q \equiv \neg P \vee Q$$

This reflects what was said in the first chapter that $P \to Q$ can be phrased as "Not P unless Q."

Contrapositive:. Suppose you know this statement is true "If Sam won his game, he is going to play in the championship game." If someone tells you that Sam is not going to play in the championship game, then you immediately conclude that Sam did not win his game. You are intuitively aware of this final logical equivalence

$$P \to Q \equiv \neg Q \to \neg P$$

The statement $\neg Q \to \neg P$ is called the *contrapositive* of $P \to Q$. You should be careful not to confuse it with the statement $\neg P \to \neg Q$ known as the *inverse* of $P \to Q$ or with $Q \to P$ known as the *converse* of $P \to Q$. These statements are not equivalent to $P \to Q$.

Basic Equivalences. Here are all of the basic equivalences together:

$P \wedge Q \equiv Q \wedge P$	commutative law
$P \vee Q \equiv Q \vee P$	commutative law
$P \wedge (Q \wedge R) \equiv (P \wedge Q) \wedge R$	associative law
$P \vee (Q \vee R) \equiv (P \vee Q) \vee R$	associative law
$\neg(\neg P) \equiv P$	idempotent law
$P \wedge P \equiv P$	idempotent law
$P \vee P \equiv P$	idempotent law
$P \equiv P \vee (P \wedge Q)$	absorption law
$P \equiv P \wedge (P \vee Q)$	absorption law
$P \wedge (Q \vee R) \equiv (P \wedge Q) \vee (P \wedge R)$	distributive law
$P \vee (Q \wedge R) \equiv (P \vee Q) \wedge (P \vee R)$	distributive law
$\neg(P \wedge Q) \equiv \neg P \vee \neg Q$	DeMorgan's Law
$\neg(P \vee Q) \equiv \neg P \wedge \neg Q$	DeMorgan's Law
$P \to Q \equiv \neg P \vee Q$	disjunctive implication
$P \to Q \equiv \neg Q \to \neg P$	contrapositive

Special Equivalences:. Suppose A is a true statement and P is any statement. A truth table for $P \wedge A$ would look like

P	A	$P \wedge A$
T	T	T
F	T	F

First of all, notice that we only need two rows since A is true. Second, notice that the truth values for $P \wedge A$ are identical to those for P. The statements P and $P \wedge A$ are equivalent if A is known to be true. We can summarize this by writing $P \wedge T \equiv P$ - where the T denotes a true statement or tautology. A similar equivalence holds for disjunction with a contradiction. It can be written as $P \vee F \equiv P$ - where F represents a contradiction.

Example 6.6. Rewrite the statement $(B \wedge D) \vee (B \wedge \neg D)$ using logical equivalences.

Solution: We can use the distributive law to "undistribute" the $B\wedge$ and re-write this as $B \wedge (D \vee \neg D)$. Now, $D \vee \neg D$ is a tautology, so this statement looks like $B \wedge T$ (where T is the tautology). This is equivalent to B. Thus, $(B \wedge D) \vee (B \wedge \neg D) \equiv B$.

Example 6.7. Use basic equivalences to show that $\neg(A \wedge \neg B)$ is equivalent to $A \to B$.

Solution: We can use the basic equivalences to show this. Notice that $\neg(A \wedge \neg B)$ is of the form of one of DeMorgan's Laws, so we will "distribute" the negation to get $\neg(A \wedge \neg B) \equiv \neg A \vee \neg(\neg B)$. Next, notice the double negation. We can use one of the idempotent laws to get $\neg A \vee \neg(\neg B) \equiv \neg A \vee B$. This last statement looks just like part of disjunctive implication, which tells us that $\neg A \vee B \equiv A \to B$. This is what we were looking for. Here is our work all together:

$$\begin{array}{rcll} \neg(A \wedge \neg B) & \equiv & \neg A \vee \neg(\neg B) & \text{DeMorgan's Law} \\ & \equiv & \neg A \vee B & \text{Idempotent Law} \\ & \equiv & A \to B & \text{Disjunctive Implication} \end{array}$$

Notice how we set up our work here. To show that $\neg(A \wedge \neg B) \equiv A \to B$, we begin with one statement on the left of an equivalence sign (here we use $\neg(A \wedge \neg B)$, but we could very well have started with the other statement). We then apply equivalences to this statement, listing the results to the right of equivalence signs until we arrive at $A \to B$.

Example 6.8. Show $A \to (P \vee R) \equiv (A \to P) \vee (A \to R)$

Solution:

$$\begin{array}{rcll} (A \to P) \vee (A \to R) & \equiv & (\neg A \vee P) \vee (\neg A \vee R) & \text{(disjunctive implication)} \\ & \equiv & \neg A \vee (P \vee (\neg A \vee R)) & \text{(associative law} \\ & \equiv & \neg A \vee ((P \vee \neg A) \vee R) & \text{(associative law)} \\ & \equiv & \neg A \vee ((\neg A \vee P) \vee R) & \text{(commutative law)} \\ & \equiv & \neg A \vee (\neg A \vee (P \vee R)) & \text{(associative law)} \\ & \equiv & (\neg A \vee \neg A) \vee (P \vee R) & \text{(associative law)} \\ & \equiv & \neg A \vee (P \vee R) & \text{(idempotent law)} \\ & \equiv & A \to (P \vee R) & \text{(disjunctive implication)} \end{array}$$

This looks a little confusing with all of the associative law applications. It is actually much simpler. If we abuse a little notation, the work looks like:

$$\begin{array}{rcll} (A \to P) \vee (A \to R) & \equiv & (\neg A \vee P) \vee (\neg A \vee R) & \text{(disjunctive implication)} \\ & \equiv & \neg A \vee P \vee \neg A \vee R & \text{(associative law)} \\ & \equiv & \neg A \vee \neg A \vee P \vee R & \text{(commutative law)} \\ & \equiv & \neg A \vee P \vee R & \text{(idempotent law)} \\ & \equiv & \neg A \vee (P \vee R) & \text{(associative law)} \\ & \equiv & A \to (P \vee R) & \text{(disjunctive implication)} \end{array}$$

Note. Usually, we will abuse notation like this and ignore parenthesis when associativiy allows it.

Example 6.9. Show $(A \vee B) \vee (A \wedge P) \vee (B \wedge P) \equiv (A \vee B)$

Solution:

$$\begin{array}{ll} (A \vee B) \vee (A \wedge P) \vee (B \wedge P) & \equiv (A \vee B) \vee (P \wedge A) \vee (P \wedge B) \\ & \text{(commutative law)} \\ & \equiv (A \vee B) \vee (P \wedge (A \vee B)) \\ & \text{(distributive law)} \\ & \equiv (A \vee B) \vee ((A \vee B) \wedge P) \\ & \text{(commutative law)} \\ & \equiv (A \vee B) \\ & \text{(absorption law)} \end{array}$$

The second step in this example can be thought of as "factoring" a common P from the last two terms. Commutativity actually lets us distribute from both directions, so we could shorten this just commuting and absorbing:

$$(A \vee B) \vee (A \wedge P) \vee (B \wedge P) \equiv (A \vee B) \vee ((A \vee B) \wedge P) \equiv (A \vee B)$$

Example 6.10. Show $(P \to B) \wedge (Q \to B) \equiv (P \vee Q) \to B$

Solution:

$$\begin{array}{lll} (P \to B) \wedge (Q \to B) & \equiv (\neg P \vee B) \wedge (\neg Q \vee B) & \text{(disjunctive implication)} \\ & \equiv (\neg P \wedge \neg Q) \vee B & \text{(distributive law)} \\ & \equiv \neg(P \vee Q) \vee B & \text{(DeMorgan's Law)} \\ & \equiv (P \vee Q) \to B & \text{(disjunctive implication)} \end{array}$$

Example 6.11. Use logical equicalences to simplify the statement "It is not true that I passed and you did not."

Solution: Let I be "I passed," and let Y be "You passed." The statement we are considering is $\neg(I \wedge \neg Y)$. Using DeMorgan's Law and then the idempotent law, we see $\neg(I \wedge \neg Y) \equiv \neg I \vee \neg\neg Y \equiv \neg I \vee Y$. Thus the original statement is equivalent to the simpler statement "Either I did not pass, or you did."

Example 6.12. Suppose you are building a machine with a warning light and you are told that "The warning light should come on if either the temperature is high while the pressure is high and the door is open or the temperature is high while it is not the case that both the pressure is not high and the door is closed." Use logical equivalences to simplify this is a baffling condition.

Solution: Let T be "The temperature is high." Let P be "The pressure is high," and let D be "The door is open." Our condition for the warning light to come on is $(T \wedge P \wedge D) \vee (T \wedge \neg(\neg P \wedge \neg D))$. This is so confusing, the circuitry to build the warning light could be quite complicated. However, notice

$$\begin{array}{lll} (T \wedge P \wedge D) \vee (T \wedge \neg(\neg P \wedge \neg D)) & \equiv T \wedge ((P \wedge D) \vee (P \vee D)) & \text{(distributive law)} \\ & \equiv T \wedge (((P \wedge D) \vee P) \vee ((P \wedge D) \vee D)) & \text{(distributive law)} \\ & \equiv T \wedge (P \vee D) & \text{(absorption law)} \end{array}$$

so the condition is equivalent to the much simpler statement "The temperature is high and either the pressure is high or the door is open."

Rewriting Implications. There are many cases in our natural language when an implication may be expressed without the words "if...then.." For example, the sentence

> The square of any even integer is even.

is logically the same as the implication

> If n is an even integer, then n^2 is even.

The statement

> All kittens are cute.

can be expressed as

> If it is a kitten, then it is cute.

The sentence

> When it rains, it pours.

can be expressed

> If it is raining, then it is pouring.

Example 6.13. Write this sentence as an implication. Then rewrite the sentence using the contrapositive, as a disjunction, using the word necessary and using the word sufficient. Finally, write the inverse and converse of the implication (which happen not to be equivalent to the implication).

> **Original:** When it rains, it pours.

Solution:

- **Implication:** If it is raining, then it is pouring.
- **Contrapositive:** If it is not pouring, then it is not raining.
- **Disjunction:** Either it is not raining, or it is pouring.
- **Necessary:** In order for it to rain, it is necessary that it pours.
- **Sufficient:** In order for it to pour, it is sufficient for it to rain.
- **Inverse:** If it is not raining, then it is not pouring.
- **Converse:** If it is pouring, then it is raining.

Fewer Logical Connectives. Not all of our logical connectives are necessary. The biconditional can be expressed with conjunction and implication. Disjunctive implication lets us express the implication with $\neg$ and $\vee$. In fact, DeMorgan's Laws let us write $\vee$ with $\neg$ and $\wedge$:

$$P \vee Q \equiv \neg(\neg P \wedge \neg Q).$$

We can also express $\wedge$ with $\neg$ and $\vee$:

$$P \wedge Q \equiv \neg(\neg P \vee \neg Q).$$

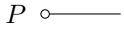
We could write all of our logical statements using fewer logical connectives. However, having all of our connectives makes it easier to translate between symbols and english.

Exercises 6.14. Answer the questions below.

1. Use the basic logical equivalences to show that these statements are equivalent.
   (a) $(P \to R) \vee (Q \to R) \equiv (P \wedge Q) \to R$
   (b) $\neg(A \vee B) \vee P \equiv (A \to P) \wedge (B \to P)$
   (c) $(A \vee B) \wedge (C \vee D) \equiv (A \wedge C) \vee (B \wedge C) \vee (A \wedge D) \vee (B \wedge D)$
   (d) $A \vee (B \vee C) \equiv (A \vee B) \vee (A \vee C)$
2. Use logical equivalences to simplify the following statements. Write your answers in words.
   (a) It is not true that both it is not cold and it is raining or snowing.
   (b) The possible combinations of toppings on the sandwich are meat and pickles and cheese, or meat and onions and cheese, or meat and pickles and tomatoes.
   (c) You will pass if either you pass both the midterm and the final, or if you do not fail both the major project and the final.
   (d) If we beat the "Cats" and the "Dogs" but not the "Penguins," or if we beat the "Cats" and the "Penguins" but not the "Dogs," or if we beat all three, then we will go to the playoffs.
3. Write each sentence as an implication. Then rewrite the sentence using the contrapositive, as a disjunction, using the word necessary and using the word sufficient. Finally, write the inverse and converse of the implication (which happen not to be equivalent to the implication).
   (a) All men are liars.
   (b) When the sun shines, she dances.
   (c) She cries when it rains.
   (d) All primes greater than 2 are odd. (Hint: "If n is...")
   (e) If you build it, he will come.
4. Express every connective in terms of $\neg$ and $\wedge$.
5. Define a new connective (called the *Sheffer Stroke*) by $P|Q = \neg(P \wedge Q)$. Express all of the logical connectives using $|$. For example, $\neg P = P|P$ and $P \wedge Q = (P|Q)|(P|Q)$. (Check these.) This means that we could do all of our logic with only one logical connective.

7 Logic Circuits

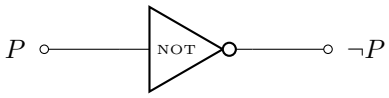
In this section, we will draw diagrams which reflect circuits. Since we will be using our tools of logic to draw these diagrams, we call them *logic circuits*. Rather than using the symbols T and F, when drawing logic circuits, we will use the symbols 0 and 1. Lines represent wires or copper traces that carry electricity. Each diagram will have one or more wires coming from the left side labeled by a Boolean variable (atomic statement) such as this:



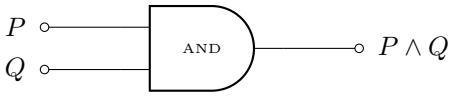
These wires we will call *inputs*. When the statement P is true, the input wire labeled P is carrying electricity. We say that it is *on* or *open* or *hot*. When the statement P is false, the input wire is not carrying electricity. We say that it is *off* or *closed*. All of the inputs to a diagram are routed through *gates* that perfom logical operations. Each diagram will have one wire exiting to the right called the *output* that looks something like this:

For each logic circuit, there is a logical statement or Boolean expression that tells, based on the inputs, when the output wire will be on or off.

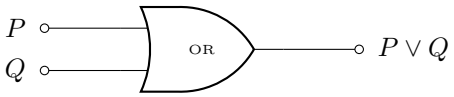
We have logic gates that mirror our logical operators $\neg$, $\wedge$, and $\vee$. A *NOT-gate* or *inverter* has one input and one output. When the input is on, the output is off. When the input is off, the output is on. We draw NOT-gates like so:



An *AND-gate* has two inputs and one output. The output is on if both inputs are on. Otherwise it is off. Here is how we draw AND-gates:



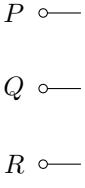
An *OR-gate* also has two inputs and one output. The output is off if both inputs are off. Otherwise, it is on. Here is a diagram:



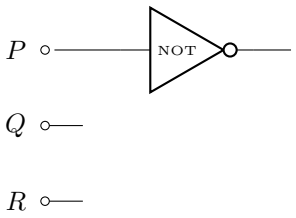
To draw logic circuits, we connect the inputs to the circuit to inputs of logic gates, and we connect the outputs of logic gates to inputs of other logic gates.

Example 7.1. Draw a logic gate for the statement $(\neg P \wedge Q) \vee R$.

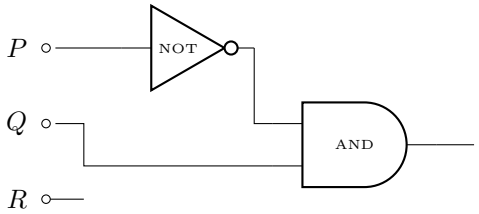
Solution: Since our statement has three variables, we begin with three inputs labeled by the variables.



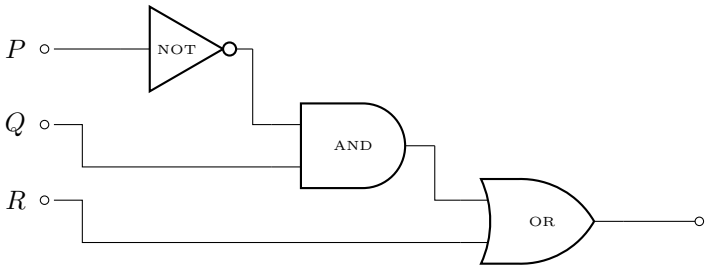
We now work from the inside out to build the statement. This is similar to how we built the columns for truth tables. The portion of the statement which is just a bit larger than what we have already draw is $\neg P$, so we add a NOT-gate and run the input for P through it.



We now have a wire for $\neg P$. We must run this and the wire for Q through an AND-gate to compute $\neg P \wedge Q$. We add an AND-gate and route the wires appropriately.



The wire farthest to the right now represents $\neg P \wedge Q$. We need to or this with R, so we add an OR-gate and run the wire for $\neg P \wedge Q$ and the wire from R through it.

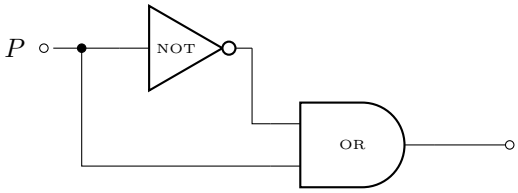


Since the output from the OR-gate is the output of our statement, we finish off the diagram with a small circle at the end of the output.

We can solder wires together so that we can split the flow from an input. This is demonstrated in the next example. Note that we only solder to split the flow of electricity. We never solder to join the flow of electricity through different wires (although, philosophically, this would simply be an or).

Example 7.2. Draw a logic circuit for $\neg P \vee P$.

Solution: We have one input P. The input needs to be fed into a NOT-gate to get $\neg P$, and it needs to be fed into an OR-gate. This means we need to split the P wire. This is indicated in the diagram by the black dot joining where the input from P separates.

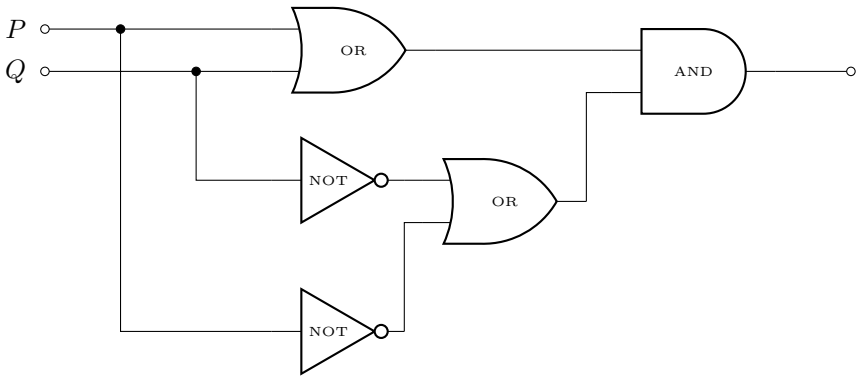


Note that since $\neg P \vee P$ is a tautology, this logic circuit is always on.

Note. It is sometimes difficult to draw logic circuits without crossing lines, especially if we want the flow of electricity always to be from left to right. If two lines in a diagram cross but do not have the circle solder mark, then we will assume the lines are not connected.

Example 7.3. Draw a logic circuit for $(P \vee Q) \wedge (\neg P \vee \neg Q)$.

Solution: A solution is pictured below.



This circuit is on if exactly one of P and Q is on, so it is a gate for exclusive or. Many texts provide XOR-gates. We are not using them. Notice that if we were to use De Morgan's Law to rewrite this statement as

$$(P \vee Q) \wedge (\neg P \vee \neg Q) \equiv (P \vee Q) \wedge \neg(P \wedge Q)$$

then we could make a slightly smaller circuit. Smaller means simpler, which in the real world means cheaper and faster.

Example 7.4. A hall light is to operated by two switches labeled P and Q. When both switches are on, the light is to be on. If the light is on and either switch is flipped, then the light should turn off. If the light is off and either switch is flipped, then the light should turn on. Design a logic circuit to operate this light.

Solution: First, we will come up with a Boolean expression (statement) that mimics the operation of the light. Then we will draw a circuit for this expression. Our statement has two variables, P and Q, so we start a truth table with two variables and all possible combination of truth values.

P	Q	?
1	1	
1	0	
0	1	
0	0	

If both switches are on, then the light is supposed to be on, so there is a 1 in the first row. We can get from the first row (light on) to either the second or third row by flipping one switch, so in both of these rows, the light should be off.

P	Q	?
1	1	1
1	0	0
0	1	0
0	0	

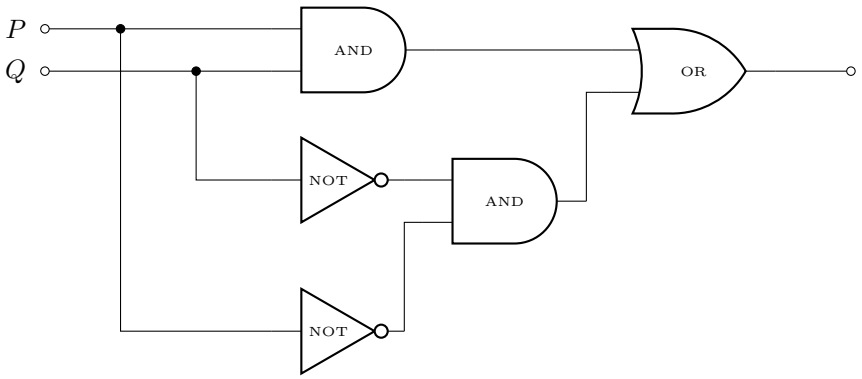
Finally, we can get from the third row (light off) to the last row by flipping the switch Q, so the light should turn on in the last row. Here is our truth table:

P	Q	?
1	1	1
1	0	0
0	1	0
0	0	1

Focusing on the 1s in the last column, a statement which has this truth table is

$$(P \wedge Q) \vee (\neg P \wedge \neg Q).$$

This is what we need to draw a logic circuit for.



Exercises 7.5. Complete the questions below.

1. Draw a logic circuit for the statement $P \wedge Q \wedge \neg R$. Note that our AND gate can only accept two inputs at a time.
2. Draw a logic circuit for the statement $P \to Q$. Note that we do not have a gate for $\to$.
3. Draw a logic circuit for the statement $(P \wedge Q) \vee \neg(Q \wedge R)$.
4. Design a logic circuit for a hall light operated by three switches P, Q, and R. The light should be on if all three switches are on, and the light should change states if any single switch is flipped.
5. When we add two one bit numbers, the result may be one bit or two bits. For example, $0 + 1 = 1$ but $1 + 1 = 10$. This question builds logic circuits to calculate these bits.
   (a) Design a logic circuit with two inputs P and Q. The inputs are interpreted as bits, and the output is the ones place of $P + Q$.
   (b) Design a logic circuit with two inputs P and Q. The inputs are interpreted as bits, and the output is the twos place of $P + Q$. (This is the carry bit.)
   (c) Combine the two circuits you just designed into one circuits with two inputs (P and Q) and two ouputs, one called S for the ones bit and one called C for the carry or twos bit.
6. We have seen that when adding two base two numbers we may end up needing to add three bits – two bits from the numbers being added along with a carry bit. So the circuit built in the last exercise does not do the full job of adding. We call it a half-adder. Treat a half-adder like a new type of logic gate. It has two inputs (the bits being added) and two outputs (the sum bit and the carry bit). Use two half-adders to design a circuit that adds three bits. The circuit should have three inputs P, Q, and R. The circuit should have two outputs. S will be the ones place of the sum, and C will be the twos place or carry bit. This circuit is a full-adder.

8 Predicates and Quantifiers

Predicates. Consider the sentence, "Bob is a man." The sentences "Larry is a man," "Lola is a man," and "Glenda is a man" are all related to this sentence. They are all of the form "x is a man." Each has a different name (or person) substituted for the letter x. The sentence "x is a man" is an example of a predicate. A **predicate** or **open statement** is a sentence involving variables which takes on a truth value once specific objects are substituted for the variables.

Predicate Notation. We use letters to represent predicates. If a predicate has a variable x, and if we want to name the predicate P, we will usually refer to the predicate as $P(x)$ (read "P of x"). The same predicate with "Bob" substituted for x would be $P(\text{Bob})$. For example, if $P(x)$ is "x is a man" then $P(\text{Bob})$ would be "Bob is a man." $P(\text{Glenda})$ is "Glenda is a man."

Predicates can have more than one variable. If $Q(x, y)$ is "x is married to y," then $Q(\text{Bob}, \text{Glenda})$ is "Bob is married to Glenda." $Q(1, 2)$ is "1 is married to 2" (which makes no sense). Suppose that $B(x, y, z)$ is "y is between x and z." Then $B(1, 2, 3)$ is "2 is between 1 and 3." $B(\text{Bob}, \text{Frank}, \text{Hank})$ is "Frank is between Bob and Hank."

Definition 8.1. The number of variables in a predicate is called the **rank** of the predicate. Here, P has rank 1, Q has rank 2, and B has rank 3. We can also say that Q is a 2-place predicate or that B is a 3-place predicate.

Quantifiers. Consider the sentence, "All men are mortal." There is clearly the predicate "x is mortal" at play here. The difference is that we are trying to substitute all men for x at the same time. We can rewrite this sentence in this way to account for all men:

> For all x, if x is a man, then x is mortal.

Here there are two predicates which have been combined: "x is a man" and "x is mortal." We also have an implication in the form of "if...then..." What is new is the "For all x." This is a quantifier. We will have two quantifiers, one to mean "For all" and one to mean "For some."

Definition 8.2. The *universal quantifier* is the symbol $\forall$. The expression $\forall x$ can be read as "For all x." If $P(x)$ is any predicate, then $\forall x P(x)$ can be read "For all x, $P(x)$."

Definition 8.3. The *existential quantifier* is the symbol $\exists$. The expression $\exists x$ can be read as "For some x" or "There exists x." If $P(x)$ is any predicate, then $\exists x P(x)$ can be read "For some x, $P(x)$" or "There exists x so that $P(x)$." (Note: Here "some" means "at least one.")

Varying Translations. As with our logical connectives, there are a variety of ways to translate quantifiers into English. Some translations of $\forall x P(x)$ are

For all x, $P(x)$.
For any x, $P(x)$.
$P(x)$, for all x.

Some translations of $\exists x P(x)$ are

For some x, $P(x)$.
For at least one x, $P(x)$.
There exists an x so that $P(x)$.
There is an x so that $P(x)$.
There is at least one x so that $P(x)$.
$P(x)$ for some x.
$P(x)$ for at least one x.

Sets:. We need some very basic ideas about sets before we continue (we will spend much more time with sets later). Even though most of mathematics is built upon the theory of **sets**, a set is something which mathematicians never define. Any definition of a set would require the use of a word such as "collection." Of course, we would then need to define "collection." This might include a word like "gathering" or "group." These words would then need to be defined. Since we only have a finite number of synonyms for any word, we would eventually circle around again to the word "set." To avoid this, we simply do not define the word and hope everyone has some intuitive idea what a set is. The things which compose a set are called the *elements* of the set. To denote that something called x is an element of a set named S, we would use the notation $x \in S$. This can be read as "x is in S," or "x is an element of S," or simply "x in S." We will denote the set of real numbers as $\mathbb{R}$ and the set of integers as $\mathbb{Z}$.

Quantifiers and Sets. The symbols $\forall x \in S$ will be used to mean "For all x in $S \ldots$" For example, if $P(x)$ is the open statement $x^2 \geq 0$ we can write "For all x in the real numbers, $x^2 > 0$" as $(\forall x \in \mathbb{R})P(x)$. The statement $(\forall x \in S)P(x)$ is true if this statement is true: "If $s \in S$, then $P(s)$."

The symbols $\exists x \in S$ will mean "There is an x in S so that..." For example, if $P(x)$ is the statement $x^2 = x$, we can write "There is an x in the real numbers so that $x^2 = x$" as $(\exists x \in \mathbb{R})P(x)$. The statement $(\exists x \in S)P(x)$ is true if $P(s)$ is true for some $s \in S$.

Example 8.4. Let $P(x)$ be the statement "x received an A," and let S is the set of people in class. Translate this sentence into words: $(\exists x \in S)P(x)$.

Solution: There are many ways to do this. Here are a few.

- Someone in class received an A.
- There is a person in class who received an A.
- There exists a student in class who received an A.

Example 8.5. Translate the the statement "2 has a square root in the real numbers" into symbols.

Solution: This sentence can be written as $(\exists x \in \mathbb{R})(x^2 = 2)$.

Abbreviations. If the set is known, we can abbreviate $\exists x \in S$ and $\forall x \in S$ simply as $\exists x$ and $\forall x$. Sometimes, we will need to use more than one quantifier. For example, the statement "For all real numbers x, y, and z, if $x \leq y$ and $y \leq z$ then $x \leq z$" really begins with three quantifiers

$$(\forall x \in \mathbb{R})(\forall y \in \mathbb{R})(\forall z \in \mathbb{R})([(x \leq y) \wedge (y \leq z)] \rightarrow (x \leq z))$$

We can abbreviate this as

$$(\forall x, y, z \in \mathbb{R})([(x \leq y) \wedge (y \leq z)] \rightarrow (x \leq z)).$$

Laziness. If all of the elements in a certain discussion are members of the same set, then we may omit the set in our quantifiers. For example, if every object being discussed is a reall number, then instead of $(\forall x \in \mathbb{R})P(x)$, we will usually write $\forall x P(x)$ and let the "in $\mathbb{R}$" be *understood*

Example 8.6. Translate the sentence into words and then find the truth value:

$$(\forall x \in \mathbb{R})(\exists y \in \mathbb{R})(y^2 = x)$$

Solution: This sentence says, "For every real number x there is a real number y so that $y^2 = x$." The equation $y^2 = x$ is declaring that y is the square root of x, so this says, "Every real number has a real square root." This sentence is false since -1 is a real number without a real square root.

Example 8.7. Let S be the set $\{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$. Translate this sentence into symbols: "There is a number in S which is less than or equal to every number in S."

Solution: The statement begins with "There is..." This is an existential quantifier, so we will have an $(\exists x \in S)$ in our translation. The rest of the statement says x is less than or equal to every number in S. The "every" is a universal quantifier, so we must have a $(\forall y \in S)$ also. We have no other quantifiers. Our statement should begin $(\exists x \in S)(\forall y \in S)$. The statement communicates that x is less than or equal to y. Thus the whole statement should be $(\exists x \in S)(\forall y \in S)(x \leq y)$.

Example 8.8. Translate this sentence into symbols. "Not every element of S has a square in S."

Solution: The "every element" is a $(\forall x \in S)$. The "has" symbolizes an existence, so we will need an $(\exists y \in S)$. Thus, we have so far $(\forall x \in S)(\exists y \in S)$. Next, the statement says that y (the thing that x "has") is the square of x, so we get $(\forall x \in S)(\exists y \in S)(x^2 = y)$. Finally, there is a "not" preceding everything, so we negate this statement to get $\neg((\forall x \in S)(\exists y \in S)(x^2 = y))$.

Example 8.9. Let $P(x)$ be the open statement "x is prime." Translate the statement into symbols: "Any number in S larger than 7 is not prime."

Solution: To begin, we translate "any number in S" as $(\forall x \in S)$. The statement communicates that if $x > 7$, then x is not prime. We can write this using $P(x)$: $(\forall x \in S)((x > 7) \to \neg P(x))$.

Example 8.10. Translate this sentences into symbols: Every dog chases some cat.

Solution: We actually give two solutions here, based on whether being a dog or cat is expressed with a predicate or a set. First, let $D(x)$ be "x is a dog." Let $C(x)$ be "x is a cat." Let $H(x, y)$ be "x chases y." Our sentence begins with "every dog." This is a universal quantifier $\forall x$, but it only applies if x is a dog. Therefore, we begin our sentence with $\forall x(D(x) \to (\cdots))$. To address the $(\cdots)$, we look to the sentence for what must be true if x is a dog. If x is a dog, then x must chase some cat. That is, there is some y so that both y is a cat (that is, $C(y)$) and x chases y (or $H(x, y)$). Our sentence is

$$\forall x(D(X) \to \exists y(C(y) \wedge H(x, y))).$$

For our second solution, instead of using predicates to declare that something is a dog or cat, we use sets. Let D be the set of all dogs. Let C be the set of all cats. Let $H(x, y)$ be "x chases y." The sentence begins wtih "every dog" so our symbols begin with $(\forall x \in D)$. If $x \in D$, then x must chase some cat. We can call that cat y, but before we can say that x chases y, we need to declare that y is "some cat." Our symbols are

$$(\forall x \in D)(\exists y \in C)H(x, y).$$

Universal Negation. If you are told, "It is not the case that everyone in class received an A," then you know immediately that someone did not get an A. This is an example of

$$\neg(\forall x \in S)P(x) \equiv (\exists x \in S)\neg P(x).$$

Existential Negation. If you are told, "It is not the case that someone received an A," you are dissapointed because you know everyone scored below an A. Intuitively, you know

$$\neg(\exists x \in S)P(x) \equiv (\forall x \in S)\neg P(x).$$

Example 8.11. Convert this sentence to symbols. Then negate the sentence and rewrite it with universal and existential negation so that no negations are outside any quantifiers. Then translate this negation back into words: There is an integer which is greater than or equal to every integer.

Solution: First, we translate to symbols. "There is an integer" gives us $\exists x \in \mathbb{Z}$. What we want to say about x is that x is greater than or equal to every integer. This "every integer" gives a universal quatification, $\forall y \in \mathbb{Z}$. Therefore, we will have our two quantifiers declaring the variables x and y followed by the statement that $x \geq y$. That is, $(\exists x \in \mathbb{Z})(\forall y \in \mathbb{Z})(x \geq y)$.

Now, the negation of this statement is:

$$\neg(\exists x \in \mathbb{Z})(\forall y \in \mathbb{Z})(x \geq y) \equiv (\forall x \in \mathbb{Z})(\exists y \in \mathbb{Z})\neg(x \geq y) \equiv (\forall x \in \mathbb{Z})(\exists y \in \mathbb{Z})(x < y).$$

In words, this is, "Every integer has an integer greater than it."

Exercises 8.12. Answer the questions below.

1. Let S be the set of numbers $\{2, 3, 4, 6, 8\}$. Let $P(x)$ be "x is prime," and let $T(x)$ be "x is a multiple of 2." Translate each of the following statements into symbols.

   (a) There is a number in S which is both prime and a multiple of two.

   (b) Every number in S is either prime or is a multiple of two.

   (c) There is a number in S which is greater than or equal to every number in S.

   (d) There is a number in S which is prime but is not a multiple of two.

   (e) There is a prime number in S which is less than or equal to every number in S.

   (f) There is a prime number x in S so that for any number y in S, if y is prime, then $y \leq x$.

2. Translate each of these English sentences into symbolic sentences using quantifiers. Use D for the set of dogs and C for the set of cats. Use $F(x)$ for "x has fleas." Use $H(x, y)$ for "x chases y."

   (a) All dogs have fleas.

   (b) Some dogs have fleas.

   (c) Not every dog has fleas.

   (d) Some dogs do not have fleas.

   (e) No dog has fleas.

   (f) Every dog chases some cat.

   (g) Every dog chases every cat.

   (h) Some dog chases every cat.

   (i) Some dog does not chase any cat.

   (j) No dog chases every cat.

3. Negate each of these statements.

   (a) $[(\forall x)P(x)] \vee [(\exists x)Q(x)]$

   (b) $(\forall x)(P(x) \to [(\forall y)Q(x, y)])$

   (c) $(\exists x)(\forall y)(P(x, y) \vee (\exists z)Q(x, y, z))$

   (d) For every $n \in \mathbb{Z}$, there is an $x \in \mathbb{R}$ so that $x^2 = n$.

4. Here are eight sets of "holes." Some are filled (the dark ones). Some are not filled.

   A. ○ ○ ○ D. ○ ● ● G. ● ● ○

   B. ○ ○ ● E. ● ○ ○

   C. ○ ● ○ F. ● ○ ● H. ● ● ●

   Let $F(x)$ be "x is filled." Translate the following statements into symbols and then decide which sets of holes satisfy the statement. You may assume that all quantifiers are over the set of holes. (This means that you can simple write $\forall x \ldots$ rather than $(\forall x \in H) \cdots$ or $\forall x(H(x) \to \cdots)$.)

(a) All holes are filled.

(b) Some holes are filled.

(c) A hole is filled.

(d) There is a hole which is filled.

(e) There is a hole which is not filled.

(f) All holes are not filled.

(g) Some holes are not filled.

(h) A hole is not filled.

(i) It is not the case that all holes are filled.

(j) It is not the case that some holes are filled.

(k) It is not the case that all holes are not filled.

(l) It is not the case that some holes are not filled.

(m) Not all holes are filled.

(n) Not all holes are not filled.

9 Sets

In any spoken language, at any point in time, there are only finitely many words. This has surprising consequences when you try to define words. Suppose we try to define the word "little." Many dictionaries will have this definition: "small in size." The same dictionaries will define "small" as "little in size." If we do not know what "little" or "small" means, these dictionaries are useless.

Sets. To avoid circular definitions, mathematicians begin with *primitives* – undefined terms. The most fundamental primitive in all of mathematics is *set*. We will not define what a set is. Presumably, *collection* is a synonym. Sets contain things which we call *elements*. To indicate that an element x is in a set A, we write $x \in A$. This notation can be read as "x is an element of A," or as "x is in A," or if necessary, "x in A." To express that x is not in A, we would write $x \notin A$.

Roster Notation. If we can list the elements of a set, we will do so between braces. For example, the set containing the symbols a, b, 1, and 2 is $\{a, b, 1, 2\}$. We can use braces to list infinite sets if there is a clear pattern. For example, the even integers are $\{\ldots, -4, -2, 0, 2, 4, 6, \ldots\}$. An element of a set can be listed within braces repeatedly without changing the set. For example, the sets $\{a, b, c\}$ and $\{a, a, b, b, c, c\}$ are the same sets. Within braces, order also does not matter. The sets $\{t, e, a\}$ and $\{a, t, e\}$ are the same set.

Special Sets of Numbers. The *natural numbers* are the numbers $\{0, 1, 2, 3, 4, \ldots\}$. This set is denoted $\mathbb{N}$. Historically, 0 was not included in the set of natural numbers. However, many modern textbooks do include 0 as it makes certain statements simpler later. The set $\{\ldots, -4, -3, -2, -1, 0, 1, 2, 3, \ldots\}$ is called the set of *integers* and is denoted as $\mathbb{Z}$. The set of all numbers which can be expressed as an integer divided by a non-zero integer is called the set of *rational numbers* and is denoted by $\mathbb{Q}$. These are precisely those numbers which in decimal form "repeat" such as $1.456121212121212\ldots$ or which "terminate" such as 1.234. The *irrational numbers* are those numbers which cannot be expressed as a fraction of integers. These are those numbers which in decimal form do not repeat or terminate. The set of all numbers which are rational or irrational is the set of *real numbers* and is denoted $\mathbb{R}$.

Set Builder Notation. Sometimes we need to describe a set which is too big or complicated to simply list. In this case, we can sometimes use **set builder** notation. This notation looks like:

$$\{x : P(x)\} \text{ or } \{x \in S : P(x)\}.$$

The colon inside the braces is read as "such that." The notation on the left is defined to mean the set of all x such that $P(x)$ is true. This means that an element x is in the set if and only if the statement $P(x)$ is true. The notation on the right is similar; however, this notation lets us restrict our attention to things which are in the set S. This is the set of all x in the set S for which the statement $P(x)$ is true.

Example 9.1. Use set builder notation to describe the set of even integers.

Solution: The even integers are those integers which are multiples of 2. A number n is a multiple of 2 if $n = 2k$ for some number k, that is, n is even if $(\exists k \in \mathbb{Z})(n = 2k)$. Therefore, our set builder notation for the set of even integers is $\{n \in \mathbb{Z} : (\exists k \in \mathbb{Z})(n = 2k)\}$.

Abuse of Notation. We can abbreviate set builder notation like we just built in the last example. This set might be written as $\{2k : k \in \mathbb{Z}\}$. This is the set of all elements that look like $2k$ for some integer k. This type of set builder notation looks like $\{f(x) : P(x)\}$ where f is some function. We will talk about functions later.

Example 9.2. Use set builder notation to describe the interval $[2, \infty)$ in $\mathbb{R}$.

Solution: This is the set of real numbers which are greater than or equal to 2, or $\{x \in \mathbb{R} : x \geq 2\}$.

Example 9.3. Use set builder notation to describe the set of real solutions to the equation $x^2 - 3x + 2 = 0$.

Solution: This is easier than it sounds: $\{x \in \mathbb{R} : x^2 - 3x + 2 = 0\}$.

Example 9.4. Let $H(x)$ be the predicate "x is a horse" and let $B(x)$ be "x is brown." Use set builder notation to describe the set of brown horses.

Solution: An element x is in this set if it is a horse – $H(x)$ – and it is brown – $B(x)$. So this is the set $\{x : H(x) \wedge B(x)\}$.

Definition 9.5. A set A is a *subset* of another set B if every element of A is also an element of B. We denote this relationship by $A \subseteq B$. This notation is read as "A is a subset of B." If $A \subseteq B$ but A is not the same set as B, then we say A is a *proper subset* of B. This is denoted as $A \subset B$.

Warning. Some older textbooks (and folks educated out of them) use the symbol $\subset$ for $\subseteq$. Be aware of this if you are getting help online or from a different text.

Caution. It is very important not to confuse the symbols $\in$ and $\subseteq$. The notation $X \in Y$ implies that Y is a set and X is a single element of that set. The notation $X \subseteq Y$ implies that X and Y are both sets and X is a subset of Y. However, there are times when X and Y might both be sets and we still have $X \in Y$. For example, this happens if $X = \{1, 2\}$ and $Y = \{\{1, 2\}, 3\}$. Note here that Y is a set with two elements. They are $\{1, 2\}$ and 3.

Definition 9.6. The *empty set* (denoted $\emptyset$) is the set which contains no elements. The empty set is a subset of every set (including itself) since the implication "If $x \in \emptyset$ then $x \in A$" is always true because $x \in \emptyset$ is always false. Note that $\emptyset = \{\,\}$.

Example 9.7. List all of the subsets of $\{a, b, c\}$.

Solution: We list the subsets from smallest to largest. First, there is one subset with zero elements – $\emptyset$. Next we move to subsets with one element. There are three of these – $\{a\}$, $\{b\}$, $\{c\}$. Next we list the two elements subsets. For a larger set, these may be difficult to list. For this set, it is easiest to focus on what element is being left out. We can leave out a, b, or c to get three subsets – $\{b, c\}$, $\{a, c\}$, $\{a, b\}$. Finally, there is one subset with three elements, the entire set – $\{a, b, c\}$. The set of subsets of $\{a, b, c\}$ (which we will name below) is

$$\{\emptyset, \{a\}, \{b\}, \{c\}, \{b, c\}, \{a, c\}, \{a, b\}, \{a, b, c\}\}.$$

Intersection. If A and B are sets, then the *intersection* of A and B (denoted $A \cap B$) is the set

$$A \cap B = \{x : (x \in A) \wedge (x \in B)\}$$

For example, if A is the set of even integers and B is the set of multiples of three, then $A \cap B$ is the set of even multiples of three. This is the set of multiples of six.

Union. If A and B are sets, then the *union* of A and B (denoted $A \cup B$) is the set

$$A \cup B = \{x : (x \in A) \vee (x \in B)\}$$

For example, if $A = \{1, 2, 3\}$ and $B = \{2, 3, 4\}$, then $A \cup B = \{1, 2, 3, 4\}$.

Difference. If A and B are sets, then the *difference*, $A - B$, of A and B is the set

$$A - B = \{x : (x \in A) \wedge (x \notin B)\}$$

For example, if $A = \{a, b, c\}$, $B = \{b, d, f\}$ and $C = \{a, b, c, d\}$, then $A - B = \{a, c\}$ and $A - C = \emptyset$. It is not uncommon for some texts to use the confusing notation $A \setminus B$ for $A - B$.

Complement. In some situations, there is a *universe of discourse* U which contains all of the elements that we are concerned with at the time. When there is such a universe, then we may want to talk about the

elements which are not in a set A. We call this the *complement* of A. That is, the complement of A is the set

$$\bar{A} = U - A = \{x \in U : x \notin A\}.$$

Some texts refer the complement as A'.

Cartesian Product. The *Cartesian product* of two sets A and B (denoted by $A \times B$) is the set of all ordered pairs (a, b) so that $a \in A$ and $b \in B$. For example, if $A = \{1, 2, 3\}$ and $B = \{a, b\}$, then

$$A \times B = \{(1, a), (1, b), (2, a), (2, b), (3, a), (3, b)\}.$$

It is important to remember that in the pairs (a, b) order matters. The Cartesian plane on which we learn to graph in elementary algebra and calculus can be thought of as $\mathbb{R} \times \mathbb{R}$. If there is some algebraic structure that we care about on the sets in a Cartesian product, then we use the words *direct product* to describe the Cartesian product with with the inherited algebraic structure. This can be confusing at times, so we will likely use both terms to refer to $A \times B$. We may simply call this the product of the sets. We will never call it the cross product. That is a product between vectors, not sets. Unfortunately, it uses the same notation.

Powerset. The *powerset* of a set A (denoted by $\mathcal{P}(A)$) is the set of all subsets of A. For example, if $A = \{a, b, c\}$, then

$$\mathcal{P}(A) = \{\emptyset, \{a\}, \{b\}, \{c\}, \{a, b\}, \{a, c\}, \{b, c\}, \{a, b, c\}\}.$$

Note. Note that the symbol for intersection resembles that for "and." This is no accident. The word "and" is important in the definition of intersection. Similarly, note that the symbol for union resembles that for "or."

Example 9.8. Working in the universe $U = \{a, b, c, d, e, f, g\}$, let $A = \{a, b, c, d, e\}$, $B = \{c, d, f\}$, and $C = \{a, f\}$. Find the indicated sets.

1. $A \cap B$
2. $B \cup C$
3. $B - A$
4. $B \times C$
5. $\mathcal{P}(C)$
6. $(A \cap B) \cup C$
7. $A - (B \cup C)$
8. $\bar{A}$

Solution:

1. To calculate $A \cap B$, we step through each element of A and ask if that element is in B also. If so, we include it in the intersection. This gives

$$A \cap B = \{c, d\}.$$

2. To calculate $B \cup C$, we list all of the elements of B and all of the elements of C in braces. Then, we remove any duplicates. For readability, we alphabetize the elements.

$$B \cup C = \{c, d, f, a, f\} = \{a, c, d, f\}.$$

3. To calculate $B - A$, we list the elements of B, and then we cross out those in the list which are in A.

$$B - A = \{\cancel{c}, \cancel{d}, f\} = \{f\}.$$

4. The set $B \times C$ is all ordered pairs where the first element comes from B and the second from C.

$$B \times C = \{(c, a), (d, a), (f, a), (c, f), (d, f), (f, f)\}$$

.

5. The powerset of C contains subsets with zero, one, or two elements.

$$\mathcal{P}(C) = \{\emptyset, \{a\}, \{f\}, \{a, f\}\}.$$

6. For $(A \cap B) \cup C$, we work inside parentheses and first find $A \cap B$. Luckily we have already done this and know that $A \cap B = \{c, d\}$. Now, we union this with C to get

$$(A \cap B) \cup C = \{c, d\} \cup \{a, f\} = \{a, c, d, f\}.$$

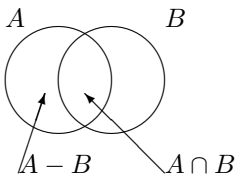
7. For $A - (B \cup C)$, we first need $B \cup C$, which we already know to be $B \cup C = \{a, c, d, f\}$. Then we find the difference.

$$A - (B \cup C) = \{a, b, c, d, e\} - \{a, c, d, f\} = \{b, e\}.$$

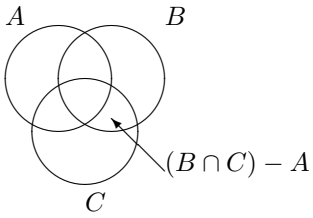
8. For $\bar{A}$ we want those elements of U which are not in A. These are

$$\bar{A} = \{f, g\}.$$

Venn Diagrams. Diagrams can be drawn to help visualize set operations. Each set is represented by an oval or a circle. The intersection, union, and difference of the sets can then be seen geometrically:



The union of A and B would be all of the area inside either circle. These diagrams of circles are called *Venn Diagrams*. Venn Diagrams can also be used to visualize more complicated set operations:



Example 9.9. Let $A = \{1, 2, 3, 4, 5, 6\}$ and $B = \{4, 5, 6, 7, 8, 9\}$. Calculate $(A - B) \cup (B - A)$.

Solution: First, $A - B = \{1, 2, 3\}$ and $B - A = \{7, 8, 9\}$. Then $(A - B) \cup (B - A) = \{1, 2, 3, 7, 8, 9\}$. These are the elements that A and B do not have in common.

Symmetric Difference. The set $(A - B) \cup (B - A)$ calculated in the last example is called the *symmetric difference* of A and B and is denoted by $A \oplus B$. Note that this is the same notation that we used for the exclusive or. We use this notation because $(A - B) \cup (B - A)$ consists of those elements which are in one set or the other but not both.

Set Equality. If A and B are sets, then $A = B$ if and only if $A \subseteq B$ and $B \subseteq A$.

Set Identities. Logical equivalences can be applied to the definitions of set operations to establish identities using those operations. The following are true for any sets A, B, and C.

$A \cap B = B \cap A$	commutative law
$A \cup B = B \cup A$	commutative law
$A \cap (B \cap C) = (A \cap B) \cap C$	associative law
$A \cup (B \cup C) = (A \cup B) \cup C$	associative law
$A \cap A = A$	idempotent law
$A \cup A = A$	idempotent law
$A \cap (A \cup B) = A$	absorption law
$A \cup (A \cap B) = A$	absorption law
$A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$	distributive law
$A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$	distributive law
$A - (B \cap C) = (A - B) \cup (A - C)$	DeMorgan's Law
$A - (B \cup C) = (A - B) \cap (A - C)$	DeMorgan's Law

Exercises 9.10. Complete the following questions.

1. Use set builder notation to describe each of the following sets.
   (a) The set of real solutions to the equation $x^3 = x$.
   (b) The set of all integer multiples of 6.
   (c) The set of all real numbers which are less than -2 or greater than 2.
   (d) The set of all numbers which can be written as an integer divided by a positive power of 2.
   (e) The set of all real numbers which are larger than their squares.
2. Let $E(x)$ be "x is even." Let $O(x)$ be "x is odd." Let $P(x)$ be "x is prime." Recall that $\mathbb{Z}$ is the set of integers, so to say "x is an integer" you can write $x \in \mathbb{Z}$. Write the following sets in set-builder notation.
   (a) The set of integer multiples of 3.
   (b) The set of odd integers less than 10.
   (c) The set of all integers which are either even or greater than 10.
   (d) The set of all integers which are prime.
   (e) The set of all even prime integers.
3. Let U be the set of all bit strings of length 4 (there are 16 of them). Let B be the set of elements of U which begin with 1. Let E be the set of elements of U which end with 1. Let T be the set of elements of U with exactly two 1s. Draw a Venn diagram for U showing B, E, and T with all sixteen elements of U in the appropriate regions.
4. List the elements of these sets
   (a) $\{x \in \mathbb{N} : x^2 < 10\}$
   (b) $\{x \in \mathbb{N} : (x < 20) \wedge \exists y[(y \in \mathbb{N}) \wedge ((x = 3y) \vee (x = 5y))]\}$
   (c) $\{n : (n \in \mathbb{N}) \wedge \exists m[(m \in \mathbb{N}) \wedge (n = m^2)] \wedge (n < 20)\}$
   (d) $\{x \in \mathbb{N} : \exists a \exists b[(a \in \mathbb{N}) \wedge (b \in \mathbb{N}) \wedge (a > 1) \wedge (b > 1) \wedge (a < 5) \wedge (b < 5) \wedge (x = ab)]\}$
5. Let $A = \{a, b, c\}$, $B = \{a, b\}$, $C = \{b, c, d\}$, and $D = \{a, b, B\}$. Fill in the blank with $\in$ or $\subseteq$ or both.
   (a) a____A
   (b) B____A
   (c) B____D
   (d) $\emptyset$____C

6. Answer the questions below about subsets.

   (a) List all of the subsets of $\emptyset$

   (b) List all of the subsets of $\{1\}$

   (c) List all of the subsets of $\{1, 2\}$

   (d) List all of the subsets of $\{1, 2, 3\}$

   (e) How many subsets should $\{1, 2, 3, 4\}$ have?

   (f) Guess at a formula for the number of subsets of an n-element set.

7. Let $A = \{a, b, c\}$, $B = \{a, b\}$, $C = \{b, c, d\}$, and $D = \{a, b, B\}$. Fill in the blank with $\in$ or $\notin$.

   (a) a____A

   (b) b____B

   (c) c____B

   (d) B____D

   (e) B____A

   (f) (a, b)____$A \times C$

   (g) (a, b)____$C \times D$

   (h) ab____$A \times C$

   (i) 4____$\mathbb{N} \times \mathbb{Z}$

   (j) 4____$\mathbb{Z}$

   (k) -4____$\mathbb{Z}$

   (l) -4____$\mathbb{N}$

8. Let A, B, and C be the sets

$$A = \{a, b, c, d, e\}$$
$$B = \{c, d, e, f, g\}$$
$$C = \{a, c, d, e, h, i, j\}.$$

   Draw a Venn diagram with the elements $\{a, b, c, d, e, f, g, h, i\}$ in the appropriate regions.

9. Shade each of these sets in Venn diagrams.

   (a) $(A \cap B) \cup (A \cap C)$

   (b) $A - (A \cap B \cap C)$

   (c) $(C - (A \cap B)) - (A \cap C)$

10 Powers and Strings

Example 10.1. Let $A = \{0, 1\}$. Calculate $(A \times A) \times A$ and $A \times (A \times A)$. Are these sets the same?

Solution: $A \times A$ is the set of order pairs of elements from A:

$$A \times A = \{(0, 0), (0, 1), (1, 0), (1, 1)\}.$$

Then $(A \times A) \times A$ is the set of ordered pairs, where the first element is from $A \times A$ and the second is from A. Note, the first element of these ordered pairs is itself an ordered pair. Each element of $A \times A$ shows up twice – once paired with 0 and once paired with 1.

$$(A \times A) \times A = \{((0, 0), 0), ((0, 1), 0), ((1, 0), 0), ((1, 1), 0), ((0, 0), 1), ((0, 1), 1), ((1, 0), 1), ((1, 1), 1), \}$$

The set $A \times (A \times A)$ is similar, except that the elements of $A \times A$ come second in the pairs.

$$A \times (A \times A) = \{(0, (0, 0)), (0, (0, 1)), (0, (1, 0)), (0, (1, 1)), (1, (0, 0)), (1, (0, 1)), (1, (1, 0)), (1, (1, 1)), \}$$

These sets are not equal to each other. They appear similar, but the parentheses are in different locations.

Note. The fact that $(A \times A) \times A \neq A \times (A \times A)$ in the last example indicates that the Cartesian product is not associative. However, we may want to find products of more than two sets without having to worry about the parentheses. Therefore, we define a general Cartesian product of more than two sets.

Definition 10.2. Suppose that $A_1, A_2, \ldots, A_n$ are sets. The *Cartesian product* of these sets is the set

$$A_1 \times A_2 \times \cdots \times A_n = \{(a_1, a_2, \ldots, a_n) : \forall i(a_i \in A_i)\}.$$

Elements of the form (a_1, a_2) are called ordered pairs. Elements of the form (a_1, a_2, a_3) are ordered triples. In general, elements of the form $(a_1, a_2, \ldots, a_n)$ are *ordered n-tuples*.

Example 10.3. Let $A = \{0, 1\}$, $B = \{a, b\}$, and $C = \{c, d\}$. Find $A \times B \times C$.

Solution: $A \times B \times C$ is the set of all ordered triple where the first element comes from A, the second from B, and the third from C.

$$A \times B \times C = \{(0, a, c), (0, a, d), (a, b, c), (a, b, d), (1, a, c), (1, a, d), (1, b, c), (1, b, d)\}.$$

Definition 10.4. If A is any set and if n is an integer greater than 1, then the *Cartesian power* A^n is the set of all ordered n-tuples of elements of A. That is

$$A^n = A \times A \times \cdot \times A = \{(a_1, a_2, \ldots, a_n) : \forall i(a_i \in A)\}.$$

For convenience, we let $A^1 = A$.

Example 10.5. Let $A = \{0, 1\}$. Find A^3.

Solution: $A^3 = \{(0, 0, 0), (0, 0, 1), (0, 1, 0), (0, 1, 1), (1, 0, 0), (1, 0, 1), (1, 1, 0), (1, 1, 1)\}$.

Definition 10.6. Suppose that A is any set and n is a positive integer. A *string of length n over A* is an element of A^n written with no parentheses or commas. The elements of A are *characters* of the string, and A is called the *alphabet*. The *empty string* is the string with no characters in it. This is denoted λ (the lowercase Greek letter lambda). The set of all strings of length n over A is denoted (rather unfortunately) as A^n. The set of all strings over A is denoted A^*. The set of all non-empty strings over A is A^+.

Note. The notation A^n for strings of length n is unfortunate because of our notation for the Cartesian power A^n. We will avoid the notation when possible. However, context will usually tell which we mean.

Example 10.7. Find all strings of length 3 over $A = \{0, 1\}$.

Solution: The strings are $\{000, 001, 010, 011, 100, 101, 110, 111\}$. This is the same as the last example with the commas and parentheses removed.

Definition 10.8. Strings over the alphabet $\{0, 1\}$ are called *bit strings*.

By identifying 0 with false and 1 with true, we can apply our logical operators to bit strings in a *bitwise* manner – that is, one bit at a time.

Example 10.9. Calculate the bitwise operations: $\neg(11010)$, $11010 \wedge 01110$, $11010 \vee 01110$, and $11010 \oplus 01110$.

Solution: We perform each operation one bit at a time. For $\neg(11010)$, we simply exchange 0s and 1s: $\neg(11010) = 00101$. We stack the rest and apply each operation vertically.

$$\begin{array}{rccccc} & 1 & 1 & 0 & 1 & 0 \\ \wedge & 0 & 1 & 1 & 1 & 1 \\ \hline & 0 & 1 & 0 & 1 & 0 \end{array} \qquad \begin{array}{rccccc} & 1 & 1 & 0 & 1 & 0 \\ \vee & 0 & 1 & 1 & 1 & 1 \\ \hline & 1 & 1 & 1 & 1 & 1 \end{array} \qquad \begin{array}{rccccc} & 1 & 1 & 0 & 1 & 0 \\ \oplus & 0 & 1 & 1 & 1 & 1 \\ \hline & 1 & 0 & 1 & 0 & 1 \end{array}$$

Exercises 10.10. Answer the following questions.

1. Calculate $A \times B \times C$ if $A = \{1\}$, $B = \{1, 2\}$, and $C = \{1, 2, 3\}$.
2. List all strings with length 1 over the alphabet $A = \{a, b, c\}$.
3. Modify your answer to the last quest to list all strings with length 2 over the alphabet $A = \{a, b, c\}$. Try to do so in a regular way that would be programmable.
4. Modify your answer to the last quest to list all strings with length 3 over the alphabet $A = \{a, b, c\}$. Try to do so in a regular way that would be programmable.
5. Let $x = 1101101$ and $y = 1010101$. Calculate $z = x \oplus y$ (do your operations bitwise). Then calculate $z \oplus y$. What do you notice?
6. The last exercise is the basis for a cipher known as the Vernam cipher. In this cipher, the plaintext (the message to be scrambled) is converted to a long bitstring. Then, an encryption bitstring is created (from a key). The plaintext bitstring and encryption bitstring are then XORed to create the ciphertext (the scrambled message). We experiment with an extremely simplified version of this system in this exercise. Note that this is a toy system, paired down so that computations by hand are easy.

   We are going to encode characters as four bits each. Since there are only 16 length 4 bitstrings, and since we have 26 letters in the alphabet, we have to identify some letters. The chart below shows how we will encode each letter when converting our plaintext to a bitstring.

letter	bitstring	letter	bitstring
A	0000	L	1000
B	0001	M, N	1001
D	0010	O	1010
E	0011	P	1011
F	0100	R	1100
G	0101	S, Z	1101
H, I, J, Y	0110	T	1110
C, K, Q	0111	U, V, W	1111

   We use a linear congruential random number generator with modulus 16 to generate our encryption bitstring. The modulus of 16 is used here to ensure bitstrings of length 4 to XOR with our plaintext bitstring. For this example, we will use $m = 3$ as a multiplier, $b = 7$ as an increment, and $x = 9$ as an initial seed. In this exercise, we encrypt the word COWS.

   (a) Convert the letters COWS to a bitstring using the table above. Leave a space every 4 bits for readability.

(b) Use the values of m, b, and x above and a modulus of 16 to calculate the next 4 random numbers using a linear congruential random number generator. (Do not include the initial seed as one of these four numbers).

(c) Convert each of the four random numbers to binary using 4 bits each (so you may have to pad some 0s) and combine them into one bitstring (from left to right). Leave a space every 4 bits for readability.

(d) XOR the two bitstrings you have computed.

(e) Use the table above to convert four bits at a time of this new bitstring to letters. When there is a choice of letters, choose whichever you like.

7. The nice thing about the encryption system we outlined in the last question is that encryption and decryption work the same way. Follow the steps above to decrypt this message: OCFO

11 Relations

Predicates express properties of and relationships between the variables involved. For example

x is taller than y.

expresses a relationship between x and y. The sentence

x robbed the bank.

expresses the property that x may have of having robbed the bank. Mathematicians use the notion of a relation to describe abstract properties and relationships.

Definition 11.1. P is a *1-ary (or unary) relation* on a set A if P is a subset of A. When thinking of a subset P as a relation, we can write $P(x)$ for $x \in P$. If we let P be the set of all people who robbed the bank. Then $P(x)$ and $x \in P$ both mean the same thing as the predicate "x robbed the bank."

Definition 11.2. P is a *2-ary (or binary) relation* on a set A if P is a subset of $A \times A$ (so P is a set of ordered pairs of A such as (x, y)). When thinking of P as a relation, we can write $P(x, y)$ or xPy for $(x, y) \in P$. We will also use the words "x is P-related to y" to express $(x, y) \in P$. If we let P be the set of all pairs of people (x, y) so that x is the father of y, then $P(x, y)$, xPy, and $(x, y) \in P$ all mean the same thing as the predicate "x is the father of y."

Definition 11.3. P is a *3-ary (or ternary) relation* on a set A if P is a subset of $A \times A \times A$ (so P is a set of ordered triples of A such as (x, y, z)). When thinking of P as a relation, we can write $P(x, y, z)$ for $(x, y, z) \in P$.

Definition 11.4. If n is a positive integer, then P is *an n-ary relation* on a set A if P is a subset of A^n so that P is a set of n-tuples of A such as $(x_1, x_2, \ldots, x_n)$. When thinking of P as a relation, we can write $P(x_1, x_2, \ldots, x_n)$ for $(x_1, x_2, \ldots, x_n) \in P$. For emphasis, we will also say "$P(x_1, x_2, \ldots, x_n)$ is true" to mean that "$(x_1, x_2, \ldots, _n) \in P$ is true."

Note. The notation we are using for relations is intended to match exactly the notation we have used for predicates and open formulas. The two concepts are intimately related. An open formula $P(x, y)$ can be used to define a set – the set of all pairs (x, y) for which $P(x, y)$ is true. In set builder notation, this is $\{(x, y) : P(x, y)\}$. This set of ordered pairs can be treated as a set or a relation. We can call both the set and the relation P. In this case, $P(x, y)$ means exactly the same thing as $(x, y) \in P$. We are intentionally blurring the lines between subset, relation, and predicate.

Infix vs. Prefix Notation. The notation $R(x, y)$ is called *prefix notation*. The notation xRy is *infix notation*. For binary relations, we will usually prefer infix notation since we commonly use this notation with relations such as $=$ and $\leq$.

Example 11.5. Suppose that $A = \{1, 2, 3, 4\}$. Define this relation on A:

$$P = \{(x, y, z) : y \text{ is strictly between } x \text{ and } z\}.$$

Is $P(1, 2, 3)$ true? Is $P(2, 1, 3)$ true? Give one other true statement using P and one other false statement.

Solution: $P(1, 2, 3)$ is true because 2 is strictly between 1 and 3. $P(2, 1, 3)$ is not true because 1 is not strictly between 2 and 3. Another true statement is $P(2, 3, 4)$. Another false statement is $P(1, 1, 1)$.

Example 11.6. Let $A = \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$. Let

$$P = \{x \in A : x \text{ is prime}\}.$$

Give a statement of the form $P(x)$ which is false and one which is true.

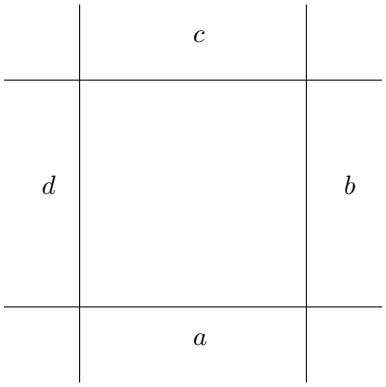
Solution: $P(1)$ is false because 1 is not prime. $P(2)$ is true because 2 is prime. We can list the elements of P as $\{2, 3, 5, 7\}$.

Example 11.7. Let A be the set $\{2, 3, 4, 5, 6\}$. Define a binary relation D on A so that for all $x, y \in A$, xDy exactly when y is a multiple of x. Use D to write a statement which is true and a statement which is false. Write the true statement using prefix notation, infix notation, and set notation.

Solution: Since 6 is a multiple of 2 we know that $D(2, 6)$. This can be written as $2D6$ or $(2, 6) \in D$. Since 6 is not a multiple of 5, $P(5, 6)$ is false. As a set of ordered pairs, D is

$$D = \{(2, 2), (2, 4), (2, 6), (3, 3), (3, 6), (4, 4), (5, 5), (6, 6)\}.$$

Example 11.8. Let A be the set of lines in this figure:



So $A = \{a, b, c, d\}$. Define a relation I on A so that xIy exactly when line x intersects line y. Give two pairs that are in I and two pairs that are not in I.

Solution: Both aIb and dIc are true so (a, b) and (d, c) are in I. However, dIb and aIc are false, so (d, b) is not in I and (a, c) is not in I.

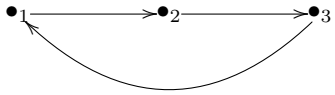
Definition 11.9. We can draw pictures of small sets with binary relations. The picture consists of one point for each element of the set. For each relation xRy, there is an arrow in the picture from x to y. Such a pictorial representation is called a *directed graph* or *digraph*.

Example 11.10. Draw a directed graph for the relation

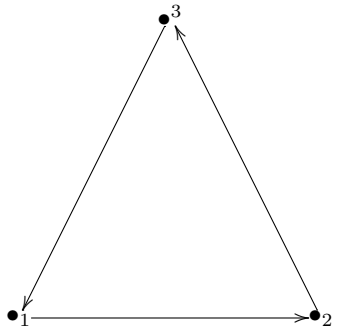
$$R = \{(1, 2), (2, 3), (3, 1)\}$$

on the set $A = \{1, 2, 3\}$.

Solution:

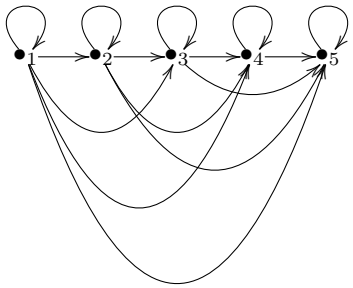


Note. What matters in directed graphs is how the points are connected by arrows, not the physical arrangement of the points in the picture. Thus, this is another depiction of the graph from the previous example:



Example 11.11. Draw a directed graph for the relation $\leq$ on the set $\{1, 2, 3, 4, 5\}$.

Solution:



Definition 11.12. Suppose that A and B are sets. A relation from A to B is a subset R of $A \times B$.

Note. A relation from A to B is just a set of ordered pairs (a, b) where $a \in A$ and $b \in B$.

Example 11.13. Let A be the set of all students at a certain college and let B be the set of all classes being offered at that college. There is a relation R from A to B so that $(a, b) \in R$ means that a is taking class b.

Exercises 11.14. Answer the questions below.

1. List two more pairs which are not in the relation I of Example 11.8 and two more pairs that are.
2. Define a relation S on $\mathbb{N}$ so that $S(x, y)$ means that $x + 1 = y$. List a few pairs in the relation S. It turns out that most of the properties of the natural numbers which we studied in arithmetic can be derived from this relation.
3. Let $A = \{1, 2, 3, 4, 5, 6\}$. Define R on A so that xRy means that $x - y \in A$. List the pairs in R.
4. Let E and F be two unary relations on $\mathbb{N}$ so that $E(x)$ means "x is even" and $F(x)$ means "x is a multiple of 4."
   (a) Find an x which makes $E(x) \wedge F(x)$ true.
   (b) Find an x which makes $E(x) \wedge \neg F(x)$ true.
   (c) Is this statement true? $\forall x(E(x) \to F(x))$
   (d) Is this statement true? $\forall x(F(x) \to E(x))$
   (e) Is this statement true? $\exists x(E(x) \to F(x))$
   (f) Is this statement true? $\exists x(F(x) \to E(x))$
5. Draw digraphs of relations $\Rightarrow$ which satisfy each of these statements.
   (a) $\forall x(x \Rightarrow x)$
   (b) $\forall x \forall y(x \Rightarrow y)$

(c) $\exists x \forall y (x \Rightarrow y)$

(d) $\exists x \exists y \exists z [(x \Rightarrow y) \wedge (y \Rightarrow z)]$

(e) $\forall x \forall y \neg [(x \Rightarrow y) \wedge (y \Rightarrow x)]$

(f) $\forall x \forall y ([(x \Rightarrow y) \vee (y \Rightarrow x)] \wedge \neg [(x \Rightarrow y) \wedge (y \Rightarrow x)])$

12 Properties of Relations

Definition 12.1. A relation R on a set A is **reflexive** if aRa for every $a \in A$. This could also be written as $(a, a) \in R$ for all $a \in A$.

Example 12.2. Is the relation

$$R = \{(a, a), (a, c), (b, d), (b, b), (c, c), (d, d)\}$$

on $S = \{a, b, c, d\}$ reflexive?

Solution: Since R contains (a, a), (b, b), (c, c), and (d, d), R is reflexive.

Example 12.3. Is the relation $<$ on the real numbers reflexive?

Solution: The relation $<$ on $\mathbb{R}$ is not reflexive because it is not the case that $1 < 1$.

Example 12.4. Is the relation $=$ on $\mathbb{R}$ reflexive?

Solution: The relation $=$ is reflexive on $\mathbb{R}$ since $x = x$ is true for all real numbers x.

Definition 12.5. A relation R on a set A is *symmetric* if for all $a, b \in A$ whenever aRb, then also bRa.

Example 12.6. Is the relation $R = \{(a, b), (b, a), (c, c), (d, c), (c, d)\}$ on the set $S = \{a, b, c, d\}$ symmetric?

Solution: This relation is symmetric because if we reverse any ordered pair in R we get another ordered pair in R.

Example 12.7. Is the relation $R = \{(a, b), (d, c), (c, d)\}$ on the set $S = \{a, b, c, d\}$ symmetric?

Solution: This relation is not symmetric because $(a, b) \in R$ but $(b, a) \notin R$.

Example 12.8. Is the relation $\leq$ on $\mathbb{R}$ symmetric?

Solution: The relation $\leq$ on $\mathbb{R}$ is not symmetric since $1 \leq 2$, but it is not the case that $2 \leq 1$.

Example 12.9. Is the relation $=$ on $\mathbb{R}$ symmetric?

Solution: The relation $=$ is symmetric on $\mathbb{R}$. If $x = y$, then x and y are actually the same number, so $y = x$.

Definition 12.10. A relation R on a set A is *anti-symmetric* if for all $a, b \in A$ whenever aRb and bRa then also $a = b$. This is equivalent to, if $a \neq b$ and $(a, b) \in R$, then $(b, a) \notin R$.

Example 12.11. Is the relation $\leq$ on $\mathbb{R}$ is anti-symmetric?

Solution: The relation $\leq$ on $\mathbb{R}$ is anti-symmetric. If $x \leq y$ and $y \leq x$, then $x = y$.

Example 12.12. Is the relation $=$ on $\mathbb{R}$ is anti-symmetric?

Solution: The relation $=$ on $\mathbb{R}$ is anti-symmetric. If $x = y$ and $y = x$, then $x = y$.

Example 12.13. Is the relation $R = \{(a, b)(b, c)(d, d)\}$ on $S = \{a, b, c, d\}$ anti-symmetric?

Solution: If we reverse any of the ordered pairs in this relation other than (d, d), we do not get another element of R. Therefore, if xRy and yRx, we have that $x = d = y$.

Definition 12.14. A relation R on a set A is *transitive* if for all $a, b, c \in A$ the relations aRb and bRc together imply aRc.

Example 12.15. Are the relations $\leq$ and $=$ on $\mathbb{R}$ transitive?

Solution: These relations are transitive. They are, in fact, the motivation behind the definition of transitivity.

Example 12.16. Is the relation $R = \{(c, d), (d, a), (c, a)\}$ on $S = \{a, b, c, d\}$ transitive?

Solution: R is transitive. The only case where we have xRy and yRx is cRd and dRa. Since cRa, R is transitive.

Example 12.17. Is the relation $R = \{(b, d), (d, a)\}$ on $S = \{a, b, c, d\}$ transitive?

Solution: This relation is not transitive. We have bRd and dRa, so to be transitive we would also need to have bRa.

Example 12.18. Is the relation $R = \{(a, b), (b, a)\}$ on $S = \{a, b, c, d\}$ transitive?

Solution: This relation is not transitive. We have aRb and bRa, so to be transitive we would also need to have aRa.

Definition 12.19. A relation R on a set A is called an *equivalence relation* if it is reflexive, symmetric, and transitive.

Example 12.20. The equality relation $(=)$ is an equivalence relation on any set. Actually, this relation is the motivation behind the concept of equivalence relation. An equivalence relation is a relation that behaves somewhat like equality.

Example 12.21. Is the relation R on the real numbers defined by xRy if $x^2 = y^2$ an equivalence relation?

Solution: R is an equivalence relation. For any real number x, $x^2 = x^2$ so xRx. This means that R is reflexive. For any real numbers x and y, if if xRx, then $x^2 = y^2$. Then $y^2 = x^2$, so yRx. Thus, R is symmetric. For any real numbers x, y, and z, if xRy an yRz, then $x^2 = y^2$ and $y^2 = z^2$, so $x^2 = z^2$. This means that xRz. Thus R is also transitive.

Note. It turns out that every equivalence relation R can be realized like the one in the previous example. There is some function so that xRy means $f(x) = f(y)$. In the example, $f(x) = x^2$. Every proof that a relation is an equivalence relation can be formulated like the last example.

Example 12.22. If n is any positive integer then congruence modulo n is an equivalence relation on $\mathbb{Z}$.

Definition 12.23. Suppose R is an equivalence relation on a set A. If $a \in A$, then the *equivalence class of a modulo R* is the set $\{x \in A : aRx\}$. The equivalence class of a modulo R is denoted as $[a]_R$ or a/R. If R is understood from context, we will sometimes just write $[a]$. Note that just by this definition the statement $x \in [a]_R$ means the same thing as aRx. The set of all equivalence classes of R is denoted A/R. This is the *factor set of A modulo R.*

Example 12.24. The equivalence classes of congruence modulo 3 on $\mathbb{Z}$ are

$$\{\ldots, -6, -3, 0, 3, 6, 9, \ldots\}$$

$$\{\ldots -5, -2, 1, 4, 7, 10, \ldots\}$$

$$\{\ldots -4, -1, 2, 5, 8, 11, \ldots\}.$$

Example 12.25. Let $A = \{1, 2, 3, 4\}$. Let $R = \{(1, 1), (2, 2), (3, 3), (4, 4), (1, 2), (2, 1), (3, 4), (4, 3)\}$. R is an equivalence relation. Find $[1]_R$, $[2]_R$, $[3]_R$, and $[4]_R$.

Solution: All of the elements to which 1 is related are $[1]_R = \{1, 2\}$. All of the elements to which 2 is related are $[2]_R = \{1, 2\}$. All of the elements to which 3 is related are $[3]_R = \{3, 4\}$. All of the elements to which 4 is related are $[4]_R = \{3, 4\}$.

Note. Notice in the last example that every element of A is in an equivalence class and that the equivalence classes are disjoint. If we select two elements of A, either they have the same equivalence class or their equivalence classes are disjoint. We have a special name for this situation.

Definition 12.26. A *partition* of a set A is a set P of nonempty subsets of A so that

- If $a \in A$, then there is a set $D \in P$ so that $a \in D$.
- If $E, F \in P$ and $E \neq F$, then $E \cap F = \emptyset$.

The elements of P are called the *partition classes* or *blocks* of P.

Example 12.27. Each of the following is an example of a partition on $\{1, 2, 3, 4, 5\}$.

$$\{\{1, 2\}, \{3, 4\}, \{5\}\}$$

$$\{\{1, 2, 3, 4, 5\}\}$$

$$\{\{1\}, \{2\}, \{3\}, \{4\}, \{5\}\}$$

$$\{\{1, 2, 3\}, \{4, 5\}\}$$

Example 12.28. This is a partition of $\mathbb{N}$: $\{\{n \in \mathbb{N} : n \text{ is even}\}, \{n \in \mathbb{N} : n \text{ is odd}\}\}$.

Equivalence Relations and Partitions. Suppose R is an equivalence relation on a set A. We know that if $a \in A$, then $a \in [a]_R$. We also know that if two equivalence classes $[a]_R$ and $[b]_R$ are different, then $[a]_R \cap [b]_R = \emptyset$. These two facts imply that the set of equivalence classes of R (the factor set A/R) is a partition of A.

Suppose P is a partition of a set A. Define a relation $\sim_P$ on A by $a \sim_P b$ if a and b are in the same partition class. Then the relation $\sim_P$ is an equivalence relation on A. The equivalence classes of $\sim_P$ are precisely the partition classes of P.

Exercises 12.29. Answer the questions below.

1. Draw one digraph of a relation on $\{a, b, c\}$ which is reflexive and one which is not reflexive. Conjecture how to decide from the digraph whether or not a relation is reflexive.
2. Draw one digraph of a relation on $\{a, b, c\}$ which is symmetric and one which is not symmetric. Conjecture how to decide from the digraph whether or not a relation is symmetric.
3. Draw one digraph of a relation on $\{a, b, c\}$ which is anti-symmetric and one which is not anti-symmetric. Conjecture how to decide from the digraph whether or not a relation is anti-symmetric.
4. Draw one digraph of a relation on $\{a, b, c\}$ which is transitive and one which is not transitive. Conjecture how to decide from the digraph whether or not a relation is transitive.
5. Determine if each of the following relations on the set $\{a, b, c, d\}$ is reflexive, symmetric, anti-symmetric, or transitive.
   (a) $R = \{(a, b), (b, c), (c, d)\}$
   (b) $R = \{(a, a), (c, c), (d, d)\}$
   (c) $R = \{(a, b), (b, a), (a, a), (c, c), (d, d)\}$
   (d) $R = \emptyset$
   (e) $R = \{(a, b), (a, a), (d, d), (b, a), (c, c), (d, c), (c, d), (b, b)\}$

(f) $R = \{(a, a), (b, b), (c, c), (d, d)\}$

6. Which of the following are equivalence relatations?

   (a) R is the relation on $\mathbb{R}$ defined by xRy when $\sin(x) = \sin(y)$.

   (b) R is the relation on $\mathbb{R}$ defined by xRy if $x^2 = y^2$.

   (c) R is the relation on $\mathbb{N}$ defined by xRy if x and y are either both even or both odd.

   (d) $R = \mathbb{N} \times \mathbb{N}$ as a relation on $\mathbb{N}$.

   (e) $R = \{(a, a) : a \in \mathbb{N}\}$ as a relation on $\mathbb{N}$.

7. Find all equivalence relations on the set $\{1, 2\}$.

8. Find all equivalence relations on the set $\{1, 2, 3\}$.

9. Find all partitions on the set $\{1, 2\}$.

10. Find all partitions on the set $\{1, 2, 3\}$.

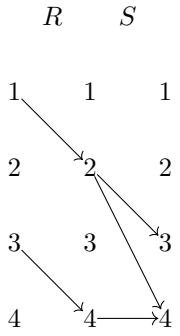
13 Operations on Relations

Definition 13.1. Suppose R and S are relations on a set A. The *composition* of R and S is the relation

$$R \circ S = \{(a, c) : (\exists b \in A)[(aRb) \wedge (bSc)]\}$$

Example 13.2. Let $R = \{(1, 2), (3, 4)\}$ and $S = \{(2, 3), (2, 4), (4, 4)\}$ be binary relations on $\{1, 2, 3, 4\}$. Find $R \circ S$.

Solution: We like to draw pictures of the relations to aid us in computing compositions.



In the picture, there are three copies of the underlying set $\{1, 2, 3, 4\}$. Between the first two are arrows depicting the relationships in R. Betwee the second two are arrows depicting the relationships in S. To compose, we follow arrows. Since we can get from 1 on the left to 3 and to 4 on the right by following arrows, $(1, 3)$ and $(1, 4)$ are in the composition. Since we can get from 3 to 4 by following arrows, $(3, 4)$ is also in the composition. Therefore, $R \circ S = \{(1, 3), (1, 4), (3, 4)\}$.

Example 13.3. Let P be the set of all people. Let R be the relation on P defined so that xRy means that "x is a parent of y." What is the interpretation of $R \circ R$?

Solution: If $(x, z) \in R \circ R$ then there is a y so that $xRyRz$. Since, xRy, then x is a parent of y. Since yRz, then y is a parent of z. Since x is a parent of a parent of z, x is a grandparent of z. $R \circ R$ is the grandparent relation.

Definition 13.4. The **converse** of a relation R on a set A is the set $R^{\cup} = \{(a, b) : bRa\}$. To calculate the converse of a relation, you simply reverse every ordered pair in the relation.

Example 13.5. Let $R = \{(1, 1), (1, 2), (2, 2), (2, 3), (3, 4)\}$. Find the converse of R.

Solution: We simply reverse every ordered pair, so $R^{\cup} = \{(1, 1), (2, 1), (2, 2), (3, 2), (4, 3)\}$.

Example 13.6. We can apply composition and converse to relations between sets too. Let S be the set of students at a college, and let C be the set of classes at the college. Let T be the relation from S to C so that xTy means that "student x is taking class y." What does the relation $T^{\cup}$ mean?

Solution: If $yT^{\cup}x$, then y is a class that x is taking.

Example 13.7. With S, C and T as in the last problem, what does the relation $T \circ T^{\cup}$ mean?

Solution: If $xT \circ T^{\cup}z$, then there is a y so that $xTyT^{\cup}z$. This means that x and z are both students, that y is a class, and that x and z are both taking the class y. Therefore, $T \circ T^{\cup}$ is the classmate relation.

Note. Some folks call $R^{\cup}$ the inverse of R and denote it as R^{-1}. This has to do with the concept of the inverse of a function which we will talk about later. However, we choose to use the more common word converse to mimic the converse of an implication.

Composition, Converse, Symmetry, and Transitivity. The operations of composition and converse are related to the properties of symmetry and transitivity. A binary relation R is symmetric if $R^{\cup} = R$. R is transitive if $R \circ R \subseteq R$.

An Aside on Matrices. Here is a very quick review of some basic matrix facts. We will generalize the notion of matrix arithmetic to perform operations with relations. A matrix is a rectangular array of numbers such as this:

$$\begin{pmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \end{pmatrix}$$

The (vertical) columns and (horizontal) rows of a matrix are numbered from the upper-left corner beginning at 1.

$$\begin{array}{c} \\ row\ 1 \\ row\ 2 \\ row\ 3 \end{array} \begin{array}{c} \begin{array}{cccc} col\ 1 & col\ 2 & col\ 3 & col\ 4 \end{array} \\ \begin{pmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \end{pmatrix} \end{array}$$

The dimensions of a matrix tell the number of rows and columns with a $\times$ between them. An $m \times n$ matrix has m rows and n columns, so the matrix above is a 3×4 matrix. The i, j-entry of a matrix is the number in row i and column j. For example, the $2, 3$-entry of the matrix above is 7. If a matrix is named M, then the notation $M_{i,j}$ is the i, j-entry of M.

The transpose of a matrix M is the matrix M^T obtained by exchanging the rows and columns of M:

$$\begin{pmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \end{pmatrix}^T = \begin{pmatrix} 1 & 5 & 9 \\ 2 & 6 & 10 \\ 3 & 7 & 11 \\ 4 & 8 & 12 \end{pmatrix}$$

Note, the i, j entry of M^T is $M^T_{i,j} = M_{j,i}$.

If two matrices have the same dimensions, then they can be added. The i, j-entry of the sum of the matrices is the sum of the i, j-entries from the two matrices. For example:

$$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix} + \begin{pmatrix} 7 & 8 & 9 \\ 10 & 11 & 12 \end{pmatrix} = \begin{pmatrix} 1+7 & 2+8 & 3+9 \\ 4+10 & 5+11 & 6+12 \end{pmatrix} = \begin{pmatrix} 8 & 10 & 12 \\ 14 & 16 & 18 \end{pmatrix}$$

If A is an $m \times n$ matrix and if B is an $n \times p$ matrix, then A and B can be multiplied. The product AB will be an $m \times p$ matrix whose i, j-entry $(AB)_{i,j}$ is computed by multiplying the entries in the i^{th} row of A one at a time times the entries in the j^{th} column of B and adding the results. For example:

$$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix} \begin{pmatrix} 7 & 10 \\ 8 & 11 \\ 9 & 12 \end{pmatrix} = \begin{pmatrix} 1 \cdot 7 + 2 \cdot 8 + 3 \cdot 9 & 1 \cdot 10 + 2 \cdot 11 + 3 \cdot 12 \\ 4 \cdot 7 + 5 \cdot 8 + 6 \cdot 9 & 4 \cdot 10 + 5 \cdot 11 + 6 \cdot 12 \end{pmatrix} = \begin{pmatrix} 50 & 68 \\ 122 & 167 \end{pmatrix}$$

Matrices and Relations. Suppose that R is a binary relation on the set $\{1, 2, \ldots, n\}$. We can represent R as an $n \times n$ matrix M^R in which all entries are 0 or 1. The i, j-entry of M^R, denoted as $M^R_{i,j}$, is 1 if iRj and is 0 otherwise. This is equivalent to saying that the i, j-entry of M^R is 1 if $(i, j) \in R$ and is 0 otherwise.

Example 13.8. Find the matrices for the relations $R = \{(1, 2), (3, 4)\}$ and $S = \{(2, 3), (2, 4), (4, 4)\}$ on the set $\{1, 2, 3, 4\}$.

Solution: Each of these should be a 4×4 matrix. M^R should have 1s in the $1,2$-entry and the $3,4$ entry:

$$M^R = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

M^S should have 1s in the $2,3$-entry, the $2,4$-entry, and the $4,4$-entry:

$$M^S = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Example 13.9. Consider this matrix:

$$\begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \end{pmatrix}$$

Treat this matrix as the matrix M^R for a binary relation R on the set $\{1,2,3\}$. Find R.

Solution: The locations of the 1s in the matrix tell us which elements are related.

$$R = \{(1,1),(1,3),(2,2),(3,1),(3,3)\}.$$

Motivation. We care about using matrices for relations for two reasons. First, it is a simple way to represent a relation in a computer. We could store a relation R in a computer as a list of ordered pairs, but then, to see if two elements i and j are related by R, we would have to search through the list for the pair (i,j). On the other hand, if we store the relation as a matrix (or an array), all we have to check is whether or not the array has a 1 in the i,j-entry. The other reason for using matrices is that we will see below that relation operations are easy to compute with matrices.

Relation Matrices and Logical Operators. If R and S are binary relations on the set $\{1,2,\ldots,n\}$, then we can apply the logical operations coordinatewise to the matrices M^R and M^S. (Here, we are treating 0 as false and 1 as true.) Then, the i,j-entry of $M^R \vee M^S$ is $M^R_{i,j} \vee M^S_{i,j}$, and the i,j-entry of $M^R \wedge M^S$ is $M^R_{i,j} \wedge M^S_{i,j}$.

Example 13.10. Calculate $M^R \wedge M^S$ and $M^R \vee M^S$ for the binary relations

$$R = \{(1,1),(2,2),(3,3),(1,2),(1,3),(2,3)\} \text{ and } S = \{(1,1),(2,1),(2,2),(3,1),(3,2),(3,3)\}$$

on $\{1,2,3\}$.

Solution: The relation matrices are

$$M^R = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix} \text{ and } M^S = \begin{pmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{pmatrix}$$

Then

$$M^R \wedge M^S = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix} \wedge \begin{pmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{pmatrix} = \begin{pmatrix} 1 \wedge 1 & 1 \wedge 0 & 1 \wedge 0 \\ 0 \wedge 1 & 1 \wedge 1 & 1 \wedge 0 \\ 0 \wedge 1 & 0 \wedge 1 & 1 \wedge 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

and

$$M^R \vee M^S = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix} \vee \begin{pmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{pmatrix} = \begin{pmatrix} 1 \vee 1 & 1 \vee 0 & 1 \vee 0 \\ 0 \vee 1 & 1 \vee 1 & 1 \vee 0 \\ 0 \vee 1 & 0 \vee 1 & 1 \vee 1 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$$

Intersections, Unions, and Coordinatewise Logical Operations. The matrix $M^R \wedge M^S$ has a 1 in the i,j-entry if and only if M^R and M^S do. This happens exactly if (i,j) is in both R and S – that is, if $(i,j) \in R \cap S$. Thus, $M^R \wedge M^S = M^{(R \cap S)}$. Similarly, the matrix $M^R \vee M^S$ has a 1 in the i,j-entry if and only if M^R or M^S does. This happens exactly if (i,j) is in R or S – that is, if $(i,j) \in R \cup S$. Thus, $M^R \vee M^S = M^{(R \cup S)}$. This gives our next theorem.

Theorem 13.11. *Suppose that R and S are binary relations on the set $\{1, 2, \ldots, n\}$. Then $M^R \wedge M^S = M^{(R \cap S)}$ and $M^R \vee M^S = M^{(R \cup S)}$.*

Converses are even easier. The converse corresponds to the transpose.

Theorem 13.12. *If R is a binary relation on the set $\{1, 2, \ldots, n\}$ then $M^{(R^\cup)} = \left(M^R\right)^T$.*

Matrix Multiplication with Logical Operations. If A and B are $n \times n$ matrices of 0s and 1s, then we can "multiply" the matrices but replace every addition with an o$\vee$ and every multiplication with an $\wedge$. When we do so, we call the resulting matrice $A \circ B$.

Example 13.13. Find $M^R \circ M^S$ for the relations from Example 13.8.

Solution: We mimic the process of matrix multiplication using $\vee$ for $+$ and $\wedge$ for $\times$:

$$M^R \circ M^S = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix} \circ \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$= \begin{pmatrix} (0 \wedge 0) \vee (1 \wedge 0) \vee (0 \wedge 0) \vee (0 \wedge 0) & (0 \wedge 0) \vee (1 \wedge 0) \vee (0 \wedge 0) \vee (0 \wedge 0) & (0 \wedge 0) \vee (1 \wedge 1) \vee (0 \wedge 0) \vee (0 \wedge 0) & (0 \wedge 0) \vee (1 \wedge 1) \vee (0 \wedge 0) \vee (0 \wedge 1) \\ (0 \wedge 0) \vee (0 \wedge 0) \vee (0 \wedge 0) \vee (0 \wedge 0) & (0 \wedge 0) \vee (0 \wedge 0) \vee (0 \wedge 0) \vee (0 \wedge 0) & (0 \wedge 0) \vee (0 \wedge 1) \vee (0 \wedge 0) \vee (0 \wedge 0) & (0 \wedge 0) \vee (0 \wedge 1) \vee (0 \wedge 0) \vee (0 \wedge 1) \\ (0 \wedge 0) \vee (0 \wedge 0) \vee (0 \wedge 0) \vee (1 \wedge 0) & (0 \wedge 0) \vee (0 \wedge 0) \vee (0 \wedge 0) \vee (1 \wedge 0) & (0 \wedge 0) \vee (0 \wedge 1) \vee (0 \wedge 0) \vee (1 \wedge 0) & (0 \wedge 0) \vee (0 \wedge 1) \vee (0 \wedge 0) \vee (1 \wedge 1) \\ (0 \wedge 0) \vee (0 \wedge 0) \vee (0 \wedge 0) \vee (0 \wedge 0) & (0 \wedge 0) \vee (0 \wedge 0) \vee (0 \wedge 0) \vee (0 \wedge 0) & (0 \wedge 0) \vee (0 \wedge 1) \vee (0 \wedge 0) \vee (0 \wedge 0) & (0 \wedge 0) \vee (0 \wedge 1) \vee (0 \wedge 0) \vee (0 \wedge 1) \end{pmatrix}$$

$$= \begin{pmatrix} 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

Matrices and Relation Composition. The relations in this last example were the same relations from Example 13.2. The final version of $M^R \circ M^S$ is the matrix for the relation $\{(1,3),(1,4),(3,4)\} = R \circ S$. This will always be the case because the i,k-entry of $M^R \circ M^S$ is 1 if and only if there is a j so that the i,j-entry of M^R is 1 and the j,k-entry of M^S is 1. This happens if and only if $iRjSk$ so $(i,k) \in R \circ S$. This implies the next theorem.

Theorem 13.14. *If R and S are binary relations on the set $\{1, 2, \ldots, n\}$, then $M^R \circ M^S = M^{(R \circ S)}$.*

Exercises 13.15. Answer the questions below.

1. Compute $R^\cup$, $R \circ S$, $R \cap S$, and $R \cup S$ in each of the following.
   (a) $R = \{(1,2),(1,3),(2,3)\}$ and $S = \{(3,4),(2,3),(3,3),(1,2)\}$
   (b) $R = \{(1,1),(2,2),(3,3)\}$ and $S = \{(3,4),(2,3),(3,3),(1,2)\}$
   (c) $R = \{(1,2),(1,1),(2,2),(3,3),(2,3)\}$ and $S = \{(1,1),(2,2),(3,3),(1,2),(2,1)\}$
2. Calculate $R \circ S$, $R \wedge S$ and $R \vee S$ for each of the following.
   (a) $R = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \end{pmatrix}$ and $S = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix}$

(b) $R = \begin{pmatrix} 1 & 0 & 1 \\ 1 & 0 & 1 \\ 1 & 0 & 1 \end{pmatrix}$ and $S = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{pmatrix}$

3. How can you tell from the matrix of a relation that the relation is reflexive?

4. How can you tell from the matrix of a relation that the relation is symmetric?

5. Let P be the set of all people, and let R be the parent relation – xRy means "x is a parent of y."

   (a) What is the interpretation of $R^{\cup}$?

   (b) What is the interpreation of $R \circ R^{\cup}$?

   (c) What is the interpretation of $R^{\cup} \circ R$?

14 Functions

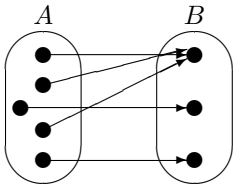
Definition 14.1. Suppose A and B are sets. A *function* or *transformation* from A to B is a subset T of $A \times B$ so that for every $a \in A$, there is precisely one $b \in B$ with $(a, b) \in T$.

Example 14.2. If $A = \{a, b, c, d\}$ and $B = \{1, 2, 3, 4, 5\}$, then $T = \{(a, 4), (b, 2), (c, 4), (d, 5)\}$ is a function from A to B.

A function from A to B is a roadmap for transforming the set A into the set B one element at a time. In our example, the transformation T says that the elements a and c of A are turned into the element 4 of B. The element b is turned into 2, and d is transformed into 5. Notice that more than one element of A can be turned into the same element of B and that not all of the elements of B are things into which elements of A are transformed.

Definition 14.3. To communicate that T is a function from A to B, we will use the notation $T : A \to B$. This is read "T is a function from A to B." In this case, the set A is called the *domain* of T. The set B is called the *codomain* of T. The set of all $b \in B$ so that there is an $a \in A$ with $(a, b) \in T$ is called the *range* or the *image* of T.

Potato Pictures. We can sometimes draw diagrams representing transformations between small sets. For example, the diagram



A transformation.

depicts a transformation from the set A with five elements on the left to the set B with three elements on the right. To determine which element of B an element of A is transformed into, you simply follow the arrows.

Some Terminology. Transformations have many names. The most common of these are transformation, *function*, *mapping*, or simply *map*. The words mapping and transformation are perhaps the most descriptive as they communicate how one set is transformed or moved into another set.

Notation 14.4. To communicate that a pair (a, b) is in a function T we will write $T(a) = b$. This notation is intended to communicate that the mapping T moves the element a to the element b. As a set a function T is precisely the set of pairs of the form $(a, T(a))$. The set of all such ordered pairs is called the *graph* of T .

Definition 14.5. Suppose that $f : A \to B$ is any function, that $C \subseteq A$ and that $D \subseteq B$. The image of C is the set of all values $f(a)$ where $a \in C$. That is, the image of C is

$$f(C) = \{f(a) : a \in C\}.$$

The *preimage* of D is the set of all a so that $f(a)$ is in D. That is, the preimage of D is

$$f^{-1}(D) = \{a \in A : f(x) \in D\}.$$

Example 14.6. Let A be the set of length four bitstrings, and let B be the set of length 3 bitstrings. Define $f : A \to B$ to be the function that "forgets" the right-most bit. That is, for any $x \in A$, we calculate $f(x)$ by removing the last bit from x. For example, $f(1011) = 101$.

1. What are the domain and codomain of f?
2. What is the image of 1010?

3. What is the image of $\{1111, 1011, 1101, 0011\}$?

4. What is the preimage of 111?

5. What is the preimage of $\{111, 000\}$?

Solution: We address the questions one at a time.

1. The domain of f is A. The codomain of f is B.

2. The image of 1010 is $f(1010) = 101$.

3. We find the image of $\{1111, 1011, 1101, 0011\}$ by applying f to each element of the set:

$$\begin{aligned} f(\{1111, 1011, 1101, 0011\}) &= \{f(1111), f(1011), f(1101), f(0011)\} \\ &= \{111, 101, 110, 001\}. \end{aligned}$$

4. The preimage of 111 is the set of all strings that begin 111. These are $\{1110, 1111\}$.

5. The preimage of $\{111, 000\}$ is the set $\{1110, 1111, 0000, 0001\}$.

Matrix Notation. A convenient notation for depicting functions out of small sets is *matrix notation for functions*. In this notation, the function is displayed as a matrix with two rows. Each element of the domain of the function is listed in the first row. The image of each element is listed directly beneath that element.

Example 14.7. The matrix for the function $f : \{1, 2, 3, 4, 5\} \to \mathbb{N}$ given by $f(x) = x^2$ is

$$\begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 1 & 4 & 9 & 16 & 25 \end{pmatrix}$$

A Function as a Rule. Rather than thinking of a function $T : A \to B$ as a roadmap for transforming A into B, we can think of it as a "rule of assignment." T provides a rule for assigning to each element a of A a unique element of B. This unique element is usually called $T(a)$. This is, in fact, a naive way of defining a function. However, it is complicated by the fact that "rule" is not well defined.

Definition 14.8. If $f : A \to B$ and $g : B \to C$ are functions, then the *composition* of f followed by g is a function $g \circ f : A \to C$ given by $g \circ f(a) = g(f(a))$ for all $a \in A$.

Example 14.9. Suppose $f : \mathbb{R} \to \mathbb{R}^+$ is the function $f(x) = x^2 + 1$ and $g : \mathbb{R}^+ \to \mathbb{R}^-$ is the function $g(x) = -\sqrt{x}$. Find $g \circ f$.

Solution:

$$\begin{aligned} g \circ f(x) &= g(f(x) \\ &= g(x^2 + 1) \\ &= -\sqrt{x^2 + 1}. \end{aligned}$$

Note. Notice that $g \circ f$ is only defined if the codomain of f is the same as the domain of g.

Example 14.10. Let A be the set of length four bitstrings, and let B be the set of length 3 bitstrings. Define $f : A \to B$ to be the function that "forgets" the right-most bit. That is, for any $x \in A$, we calculate $f(x)$ by removing the last bit from x. For example, $f(1011) = 101$. Let $g : B \to \mathbb{N}$ be defined so that $g(x)$ is the number of 1s in the string x. For example, $g(101) = 2$.

1. Find the preimage of 1 under g.

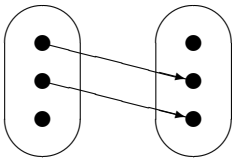
2. Find $g \circ f(1101)$.

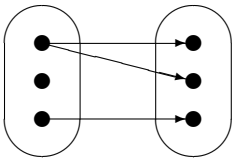
Solution: We address one question at a time.

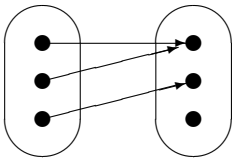
1. The preimage of 1 is the set of strings in B which contain exactly one 1. These are $\{100, 010, 001\}$.
2. $g \circ f(1101) = g(f(1101)) = g(110) = 2$.

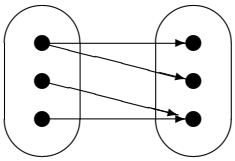
Exercises 14.11. Answer the questions below.

1. Which of the following are functions?

(a) 

(c) 

(b) 

(d) 

2. Is T a function from A to B if $A = \{a, b, c\}$, $B = \{1, 2, 3, 4\}$, and $T = \{(a, 1), (b, 1), (c, 1)\}$?
3. Is T a function from A to B if $A = \{a, b, c\}$, $B = \{1, 2, 3, 4\}$, and $T = \{(a, 2), (b, 1), (a, 3), (c, 3)\}$?
4. Is T a function from A to B if $A = \{a, b, c\}$, $B = \{1, 2, 3, 4\}$, and $T = \{(a, 2), (b, 1), (a, 2), (c, 3)\}$?
5. Is T a function from A to B if $A = \{a, b, c\}$, $B = \{1, 2, 3, 4\}$, and $T = \{(1, a), (2, b), (3, c)\}$?
6. Let $A = \emptyset$, $B = \{1, 2\}$, and $T = \emptyset$. Is T a transformation from A to B?
7. Let $B = \emptyset$, $A = \{1, 2\}$, and $T = \emptyset$. Is T a transformation from A to B?
8. Let A be the set of length four bitstrings, and let B be the set of length three bitstrings. Define $f : A \to B$ so that $f(x)$ is x with the last bit removed. Define $g : B \to A$ so that $g(x) = x1$. That is, $g(x)$ is calculated by concatenating a 1 onto the end of x.
   (a) Calculate $g \circ f(1111)$ and $g \circ f(1110)$.
   (b) Calculate $f \circ g(111)$ and $f \circ g(010)$.
   (c) Find the image of $\{101, 010\}$ under the function g.
   (d) Find the preimage of $\{1111, 1011, 0001\}$ under the function g.
   (e) Find the preimage of 0000 under the function g.
   (f) Find the preimage of 010 under the function f.

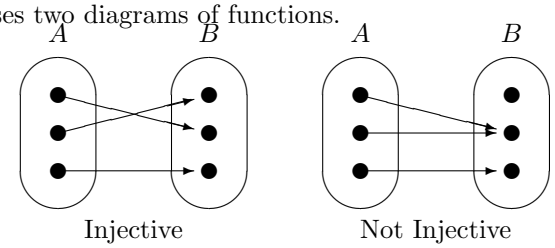
15 Properties of Functions

Definition 15.1. A function (transformation) $T : A \to B$ is *injective* (or *one-to-one*) if for all x and y in A if $T(x) = T(y)$ then $x = y$.

Example 15.2. If T is injective, then different elements of A are mapped by T to different elements of B. For example, the transformation $f : \mathbb{R} \to \mathbb{R}$ given by $f(x) = x^3$ is injective. On the graph of this function, no two x-values give the same y-value.

Example 15.3. The function $g : \mathbb{R} \to \mathbb{R}$ given by $g(x) = x^2$ is not one-to-one since $g(-1) = g(1)$ but $-1 \neq 1$.

Example 15.4. Consider theses two diagrams of functions.



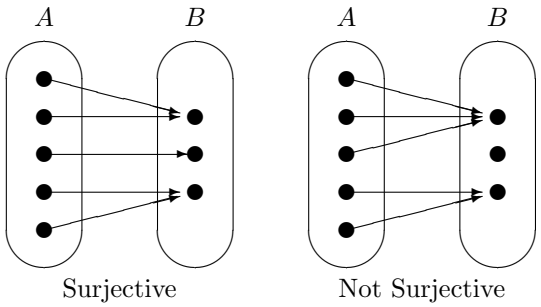
The first function is injective. You can see this because each element of B has at most one arrow pointing at it. The second function is not injective, because one element of B has more than one arrow pointing at it.

Definition 15.5. A transformation $T : A \to B$ is *surjective* (or *onto*) if for every $b \in B$ there is an $a \in A$ so that $T(a) = b$.

Example 15.6. The function $f(x) = 3x + 1$ mapping $\mathbb{R}$ to $\mathbb{R}$ is onto because if $b \in \mathbb{R}$ and $a = \dfrac{b-1}{3}$, then $f(a) = 3a + 1 = 3(\dfrac{b-1}{3}) + 1 = (b-1) + 1 = b$.

Example 15.7. The function $g : \mathbb{R} \to \mathbb{R}$ given by $g(x) = x^2 + 1$ is not onto because there is no $x \in \mathbb{R}$ with $x^2 + 1 = 0$.

Example 15.8. Consider these two diagrams of functions.



The first function is surjective because every element of B has an arrow pointing at it. The second is not surjective since there is an element of B without an arrow pointing at it.

Definition 15.9. A function which is both injective (one-to-one) and surjective (onto) is called **bijective**.

Theorem 15.10. *Suppose that $f : A \to B$ and $g : B \to C$ are functions.*

- *If f and g are both injective, then $g \circ f$ is injective.*
- *If f and g are both surjective, then $g \circ f$ is surjective.*
- *If f and g are both bijective, then $g \circ f$ is bijective.*

Example 15.11. Let A be the set of all bitstrings of length four, and let B be the set of all bitstrings of length 3. Define $f : A \to B$ so that $f(x)$ is the bitstring obtained by removing the last bit from x. Is f injective? Is f surjective?

Solution: The function f is not injective since $f(1111) = f(1110)$. The function is surjective. If y is any bitstring in B, then $f(y1) = y$. Here, $y1$ is the concatenation of y and 1. For example, if $y = 101$, then $y1 = 1011$.

Example 15.12. Let A be the set of all bitstrings of length four, and let B be the set of all bitstrings of length 3. Define $g : B \to A$ so that $g(x)$ is the bitstring obtained by cocatenatig x with 1. For example, if $x = 101$, then $g(x) = 1011$. Is g injective? Is g surjective?

Solution: The function g is injective. If $x, y \in B$, and if $g(x) = g(y)$, then $x1 = y1$. Removing the $1s$ gives $x = y$, so if $g(x) = g(y)$, then $x = y$. The function g is not surjective. There is no x with $g(x) = 0000$.

Definition 15.13. Let A be any set. The *identity function* on A is the function $1_A : A \to A$ given by $1_A(x) = x$.

Definition 15.14. Suppose $f : A \to B$ is a function. A function $g : B \to A$ is an *inverse* of f if $g \circ f = 1_A$ and $f \circ g = 1_B$. This is equivalent to saying that for all $a \in A$ $g(f(a)) = a$ and for all $b \in B$ $f(g(b)) = b$. (You can imagine that the functions g and f "unwrap" each other.) If f has an inverse, it has only one, and we denote it f^{-1}.

Example 15.15. Show that the function $g : \mathbb{R} \to \mathbb{R}$ given by $g(x) = \dfrac{x-1}{2}$ is the inverse of the function $f : \mathbb{R} \to \mathbb{R}$ given by $f(x) = 2x + 1$.

Solution: Let $a \in \mathbb{R}$. We calculate $g(f(a))$.

$$g(f(a)) = g(2a+1) = \frac{(2a+1)-1}{2} = \frac{2a}{2} = a.$$

Next, let $b \in \mathbb{R}$. We calculate $f(g(b))$.

$$f(g(b)) = f(\frac{b-1}{2}) = 2\frac{b-1}{2} + 1 = (b-1) + 1 = b.$$

Since $g(f(a)) = a$ and $f(g(b)) = b$ for all a and b in $\mathbb{R}$, g is the inverse of f.

Example 15.16. Are the functions f and g from Examples 15.11 and 15.12 inverses?

Solution: If $x \in B$, then $f(g(x)) = f(x1) = x$, so things look hopeful. However, $g(f(0000)) = g(000) = 0001$, so g and f are not inverses.

Finding Inverses. To find the inverse of a function $f : A \to B$, you can attempt to solve the equation $b = f(a)$ for a. This will yield an equation $a = g(b)$. If $g(b) \in A$ for all b, and if $g(f(a)) = a$ for all $a \in A$, then g is the inverse of f.

Example 15.17. Let $f(x) = 3x - 6$. Consider f first as a function from $\mathbb{R}$ to $\mathbb{R}$. Find an inverse function for f.

Solution: First, we set up the equation $b = f(a)$ and solve. The equation is $b = 3a - 6$. When we solve, we get $a = \dfrac{1}{3}b + 2$. Thus it appears our inverse should be $g(b) = \dfrac{1}{3}b + 2$. We must check two things. First, note that for any real number b, $g(b) \in \mathbb{R}$. Second, we must see if $g(f(a)) = a$ for all real numbers a. We check:

$$g(f(a)) = g(3a-6) = \frac{1}{3}(3a-6) + 2 = a - 2 + 2 = a$$

Hence, we have found the inverse of f.

Example 15.18. Let $f(x) = 3x - 6$. Consider f as a function from $\mathbb{Z}$ to $\mathbb{R}$. Try to find an inverse for f and explain what goes wrong.

Solution: If we set up the equation above, we still get the same g, and we still get that $g(f(a)) = a$ for all $a \in \mathbb{Z}$. However, notice that $g(1) = 7/3$, so g is not even a function from $\mathbb{R}$ to $\mathbb{Z}$, so it cannot be an inverse for f if we consider f to be a function from $\mathbb{Z}$ to $\mathbb{R}$. In this form, f has no inverse.

Theorem 15.19. *A function $f : A \to B$ has an inverse if and only if f is bijective.*

Definition 15.20. A **permutation** of a set A is a bijection from A to A. We will denote the set of all permutations on a set A by S_A.

Example 15.21. Find all permutations on the set $\{a, b, c\}$.

Solution: We write our solutions using matrix notation. The element a must map to a, b, or c, so we have three "types" of permutations:

$$\begin{pmatrix} a & b & c \\ a & & \end{pmatrix} \text{ and } \begin{pmatrix} a & b & c \\ b & & \end{pmatrix} \text{ and } \begin{pmatrix} a & b & c \\ c & & \end{pmatrix}$$

In the first case, since a is already mapped to, then b must map to b or to c to maintain injectivity. This gives two options:

$$\begin{pmatrix} a & b & c \\ a & b & c \end{pmatrix} \text{ and } \begin{pmatrix} a & b & c \\ a & c & b \end{pmatrix}$$

In the second case, since b is already mapped to, then b must map to a or to c to maintain injectivity. This gives two options:

$$\begin{pmatrix} a & b & c \\ b & a & c \end{pmatrix} \text{ and } \begin{pmatrix} a & b & c \\ b & c & a \end{pmatrix}$$

In the third case, since c is already mapped to, then b must map to a or to b to maintain injectivity. This gives two options:

$$\begin{pmatrix} a & b & c \\ c & a & b \end{pmatrix} \text{ and } \begin{pmatrix} a & b & c \\ c & b & a \end{pmatrix}$$

Cryptography. The notion of invertible function is essential to cryptography. Let P be the set of all messages which we might want to encode (P is for *plaintext*), and let C be a set which contains all encrypted messages (C is for *ciphertext*). Cryptography works with two functions, an encryption function $E : P \to C$ and a decryption function $D : C \to P$. To encrypt a piece of plaintext x, we calculate $E(x)$. To decrypt a piece of ciphertext y, we calculate $D(y)$. The two functions should satisfy the condition that for all plaintext messages $x \in P$, if we encrypt x and then decrypt x, we should arrive back at x. In symbols, this is $D(E(x)) = x$ or $D \circ E(x) = x$ or $D \circ E = 1_P$. This does not quite require that E and D are inverses. For the functions to be true inverses, we would also need $E \circ D = 1_C$. Ideally, the function E is easy to calculate, while the function D is difficult to calculate. In practice, the set P is not truly the set of all messages but a set of *blocks* from which messages could be built. For example, for the traditional affine cipher mentioned earlier, P would be the set of letters. For more general ciphers, P would be the set of bitstrings of a particular length. For example, you might convert a message to binary and then divide the binary message up into blocks of 256 bits.

Keys. The encryption and decryption functions in any cryptographic system actually take an additional argument called a key, so E and D are actually functions of more than one variable. For example, if E and D both require one key variable k, then to encrypt a piece of plaintext x, we calculate $E(x, k)$, and to decrypt a piece of ciphertext y, we calculate $D(y, k)$. What is required with this notation is that $D(E(x, k), k) = x$ for every $x \in P$ and for every key k. Ideally, there are so many keys k that one could not decrypt y by simply trying every key until one works.

Hash Functions. Suppose that we want to store records for employees or customers or citizens efficiently in a way that they can be retrieved quickly. Assume that for each record there is a key or identification number (such as a social security number). There may be so many potential identification numbers that allotting a memory location for each number is not feasible from a memory standpoint. Storing all of the data in a list could be just as impractical because searching the list for a single record may take too much time. Instead,

we can use a *hash function* to map each identification number to a code which is used to locate a place for the record in memory. An extremely simple example of this would be to mod the identification number by a modulus such as 1000 to obtain a code. The code is then linked to a memory location. It could be that many identification numbers have the same hash code. This is called collision. One way to account for collision is to have each memory location corresponding to a hash value contain a pointer to a list of records which have that same hash code. If identification codes are uniformly distributed, then a solution such as this which mods by 1000 would create a collection of lists which each would be about 1/1000 of the length of the entire list of identification numbers. This speeds up searches by a factor of 1000. Instead of having each memory location corresponding to a hash value contain a pointer to a list of records that have the same hash code, a hash function might mod by a large number (1000 would be too small for this approach) and then begin at that location in memory searching for the first open slot in memory to store the record.

Exercises 15.22. Answer the questions below.

1. Let $A = \{1, 2, 3\}$, $B = \{a, b, c\}$, $C = \{x, y, z, w\}$, and $D = \{u, v\}$.
   (a) Use matrix notation to give examples of injective functions from A to B and A to C.
   (b) Explain why there can be no injective function from A to D.
   (c) Use matrix notation to give examples of surjective functions from C to B and C to A.
   (d) Explain why there can be no surjective function from D to A.
   (e) Use matrix notation to give an injective function from A to A. Is this function surjective?
   (f) Use matrix notation to give an surjective function from A to A. Is this function injective?
2. Find all permutations on the set $\{1, 2\}$. Write your solutions in matrix form.
3. Find all permutations on the set $\{1, 2, 3, 4\}$. Write your solutions in matrix form.
4. Find an injection from $\mathbb{N}$ to $\mathbb{N}$ which is not surjective. Use a picture if necessary.
5. Find a surjection from $\mathbb{N}$ to $\mathbb{N}$ which is not injective. Use a picture if necessary.
6. Is there a bijection from $\mathbb{N}$ to $\mathbb{Z}$? Explain.
7. Let B be the set of all bitstrings. Define $f : B \to \mathbb{N}$ so that $f(x)$ is the length of the bitstring x. Is f injective? Is f surjective?
8. Draw potato pictures of functions f and g which can be composed in the order $g \circ f$ so that:
   (a) $g \circ f$ is injective but g is not.
   (b) $g \circ f$ is surjective but f is not.

16 Sequences

Definition 16.1. A *sequence* is a function whose domain is a set of integers of the form $\{n \in \mathbb{Z} : n \geq k\}$ for some fixed k. Usually $k = 0$ (in which case the domain is $\{0, 1, 2, \ldots\}$) or $k = 1$ (in which case the domain is $\{1, 2, 3, 4, \ldots\}$). We call the values of the sequence *terms*. The value of the sequence on an input n is called the n^{th} term.

Sequence Notation. We can think of a sequence f as an infinitely long ordered list $(f(0), f(1), f(2), f(3), \ldots)$. Usually, rather than traditional function notation, we use subscripts when referencing the terms of a sequence f. That is, rather than $(f(0), f(1), f(2), f(3), \ldots)$ we would write $(f_0, f_1, f_2, f_3, \ldots)$.

Sequences and Strings. Some books will call strings and elements of Cartesian powers sequences. It is correct to think of these objects as functions. For example, an element of $\mathbb{R}^3$ may look like (x_1, x_2, x_3) and may be though of as a function from $\{1, 2, 3\}$ to $\mathbb{R}$. However we will reserve the term sequence for "infinite ordered lists."

Example 16.2. List the first five terms of the sequence s given by $s_n = (-1)^n \dfrac{n}{n+1}$ for $n \geq 0$.

Solution: We just plug in $n = 0, 1, 2, 3, 4$.

$$\begin{aligned} s_0 &= (-1)^0 \frac{0}{0+1} = 0 \\ s_1 &= (-1)^1 \frac{1}{1+1} = -\frac{1}{2} \\ s_2 &= (-1)^2 \frac{2}{2+1} = \frac{2}{3} \\ s_3 &= (-1)^3 \frac{3}{3+1} = -\frac{3}{4} \\ s_4 &= (-1)^4 \frac{4}{4+1} = \frac{4}{5} \end{aligned}$$

Alternating Sequences. Notice how the signs in this sequence alternate between positive and negative. We call this an *alternating sequence*. The simplest way to make a sequence alternate is to include a factor of $(-1)^n$.

Multiple Formulas. We could also have defined the sequence in the last example using $s_n = (-1)^{n-1} \dfrac{n-1}{n}$ for $n \geq 0$. Most sequences can be described in more than way.

Example 16.3. Find a formula for the n^{th} term of this sequence: $\left(2, \dfrac{3}{4}, \dfrac{4}{9}, \dfrac{5}{16}, \dfrac{6}{25}, \ldots\right)$

Solution: Name the function a_n. We notice that the tops of the fractions are increasing $2, 3, 4, 5, 6, \ldots$. So we could have $a_n = \dfrac{n}{?}$ for $n \geq 2$. Now the bottoms of the fractions are perfect squares (treating the first term as $\dfrac{2}{1}$). However, it is not the tops that are being squared, but one less than the top. So we could use $a_n = \dfrac{n}{(n-1)^2}$ for $n \geq 2$. An alternative answer would be $a_n = \dfrac{n+1}{n^2}$ for $n \geq 1$.

Recursively Defined Sequences. Sometimes, rather than a formula for each term of a sequence, we are given a way to calculate one term based on the previous term. For example, suppose that s is a sequence and we know that $s_0 = 1$ and that whenever we know s_n, then $s_{n+1} = 2 \cdot s_n + 1$. Then

$$s_0 = 1$$

$$s_1 = 2 \cdot s_0 + 1 = 3$$
$$s_2 = 2 \cdot s_1 + 1 = 7$$
$$s_3 = 2 \cdot s_2 + 1 = 15$$
$$s_4 = 2 \cdot s_3 + 1 = 31$$

and so on. Such a definition of a sequence is called a *recursive definition.* The equation $s_{n+1} = 2 \cdot s_n + 1$ is called a *recursive relation* or *recurrence relation.* Rather than defining s_{n+1} in terms of s_n, we could also define s_n in terms of s_{n-1}. Then this sequence would be defined as: $s_0 = 1$ and $s_n = 2 \cdot s_{n-1} + 2$ for $n > 0$.

Example 16.4. Find the first five terms of the sequence defined by

$$s_0 = 5 \text{ and } s_{n+1} = \frac{1 + s_n}{2} \text{ for } n \geq 0.$$

Solution: We plug in repeatedly:

$$s_0 = 5$$
$$s_1 = \frac{1 + s_0}{2} = 3$$
$$s_2 = \frac{1 + s_1}{2} = 2$$
$$s_3 = \frac{1 + s_2}{2} = \frac{3}{2}$$
$$s_4 = \frac{1 + s_3}{2} = \frac{5}{4}$$

Example 16.5. Recursive definitions may refer to more than one previous term. Consider the sequence F_n defined by

$$F_0 = 0 \text{ and } F_1 = 1 \text{ and } F_n = F_{n-1} + F_{n-2} \text{ for } n \geq 2.$$

List the first six terms of F_n.

Solution: Again, we just plug in.

$$F_0 = 0$$
$$F_1 = 1$$
$$F_2 = F_1 + F_0 = 1$$
$$F_3 = F_2 + F_1 = 2$$
$$F_4 = F_3 + F_2 = 3$$
$$F_5 = F_4 + F_3 = 5$$

This sequence is known as the Fibonacci sequence.

Definition 16.6. An *arithmetic* sequence is a sequence a_n whose n^{th} term is given by $a_n = m \cdot n + b$ for $n \geq 0$. Such a sequence has the recursive definition

$$a_0 = b \text{ and } a_{n+1} = a_n + m \text{ for } n \geq 0.$$

Here the number m is called the *common difference.*

Example 16.7. List the first five terms of the arithmetic sequence $a_n = 4n - 7$ for $n \geq 0$.

Solution: We plug in $n = 0, 1, 2, 3, 4$.

$$s_0 = 4 \cdot 0 - 7 = -7$$
$$s_1 = 4 \cdot 1 - 7 = -3$$
$$s_2 = 4 \cdot 2 - 7 = 1$$
$$s_3 = 4 \cdot 3 - 7 = 5$$
$$s_4 = 4 \cdot 4 - 7 = 9.$$

Notice how each term is 4 more than the previous term.

Example 16.8. Find a formula for the n^{th} term of the arithmetic sequence $(2, 5, 8, 11, 14, \ldots)$.

Solution: The formula is $s_n = mn + b$. The number m is the common difference, which is 3. The number b is the 0^{th} term, which is 2. So the formula is $s_n = 3n + 2$.

Definition 16.9. A *geometric sequence* is a sequence b_n whose n^{th} term is given by $b_n = a \cdot r^n$ for $n \geq 0$. Such a sequence has the recursive definition

$$b_0 = a \text{ and } b_{n+1} = r \cdot b_n \text{ for } n \geq 0.$$

Here the number r is called the *common ratio.*

Example 16.10. List the first five terms of the geometric sequence $b_n = 3 \cdot 2^n$.

Solution: We simply plug in $n = 0, 1, 2, 3, 4$.

$$s_0 = 3 \cdot 2^0 = 3$$
$$s_1 = 3 \cdot 2^1 = 6$$
$$s_2 = 3 \cdot 2^2 = 12$$
$$s_3 = 3 \cdot 2^3 = 24$$
$$s_4 = 3 \cdot 2^4 = 48.$$

Notice how each term is twice the previous term.

Example 16.11. Find a formula for the sequence s_n which begins $\left(8, 4, 2, 1, \frac{1}{2}, \frac{1}{4}, \ldots\right)$.

Solution: The formula is $s_n = a \cdot r^n$. The number a is the 0^{th} term, 8, and the number r is the common ratio, $\frac{1}{2}$. So the formula is $s_n = 8 \cdot \left(\frac{1}{2}\right)^n$.

Exercises 16.12. Answer the questions below.

1. List the first five terms of the sequences below.
   (a) $a_n = 1 + \dfrac{(-1)^n}{n}$ for $n \geq 1$
   (b) $b_n = \left(1 + \dfrac{1}{n}\right)^n$. Find exact answers for each term and then give a decimal approximation to the last term.
2. List the first four terms of the sequence below. Find exact answers for each term and then give a decimal approximation to the last term.
   (a) $a_0 = 1$ and $a_{n+1} = \dfrac{a_n^2 + 2}{2a_n}$ for $n \geq 0$

(b) $b_1 = 1$, $b_2 = 1$, and $b_{n+1} = b_n - \frac{1}{n} b_{n-1}$ for $n > 2$.

3. Find a formula for the n^{th} term of the sequence.

   (a) $a = (7, 10, 13, 16, 19, 22, \ldots)$

   (b) $b = (2, 6, 18, 54, 162, 486, \ldots)$

   (c) $c = (3, 8, 15, 24, 35, 48, 63, 80, 99, \ldots)$

   (d) $d = \left(\frac{1}{4}, -\frac{1}{10}, \frac{1}{28}, -\frac{1}{82}, \frac{1}{244}, \ldots\right)$

4. Find initial conditions and a recurrence relation for the sequence $(1, 4, 13, 40, 121, 364, 1093, \ldots)$.

5. Suppose that you invest \$1000 at 3% simple interest compounded anually. Write initial conditions and an recurrence relation giving the value a_n of your investment after n years.

6. In this exercise, we will find a recurrence relation for the number of ways that you can walk up a flight of n stairs assuming you can take either 1 or 2 stairs at a time. Let a_n be the number of ways you can walk up a flight of n stairs assuming you can take either 1 or 2 stairs at a time. Note that a_n should be defined for $n \geq 1$.

   (a) What is a_1? That is, how many ways can you walk up one step if you are allowed to take 1 or 2 steps at a time?

   (b) What is a_2? That is, how many ways can you walk up two steps if you are allowed to take 1 or 2 steps at a time?

   (c) Suppose now that you want to walk up n steps. You can start one of two ways, either taking one step or two. If you take one step, how many steps are left? What is the symbol for the number of ways you can take these steps?

   (d) Suppose now that you want to walk up n steps. You can start one of two ways, either taking one step or two. If you take two steps, how many steps are left? What is the symbol for the number of ways you can take these steps?

   (e) Write a recurrence relation and initial conditions for the number a_n of ways you can walk up a flight of n stairs assuming you can take either 1 or 2 stairs at a time.

7. In this exercise, we will find a recurrence relation for the number of bitstrings of length n which do not contain two consecutive 0s. Let a_n be the number of bitstrings of length n which do not contain two consecutive 0s.

   (a) What is a_1? That is, how many bitstrings of length 1 do not contain two consecutive 0s?

   (b) What is a_2? That is, how many bitstrings of length 2 do not contain two consecutive 0s?

   (c) Suppose that $n > 2$. Consider the bitstrings of length n which do not contain two consecutive 0s and which end in 1. The first $(n-1)$ bits of such a bitstring cannot contain two consecutive 0s. How many such bitstrings are there?

   (d) Suppose that $n > 2$. Consider the bitstrings of length n which do not contain two consecutive 0s and which end in 0. Since these bitstrings cannot end with two consecutive 0s, the $(n-1)^{st}$ bit in each of these must be 1. This means that the first $(n-1)$ bits form a bitstring of length $(n-1)$ that ends in 1 and that does not contain two consecutive 0s. How many such bitstrings are there?

   (e) Now write a recurrence relation and initial condistions for a_n.

8. **Towers of Hanoi** This is a puzzle involving three pegs mounted on a board and a collection of disks with holes in the center. The disks are all of different sizes, and they begin all stacked on the first peg, in order of size, with the largest disk on bottom. The objective of the puzzle is to move the disks one at a time until they are all on the second peg with the restriction that no disk can be placed on top of a smaller disk. Let H_n be the number of steps necessary to move n disks from the first peg to the second peg according to this restriction.

(a) What is H_1? That is, how many steps are necessary to move 1 disk from the first peg to the second?

(b) What is H_2? That is, how many steps are necessary to move 2 disks from the first peg to the second?

(c) What is H_3? That is, how many steps are necessary to move 3 disks from the first peg to the second?

(d) Find a recurrence relation for H_n. Hint: Begin by moving all but the largest (bottom) disk to the third peg.

(e) Use your recurrence relation to list several values of H_n. Guess a formula for H_n.

(f) The original Tower of Hanoi puzzle included the myth about monks solving the problem with 64 gold disks. According to the myth, when the monks finish, the world will end. If they can transfer 1 disk per second, how long would it take to move all 64 disks?

17 Summations

Definition 17.1. The notation

$$\sum_{i=1}^{n} a_i$$

is called *summation notation*. The symbol $\sum$ is a capital Greek *sigma*, which means (for us) to sum. The decorations $i = 1$ and n are *limits of the summation*. The variable i is the *index*. The symbol a_i is a function using the subscript notation we used for sequences. The notation is short-hand for a sum

$$\sum_{i=1}^{n} a_i = a_1 + a_2 + \cdots + a_n.$$

The summation is calculated by plugging in $1, 2, 3, \ldots, n$ for the index i in the function a_i and then adding the results. The limits of the summation tell where to begin and end the values plugged into the index.

A summation as a for-loop. You could think of the summation $\sum_{i=1}^{n} a_i = a_1 + a_2 + \cdots + a_n$ as calculating the value of the variable S in the following code.

$S = 0;$
for $(i = 1; i <= n; i++)$
    $S = S + a_i;$

Example 17.2. Calculate the summation $\sum_{i=1}^{5} \left(\frac{i}{i+1}\right)$.

Solution: We plug in and add:

$$\sum_{i=1}^{5} \left(\frac{i}{i+1}\right) = \frac{1}{2} + \frac{2}{3} + \frac{3}{4} + \frac{4}{5} + \frac{5}{6} = \frac{71}{20}.$$

Example 17.3. Calculate the summation $\sum_{k=-1}^{2} (2k + 1)$.

Solution: Again, we just plug in:

$$\sum_{k=-1}^{2} (2k+1) = (2 \cdot (-1) + 1) + (2 \cdot 0 + 1) + (2 \cdot 1 + 1) + (2 \cdot 2 + 1) = 8.$$

Summation Equations. There are a few equations that summations satisfy that can help in computations. These follow from the commutative, associative, and distributive properties of addition.

$$\sum_{i=1}^{n} c \cdot a_n = c \cdot a_1 + c \cdot a_2 + \cdots c \cdot a_n = c \cdot (a_1 + a_2 + \cdots a_n) = c \sum_{i=1}^{n} a_n$$

$$\sum_{i=1}^{n} (a_n + b_n) = (a_1 + b_1) + (a_2 + b_2) + \cdots (a_n + b_n) = (a_1 + a_2 + \cdots a_n) + (b_1 + b_2 + \cdots b_n) = \left(\sum_{i=1}^{n} a_n\right) + \left(\sum_{i=1}^{n} b_n\right)$$

$$\sum_{i=1}^{n} (a_n - b_n) = (a_1 - b_1) + (a_2 - b_2) + \cdots (a_n - b_n) = (a_1 + a_2 + \cdots a_n) - (b_1 + b_2 + \cdots b_n) = \left(\sum_{i=1}^{n} a_n\right) - \left(\sum_{i=1}^{n} b_n\right)$$

These are called the *linear* properties of summations. In addition, there are some standard formulas for adding up simple functions of the index:

$$\sum_{i=1}^{n} c = c + c + \cdots + c = n \cdot c$$

$$\sum_{i=1}^{n} i = 1 + 2 + 3 + \cdots + n = \frac{n(n+1)}{2}$$

$$\sum_{i=1}^{n} i^2 = 1^2 + 2^2 + 3^2 + \cdots + n^2 = \frac{n(n+1)(2n+1)}{6}$$

$$\sum_{i=1}^{n} i^3 = 1^3 + 2^3 + 3^3 + \cdots + n^3 = \left(\frac{n(n+1)}{2}\right)^2$$

$$\sum_{i=0}^{n} r^i = 1 + r + r^2 + r^3 + \cdots + r^n = \frac{r^{n+1} - 1}{r - 1}$$

Example 17.4. Calculate $\sum_{i=1}^{100}(i^2 + 2i + 3)$.

Solution: We could plug in $i = 1, 2, 3, \ldots, 100$ and add. That would be tedious. Instead, we apply the equations above.

$$\begin{aligned}
\sum_{i=1}^{100}(i^2 + 2i + 3) &= \left(\sum_{i=1}^{100} i^2\right) + \left(\sum_{i=1}^{100} 2i\right) + \left(\sum_{i=1}^{100} 3\right) \\
&= \left(\sum_{i=1}^{100} i^2\right) + \left(2\sum_{i=1}^{100} i\right) + \left(\sum_{i=1}^{100} 3\right) \\
&= \frac{100 \cdot 101 \cdot 201}{6} + 2 \cdot \frac{100 \cdot 101}{2} + 100 \cdot 3 \\
&= 338350 + 10100 + 300 \\
&= 348480.
\end{aligned}$$

Example 17.5. How many multiplications are performed in this piece of code?

```
c = 0
for (i = 1; i ≤ 100; i++) {
    c = c · c;
    for (j = 1; j ≤ i; j++) {
        c = c · c + 2 · c + 3;
    }
}
```

Solution: We begin with the inside loop. For a fixed value of i, this loop is executed i times. For each execution of the loop, there are two multiplications, so for each i, there are $2i$ multiplications done in the inside loop. Now we address the outside loop. Each time this loop is executed, there are $2i$ multiplications in the inside loop and one multiplication outside the inside loop. Therefore, there are a total of

$$\sum_{i=1}^{100}(2i + 1)$$

multiplications here. We can use the formulas above to calculate this value:

$$\begin{aligned}\sum_{i=1}^{100}(2i+1) &= \left(\sum_{i=1}^{100}2i\right)+\left(\sum_{i=1}^{100}1\right)\\ &= \left(2\cdot\sum_{i=1}^{100}i\right)+\left(\sum_{i=1}^{100}1\right)\\ &= 2\cdot\frac{100\cdot 101}{2}+100\cdot 1\\ &= 10200.\end{aligned}$$

When the code is executed, there are 10200 multiplications.

Here is another solution: We can turn the for-loops into summations. The outside for-loop becomes

$$\sum_{i=1}^{100}\cdots$$

Inside this for-loop, there is one multiplication plus the inside loop. The inside loop is also a summation:

$$\sum_{i=1}^{100}\left(1+\sum_{j=1}^{i}\cdots\right)$$

Within the inside loop, there are two multiplications (well, one could be done as a shift). The number of multiplications is

$$\sum_{i=1}^{100}\left(1+\sum_{j=1}^{i}2\right)$$

Now we can use the formulas:

$$\begin{aligned}\sum_{i=1}^{100}\left(1+\sum_{j=1}^{i}2\right) &= \sum_{i=1}^{100}(1+2i)\\ &= \left(\sum_{i=1}^{100}1\right)+\left(\sum_{i=1}^{100}2i\right)\\ &= \left(\sum_{i=1}^{100}1\right)+\left(2\sum_{i=1}^{100}i\right)\\ &= 100+2\cdot\frac{100\cdot 101}{2}\\ &= 10200.\end{aligned}$$

When the code is executed, there are still 10200 multiplications.

Example 17.6. Calculate this sum:

$$\sum_{i=3}^{100}(i+1)$$

Solution: The tricky part here is that the summation starts at 3 rather than 1. If the summation started at 1, then we would have

$$\sum_{i=1}^{100}(i+1)=\left(\sum_{i=1}^{100}i\right)+\left(\sum_{i=1}^{100}1\right)=\frac{100\cdot 101}{2}+100\cdot 1=5150.$$

However, this includes terms where $i = 1$ and $i = 2$. It includes:

$$\sum_{i=1}^{2}(i+1) = (1+1) + (1+2) = 2 + 3 = 5.$$

To find the sum we want, we just subtract:

$$\sum_{i=3}^{100}(i+1) = \sum_{i=1}^{100}(i+1) - \sum_{i=1}^{2}(i+1) = 5150 - 5 = 5145.$$

Exercises 17.7. Answer the questions below.

1. Calculate each of these summations by plugging in.

   (a) $\displaystyle\sum_{i=-2}^{2} \frac{i}{i^2+1}$

   (b) $\displaystyle\sum_{i=0}^{4} \left(\frac{1}{2}\right)^i$

   (c) $\displaystyle\sum_{i=0}^{4}(2i^3 - 3i^2)$

2. Calculate each of these summations by using the summation formulas in the section.

   (a) $\displaystyle\sum_{i=1}^{50}(3i+4)$

   (b) $\displaystyle\sum_{i=1}^{50}(i^3 - i^2)$

   (c) $\displaystyle\sum_{i=0}^{10}(2i + 2^i)$ (Does the initial value $i = 0$ instead of $i = 1$ matter?)

   (d) $\displaystyle\sum_{i=1}^{10}(2i + 2^i)$ (Does the initial value $i = 1$ instead of $i = 0$ matter?)

   (e) $\displaystyle\sum_{i=0}^{10}\left(\left(\frac{1}{2}\right)^i + \frac{1}{2}\right)$

   (f) $\displaystyle\sum_{i=5}^{100} i^2$

3. The code below finds the maximum number in the list $a_1, a_2, \ldots, a_n$. How many times is the comparison in line 3 executed?

   $max = a_1$;
   for $(i = 2; i \leq n; i++)$
   if $(a_i > max)$
   $max = a_i$;

4. The code below evaluates the polynomial $a_0 + a_1x + a_2x^2 + \cdots a_nx^n$. How many multiplications are done in this code?

```
//This variable is an accumulator that will contain the value of the polynomial.
v = 0;
//The variable i will be the exponent.
for (i = 0; i ≤ n; i++) {
    //The variable t will contain the next term to add to the polynomial.
    //We initialize t to the coefficient a_i.
    t = a_i;
    //Then we multiply by i copies of x to get the right exponent.
    for (j = 0; j < i; j++)
        t = t · x;
    //Add the new term onto v.
    v = v + t;
}
```

5. The code below also evaluates the polynomial $a_0 + a_1x + a_2x^2 + \cdots a_nx^n$, but it uses something called Horner's Algorithm. How many multiplications are done in this code?

```
//This variable is an accumulator that will contain the value of the polynomial.
v = a_n;
for (i = 1; i ≤ n; i++)
    v = v · x + a_{n-i};
```

6. The code below multiplies an $n \times n$ matrix A times an $n \times n$ matrix B and places the result in an $n \times n$ matrix C. The notation $A_{i,j}$ is the entry of A in the i^{th} row and j^{th} column. How many multiplications are executed in the code?

```
//The variable i will traverse the rows of A.
for (i = 1; i ≤ n; i++) {
    //The variable k will traverse the columns of B.
    for (k = 1; k ≤ n; k++) {
        //Initialize C_{i,k} to 0.
        C_{i,k} = 0;
        //The variable j will be used to traverse the i^th row of A and the k^th column of B.
        for(j = 1; j ≤ n; j++)
            C_{i,k} = C_{i,k} + A_{i,j} · B_{j,k};
    }
}
```

7. The code below performs a simple bubble sort to place the numbers $a_1, a_2, \ldots, a_n$ in order from least to greatest. How many times is the if statement in the third line executed?

```
for (i = 1; i ≤ n; i++)
    for (j = i + 1; j ≤ n; j++)
        if (a_i > a_j)
            swap(a_i, a_j);
```

8. The code below checks objects $a_1, a_2, \ldots, a_n$ for collisions and "does something" if there is a collision. How many times is the function *tooClose*() called?

```
for (i = 1; i ≤ n; i++)
    for (j = i + 1; j ≤ n; j++)
        if (tooClose(a_i, a_j))
            doSomething(a_i, a_j);
```

9. The code below performs some weird calculation. How many multiplications does it perform?

```
x = 0;
for (i = 0; i < n; i++)
    for (j = 0; j < i; j++)
        for (k = 0; k < j; k++)
            x = x + i · j + j · k;
```

10. The code below performs some weird calculation. How many multiplications does it perform?

```
x = 0;
for (i = 0; i < n; i++)
    for (j = 0; j < i; j++)
        for (k = 0; k < i; k++)
            x = x + i · j + j · k;
```

18 Cardinality

Bijections and Finite Sets. Suppose that A and B are finite sets and that $f : A \to B$ is any function. If f is injective (one-to-one) then all of the elements in B of the form $f(a)$ where $a \in A$ are distinct. This means that B must have at least as many elements as A. If f is surjective (onto), then for each $b \in B$, there is at least one $a \in A$ with $f(a) = b$. Since every element of b can be paired with at least one element of A, there are at least as many elements in A as in B. Then, if f is injective the number of elements of A is less than or equal to the number of elements of B, and if f is surjective the number of elements of A is greater than or equal to the number of elements of B. Therefore, if f is a bijection, the number of elements of A is equal to the number of elements of B.

We would like to extend this observation to infinite sets. We would like to be able to talk about the "number" of elements in a set and to compare these "numbers" even when the sets are infinite. To do so, we cannot use "numbers" in the usual sense. When we talk about the "size" of a set or the "number" of elements of a set, we will use a different number system called *cardinal numbers*. We will use the symbol $|A|$ for the number of elements in a set, and we will call this the *cardinality* of the set. The following are assumptions we make about cardinal numbers and cardinality.

Axiom 18.1. *We assume the following about cardinal numbers and cardinalities of sets.*

1. *To every set A is associated a unique cardinal number $|A|$ called the cardinality of A.*
2. *For every cardinal number κ, there is a set A with $|A| = \kappa$.*
3. *For every positive integer n, $|\{0, 1, 2, \ldots, (n-1)\}| = n$.*
4. *For any set A, $|A| = 0$ if and only if $A = \emptyset$.*
5. *For any sets A and B, $|A| = |B|$ if and only if there is a bijection from A to B.*
6. *For any sets A and B, $|A| \leq |B|$ if and only if there is an injection from A to B.*
7. *For any sets A and B, $|A| < |B|$ if and only if $|A| \leq |B|$ but $|A| \neq |B|$.*

Terminology. When there is a bijection between two sets so that the sets have the same cardinality, some texts will say that the two sets are equipotent, equinumerable, or bijective. We choose simply to say they have the same cardinality.

Since the composition of bijections is a bijection, and since every bijection has an inverse which is a bijection, most of the following theorem is not hard to prove.

Theorem 18.2. *Suppose that A, B, and C are sets.*

1. *$|A| = |A|$ (The relationship of having the same cardinality is reflexive.)*
2. *If $|A| = |B|$, then $|B| = |A|$. (The relationship of having the same cardinality is symmetric.)*
3. *If $|A| = |B|$ and $|B| = |C|$, then $|A| = |C|$. (The relationship of having the same cardinality is transitive.)*
4. *$|A| \leq |A|$ (The less than or equal to relation for cardinalities is reflexive.*
5. *If $|A| \leq |B|$ and $|B| \leq |A|$, then $|A| = |B|$. The less than or equal to relation for cardinalities is antisymmetric.)*
6. *If $|A| \leq |B|$ and $|B| \leq |C|$, then $|A| \leq |C|$. The less than or equal to relation for cardinalities is transitive.)*

Cantor-Schroeder-Bernstein Theorem. The only portion of this theorem which is difficult to prove is part 5. This is known as the Cantor-Schroeder-Bernstein Theorem. It says that we can prove two sets A and B have the same cardinality by proving there is an injection from A to B and an injection from B to A.

Example 18.3. Let n be a positive integer. Show that $|\{1, 2, \ldots, n\}| = n$.

Solution: Let $A = \{0, 1, 2, \ldots, (n-1)\}$ and $B = \{1, 2, \ldots, n\}$. From Axiom 18.1 part 3, we know that that $|A| = n$. Define $f : A \to B$ by $f(x) = x + 1$. Then f is clearly a bijection, so $|B| = |A| = n$.

Example 18.4. Let $A = \{0, 1, 2, 3, \ldots\}$ and let $B = \{1, 2, 3, \ldots\}$. Show that $|A| = |B|$.

Solution: We just have to find a bijection between A and B. The function $f : A \to B$ given by $f(x) = x + 1$ is a bijection. (We know it is a bijection because it has an inverse which consists of subtracting 1.) Since $f : A \to B$ is a bijection, $|A| = |B|$.

Example 18.5. Let $C = [0, 1]$ be the interval in $\mathbb{R}$ from 0 to 1, and let $D = [0, 17]$ be the interval in $\mathbb{R}$ from 0 to 17. Show that $|C| = |D|$.

Solution: Again, all we need is a bijection from C to D. The function $f : C \to D$ given by $f(x) = 17x$ will do.

Wait. These last two examples should make you pause to ponder. In Example 18.4, the set A seems larger because it has one element that B does not. In Example 18.5, the set D seems larger since it is longer. There are two morals here. First, weird things can happen with infinite sets. Second, cardinality has nothing to do with length or area or volume.

Definition 18.6. Suppose that A is a set.

- A is *finite* if $|A|$ is a natural number.
- A is *infinite* if A is not finite.

Theorem 18.7. *A set A is infinite if and only if there is an injection $f : A \to A$ which is not surjective.*

Example 18.8. Let S be the set of sequences of 0s and 1s. Show that there is no surjection from $\mathbb{N}$ to S.

Solution: Suppose that $f : \mathbb{N} \to S$ is any function. We will show that f is not surjective. To do so, we will find a sequence of 0s and 1s which is not in the range of f. The tricky part is the notation. For each $n \in \mathbb{N}$, $f(n)$ is a sequence. We will denote this sequence by f^n, and we will index the sequence with natural numbers. Each f^n is a sequence. We denote the i^{th} term of f^n as f^n_i. We now define a new sequence s. For each $n \in \mathbb{N}$ we let $s_n = 1 - f^n_n$. That is, if $f^n_n = 0$, then $s_n = 1 - 0 = 1$, and if $f^n_n = 1$, then $s_n = 1 - 1 = 0$. The term s_n is the opposite of the n^{th} term in f^n. The sequence s cannot be in the range of f. If s were in the range of f, then there would be some n so that $s = f(n) = f^n$. However, we know that $s \neq f^n$ because the terms s_n and f^n_n are different. Since there is a sequence $s \in S$ which is not in the range of f, the function f is not surjective. This applies to all $f : \mathbb{N} \to S$, so there is no surjection from $\mathbb{N}$ to S.

This theorem now follows.

Theorem 18.9. *The set S of sequences of 0s and 1s is not the same cardinality as the set $\mathbb{N}$ of natural numbers.*

Two Different Infinite Cardinals. In the last example, S and $\mathbb{N}$ are both infinite. However, they have different cardinalities. This gives two different infinite cardinals! This should begin to bother you.

Definition 18.10. Suppose that A is any set.

- A is *countably infinite* if $|A| = |\mathbb{N}|$.
- A is *countable* if A is finite or countably infinite.
- A is *uncountably infinite* or *uncountable* if A is not countable. (This means that A is not finite and not the same cardinality as $\mathbb{N}$.)

Example 18.11. We saw in Example 18.8 that the set S of sequences of 0s and 1s is uncountable.

Example 18.12. Let P be the powerset of $\mathbb{N}$ (the set $\mathcal{P}(\mathbb{N})$ of subsets of $\mathbb{N}$). Then P is uncountable.

Solution: There is a bijection from P to the set S of sequences of 0s and 1s that maps any subset $A \subseteq \mathbb{N}$ to the sequence s defined so that $s_n = 1$ if and only if $n \in A$. Since there is such a bijection, $|P| = |S|$. Since S is uncountable, so is P.

Note. The sequence s_n related to the set $A \subseteq \mathbb{N}$ in the last example is called the *characteristic function* of A. In general if $X \subseteq Y$ are sets, the characterisitc function of X is the function $\chi : Y \to \{0, 1\}$ so that $\chi(y) = 1$ if and only if $y \in X$.

Powersets. Example 18.12 is actually a special case of a theorem by Cantor that the powerset of a set always has a larger cardinality.

Theorem 18.13. Cantor's Theorem *If A is any set, then $|A| < |\mathcal{P}(A)|$.*

This theorem has disturbing consequences. we know $|\mathbb{N}| < |\mathcal{P}(\mathbb{N})|$, but $\mathcal{P}(\mathbb{N})$ is just a set, so we can apply Cantor's Theorem to it also to get $|\mathcal{P}(\mathbb{N})| < |\mathcal{P}(\mathcal{P}(\mathbb{N}))|$. In fact, we can apply the theorem repeatedly to find an infinite sequence of increasingly larger infinite cardinals:

$$|\mathbb{N}| < |\mathcal{P}(\mathbb{N})| < |\mathcal{P}(\mathcal{P}(\mathbb{N}))| < |\mathcal{P}(\mathcal{P}(\mathcal{P}(\mathbb{N})))| < |\mathcal{P}(\mathcal{P}(\mathcal{P}(\mathcal{P}(\mathbb{N}))))| < |\mathcal{P}(\mathcal{P}(\mathcal{P}(\mathcal{P}(\mathcal{P}(\mathbb{N})))))| \cdots$$

Are these the only infinite cardinal numbers? In particular, is there an infinite cardinal number between $|\mathbb{N}|$ and $|\mathcal{P}(\mathbb{N})|$? The thought for a long time was no:

The Continuum Hypothesis: There is no set A with $|\mathbb{N}| < |A| < |\mathcal{P}(\mathbb{N})|$.

However, it was proven in the 1960s that the Continuum Hypothesis is independent of the current axioms of mathematics – we can assume it is true or that it is false without affecting the consistency of mathematics. That should really be disturbing.

Example 18.14. Let B be the set of all real numbers in the interval $[0, 1)$ whose decimal expansions use only 0s and 1. B contains numbers like 0, 0.1 $0.\overline{1}$, and $0.\overline{01}$. The set B is uncountable.

Solution: There is a bijection from B to S from Example 18.8 which maps any number $0.b_1b_2b_3...$ to the sequence $(b_1, b_2, b_3, \ldots)$, so $|B| = |S|$. Since S is uncountable, so is B.

Example 18.15. Since the set B in the previous example is uncountable, so is any set which contains it. This means that $[0, 1]$ is uncountable and $\mathbb{R}$ is uncountable. Because of Example 18.5, every interval in $\mathbb{R}$ is uncountable.

Example 18.16. The integers have the same cardinality as the natural numbers.

Solution: We can draw a picture of a bijection between the two sets:

$$\begin{array}{cccccccccc} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & \cdots \\ \updownarrow & \updownarrow & \updownarrow & \updownarrow & \updownarrow & \updownarrow & \updownarrow & \updownarrow & \updownarrow & \cdots \\ 0 & 1 & -1 & 2 & -2 & 3 & -3 & 4 & -4 & \cdots \end{array}$$

Since such a bijection exists, $|\mathbb{Z}| = |\mathbb{N}|$. This implies, by the way, that $\mathbb{Z}$ is countably infinite.

Listable. The manner in which we listed the elements of $\mathbb{Z}$ in this example with one ellipsis is important. Essentially, if we can list the elements of a set in a *reasonable* way then that set will be countable.

Example 18.17. The set $\mathbb{Z} \times \mathbb{Z}$ is countably infinite.

Solution: The elements of $\mathbb{Z} \times \mathbb{Z}$ are ordered pairs (x, y) where both x and y are integers. These are exactly those points on the plane which have integer coordinates. What we need is a way to number these points to set up a bijection with the natural numbers. We can do this by assigning 0 to the origin $(0, 0)$ and spiraling outwards, assigning a natural number to each integer point, in a pattern like so:

$$
\begin{array}{ccccccccccccc}
36 & \leftarrow & 35 & \leftarrow & 34 & \leftarrow & 33 & \leftarrow & 32 & \leftarrow & 31 & \leftarrow & 30 \\
\downarrow & & & & & & & & & & & & \uparrow \\
37 & & 16 & \leftarrow & 15 & \leftarrow & 14 & \leftarrow & 13 & \leftarrow & 12 & & 29 \\
\downarrow & & \downarrow & & & & & & & & \uparrow & & \uparrow \\
38 & & 17 & & 4 & \leftarrow & 3 & \leftarrow & 2 & & 11 & & 28 \\
\downarrow & & \downarrow & & \downarrow & & & & \uparrow & & \uparrow & & \uparrow \\
39 & & 18 & & 5 & & 0 & \rightarrow & 1 & & 10 & & 27 \\
\downarrow & & \downarrow & & \downarrow & & & & & & \uparrow & & \uparrow \\
40 & & 19 & & 6 & \rightarrow & 7 & \rightarrow & 8 & \rightarrow & 9 & & 26 \\
\downarrow & & \downarrow & & & & & & & & & & \uparrow \\
41 & & 20 & \rightarrow & 21 & \rightarrow & 22 & \rightarrow & 23 & \rightarrow & 24 & \rightarrow & 25 \\
\downarrow & & & & & & & & & & & & \\
42 & \rightarrow & 43 & \rightarrow & 44 & \rightarrow & 45 & \rightarrow & 46 & \rightarrow & 47 & \rightarrow & 48 & \cdots
\end{array}
$$

Since such a bijection exists, we know that $|\mathbb{N}| = |\mathbb{Z} \times \mathbb{Z}|$.

Example 18.18. The set $\mathbb{Q}$ is countably infinite.

Solution: It is difficult to come up with a single function to show that these sets have the same cardinality. Since $\mathbb{N} \subseteq \mathbb{Q}$, we know that $|\mathbb{N}| \leq |\mathbb{Q}|$. We can define a function $f : \mathbb{Q} \to \mathbb{Z} \times \mathbb{Z}$ so that $f(0) = (0, 0)$ and for $x \neq 0$, $f(x) = (a, b)$ where $x = \dfrac{a}{b}$ is in lowest terms with $b > 0$. This f is an injection, so $|\mathbb{Q}| \leq |\mathbb{Z} \times \mathbb{Z}|$. We know from the last example that $|\mathbb{Z} \times \mathbb{Z}| = |\mathbb{N}|$, so we have:

$$|\mathbb{N}| \leq |\mathbb{Q}| \leq |\mathbb{Z} \times \mathbb{Z}| = |\mathbb{N}|.$$

Therefore, all of these cardinalities are equal.

Example 18.19. The set T of all bitstrings is countably infinite.

Solution: We need to find a way to list all of the bitstrings in T in an organized way so that we can assign a natural number to each. We will do so by length. First, we will list the bitstrings of length 0, then those of length 1, and then those of length 2, and then 3, and so on. Within each length, we will order the bitstrings as if they are base 2 numbers. This gives the following list and implied bijection.

$$
\begin{array}{rcl}
0 & \leftrightarrow & \lambda \\
1 & \leftrightarrow & 0 \\
2 & \leftrightarrow & 1 \\
3 & \leftrightarrow & 00 \\
4 & \leftrightarrow & 01 \\
5 & \leftrightarrow & 10 \\
6 & \leftrightarrow & 11 \\
7 & \leftrightarrow & 000 \\
8 & \leftrightarrow & 001 \\
9 & \leftrightarrow & 010 \\
10 & \leftrightarrow & 011 \\
11 & \leftrightarrow & 100 \\
12 & \leftrightarrow & 101 \\
13 & \leftrightarrow & 110 \\
14 & \leftrightarrow & 111 \\
15 & \leftrightarrow & 0000 \\
16 & \leftrightarrow & 0001 \\
\vdots & \vdots & \vdots
\end{array}
$$

Example 18.20. The set of strings over any finite alphabet is countably infinite.

Solution: Suppose that A is any finite alphabet (a set). We can mimic the last example and list the strings of length 0, then those of length 1, and then those of length 2, and then 3, and so on. This gives a list which can be paired with the natural numbers.

Example 18.21. Programming Languages. Any reasonable programming language has finitely many symbols in it. A program in that language is a string using those symbols which adheres to some grammatical rules. By the last example, there are a countably infinite number of possible strings. This means that the number of programs in any reasonable language is countably infinite.

Cardinality Examples. Here is a summary of cardinality examples from the section in table form.

Finite	Countably Infinite	Uncountable
$\emptyset$	$\mathbb{N}$, $\mathbb{Z}$, $\mathbb{Q}$, $\mathbb{Z} \times \mathbb{Z}$	$\mathbb{R}$, intervals
$\{0, 1, 2\}$	bitstrings	sequences of 0s and 1s
Symbols in any programming language	programs in any language	subsets of $\mathbb{N}$
	strings over a finite alphabet	

Consequences. The fact that are only countable many programs in any language but that there are uncountably many subsets of $\mathbb{N}$ and uncountably many functions from $\mathbb{N}$ (sequences) to $\{0, 1\}$ has some surprising consequences. Each statement below follows from the fact that there are more subsets of $\mathbb{N}$ and more functions from $\mathbb{N}$ to $\{0, 1\} \subseteq \mathbb{N}$ than there are programs.

- There are functions $f : \mathbb{N} \to \mathbb{N}$ which are not computable. This means there is no program which inputs a natural number n and outputs $f(n)$ correctly.
- There are subsets $A \subseteq \mathbb{N}$ for which it is impossible to write a program to decide if a number n is in A.

Treating a program as some sort of algorithmic description:

- There are functions $f : \mathbb{N} \to \mathbb{N}$ which cannot be described algorithmically.
- There are subsets $A \subseteq \mathbb{N}$ which cannot be described algorithmically.

19 Diagonalization and Undecidability

- Liar's Paradox
- Russel's Paradox
- Cantor's Theorem
- Halting Problem
- Uncomputable functions and sets
- Continuum Hypothesis

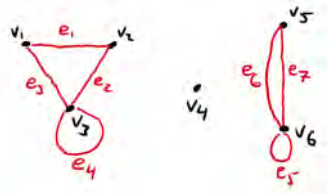
20 Graphs

Definition 20.1. A *graph* G consists of a nonempty set G_V called the *vertices* (or nodes) of G and a set G_E called the *edges* of G. Each edge is associated with a set of either one or two vertices called its *endpoints*.

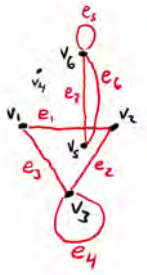
Example 20.2. The table below describes a graph G with vertices $G_V = \{v_1, v_2, v_3, v_4, v_5, v_6\}$.

Edge	Endpoints
e_1	$\{v_1, v_2\}$
e_2	$\{v_2, v_3\}$
e_3	$\{v_3, v_1\}$
e_4	$\{v_3\}$
e_5	$\{v_6\}$
e_6	$\{v_5, v_6\}$
e_7	$\{v_5, v_6\}$

Diagrams of Graphs. We can draw diagrams representing a graph G with vertex set G_V and edge set G_E. The diagram consists of a point for each vertex and a curve between the endpoints of each edge. A diagram of the graph in Example 20.2 is below.



Graphs can usually be depicted in more than one way. The diagram below represents the same graph as the diagram above. All that matters is that the right edges connect the right vertices.

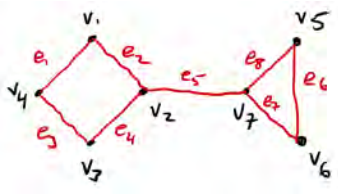


Directed Graphs. Our graphs will be almost exclusively *undirected*. That is, an edge is represented as an unordered pair $\{i, j\}$ between vertices i and j. In a *directed graph* edges are represented as ordered pairs (i, j). When drawing such edges, we draw an arrow from vertex i to vertext j. If we are going to work with a directed graph, we will explicitly state so. Unless otherwise stated, "graph" for us means "undirected graph."

Finite Graphs. If the vertex set of a graph is infinite, we say the graph is infinite. Almost every graph we will work with will be finite, so we assume that, unless otherwise stated, graphs are finite.

Definition 20.3. Two vertices in a graph are *adjacent* if there is an edge between them. In the graph above, v_1 is adjacent to v_2. An edge with one endpoint is a *loop*. In the graph above, e_4 and e_5 are loops. Any edge is said to be *incident* with its endpoints. A graph which has no loops and which has at most one edge between any two vertices is a *simple graph*.

Example 20.4. The graph in Example 20.2 is not a simple graph because it contains loops and because there are two edges between v_5 and v_6. The graph below is a simple graph.

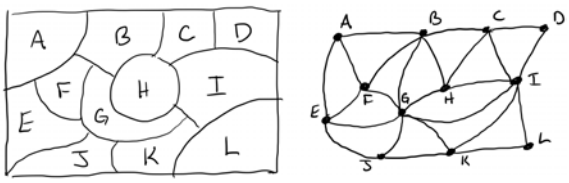


Binary Relations. You should be wondering whether or not graphs (or directed graphs) are simply binary relations. The answer is, almost. Directed graphs are almost identical to the concept of digraph that we drew for binary relations. However, with binary relations, we can have at most one edge/arrow between vertices i and j. We allow multiple edges between vertices in graphs. Simple graphs are essentially symmetric binary relations without loops. Directed graphs with no multiple edges are essentially binary relations.

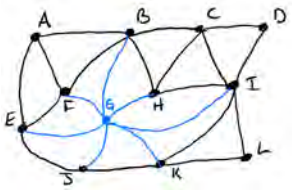
Definition 20.5. An edge in a graph is said to be *incident* with its edges. The *degree* of a vertex v, denoted $\deg(v)$, is the number of edges incident with v, with loops being counted twice.

Example 20.6. In the graph of Example 20.2, $\deg(v_1) = 2$, $\deg(v_3) = 4$, and $\deg(v_4) = 0$.

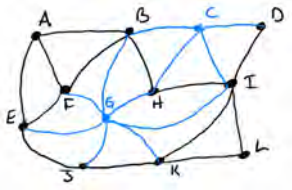
Map Coloring. Graphs can be used to model states or countries on a map. The vertices of the graph associated with a map in this way are the states or countries. There is an edge between two vertices if the states they represent share a border. For example, on the left below is a map of several "states." On the right is a graph depicting the map. To make the graph, simply place a dot in the center of each state and draw edges across state boundaries to connect the dots. Then, erase the map.



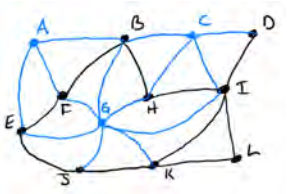
When drawing a map, it is ideal to color the map so that no two adjacent states have the same color. Reprsenting states as a graph can help with coloring the map with the fewest number of colors possible. We illustrate this here. It is helpful to work in the order of degrees, from highest to lowest. The vertex with highest degree is G, so we color G blue. None of the vertices adjacent to G can be blue. To help our process, we temporarily color G blue along with all edges incident to G.



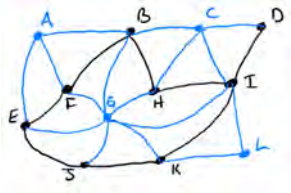
Among the vertices not touching anything blue, C has the highest degree, so we color C and its incident edges blue.



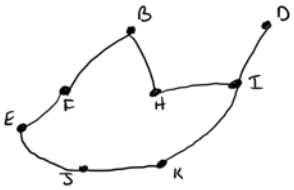
Among the vertices not touching anything blue, A has the highest degree, so we color A and its incident edges blue.



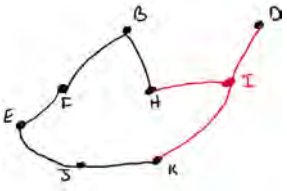
There is now one vertext not incident to any blue, L. We color L and its incident edges blue.



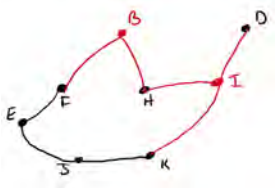
We are done with blue. To make the graph easier to look at, we remove all of the blue vertices and edges. What is left must be colored with colors other than blue. What is left could be called a *subgraph*.



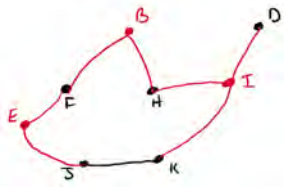
In this graph, the vertex with highest degree is I, so we color I and the edges incident to I red.



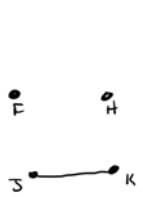
Among the vertices not incident to any red, B, E, F, and J all have degree 2. Among these, we *randomly* select B to color red.



Among the vertices not incident to any red, E and J all have degree 2. Among these, we *randomly* select E to color red.



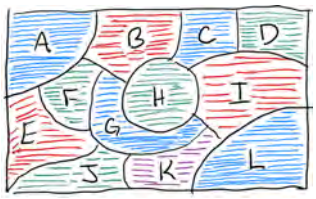
At this point all of the remaining vertices are incident to something red. We delete the red and see what is left.



The graph is simple enough at this point that we could figure out the coloring without an algorithm, but we persist with the algorithm. The vertices with highest degree are J and K. We arbitrarily choose to color J green.

The vertices not incident to green are all tied with degrees of 0, so we color them all green.

There is one vertex, K, not colored. We color it purple. Our coloring, then, is Blue: A, C, G, L; Red: B, E, I; Green: D, F, H, J; Purple: K. The colored map is below.



Definition 20.7. Since we used this word in our coloring exercise, we will define it. A graph H is a *subgraph* of a graph G if the vertices of H form a subset of the vertices of G and the edges of H form a subset of the edges of G.

We were able to color this map with just four colors. This is not an accident. The reason lies in a special property of the graphs derived from state borders.

Definition 20.8. A graph is *planar* if it can be drawn in the plane with no edges intersecting other than at vertices.

Every graph derived from state borders is planar, and one of the most famous theorems in graph theory is:

Theorem 20.9. Four Color Theorem *Every planar graph can be colored with four or fewer colors in such a way that no two adjacent vertices are the same color.*

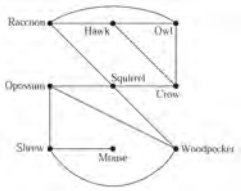
This theorem was the first example of a mathematical theorem which was proven with the assistance of computers. Incidentally, map makers "knew" this for centuries before there was a mathematical proof. We will see in the exercises that the method we used above for coloring maps can also be used for scheduling. One of the strengths of graph theory is that the same tools can be used to solve a variety of problems.

Road Maps. Another way to derive a graph from a map is to treat intersections, dead ends, and cul-de-sacs in the map as vertices. Edges in the map are roads between the vertices. There are variants of this type of graph where each side of a road is an edge or where each lane on a road is an edge. What we consider edges depends on what we want to accomplish. A police office may want to travel through the graph and observe everything. He or she would only need to traverse each road once, so roads are adequate for edges. A mail carrier may need to travel down each side of the road once, so edges for a mail carrier may be sides of roads (and some roads without mailboxes may be disregarded). An ice truck may need to travel down each lane of a road. Representing maps as graphs is important for such tasks as finding the shortest route through a city or the shortest route that visits several cities once each.

Tournaments. We can draw a graph to represent a set of teams playing each other in a tournament (or in a season). Vertices are the teams, and there is an edge between two teams if they play each other.

Association Graphs. For any set of people, we can make an association graph. The vertices are the people, and there is an edge between two vertices if those people are "associates." Here, *associate* may have different meanings depending on our purpose. If we are working in epidemiology, then two people may have an edge between them if they have any contact (that might spread disease). A question about a graph like this might relate to vaccination. When a person is vaccinated, remove them (and their incident edges). How many people need to be removed before we are reasonably sure the graph is disconnected (so disease cannot be spread). If we are letting nature take its course rather than vaccinating, then the question is still how many people have to be removed from the graph; however, "removal" means something different. Association graphs can also be used to model crime rings. Vertices are people, and edges are lines of communication. Here, the question is how many vertices must be removed before we are reasonably sure the lines of communication are broken. We could also build a graph like this from social media. Vertices are people, and edges are connections through social media. Here, applications may have to do with advertising. How many people in a community must I advertise to to be reasonably sure that the entire community will be exposed through social media?

Niche Graph. In biology, we can draw niche overlap graphs. Vertices are species. Two vertices are adjacent if those species compete (for the same food resources, for example).



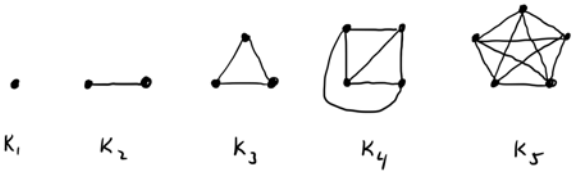
Summing Degrees. Each edge in a graph contributes 1 to the degree of each endpoint of that edge (or 2 in the case of loops). Therefore, if we add the degrees of all of the vertices in a graph, then edge will be represented in that sum twice. That is, the sum of the degrees is twice the number of edges.

Theorem 20.10. Handshaking Theorem *The sum of the degrees in a graph is equal to twice the number of edges in the graph.*

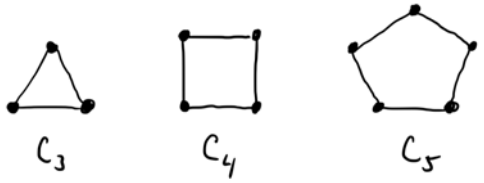
A consequence of this theorem is that the sum of the degrees of a graph must be even. This forces the number of vertices in the graph with odd degrees to be even, since a sum of an odd number of odd numbers is an odd number.

Lemma 20.11. *Any graph has an even number of vertices of odd degree.*

Complete Graphs. The *complete graph* on n vertices is a simple graph with n vertices so that there is exactly one edge between every pair of distinct vertices. We denote the complete graph on n vertices as K_n. K_1, K_2, K_3, K_4, and K_5 are shown below.

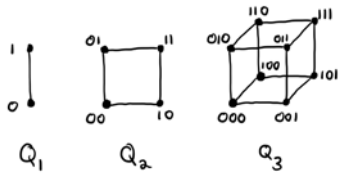


Cycles. The *cycle* on n vertices for $n \geq 3$ is a graph with n vertices $v_0, v_1, \ldots, v_{n-1}$ with an edge between v_i and $v_{(i+1)\%n}$ for each i. The cycle on n vertices is denoted C_n. C_3, C_4 and C_5 are shown below.

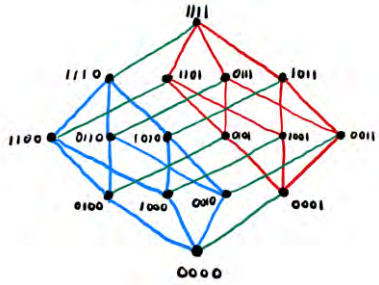


Wheels. If we add a "center" vertex to C_n which is adjacent to every original vertex, we get the wheel W_n. W_3, W_4, and W_5 are shown below.

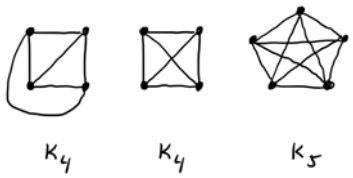
n-Cubes. The n-dimensional hypercube or *n-cube* is a graph Q_n whose vertices are the 2^n bitstrings of length n. Two vertices are adjacent if the corresponding bitstrings differ at exactly one place. Q_1, Q_2, and Q_3 are shown below.



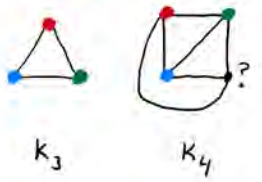
Q_4 is a bit more challenging to draw. We have included colors to show the structure. Q_4 contains two copies of Q_3, here in red and blue. (Actually, there are four copies, but we have shown two.) The blue copy includes those strings ending in 0. The red includes those ending in 1. There are green edges connecting the blue Q_3 to the corresponding vertices in the red Q_3.



Planar Graphs. A graph is planar if it can be drawn in the plane in such a way that none of its edges cross each other except at vertices. That K_4 is planar can be seen in the picture below on the left. However, if we draw K_4 as in the middle picture, planarity is not obvious. The graph K_5 on the right is not planar.

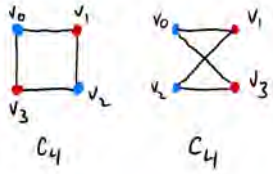


3-Colorable Graphs. A simple graph is *3-colorable* if 3 or fewer colors can be assigned to its vertices in such a way that no two adjacent vertices are assigned the same color. The graph K_3 is 3-colorable. The graph K_4 is not.



Determining if an arbitrary graph is 3-colorable is, in general, computationally complex.

Bipartite Graphs. A simple graph is *bipartite* if its vertices can be partitioned into two disjoint sets V_1 and V_2 so that every edge in the graph connects a vertex from V_1 and a vertex in V_2. This implies there is no edge connecting two vertices in V_1 or two vertices in V_2. The graph C_3 is not bipartite. The graph C_4 is bipartite, as is seen in the picture below.

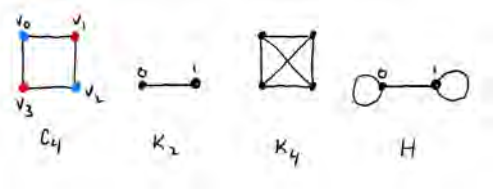


A graph is bipartite if and only if it is 2-colorable – two colors can be assigned to the vertices in such a way that no two adjacent vertices are the same color.

Homomorphisms and Isomorphisms. Graph colorability can be described using the notion of homomorphism.

Definition 20.12. A *homomorphism* from a graph G to a graph H is a function $f : G_V \to H_V$ from the vertext set of G to the vertex set of H so that if there is an edge between vertices i and j in G then there is an edge between vertices $f(i)$ and $f(j)$ in H.

In the picture below, there is a homomorphism f from C_4 to K_2 that maps $f(v_0) = f(v_2) = 0$ and $f(v_1) = f(v_3) = 1$. However, there is no homomorphism from K_4 to K_2. Any function f from K_4 to K_2 would have to identify two vertices i and j. Since there is an edge between i and j, this means that there would be an edge between $f(i)$ and $f(j)$. However, if $f(i) = f(j)$, then this would be a loop, and there are no loops in K_2. There is however, a homomorphism from K_4 to the graph of H below.



The relationship between coloring and homomorphisms is this:

Theorem 20.13. *A graph G is 2-colorable if and only if there is a homomorphism from G to K_2. A graph G is 3-colorable if and only if there is a homomorphism from G to K_3.*

Definition 20.14. As *isomorphism* from a graph G to a graph H is a bijective function $f : G_V \to H_V$ from the vertex set of G to the vertex set of H so that there is an edge between vertices i and j in G if and only if there is an edge between vertices $f(i)$ and $f(j)$ in H. If there is an isomorphism between G and H, then we say that G and H are *isomorphic*.

Example 20.15. The graphs graphs K_2 and Q_1 are isomorphic. The graphs Q_2 and C_4 are isomorphic.

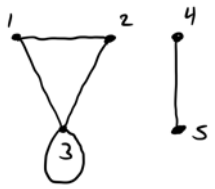
Isomorphic graphs are identical except for the names of the elements.

Adjacency Matrices. For a graph without multiple edges between the same vertices, we can make an *adjacency matrix*. Number the vertices of a graph G as $1, 2, \ldots, n$. The adjacency matrix of G is an $n \times n$ matrix M^G so that the i, j-entry $M^G_{i,j}$ is 1 if and only if there is an edge between i and j in G and is 0 otherwise. Note that since our graphs are undirected, M^G is symmetric.

Example 20.16. Find the graph with this adjacency matix:

$$\begin{pmatrix} 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

Solution: In this graph, 1 must be adjacent to 2 and 3, 2 is adjacent to 1 and 3, and 3 is adjacent to 1, 2, and 3. In addition, 4 and 5 are adjacent.

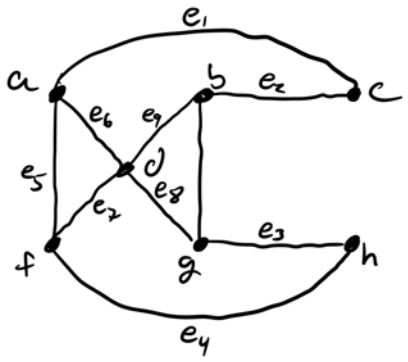


Exercises 20.17. Please answer the questions below.

1. Draw a graph with this edge table. Draw your graph with no edges crossing.

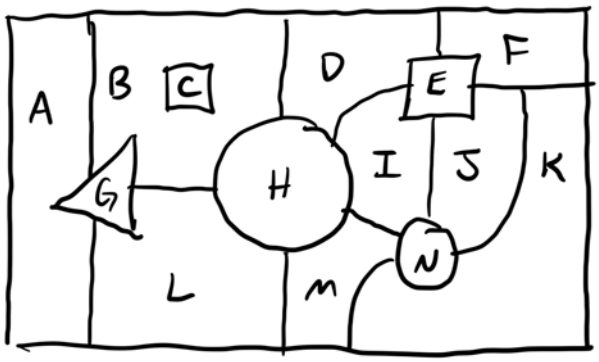
Edge	Endpoints
e_1	$\{v_1, v_2\}$
e_2	$\{v_1, v_3\}$
e_3	$\{v_2, v_5\}$
e_4	$\{v_2, v_3\}$
e_5	$\{v_1, v_4\}$
e_6	$\{v_4, v_5\}$
e_7	$\{v_2\}$

2. Consider this graph:



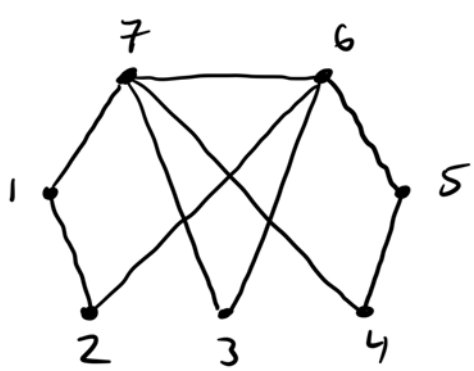
   (a) Which vertices are adjacent to b?
   (b) Which edges are incident to g?
   (c) Find the degree of each vertex.

3. Consider this map:



   (a) Draw a graph describing the states and borders in this map.
   (b) Color the map with four or fewer colors in such a way that no two adjacent states have the same color.

4. Final exam times need to be selected for Math 1, Math 2, Math 3, Math 4, Math 5, Math 6, and Math 7. The department wants to use as few times as possible. However, the exams cannot all be at the same time because some of the classes share students. Below is a graph with the classes as vertices. An edge between two classes means there is a student in both classes.



Color the graph with as few colors as possible. How can this coloring be helpful in scheduling the exams?

5. Ten people are at a party: Alice, Bob, Cal, Doug, Eunice, Fran, Grace, Hank, Ian, and Jack. They do not all know each other. Here is a table of who knows whom.

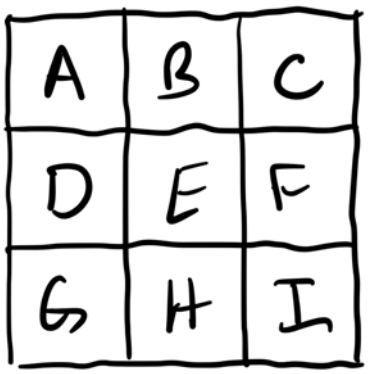
Person	Knows
A	B, D, E, H
B	A, C, F, H, I
C	B, D, E, F, G
D	A, C, E
E	A, C, D
F	B, C, G, I
G	C, F, H
H	A, B, G
I	B, F, J
J	I

Draw a graph in which the vertices are the people at the party. An edge between two vertices means they know each other.

6. In the niche graph in this section, what is the degree of the vertex labeled squirrel? What is the meaning of this number?

7. Draw a planar representation of the graph Q_3.

8. Consider this 3×3 chess board.



A knight on the chess board can legally move in an L-shape. It can move two steps forward (in any direction) followed by one step to either side. For example, if the knight is at D, it can move two steps

right followed by a step up to reach C, or it can move two steps right followed by a step down to I. Draw a graph whose vertices are the squares of the chess board. Draw an edge between two vertices if a knight can move from one vertex to the other. Draw a planar version of this graph (that is possible). Is the graph connected?

9. The graph Q_3 is 2-colorable. Color it with 2 colors.

10. The graph Q_4 is 2-colorable. Color it with 2 colors.

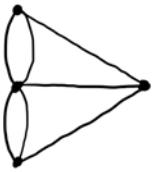
11. Draw the graph represented by this adjacency matrix.

$$\begin{pmatrix} 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \end{pmatrix}$$

21 Euler Paths

Graph theory was invented in 1736 by Leonard Euler to solve the Konigsberg Bridge Problem. In eighteenth century Konigsberg, there were seven bridges. Apparently, the good folk of Konigsberg spent their spare time walking around town attempting to walk across every bridge exactly once while starting and stopping on the same land mass.

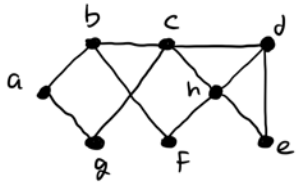
Euler abstracted the problem by creating a graph whose vertices represented the land masses of town and whose edges were the bridges.



Based on this use of graph theory (and on many modern applications) we introduce terminology to describe walking around a graph. We follow the terminology of Rosen because it is as simple of terminology as we can use and still be considered somewhat standard. However, be aware that this terminology varies dramatically, especially across different fields of mathematics and computer science.

Definition 21.1. A *path of length* n in a graph G is an ordered list of n edges, $e_1e_2\dots e_n$ so that there are vertices $x_0, x_1, \dots, x_n$ with $e_i = \{x_{i-1}, x_i\}$ for $i = 1, 2, \dots, n$. That is $e_1 = \{x_0, x_1\}$, $e_2 = \{x_1, x_2\}$, all the way to $e_n = \{x_{n-1}, x_n\}$. We say that such a path is a path from x_0 to x_n. If $x_0 = x_n$, the we say that the path is a *circuit*. A path or circuit which does not contain the same edge more than once is *simple*. If G has no multiple edges, then we describe the path here as $x_0, x_1, \dots, x_n$ since each pair x_{i-1}, x_i uniquely determines each edge e_i.

Example 21.2. Consider this graph:

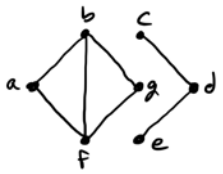


In this graph, a, g, c, h is a simple path of length 3 from a to h. Also, a, g, c, h, d, c, h, e is a path of length 7 from a to e which is not simple. The list a, g, f, e is not a path because there are not edges between g and f or between f and e. Finally, $a, b, c, d, e, h, c, g, a$ is a simple circuit.

Rotating Circuits. Consider the circuit $a, b, c, d, e, h, c, g, a$ in the last example, if we start at b rather than a but follow the same steps, we get $b, c, d, e, h, c, g, a, b$, which is also a circuit. We can rotate circuits in this manner to start at different vertices.

Definition 21.3. If there is a path between every pair of distinct vertices in a graph G, then G is *connected*. If v is a vertex in a graph G, then the *connected component* containing v consists of v along with of all vertices u for which there is a path from v to u.

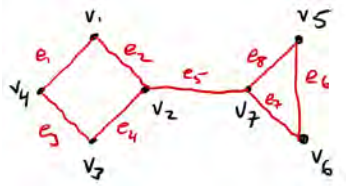
Example 21.4. Consider the graph below. This graph is not connected because, for example, there is no path from a to c.



The connected component containing a is $\{a, b, g, f\}$. The connected component containing c is $\{c, d, e\}$.

Definition 21.5. If e is an edge in a connected graph G, and if G becomes disconnected if e is removed, then e is called a *bridge* or *cut edge*. If v is a vertex in a connected graph G, and if G becomes disconnected if v is removed, then v is a *cut vertex*.

Example 21.6. In this graph e_5 is a cut edge. The vertices v_2 and v_7 are cut vertices.



The Konigsberg Bridges problem asks for a simple circuit in a graph which uses every edge. We call such circuits Euler circuits.

Definition 21.7. A path in a graph which uses every edge of the graph exactly once is an *Euler path*. A circuit in a graph which uses every edge of the graph exactly once is an *Euler circuit*.

Note. An Euler path is by necessity simple. If a path is not simple, then it uses an edge more than once. This would make it not an Euler circuit.

Street Sweepers. Suppose we want to find a route for street sweepers to follow in a small town. The street sweepers must travel down both sides of each street in town and should start and stop at the same location. Let the intersections of the town be the vertices of a graph, and let the sides of the roads be edges. The most efficient route for the mail carrier to follow is an Euler circuit.

Salt Trucks. Suppose that we need to find an efficient route for salt trucks to travel to salt all of the bridges in town. Let the land masses (or regions of town connected by bridges) be the vertices of a graph, and let the bridges be the edges in the graph. The trucks need to cross every bridge exactly once, starting and stoping in the same location. The most efficient route for them to follow is an Euler circuit.

Tour Guides. Suppose that you are giving an outdoor tour of campus. Let the sidewalks on campus be edges in a graph whose vertices are the intersections of sidewalks. An efficient route through campus would be an Euler path. If you want to start and stop at the same location, then you would want an Euler circuit.

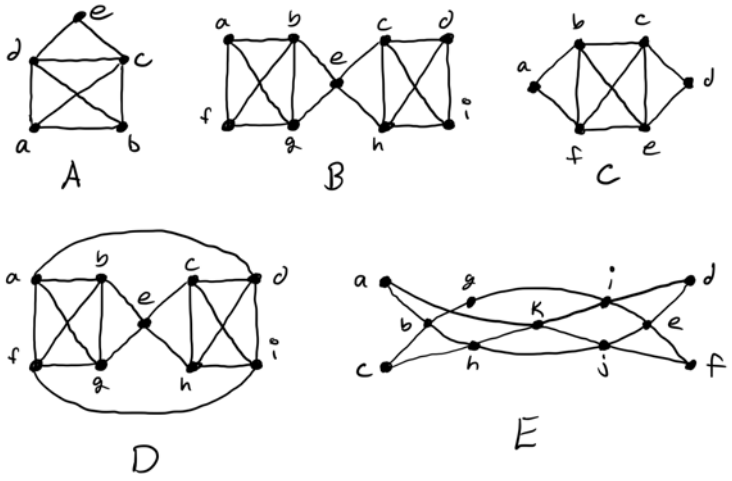
Security. Suppose that a security guard wants to patrol a neighborhood. He or she wants to follow a path that patrols every road in the neighborhood exactly once. In a graph where the vertices are intersections (or dead ends and cul-de-sacs) and the edges are roads, the most efficient patrol route is an Euler path. If the security guard wants to start and stop at the same location, the an Euler circuit would be ideal.

Ins and Outs. Suppose that we are walking along an Euler circuit of a graph G beginning at a vertex v. We begin by leaving v. Since we must begin and end at v (this being circuit), we must return to v at some point. Therefore, the initial edge we use to leave v is paired with the final edge that we use to return to

v. At some point during the circuit, we might return to v prior to the end of the circuit. If this happens, then we must also leave v again. Therefore, every edge coming into v must be paired with an edge to leave v along. If this is the case, then the degree of v must be even. Since we can follow a circuit by beginning at any vertex, the degree of every vertex must be even. Now, if we are following an Euler path which is not an Euler circuit, then we will not return to v at the end. This means that the initial outward edge at v is not paired with a return edge. Since v is missing an edge, its degree is odd. Similarly, wherever we end up with at the end of the path must be missing an edge to leave by and must have odd degree. This should make the next theorem (due to Euler) seem like it might be reasonable.

Theorem 21.8. *A connected graph has an Euler circuit if and only if the degree of every vertex in the graph is even. A connected graph has an Euler path but not an Euler circuit if the graph has exactly two vertices of odd degree.*

Example 21.9. Which of the graphs below have Euler circuits? Which have Euler paths which are not circuits?



Solution: In graph A, vertices a and b have odd degree, but all other vertices have even degree. Therefore, A has an Euler path which is not a circuit. In graph B, vertices a, d, i, and f have odd degree. Since there are more than two odd vertices, B has no Euler circuit and no Euler path. In graph C, every vertex has even degree, so C has an Euler circuit. In graph D, every vertex has even degree, so D has an Euler circuit. In graph E, every vertex has even degree, so E has an Euler circuit.

Fleury's Algorithm. Finding Euler paths and circuits is oddly easy to do and requires almost no forethought. The technique we use is called Fleury's Algorithm. The algorithm selects a starting vertex and progressively follows edges from that vertext to other vertices. After an edge is followed, it is removed from the graph. The only restriction is that we should not follow a cut edge (bridge) until necessary. Here is the algorithm for finding an Euler path or circuit in a graph G. The algorithm builds an ordered list P of vertices which describes the path through G. Along the way, the graph G is dismantled.

1. If G has no odd degree vertices, then select a random vertex v_0 to begin at. This is your current vertex. If G has exactly two odd degree vertices, select one of them as the starting vertex v_0. This is your current vertex. Let $P = (v_0)$ (an ordered list of one vertex).

2. While G still contains edges, do the following:

   (a) Let v be the current vertex.

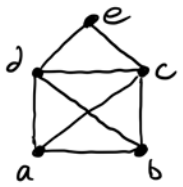
   (b) If there are one or more edges incident to v which are not cut edges, randomly select one of them. If the only edge incident to v is a cut edge, select that edge. Let e be the selectd edge. Let w be the other endpoint of e.

(c) Add the vertex w onto the end of P.

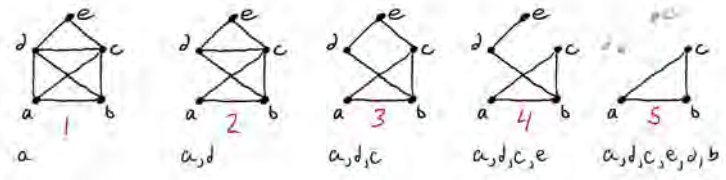
(d) Delete the edge e from G. If this isolates v, delete v from G.

At the end of these steps, P will be an ordered list of vertices which describes the path through the original G.

Example 21.10. Use Fleury's Algorithm to find an Euler path in this graph.

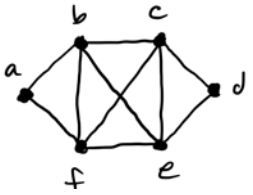


Solution: Since the vertices a and b have odd degree, we must begin at one of them. We choose to begin at a. Most of the steps in building that Euler path are illustrated below.

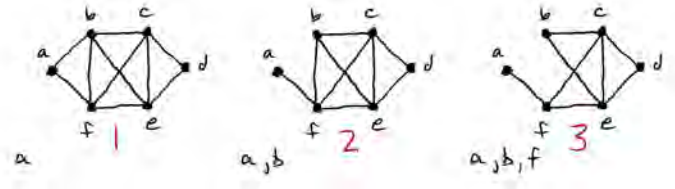


We begin with $P = (a)$ in step 1. From a, we can move to b, c, or d without following a cut edge. We randomly select d and then remove the edge from a to d in step 2. Now $P = (a, d)$, and the current vertex is d. From d we can go to b, c, or e without following a cut edge. We choose c and delete the edge from d to c in step 3. Now $P = (a, d, c)$, and c is the current vertex. From c, we can go to a, b, or e without following a cut edge. We choose e and delete the edge from c to e in step 4. Now $P = (a, d, c, e)$, and e is the current vertex. The only edge from e is a cut edge to d, so we must follow it. From d, the only edge is a cut edge to b, so we must follow it. After following these two edges and deleting them, we also delete e and d. Now $P = (a, d, c, e, d, b)$, and the current vertex is b. From b we can go to a or to c. We go to c, and then the rest of the path is completely determined, so our Euler path is $P = (a, d, c, e, d, b, c, a, b)$.

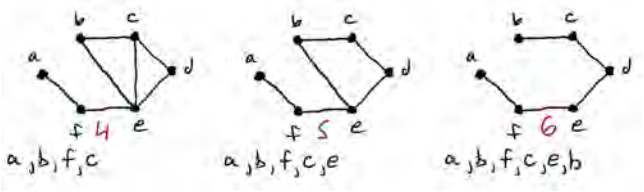
Example 21.11. Use Fleury's algorithm to find an Euler circuit in this graph.



Solution: Since all of the vertices have even degree, we can begin at any vertex. We start at a with $P = (a)$. From a we can go to b or f without following a cut edge, so we move to b and delete the edge from a to b in step 2. Now $P = (a, b)$, and b is the current vertex.



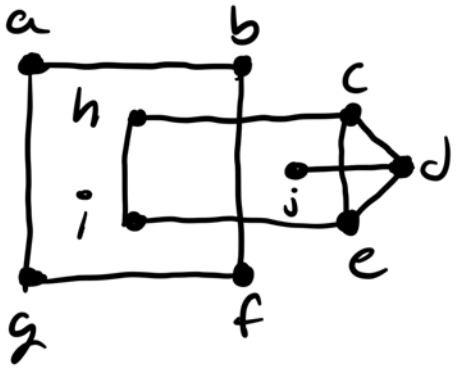
From b we can go to c, e, or f without following a cut edge. We choose f and delete the edge from b to f in step 3. Now $P = (a, b, f)$, and f is the current vertex. There at three edges incident to f. We cannot follow the one to a because it is a cut edge and we have edges which are not cut edges to choose from. We can move to c or e without following a cut edge. We choose to move to c and delete the edge from f to c in step 4. Now $P = (a, b, f, c)$, and c is the current vertex.



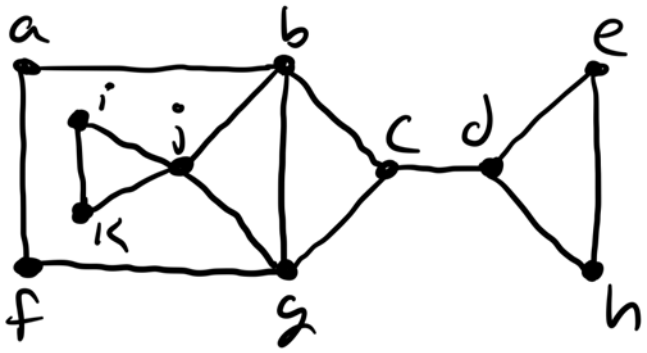
From c, we can move to b, d, or e without following a cut edge. We choose to move to e and delete the corresponding edge in step 5. Now $P = (a, b, f, c, e)$, and e is the current vertex. From e, we cannot yet follow the edge to f since this is a cut edge and we have edges which are not cut edges to choose from. We can move to b or d without using a cut edge. We choose to move to b and delete the corrsponding edge in step 6. Now $P = (a, b, f, c, e, b)$. From this point, the rest of the circuit is uniquely determined. The final circuit is $P = (a, b, f, c, e, b, c, d, e, f, a)$.

Exercises 21.12. Answer the questions below.

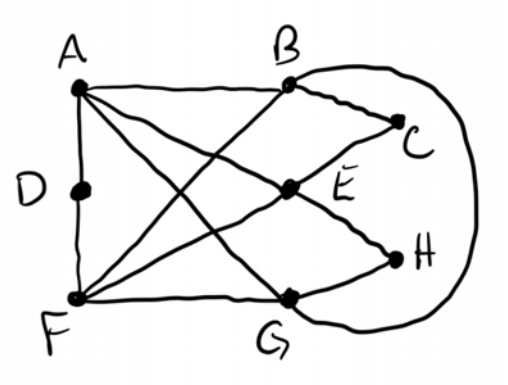
1. This is a circuit in some graph: $acbdfehgjia$. Rewrite the circuit beginning at f.
2. Find the connected component of g in this graph.



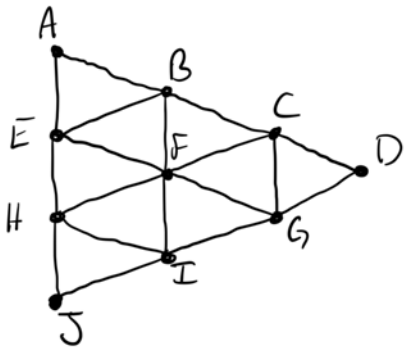
3. Find all cut edges (bridges) and cut vertices in this graph.



4. Use Fleury's Algorithm to find an Euler circuit in this graph.

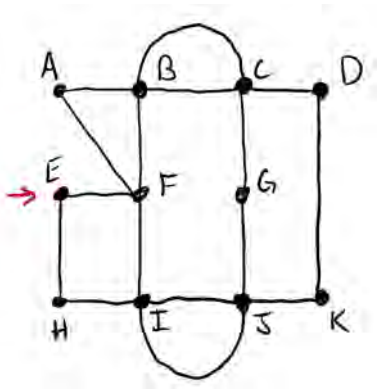


5. Use Fleury's Algorithm to find an Euler circuit in this graph.



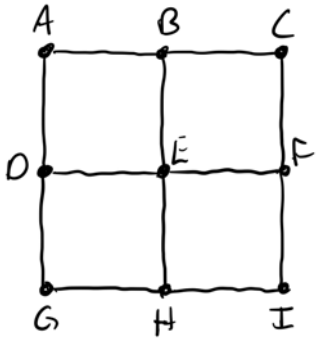
6. For which n does Q_n have an Euler circuit?

7. The graph below represents a neighborhood. Edges are roads, and vertices are intersections.



A security guard wants to enter the neighborhood at the red arrow (E), drive every street in the neighborhood once, and exit again at the red arrow. What is a path he could follow?

8. The graph below represents a neighborhood. Edges are roads, and vertices are intersections.



This graph has no Euler circuit. Find a circuit through the neighborhood that reuses as few edges as possible.

22 Hamilton Paths

It is not the case that every time we want to travel through a graph we want to use every edge. For example, a mail carrier may not need to visit every road in a town because there may be some roads without mailboxes. If the mailboxes are vertices, then what the mail carrier wants to do is visit every vertex exactly once.

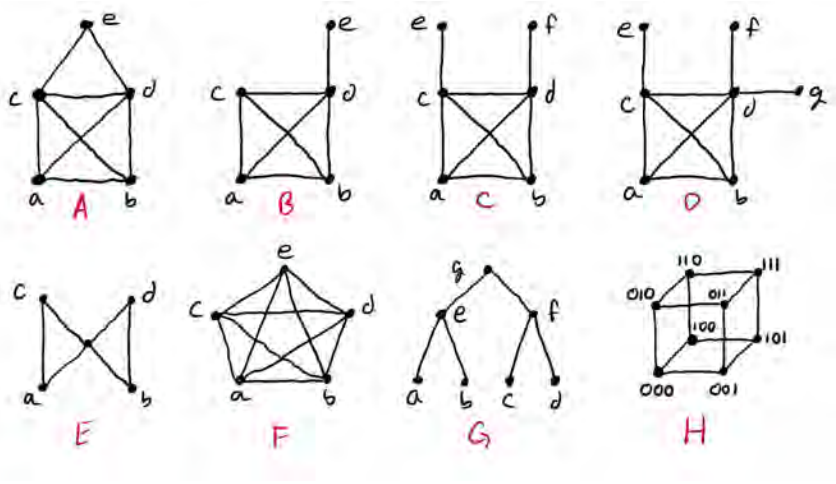
Definition 22.1. Suppose that G is a graph. A *Hamilton path* in G is a path in G which passes through every vertex exactly once. A *Hamilton circuit* in G is a circuit in G which passes through every vertex exactly once (except that the first and last vertices are the same).

Note. A Hamilton path is necessarily simple. If a path uses any edge more than once, it would use the endpoints of that edge more than once.

Icosian Puzzle. The concept of Hamilton path comes from a game, the Icosian Puzzle, invented by Irish mathematician Sir William Rowan Hamilton in 1857. The game consisted of a dodecahedron (a polyhedron with 12 pentagonal sides) with a peg at each of its 20 vertices. The object of the game was to find a path beginning at one vertex, following edges, visiting every vertex once, and returning to the initial vertex – to find a Hamilton circuit treating the vertices and edges of the polyhedron as a graph.

Security Guards. Suppose that a security guard on a college campus must walk a route that visits every building on campus. Let the vertices of a graph be the buildings on campus, and let the edges be the sidewalks between them. An efficient route for the security guard is a Hamilton path in the graph. If the guard wants to start and stop at the same location, a Hamilton circuit would be appropriate.

Example 22.2. Which of the graphs below have Hamilton circuits? Which have Hamilton paths but not circuits?



Solution: Graph A has a Hamilton circuit: a, b, d, e, c, a. Graph B cannot have a Hamilton circuit because such a circuit would have to use the edge from d to e (and hence the vertices d and e) twice. However, B does have a Hamilton path: a, b, c, d, e. Graph C cannot have a Hamilton circuit because such a circuit would have to use the edges from d to f and from c to e twice. However, C does have a Hamilton path: e, c, a, b, d, f. Graph D cannot have a Hamilton circuit because such a circuit would have to use the edges from e to c, from f to d, and from d to g twice. This graph also cannot have a Hamilton path because such a path would have to use one of these edges twice (while it could begin and end with the other two). Graph E cannot have a Hamilton circuit. Any such circuit would have to use the center vertex twice. This graph does have a Hamilton path: a, c, e, d, b (where e is the center vertex). Graph F has many Hamilton circuits. In fact, every rearrangement of the vertices of F gives a Hamilton circuit. This is because F is a complete graph and has edges between every pair of vertices. Graph G has no Hamilton circuit because it has no simple circuits! This graph also has no Hamilton path. Any Hamilton path would have to use at least one of the edges incident to a, b, c, or d twice. Graph H has a Hamilton cycle: $000, 001, 011, 111, 101, 100, 110, 010, 000$

Conditions. There is no easy test to determine if a graph has a Hamilton circuit like there is for Euler circuits. There are some theorems that guarantee the existence of a Hamilton circuit in some cases. These theorems essentially state, "If a simple graph has *enough* edges, then it has a Hamilton circuit." Here are two examples.

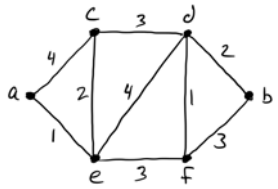
Theorem 22.3. Dirac's Theorem *If G is a simple graph with $n \geq 3$ vertices such that the degree of every vertex is at least $n/2$, then G has a Hamilton circuit.*

Theorem 22.4. Ore's Theorem *If G is a simple graph with $n \geq 3$ vertices such that $\deg(a) + \deg(b) \geq n$ for every pair of nonadjacent vertices a and b, then G has a Hamilton circuit.*

Weighted Graphs. Graphs which model certain situations in the real world (maps and computer networks, for instance) have natural ways to assign distances or costs to each edge.

Definition 22.5. A *weighted graph* is a graph with numbers called *weights* on each edge. The *weight of a path* in a weighted graph is the sum of the weights of the edges in that path. The *shortest path* between two vertices is the path with the least weight among all paths between the vertices.

Example 22.6. Find the weight of the circuit a, c, d, b, f, e, a in this graph.



Solution: The weights of the edges in this circuit are 4, 3, 2, 3, 3, and 1, so the weight is 4+3+2+3+3+1 = 16.

Dijkstra's Algorithm. Dijkstra's Algorithm is an algorithm for finding the length of the shortest path from a specified vertex to any other vertex in a connected weighted graph G. The algorithm operates on these values at each step:

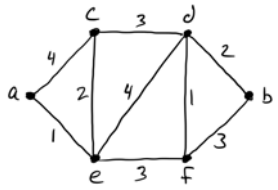
- $w(u, v)$ is the weight of the edge from u to v.
- $D(v)$ is the (potential) minimum distance from the initial vertex to a vertex v. This value is initially set to ∞ and is modified at each step of the algorithm.
- V is the set of vertices that have already been visited. These are the vertices for which a minimum value of $D(v)$ has already been determined.
- $A(v)$ is a vertex in V adjacent to v which is in the path from the initial vertex to v.
- H is a graph which is built during the process from which the minimal path from the inital vertex to any other vertex can be derived.

Dijkstra's Algorithm beginning at a vertex a follows these steps:

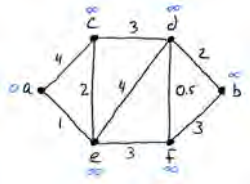
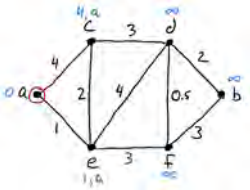
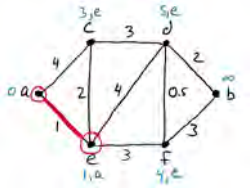
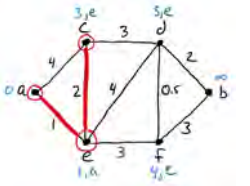
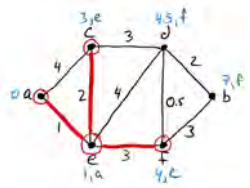
- Let the vertices of H be the vertices of G, and let the edge set of H be $H_E = \{\}$.
- Let $D(a) = 0$ and let $D(v) = \infty$ for all vertices $v \neq a$.
- Initialize $V = \{\}$.
- While there are vertices outside of V
  - Let v be a vertex not in V for which $D(v)$ is minimal. (On the first pass, this will result in $v = a$ since $D(a) = 0 < \infty$.)

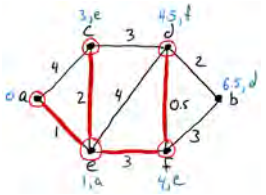
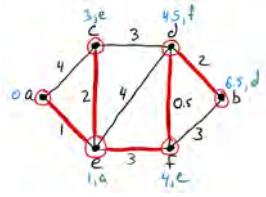
– Add v to V.
– If $v \neq a$, add the edge $\{v, A(v)\}$ to H.
– For each vertex u adjacent to v, if $D(v)+w(v,u) < D(u)$, let $D(u) = D(v)+w(v,u)$ and $A(u) = v$. (If the path through v to u is shorter than the previously known path, log the shorter distance and remember which vertex $A(u)$ gave that distance.)

Example 22.7. Apply Dijkstra's Algorithm in this graph beginning at a.



Solution: We will do this pictorially on the graph. Values of D will be drawn in blue. Values of A will be drawn in green. Vertices in V will be circled in red, and edges in H will be colored red.

First we initialize the distances. $D(a) = 0$, and the other distances are ∞.	
When we enter the loop, we place a in V and update the distances to e and c. We set both $A(c)$ and $A(e)$ equal to a since they are adjacent to a.	
On the next pass through the loop, e has the minimum distance from a, so we put e in V and add the edge from a to e to H. We then update distances and adjacency for c, d, and f.	
Next, the vertex with the minimum distance is c, so we add c to V and draw the edge from c to e. The distance to d through c is not shorter than the current value of $D(d)$, so we leave distances and adjacencies unchanged.	
The next minimum value of D is at f, so we add f to V and put the edge from e to f in H. The distance to d through f is less than the current distance, so we update the distance and adjacency for d along with b.	

The next shortest distance is to d, so we add d to V and add the edge from f to d to H. We also update the distance and adjacency for b.	
There is one vertex, b, left, so we add it to V and we add the edge to H.	

We now have calculated the minimum distance from a to every vertex of G. The function A (or the graph H) can be used to actually retreive the paths. For example, if we want the path from a to b, we calculate:

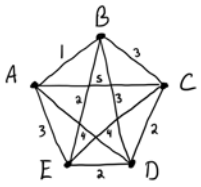
$$A(b) = d \text{ and } A(d) = f \text{ and } A(f) = e \text{ and } A(e) = a.$$

The path, then, is a, e, f, d, b.

Complete Graphs and Traveling Salespersons. Every complete graph has many Hamilton circuits. In fact, a complete graph on n vertices has $(n-1)!$ Hamilton circuits since every rearrangement of the vertices gives a Hamilton circuit (we will discuss this number in detail in the counting sections). For a weighted complete graph, the challenge is to find the Hamilton circuit with the lowest weight. This is known as the Traveling Salesperson Problem (TSP). The idea is that a salesperson must travel to visit every town in a region, visiting every town exactly once, and returning to the first location at the end.

Brute Force. One way to solve the TSP is by brute force. Simply list every Hamilton circuit, find all of their weights, and select the lowest weight.

Example 22.8. Use brute force to find the minimum weight Hamilton circuit in this graph.



Solution: Since we can rotate a circuit without changing its weight, we arbitrarily choose to begin at A. There are $4! = 24$ different Hamilton circuits beginning at the vertex A. The circuits and their weights are in the table below.

Circuit	*Weight*	*Circuit*	*Weight*	*Circuit*	*Weight*	*Circuit*	*Weight*
$ABCDEA$	11	$AEDCBA$	11	$ACBDEA$	16	$AEDBCA$	16
$ABCEDA$	14	$ADECBA$	14	$ACBEDA$	16	$ADEBCA$	16
$ABDCEA$	13	$AECDBA$	13	$ADBCEA$	17	$AECBDA$	17
$ABDECA$	15	$ACEDBA$	15	$ADBECA$	18	$ACEBDA$	18
$ABECDA$	13	$ADCEBA$	13	$AEBCDA$	14	$ADCBEA$	14
$ABEDCA$	12	$ACDEBA$	12	$AEBDCA$	15	$ACDBEA$	15

The minimum weight is 11. This weight actually shows up twice in the circuits A, B, C, D, E, A and A, E, D, C, B, A. Note that these are actually the same circuit followed in reverse order. Actually, all of the weights in the table come in pairs for the same circuit followed forwards and backwards.

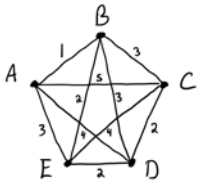
The brute force approach is unreasonable even for small sets. Below is a table showing how long it would take to solve the TSP problem using brute force for 11 to 20 vertices assuming we can check one million paths every second.

Vertices	Time required
11	3.62 seconds
12	39.91 seconds
13	7.98 minutes
14	1.73 hours
15	1.01 days
16	15.14 days
17	242.16 days
18	11.28 years
19	203.02 years
20	3857.34 years

Nearest Neighbor. Since brute force is not a viable approach to solving the TSP, a variety of simpler algorithms have been developed which find good, but possibly not optimum, solutions. The simplest of these is the Nearest Neighbor Algorithm:

- Select an initial vertex v and set the path P equal to (v) (an ordered list of one vertex) and set the current vertex equal to v.
- While there are still vertices not in P:
  - Among the edges incident to the current vertex which connect to vertices not in P, select one with minimum weight. Let u be the endpoint of this edge other than the current vertex.
  - Add u to the end of P.
  - Set the current vertex to u.
- Add the original vertex v to the end of P.

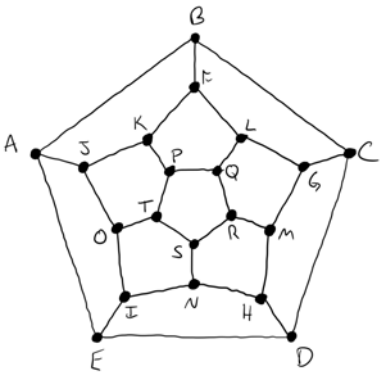
Example 22.9. Use the Nearest Neighbor algorithm to find a short Hamilton circuit starting at A in this graph.



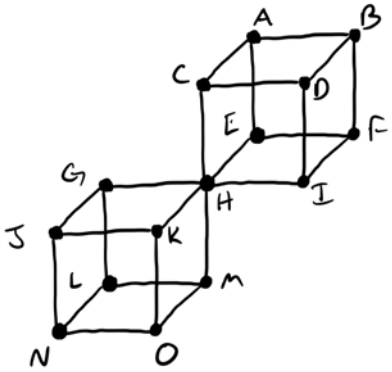
Solution: The minimum weight edge incident to A is the edge to B with weight 1, so we follow that edge to B. The minimum weight edge incident to B which does not lead to A is the one to E with weight 2, so we follow that edge to E. The minimum weight edge incident to E that does not lead to A or B is the one to D with weight 2, so we move to D. At this point, the only edge we have not visited is C, so we follow the edge of weight 2 to C. We have visited all of the edges at this point, so we follow the edge of weight 5 back to A. Our path is A, B, E, D, C, A with weight 12. This is the same graph as in Exercise 22.8. Notice that the Nearest Neighbor algorithm found a good circuit but not the best circuit.

Exercises 22.10. Answer the questions below.

1. Solve Hamilton's Icosian Puzzle by finding a Hamilton circuit in this graph which is equivalent to a dodecahedron.

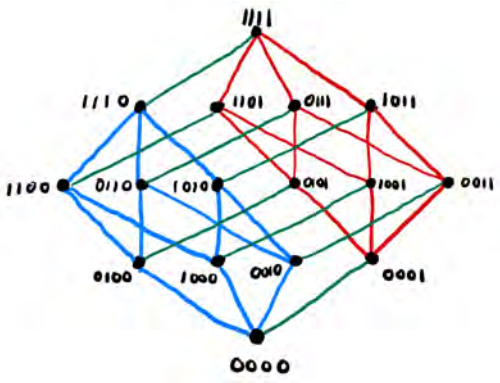


2. Find a Hamilton path in this graph which consists of two copies of Q_3 glued together.

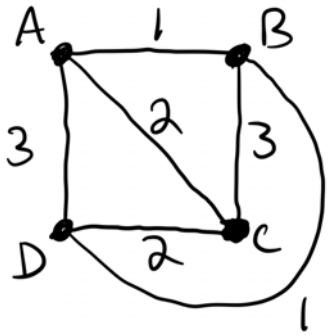


   Explain why the graph has no Hamilton circuit.

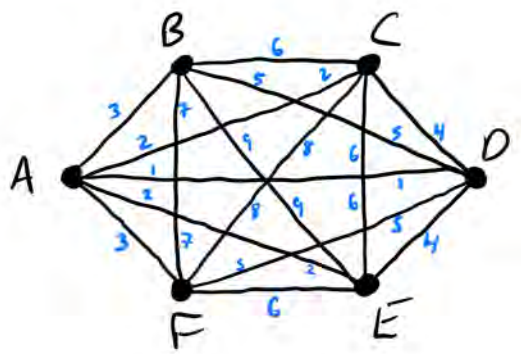
3. Does Q_4 have a Hamilton circuit?



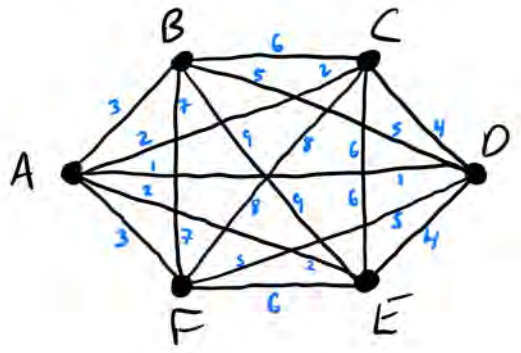
4. Use brute force to find a TSP solution for this graph.



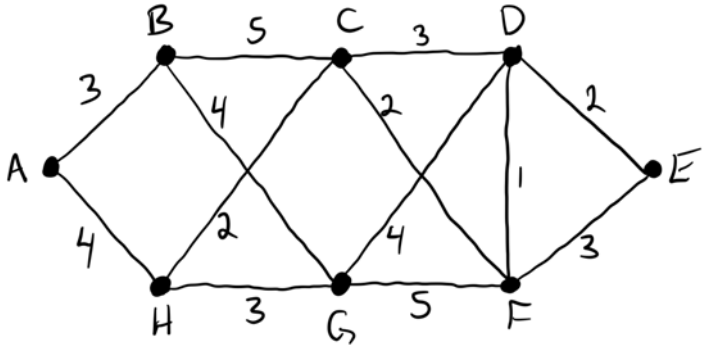
5. Use the nearest neighbor algorithm to find a short Hamilton circuit in this complete graph starting at vertex A.



6. Use the nearest neighbor algorithm to find a short Hamilton circuit in this complete graph starting at vertex B.



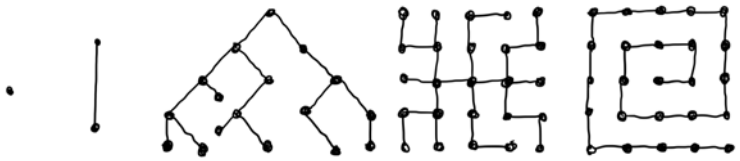
7. Use Dijkstra's Algorithm to find the minimum distance from A to every other vertex in this graph.



23 Trees

Definition 23.1. A connected graph with no simple circuits is called a *tree*.

Example 23.2. Each of these five graphs is a tree.



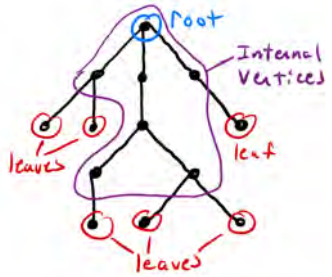
Neither of these graphs is a tree. The graph on the left has a circuit. The graph on the right is not connected. One might call the graph on the right a "forest" since it is a collection of trees.

Minimal Connectedness. Trees are graphs with the absolute minimum amount of connectedness.

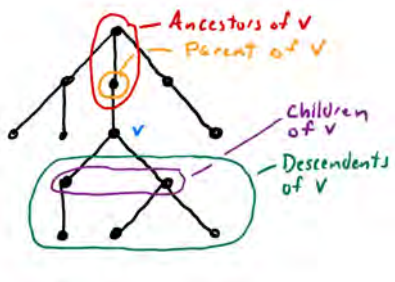
Theorem 23.3. *A connected graph with n vertices is a tree if and only if it has exactly $n-1$ edges.*

Theorem 23.4. *A graph is a tree if and only if there is a unique simple path between any two of its vertices.*

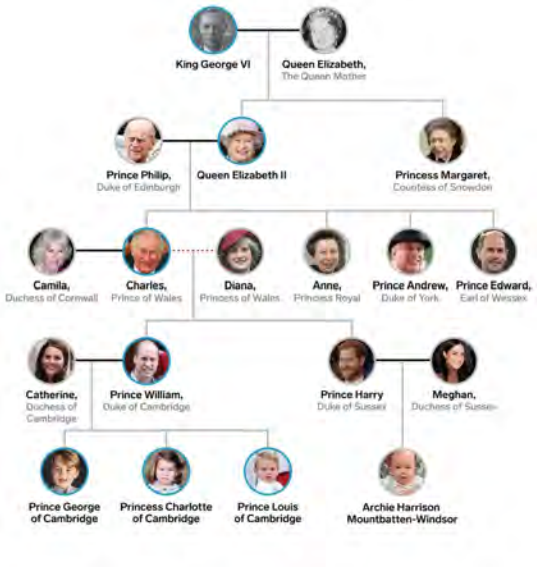
Definition 23.5. A *rooted tree* is a tree with a single vertex designated as a root. In a rooted tree, the vertices of degree 1 are called *leaves*. The vertices which are not leaves are *internal vertices*.



If v is a vertex other than the root in a rooted tree G, then there is a unique vertex u in the path from the root of G which is adjacent to v. The vertex u is the *parent* of v, and v is a *child* of u. Vertices with the same parent are called *siblings*. Any vertex in the unique path from the root of G to v is an *ancestor* of v. The *descendents* of v are those vertices which have v as an ancestor.

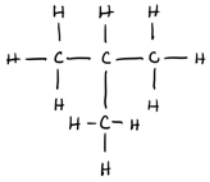


Genealogies. Family trees are actual trees for a few generations.



Are family trees actual trees? You have two biological parents. Each of them had two biological parents, and each of them had two biological parents. As we proceed backwards like this through your family, if everyone is distinct, then each generation doubles in size. Let us suppose that a generation takes twenty years. If we go back in time 1000 years, that is 50 generations. If all of your ancestors were distinct, this would require 1.13×10^{15} people to be alive around the year 1024. This is more than *four million times* the estimated population of the Earth in the year 1000. This implies that your family tree actually is not a tree.

Saturated Hydrocarbons. Graphs can be used to represent molecules. Vertices represent atoms, and edges represent bonds betweent he atoms. A compound of the form C_nH_{2n+2} is a saturated hydrocarbon. Each carbon atom C is a vertex of degree four. Each hydrogen atom H is a vertex of degree one. The nonisomorphic trees with n vertices of degree 4 and $2n + 2$ vertices of degree 1 represent the isomers of C_nH_{2n+2}. Here is a representation of an isomer of C_4H_{10}.

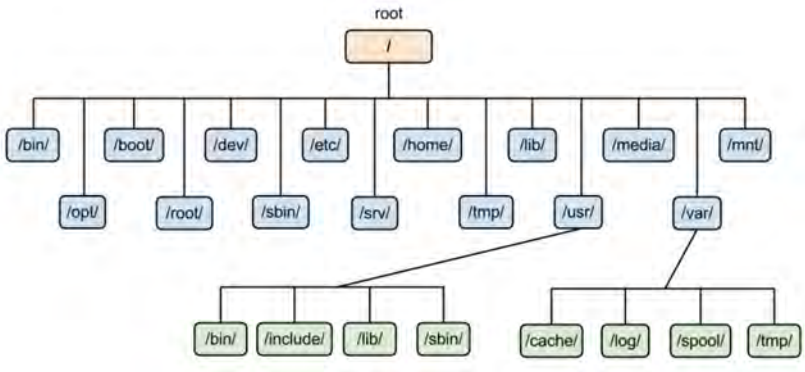


Minimum Path Trees. The subgraph H constructed in Dijkstra's Algorithm is a rooted tree so that the path from the root to any vertex has minimal weight. Since every vertex of the original graph G is in H, H is called a *spanning tree* of G.

Organizational Chart. Large organizations usually have a chain of command that forms a rooted tree. For example, here is an organizational chart for the Department of Justice (taken from their website).

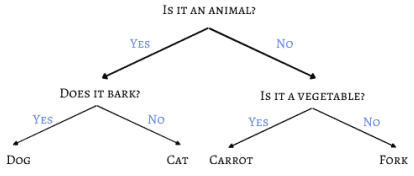
The DOJ often uses such charts to model criminal organizations.

File Systems. File systems can be modeled after trees. Here is a depiction of the standard Linux file system.



There have been some files systems proposed that are not based on trees, and you can link folders in some file systems to more than one location to mimic such a structure. However, tree-based files systems have benefits for searching for and storing data.

Decision Trees. Trees can be used to model the decision making process. In the game 20 Questions, one player thinks of an object - an animal, vegetable, or mineral. The other player then can ask up to 20 yes/no questions to determine what the object is. Treating the questions as vertices, we can draw a rooted tree to guide someone in asking the questions.



This type of tree could theoretically guide a player to $2^{20} = 1,048,576$ different guesses. In this tree, each internal vertex has two children. We give such a tree a name.

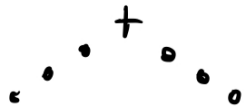
Definition 23.6. A *binary tree* is a rooted tree in which each internal vertex has at most two children. A *full binary tree* is a rooted tree in which each internal vertex has exactly two children.

Of course, there is not much special about two here. We can also have ternary trees or 4-ary trees. Or 184-ary trees.

Derivation Trees. A *parse tree* or *derivation tree* for an expression is a tree that describes how an expression in a language is derived from the grammar of that language. As an example, we show a derivation tree for an algebraic expression. Consider the algebraic expression

$$(3 \times (a + b)) + ((a \times b) + (a + b))$$

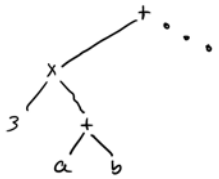
If we were to compute this expression, the last thing we would do is the sum in the middle, so we start our derivation tree off with a + and space for two children.



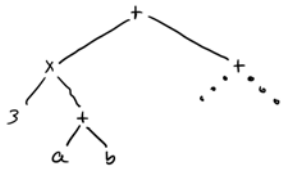
The left child will represent the expression in the left set of parentheses. This is a product, so we put a product as the left child. This product should have two children.

The left side of the product is just a 3, and the right side is a sum.

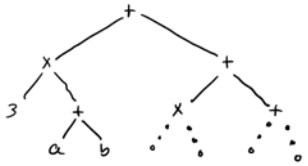
The children of the sum are a and b.



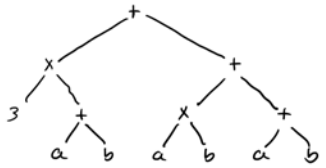
Now we work on the other branch of this tree. This operation at the top branch is the main operation in the right set of parentheses. This is a sum.



The arguments of the sum are a product and a sum.



The arguments to the product and the sum are a and b.



This is finally a derivation tree for our initial expression.

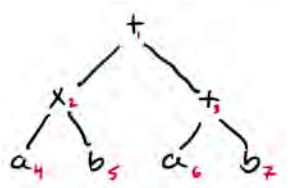
One reason we may want to draw a derivation tree for an expression is to demonstrate in a more complex language that an expression is grammatically correct or well-formed. For more complicated computations, a derivation tree could also help with scheduling for parallel processing. The tree we just drew has 6 operations. If we could perform the three additions near the bottom of the tree in parallel, and then the multiplication and addition in the middle in parallel, and then the addition at the top, then we could perform the operation in only 3 steps rather than 6, cutting the time to compute in half.

Polish Notation. There are at least three ways to convert the derivation tree for an expression back into an expression. These approaches input the tree for an expression and output a string representing the expression. All three approaches are recursive. To describe them, we will assume that all of our operations are binary.

Infix Notation: The first approach reads the left side of a tree, then the right side of the tree, and returns a string which has the left side followed by an operation symbol, followed by the right side. Since the operation is written between the operands, this is called infix notation. The function accepts as an argument a vertex of the derivation tree. If the vertex is a leaf, then the variable or number represented by that leaf is returned. Otherwise, the vertex represents an operation $*$ and has has a left child and a right child. A string $left$ is constructed to represent the left child. A string $right$ is constructed to represent the right child. Then the string $(left) * (right)$ is returned. In the code below, if v is a leaf, then $v.value$ is the number or variable that v represents. If v is an internal vertex, then $v.op$ is the operation that v represents, $v.left$ is the left child of v, and $v.right$ is the right child of v.

```
function infix(v) {
    If v is a leaf
        return(v.value);
    else
        return("("+infix(v.left)+")"+v.op+"("+infix(v.right)+")");
}
```

We illustrate running this function on the derivation tree below. The labels of our vertices have been subscripted to make it clear which vertex we are looking at.



We want to calculate the string $f(+_1)$. Since this vertex is not a leaf, we first calculate $f(\times_2)$ and then $f(+_3)$. For $f(\times_2)$, this vertex is not a leaf, so we need to calculate $f(a_4)$ and $f(b_5)$. Since a_4 and b_5 are leaves, $f(a_4)$ returns "a" and $f(b_5)$ returns "b". The call $f(\times_2)$ now returns "$(a) \times (b)$". (Note that the function adds the extra parentheses.) Now we go back to $f(+_3)$. Since $+_3$ is not a leaf, we nave to call f on the children $f(a_6)$ and $f(b_7)$. Since a_6 and b_7 are leaves, $f(a_6)$ returns "a" and $f(b_7)$ returns "b". The call $f(+_3)$ now returns "$(a) + (b)$". Now that $f(\times_2)$ and $f(+_3)$ have returned, $f(+_1)$ can combine their return values to return "$((a) \times (b)) + ((a) + (b))$". We would usually write this as, $(a \times b) + (a + b)$.

Prefix Notation: The second approach reads the left side of a tree and the right side of the tree, and returns a string which has the operation symbol followed by the left side of the tree followed by the right side of the tree. Since the operation is written before the operands, this is called prefix notation (or Polish notation since it was championed by the Polish mathematician Łukasiewicz). The function accepts as an argument a vertex of the derivation tree. If the vertex is a leaf, then the variable or number represented by that leaf is returned. Otherwise, the vertex represents an operation $*$ and has has a left child and a right child. A string $left$ is constructed to represent the left child. A string $right$ is constructed to represent the right child. Then the string $*\,left\,right$ is returned.

```
function prefix(v) {
    If v is a leaf
        return(v.value);
    else
        return(v.op+prefix(v.left)+prefix(v.right));
}
```

We illustrate running this function on the derivation tree above. We want to calculate the string $f(+_1)$. Since this vertex is not a leaf, we first calculate $f(\times_2)$ and then $f(+_3)$. For $f(\times_2)$, this vertex is not a leaf, so we need to calculate $f(a_4)$ and $f(b_5)$. Since a_4 and b_5 are leaves, $f(a_4)$ returns "a" and $f(b_5)$ returns "b". The call $f(\times_2)$ now returns "$\times ab$". Now we go back to $f(+_3)$. Since $+_3$ is not a leaf, we nave to call f on the children $f(a_6)$ and $f(b_7)$. Since a_6 and b_7 are leaves, $f(a_6)$ returns "a" and $f(b_7)$ returns "b". The call $f(+_3)$ now returns "$+ab$". Now that $f(\times_2)$ and $f(+_3)$ have returned, $f(+_1)$ can combine their return values to return "$+ \times ab + ab$". Notice how this notation does not need parentheses!

Postfix Notation: The third approach reads the left side of a tree and the right side of the tree, and returns a string which has the left side of the tree followed by the right side of the tree followed by the operation symbol. Since the operation is written after the operands, this is called postfix notation (or reverse Polish notation). The function accepts as an argument a vertex of the derivation tree. If the vertex is a leaf, then the variable or number represented by that leaf is returned. Otherwise, the vertex represents an operation $*$ and has has a left child and a right child. A string $left$ is constructed to represent the left child. A string $right$ is constructed to represent the right child. Then the string $left\,right\,*$ is returned.

```
function postfix(v) {
    If v is a leaf
        return(v.value);
    else
        return(postfix(v.left)+postfix(v.right) + v.op);
}
```

We illustrate running this function on the derivation tree above. We want to calculate the string $f(+_1)$. Since this vertex is not a leaf, we first calculate $f(\times_2)$ and then $f(+_3)$. For $f(\times_2)$, this vertex is not a leaf, so we need to calculate $f(a_4)$ and $f(b_5)$. Since a_4 and b_5 are leaves, $f(a_4)$ returns "a" and $f(b_5)$ returns "b". The call $f(\times_2)$ now returns "$ab\times$". Now we go back to $f(+_3)$. Since $+_3$ is not a leaf, we nave to call

f on the children $f(a_6)$ and $f(b_7)$. Since a_6 and b_7 are leaves, $f(a_6)$ returns "a" and $f(b_7)$ returns "b". The call $f(+_3)$ now returns "$ab+$". Now that $f(\times_2)$ and $f(+_3)$ have returned, $f(+_1)$ can combine their return values to return "$ab \times ab++$".

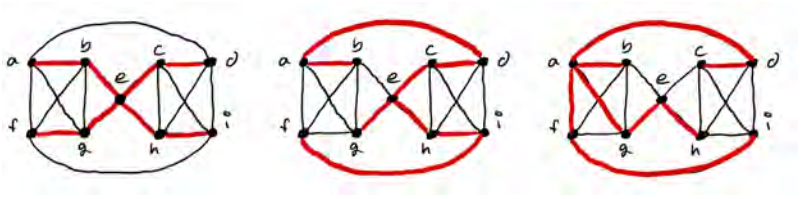
Advantages of (reverse) Polish Notation: Polish notation and reverse Polish notation have some advantages over infix notation.

- First, not all operations are binary. In the presence of unary or ternary operations, infix notation must be mixed with either prefix or postfix notation, so infix notation cannot usually be used exclusively.
- Prefix and postfix notation need not parentheses!
- It is easier to write code to parse prefix and postfix notation than infix notation using a stack.
- Prefix and postfix notation are more convenient for writing proofs about the language of algebraic expressions.

Spanning Trees.

Definition 23.7. A *spanning tree* of a graph G is a subgraph of G which contains all the vertices of G and which is a tree.

All connected graphs have spanning trees. Most have many. Below in red are highlighted multiple spanning trees of the same graph.

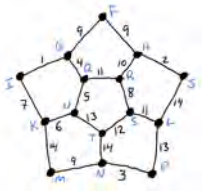


Dijkstra's Algorithm. The subgraph H built in Dijkstra's Algorithm is a spanning tree of the original graph G.

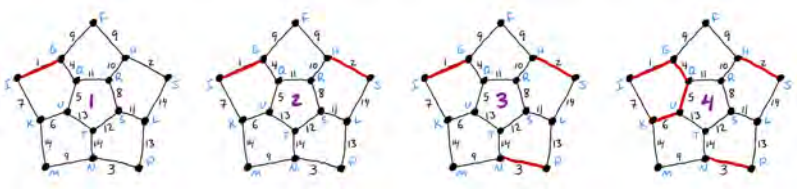
Kruskal's Algorithm. In a weighted graph, the objective is often to find a spanning tree which has the minimum total weight. The process for doing this is surprisingly simple. Essentially, we simply add vertices in the order of weight (from least to greatest) until we have connected all of the vertices. The one thing to avoid is adding an edge that creates a circuit. The algorithm below, known as Kruskal's Algorithm accomplishes this task.

- Let G be the input graph.
- Initialize U to be the empty set. This will be the set of used or discarded edges
- Initialize H as a subgraph of G. Let each vertex of G be a vertex of H. H initially has no edges.
- While H is not connected
  - Find a minimum weight edge e of G which is not in U.
  - If adding e to H will not create a circuit in H, add e to H.
  - Add e to U.

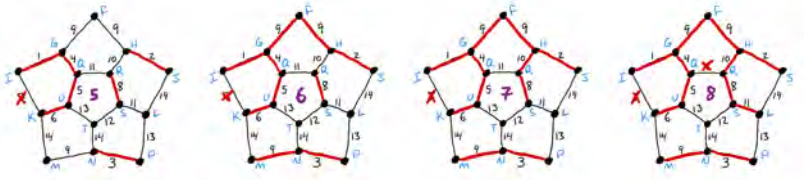
Example 23.8. Use Kruskal's Algorithm to find a minimum weight spanning tree of this graph.



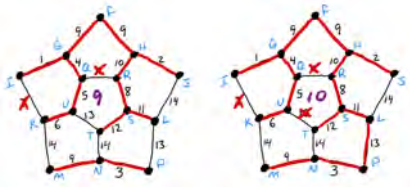
Solution: We illustrate this process below. As we construct the subgraph H, we highlight the edges of H on the graph in red. The steps are numbered in the center of the graph in purple. First, there is one edge of weight 1, so we add that to H in step 1. Next, there is one edge of length 2, so we add that to H in step 2. Next, there is one edge of length 3, so we add that in step 3. The same continues for weights 4, 5, and 6, so we add those in step 4.



After adding the edge of weight 6, there is one edge of weight 7, the one between I and K. Adding this edge to H would create a circuit, so we skip that edge and add the one edge of weight 8 in step 5. We cross out the edge from I to K to remind ourselves that we have skipped that edge. There are three edges of weight 9. Adding them does not create any circuits, so we add all three in step 6. Then we add the one edge of weight 10 in step 7. There are two edges of weight 11. Adding the one from S to L does not create a circuit, so we add it. However, the one between Q and R would create a circuit, so we do not add it.



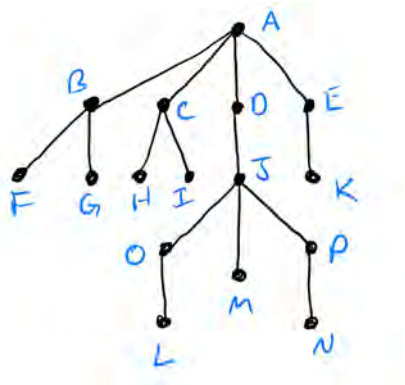
There is one edge of weight 12, so we add it without creating a circuit in step 9. Finally, there are two edges of weight 13. The one between U and T would create a circuit, so we cannot use it. The one between L and P is safe to add.



At this point, the subgraph H (with red edges) is connected, so we can stop.

Exercises 23.9. Answer the quetions below, please.

1. Draw two different polymers of C_6H_{14}.
2. Consider this rooted tree.



(a) What is the root?

(b) What are the leaves?

(c) What are the ancestors of J?

(d) What are the descendents of J?

3. Draw a derivation tree for the expression $(2 \times a) + b$

4. Draw a derivation tree for the expression $b + (a \times ((a + b) \times (b + c)))$

5. Write this expression in prefix notation: $(2 \times a) + b$

6. Write this expression in postfix notation: $(2 \times a) + b$

7. Write this expression in prefix notation: $(2 \times a) + (3 \times b)$

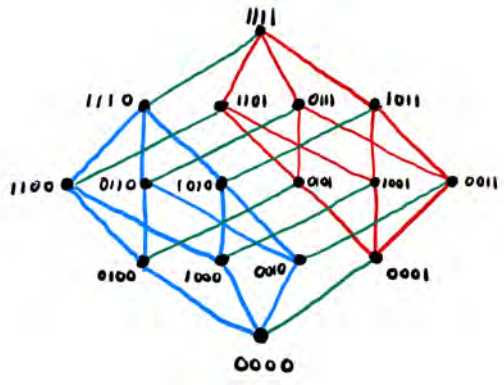
8. Write this expression in postfix notation: $(2 \times a) + (3 \times b)$

9. Write this postfix expression in infix notation: $2ab + \times 3ab \times \times +$

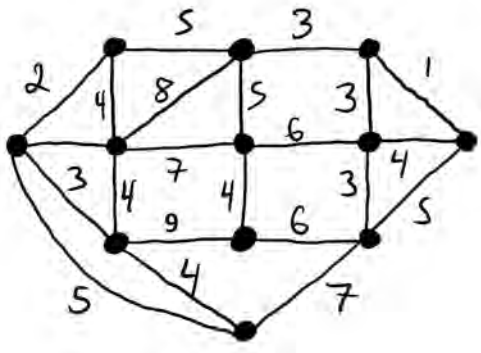
10. Write this prefix expression in infix notation: $+3 + \times aa \times 2a$

11. Find a spanning tree for Q_3. Shade your answer in the graph.

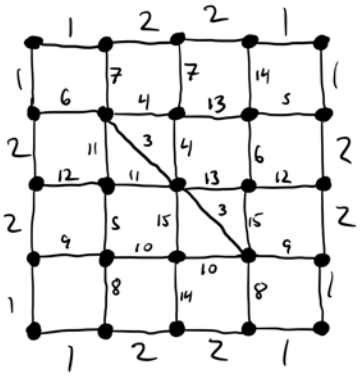
12. Find a spanning tree for Q_4. Shade your answer in the graph.



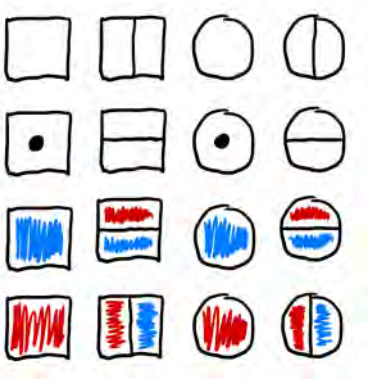
13. The graph below depicts several small towns as vertices. Edges are roads between the towns. Weights on the edges are costs in millions of dollars to pave the raods. Which roads should be paved so that all of the towns are connected by pavement, but the least money is spent? Shade your answer in the graph.



14. Find a minimum weight spanning tree in this graph. Shade your answer in the graph.



15. Consider the shapes below. In a game, the first player selects a shape and keeps it secret. The second player can then ask the first player yes/no questions to determine which shape was selected. Draw a decision tree to help the second player determine the shape in as few questions as possible. This will be a binary tree. Each vertex should be a question. Follow the left branch on answers of yes and the right on answers of no.



24 Basic Counting

Multiplication Rule:. Suppose that one procedure can end in m outcomes and that for each of these outcomes a second procedure can end in n outcomes. The two procedures combined can end in a total of mn outcomes.

Example 24.1. The chairs in a stadium are labeled with capital letter followed by a positive integer less than 100. How many such labels are there?

Solution: We can treat this as two procedures – first picking the letter, and then picking the integer. There are 26 letters to choose from, and there are 99 positive integers less than 100. According to the Multiplication Rule, there are $26 \cdot 99 = 2574$ possible chair labels.

The multiplication rule can be extended to more than two procedures.

Example 24.2. Suppose that license plates in a certain state consist of two nonzero digits followed by four letters. If all letter cominations are allowed (rather than disallowing offensive four-letter words like MATH), then how many such license plates are there?

Solution: There are nine nonzero digits - 1, 2, 3, 4, 5, 6, 7, 8, 9 - so there are 9 ways of choosing each digit. There are 26 letters, so there are 26 ways of choosing each letter. The total number of license plates is

$$9 \cdot 9 \cdot 26 \cdot 26 \cdot 26 \cdot 26 = 37015056.$$

Example 24.3. How many bitstrings are there of length 5?

Solution: There are two choices (0 or 1) for each bit in the bitstring. According to the Multiplication Rule, there are $2^5 = 32$ length 5 bitstrings.

Example 24.4. How many functions are there from a set of 4 elements into a set of 8 elements?

Solution: Suppose that $A = \{1, 2, 3, 4\}$ and that $|B| = 8$. We want to count the functions $f : A \to B$. There are 8 possible values for $f(1)$. There are 8 possible values for $f(2)$. There are 8 possible values for $f(3)$. There are 8 possible values for $f(4)$. By the multiplication rule, there are $8^4 = 4096$ possibilities for f.

Example 24.5. How many injective functions are there from a set of 4 elements into a set of 8 elements?

Solution: Suppose that $A = \{1, 2, 3, 4\}$ and that $|B| = 8$. We want to count the injective functions $f : A \to B$. There are 8 possible values for $f(1)$. Suppose that $f(1)$ is selected. Since f is supposed to be injective, the value of $f(1)$ cannot be equal to any other values of f. Therefore, there are only 7 possible values for $f(2)$. Now that we have already used two values from B, to keep f injective, we have only 6 possible values for $f(3)$. Finally, there are only 5 values left over for $f(4)$. By the multiplication rule, there are $8 \cdot 7 \cdot 6 \cdot 5 = 1680$ injective functions from A to B.

Example 24.6. How many subsets does an n element set have?

Solution: For each of the n elements, there are two options - either the element is in the subset, or it is not. Therefore, the Multiplication Rule says that there should be $2 \cdot 2 \cdots 2 = 2^n$ possible subsets.

The Sum Rule:. If A and B are finite sets, then $|A \cup B| = |A| + |B| - |A \cap B|$. If $A \cap B = \emptyset$ then $|A \cup B| = |A| + |B|$.

Note. You might think here that $|A \cup B|$ should be $|A| + |B|$. However, if we just add the sizes of A and B together, then we are counting the elements of $A \cap B$ twice - once when we count A and once when we count B. Many books call this the *inclusion-exclusion principle*.

Example 24.7. Suppose that a certain code consists of at most two characters. The characters must either be digits or uppercase letters. The first character must be a letter. How many such codes are there?

Solution: Any code which is at most two characters must be either one character or two characters. We apply the Sum Rule by counting the one-character codes and the two-character codes and then adding. Since every code must begin with a letter, a one-character code is just a letter. There are 26 of these. For two-character codes, the first character must be a letter (of which there are 26), but the second can be a letter or a digit (of which there are $26 + 10 = 36$ – which is another application of the Sum Rule). Therefore, there are $26 \cdot 36 = 936$ two-character codes. Since there are 26 one-character codes and 936 two-character codes, there are a total of $26 + 936 = 962$ codes.

Example 24.8. Suppose that a password must be exactly 8 characters long, that each character must be either an uppercase letter, a lowercase letter, or a digit, and that at least one character must be a digit. How many such passwords are there?

Solution: Counting problems involving "at least one" are usually difficult to count directly. Instead, we usually use a trick. We will count all of the strings involved in the problem, and we will count all of the strings that contain no digits. Then, we will subtract. The key is that each string either contains at least one digit or it contains no digits. Let A be the set of all 8-character strings consisting of uppercase letters, lowercase letters, and digits. Let B be the set of all such strings which contain at least one digit. Let C be the set of all such strings which contain no digits. Then $A = B \cup C$ and $B \cap C = 0$. By the Sum Rule, $|A| = |B| + |C|$. We are interested in $|B|$, so solving gives $|B| = |A| - |C|$. To count the elements of A, we can select one digit at a time. There are 26 lowercase letters, 26 uppercase letters, and 10 digits, for a total of 62 possibilities for each character. That gives $|A| = 62^8 = 218340105584896$. To count elements of C, we also select one character at a time. However, since elements of C contain no digits, there are only $26 + 26 = 52$ possibilities for each character. This gives $|A| = 52^8 = 53459728531456$. Now

$$|B| = |A| - |C| = 62^8 - 52^8 = 164880377053440.$$

Example 24.9. How many length 10 bitstrings are there that begin with 1 or end with 1?

Solution: Let A be the set of all length 10 bitstrings that begin with 1. Let B be the set of all length 10 bitstrings that end with 1. We want $|A \cup B|$. By the Sum Rule, this is $|A| + |B| - |A \cap B|$. The set $A \cap B$ is the set of length 10 bitstrings which begin with 1 and end with 1.

Any bitstring in A begins with 1 and is followed by 9 bits. For each of these 9 bits, there are only two options – 0 or 1 – so there are $2^9 = 512$ bitstrings in A.

Any bitstring in B is a 1 preceded by 9 bits. For each of these 9 bits, there are two options – 0 or 1 – so there are $2^9 = 512$ bitstrings in B.

Any bitstring in $A \cap B$ consists of two 1s with 8 bits in between. There are $2^8 = 256$ of these. Therefore

$$|A \cup B| = |A| + |B| - |A \cap B| = 512 + 512 - 256 = 768.$$

Example 24.10. Among 25 students, 15 took calculus, 19 took discrete, and 2 took neither. How many took calculus and discrete?

Solution: Let C be the set of students who took calculus, and let D be the set of students who took Discrete. We want $|C \cap D|$. We know that $|C \cup D| = 25 - 2 = 23$. We also know from the Sum Rule that

$$|C \cup D| = |C| + |D| - |C \cap D|.$$

Plugging in the numbers we know gives

$$23 = 15 + 19 - |C \cap D|.$$

We can now solve for $|C \cap D|$ to get

$$|C \cap D| = 15 + 19 - 23 = 11.$$

Eleven students took both classes.

Example 24.11. Among 20 students, 16 took discrete, and 8 took knitting. What are the largest and smallest that the number of students who took both classes might be?

Solution: Let D be the set of students who took discrete. Let K be the set of students who took knitting. We want to know the largest and smallest that $D \cap K$ can be. The reason that $D \cap K$ is unknown in this example (compared to the last) is that we do not have a way to determing $D \cup K$. $D \cap K$ is at its largest if one of the sets is actually a subset of the other. If $K \subseteq D$, then $|D \cap K| = |K| = 8$. in this case, among the 20 students, 16 took discrete, and 8 of those 16 took knitting. That leaves 4 students who took neither. To discover the smallest that $D \cap K$ can be, consider the sum rule

$$|D \cup K| = |D| + |K| - |D \cap K|.$$

If we solve for $|D \cap K|$, we get

$$|D \cap K| = |D| + |K| - |D \cup K|.$$

To make $|D \cap K|$ as small as possible, we make $|D \cup K|$ as large as possible. Since there are only 20 students, the largest $|D \cup K|$ might be is 20, so the smallest value of $|D \cap K|$ is

$$|D \cap K| = 16 + 8 - 20 = 4.$$

The number of students who took both classes is somewhere between 4 and 8, inclusive.

Inclusion-Exclusion with Three Sets. The Sum Rule can be extended to three sets.

$$\begin{aligned}
|A \cup B \cup C| &= |A \cup (B \cup C)| \\
&= |A| + |B \cup C| - |A \cap (B \cup C)| \\
&= |A| + |B \cup C| - |(A \cap B) \cup (A \cap C)| \\
&= |A| + |B \cup C| - (|(A \cap B)| + |(A \cap C)| - |(A \cap B) \cap (A \cap C)|) \\
&= |A| + |B \cup C| - (|(A \cap B)| + |(A \cap C)| - |A \cap B \cap C|) \\
&= |A| + |B \cup C| - |(A \cap B)| - |(A \cap C)| + |A \cap B \cap C| \\
&= |A| + |B| + |C| - |B \cap C| - |(A \cap B)| - |(A \cap C)| + |A \cap B \cap C| \\
&= |A| + |B| + |C| - |(A \cap B)| - |(A \cap C)| - |B \cap C| + |A \cap B \cap C|
\end{aligned}$$

The way to think of this is that we should add $|A| + |B| + |C|$, but this counts the intersections too many times, so we subtract off the pairwise intersections. However, $|A| + |B| + |C|$ counts $|A \cap B \cap C|$ three times, and subtracting off the pairwise intersections subtracts $|A \cap B \cap C|$ three times. Therefore, we have to add it back at the end.

Example 24.12. Among 40 students, 16 took calculus, 19 took discrete, and 16 took Greek. Seven students took both calculus and discrete. Five took both calculus and Greek, and 6 took both discrete and Greek. Five students took none of the classes. How many took all three?

Solution: Let C be the set of students who took calculus, D the set of students who took discrete, and G the number of students that took Greek. We want $|C \cap D \cap G|$. We know:

$$\begin{array}{ll} |C| = 16 & |C \cap D| = 7 \\ |D| = 19 & |C \cap G| = 5 \\ |G| = 16 & |D \cap G| = 6 \end{array}$$

Additionally, we know that $|C \cup D \cup G| = 40 - 5 = 35$. Plugging into the Sum Rule equation for three sets gives:

$$35 = 16 + 19 + 16 - 7 - 5 - 6 + |C \cap D \cap G|.$$

Solving for $|C \cap D \cap G|$ gives

$$|C \cap D \cap G| = 35 - (16 + 19 + 16 - 7 - 5 - 6) = 2.$$

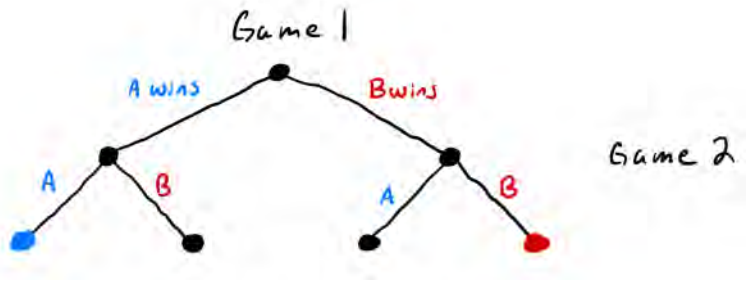
Two students took all three classes.

We can sometimes draw trees to help with counting.

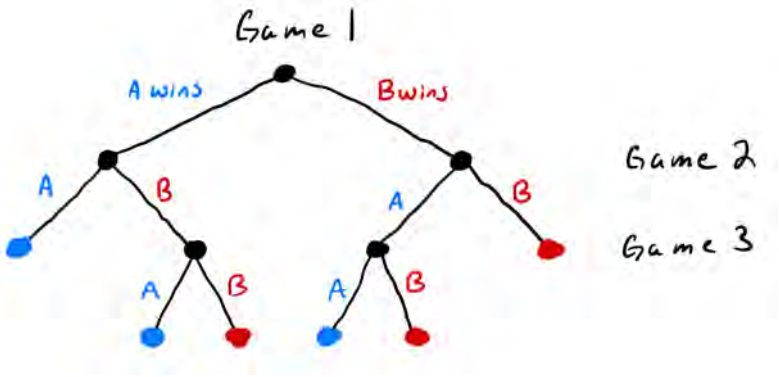
Example 24.13. Suppose that two teams, A and B, are playing in a three game series. The first team to win two games wins the series. How many different ways could the series play out?

Solution: First, we draw a vertex (representing Game 1) which is the root of our tree. This vertex has two edges branching off of it – one representing a win by A and one representing a win by B.

The children of the first vertex represent possible Game 2s. Each of these might be won by A or by B, so we add branches to each.



Now, the left-most branch indicates a situation in which team A has won two games. In this case, A wins the series. Similarly, the right-most branch indicates a situation in which team B has won two games. In this case, B wins the series. These two branches are done growing. However, in the middle two branches, we still do not have a winner, so the two middle vertices represent possible Game 3s. We add branches to these to indicate the possible winners.



At this point, a team has one two games along any branch, and the series is over. The number of possible series is the number of leaves - 6. Then number of ways that A can with is 3. The number of ways that B can win is 3. The number of series that end after two games is 2. The number of series that end after three games is 4. Notice that the team to win the last game is the team that wins the series.

The Pigeonhole Principle. If $k+1$ objects are placed into k boxes, then at least one box contains at least two objects.

The Generalized Pigeonhole Principle. If n objects are placed into k boxes, then at least one box contains at least $\lceil n/k \rceil$ objects.

Example 24.14. Among 100 people, at least how many were born in the same month?

Solution: We are placing the 100 people in 12 "boxes" or months. By the Generalized Pigeonhole Principle, at least one box contains $\lceil 100/12 \rceil = \lceil 10.1\overline{66} \rceil = 11$ people. So at least 11 people were born in the same month.

Example 24.15. What is the least number of students necessary in a class to guarantee that at least 4 of them have the same grade from A, B, C, D, F?

Solution: Let n be the number of students in a class. Since there are five grade categories, we need for $\lceil n/5 \rceil = 4$. Consider multiples of 5. Note that $\lceil 15/5 \rceil = \lceil 3 \rceil = 3$ and $\lceil 20/5 \rceil = \lceil 4 \rceil = 4$, so 15 students is too few, but 20 is (probably more than) enough. We raise 15 by 1 and try 16 to see that $\lceil 16/5 \rceil = \lceil 3.2 \rceil = 4$. Therefore, among 16 students at least 5 have the same grade.

Exercises 24.16. Please solve the following problems.

1. A code consists of two distinct digits followed by two distinct capital letters followed by four characters that can each be any digit or any capital letter. How many such codes are there?
2. How many length five strings are there over the alphabet $\{a, b, c, d, e, f\}$?
3. How many length five strings are there over the alphabet $\{a, b, c, d, e, f\}$ which have no repeated characters?
4. How many length five strings are there over the alphabet $\{a, b, c, d, e, f\}$ which have at least one repeated character?
5. How many length five strings are there over the alphabet $\{a, b, c, d, e, f\}$ which do not contain a?
6. How many length five strings are there over the alphabet $\{a, b, c, d, e, f\}$ which do contain a?
7. How many length 10 bitstrings are there which begin with 11 or end with 00?
8. Among 20 students, 10 took calculus, 10 took discrete, and 10 took algebra. Five took calculus and discrete. Five took calculus and algebra, and five took algebra and discrete. If 3 students took none of the classes, how many students took all three classes?
9. Among 20 students, 15 took calculus and 12 took discrete. What is the largest number that might have taken both classes?
10. Among 20 students, 15 took calculus and 12 took discrete. What is the smallest number that might have taken both classes?
11. Draw a tree showing the possible outcomes of a best of 5 games series.
12. Among a class of 30 students, at least how many were born on the same day of the week?
13. A certain type of beetle can be red, purple, or green. How many beetles have to be caught to guarantee you have at least 10 of the same color?
14. How many four-digit pin numbers are there which have at least one repeated digit?

25 Permutations and Combinations

Definition 25.1. An ordered arrangement of the elements of a set is a **permutation** of the elements. An ordered arrangement of r elements from a set is an r-**permutation**. The symbols $P(n,r)$ and ${}_nP_r$ represent the number of r-permutations of an n element set.

Example 25.2. How many 3-permutations are there of the letters A, B, C, D, E, F, G?

Solution: We will use the Multiplication Rule to answer this. We have three procedures here – selecting the first, second, and third letters. There are seven choices for the first letter. Once that letter has been chosen, there are six choices for the second letter. After two have been chosen, there are five choices left for the third letter. Therefore, the number of 3-permutations is $7 \cdot 6 \cdot 5 = 210$.

Note. The product $7 \cdot 6 \cdot 5$ looks like the beginning of 7!. In fact, it happens to be that

$$7 \cdot 6 \cdot 5 = \frac{7 \cdot 6 \cdot 5 \cdot 4 \cdot 3 \cdot 2 \cdot 1}{4 \cdot 3 \cdot 2 \cdot 1} = \frac{7!}{4!} = \frac{7!}{(7-3)!}.$$

Example 25.3. How many permutations are there of the letters A, B, C, D, E, F, G?

Solution: We can proceed here just like we did in the last example. However, this time we need to select 7 numbers. The count is $7 \cdot 6 \cdot 5 \cdot 4 \cdot 3 \cdot 2 \cdot 1 = 7!$ or 5040.

The process we used to solve the previous examples hints at this theorem.

Theorem 25.4. *There are $P(n,n) = n!$ permutations of n objects and $P(n,r) = \dfrac{n!}{(n-r)!}$ r-permutations of n objects.*

Example 25.5. How many permutations of A, B, C, D, E, F contain the string AB?

Solution: The trick to this problem is to treat AB as a single character. Therefore, we are looking at permutations of AB, C, D, E, F – which is five characters. There are $5! = 120$ such permutations.

Example 25.6. How many permutations of A, B, C, D, E, F contain the string AB or the string BA?

Solution: Here, we use the Sum Rule. No permutation can contain both AB and BA, so we merely count those that contain AB (there are 120 of these by the last example) and those that contain BA (there are 120 these by mimicing the last example). There should be a total of $120 + 120 = 240$ permutations containing AB or BA.

Example 25.7. How many permutations of A, B, C, D, E, F contain the string AB or the string CD?

Solution: This problem is different from the last because a permutation might contain both AB and CD. Let X be the set of all permutations containing AB, and let Y be the set of all permutations containing CD. We want $|X \cup Y| = |X| + |Y| - |X \cap Y|$. We know that $|X| = 120$ from the previous examples. It is also the case that $|Y| = 120$ for similar reasons. We need only calculate $|X \cap Y|$. This is the number of permutations which contain both AB and CD. For this, we treat AB and CD as characters, so that we are looking for permuations of AB, CD, E, F. There are $4! = 24$ of these. Therefore, the number of permutation containing AB or CD is $120 + 120 - 24 = 216$.

Definition 25.8. An r-**combination** of a set of n elements is an r-element subset of the n element set. This can be thought of an an unordered selection of elements from the set. The number of r-combinations of an n element set is denoted $C(n,r)$ or ${}_nC_r$ or $\begin{pmatrix} n \\ r \end{pmatrix}$. This is read as "$n$ choose r."

Example 25.9. How many 3-element subsets are there of the 5-element set $\{A, B, C, D, E\}$? (That is, find $C(5,3)$.)

Solution: We first try to work this problem like we did the permutation problems. We can select the first character, then the second, and then the third. There are $P(5,3) = \dfrac{5!}{2!}$ ways to do this. However, this counts some subsets too many times. The selections ABC, ACB, BAC, BCA, CAB, CBA all give the same set. We must divide our count by 3! to avoid counting every rearrangement of our set separately. Therefore, $C(5,3) = P(5,3) \div 3! = \dfrac{5!}{2!} \div 3! = \dfrac{5!}{3!2!} = 10$.

This example hints at the following theorem.

Theorem 25.10. *The number of r-combinations of an n element set is* $C(n,r) = \dfrac{n!}{r!(n-r)!}$.

Example 25.11. How many committees of three people can be formed in a department of ten people?

Solution: We simply need to select 3 people from the ten. The order we select them does not matter, just the group of three. Therefore, the number of committees is

$$C(10,3) = \frac{10!}{3!7!} = \frac{10 \cdot 9 \cdot 8 \cdot 7!}{3!7!} = \frac{10 \cdot 9 \cdot 8}{3 \cdot 2 \cdot 1} = 120.$$

Example 25.12. How many ways can we select a president, a secretary, and a treasurer from ten people?

Solution: We select the people one at a time – president, secretary, treasurer. What makes this different from the last question is that the order we select them matters. Therefore, we want $P(10,3) = 720$.

Example 25.13. How many bitstrings of length 10 contain exactly 3 1s?

Solution: To form a bitstring of length 10 with exactly 3 1s, we simply need to select the three places to put the 1s and put 0s everywhere else. Therefore, the number of length 10 bitstrings with exactly 3 1s is

$$C(10,3) = \frac{10!}{3!7!} = \frac{10 \cdot 9 \cdot 8 \cdot 7!}{3!7!} = \frac{10 \cdot 9 \cdot 8}{3 \cdot 2 \cdot 1} = 120.$$

Example 25.14. A math department with 6 members and a computer science department with 4 members are going to create a committee of 4 to design a new class. If the committee must contain 2 people from each department, then how many such committees are there?

Solution: We select the 2 math people and then the two computer science people and apply the Multiplication Rule.

$$\begin{aligned} \#\text{committees} &= (\#\text{ways to select the math}) \times (\#\text{ways to select the CS}) \\ &= C(6,2) \times C(4,2) \\ &= 15 \times 6 \\ &= 90. \end{aligned}$$

There are 90 ways to form the committee.

Example 25.15. How many ways can we rearrange the letters in MISSISSIPPI?

Solution: What makes this problem different from earlier problems is that some of the letters are repeated – we have some *indistinguishable objects*. We will solve this problem twice to see two different philosophical approaches. We are arranging letters into 11 places. All we have to do is

1. Select 4 places of the 11 for the Ss.
2. Select 4 places of the remaining 7 for the Is.
3. Select 2 places of the remaining 3 for the Ps.

4. Select 1 place of the remaining 1 for the M.

The number of ways to do this is:

$$\begin{aligned} C(11,4) \times C(7,4) \times C(3,2) \times C(1,1) &= \frac{11!}{4!7!} \times \frac{7!}{4!3!} \times \frac{3!}{2!1!} \times 1 \\ &= \frac{11!}{4!4!2!} \\ &= 34650. \end{aligned}$$

Our second approach will give more meaning to the fraction in our last step. Thee are 11! ways to rearrange 11 letters. However, since we have 4 Ss, we divide by 4! so that we do not count all of the rearrangements of the Ss separately. Similarly, we divide by 4! to identify the Is with each other, and we divide by 2! to identify the Ps.

Permutations with Repetition. In Example 24.3 we counted bitstrings of length 5. We would like to revisit that example. In constructing a bitstring of length 5, we can imagine that we are selecting 5 objects in order from the set $\{0, 1\}$. Some books would call this a 5-permutation with repetitions allowed of a 2-element set. We use the word permutation here because order matters. Repetition must be allowed because there are only 2 elements to pick from. In general, then number of r-permutations with repetition allowed selected from n elements is just n^r. These are simply the strings of length r over the alphabet consisting of the n objects.

Example 25.16. Sam is painting Easter eggs. He has 12 eggs, and he has five colors: red, orange, yellow, green, and blue. If each egg is painted one color and if all of the eggs are considered identical, in how many ways can he paint the eggs?

Solution: Sam is going to make 12 choices from the colors red, orange, yellow, green, and blue. The choices must allow repetition since he has fewer colors than the number of choices. The order of the choices does not matter since, for example, painting one egg red and then one blue is the same as painting one blue and then one red. To decide how to do this, we draw a diagram. The categories corresponding to colors below are separated by vertical bars.

red	orange	yellow	green	blue
|	|	|	|	

We illustrate a selection of colors for all 12 eggs by placing the eggs in the color categories. We represent each egg with a $*$. This selection represents 4 red, 4 orange, and 4 yellow.

red	orange	yellow	green	blue
$****$ |	$****$ |	$****$ |	|	

This selection represents 2 red, 2 orange, 2 yellow, 3 green, and 3 blue.

red	orange	yellow	green	blue
$**$ |	$**$ |	$**$ |	$***$ |	$***$

Every possible selection here is merely a rearrangement of 12 $*$s and 4 |s. There are

$$\frac{(12+4)!}{12!4!} = \frac{16!}{12!4!} = C(16, 12) = 1820.$$

There are 1820 ways for Sam to paint the eggs.

Stars and Bars. In this last example, we are selecting 12 colors with repetition with no regard for order from a set of 5 colors. This is called a 12-combination with repetition selected from a 5-element set. Some books will use the word *multiset* to describe such a combination. To count combinations with repetition, we usually use the process above with $*$s and |s. The name of this approach is *stars and bars*. The number of $*$s

is the number of selections being made. The number of |s is one less than the number of categories or bins or objects from which we are selecting. If we are selecting an r-combination with repetition from n objects, the number of $*$s is r, and the number of |s is $n-1$. The number of rearrangments of these $*$s and |s is

$$\frac{r+(n-1)}{r!(n-1)!} = C(r+(n-1), r).$$

Example 25.17. How many nonnegative integer solutions are there to the equation $x_1+x_2+x_3+x_4 = 10$?

Solution: The secret to questions such as this is to think of the variables x_1, x_2, x_3, and x_4 as categories or bins in which to place 10 $*$s. The four bins will be separated by 3 |s, so we are arranging 10 $*$s and 3 |s. The number of ways to do this is

$$\frac{(10+3)!}{10!3!} = \frac{13!}{10!3!} = C(13,3) = 286.$$

Incidentally, you could call what we are counting here 10-combinations with repetition of 4 objects.

Example 25.18. How many positive integer solutions are there to the equation $x_1 + x_2 + x_3 + x_4 = 10$?

Solution: The difference here is that each of the variables must be at least 1. We can have no empty bins as might have happened in the last problem. Again, we start with 10 $*$s and 4 bins or 3|s. Since every bin must contain at least one $*$, we set aside 4 $*$s which will be placed into the bins at the end. Then we distribute the remaining 6 $*$s in the 4 bins without having to think about whether or not the bins are empty. At the end, we add one of the 4 $*$s we set aside to each bin. Therefore, we only need to count how to place 6 $*$s in 4 bins separated by 3 |s. The number of ways to do this is

$$\frac{(6+4)!}{6!4!} = \frac{10!}{6!4!} = C(10,6) = 210.$$

Example 25.19. How many ways can 11 identical books be distributed among 4 shelves?

Solution: We have 11 objects or $*$s to be placed in 4 bins. The bins would be separated by 3 dividers or |s. The number of ways to do this is

$$\frac{(11+3)!}{11!3!} = \frac{14!}{11!3!} = C(14,11) = 364.$$

Example 25.20. How many ways can 11 different books be arranged on 4 shelves?

Solution: We do this through two procedures. First, we decide how to distribute the books on the shelves, then we decide which order to put the books in. There are 364 ways to select the shelves by the previous example. There are 11! ways to arrange the books, so the number of ways to arrange the books on the shelves is $364 \times 11! = 14,529,715,200$.

Exercises 25.21. Please complete the following exercises.

1. How many permutations are there of the digits 0, 1, 2, 3, 4, 5, 6, 7, 8, 9?
2. How many permutations are there of the digits 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 that do not contain the pair 12?
3. How many permutations are there of the digits 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 that contain the pair 12 or the pair 23? (Be careful.)
4. How many 5-permutations are there of the digits 0, 1, 2, 3, 4, 5, 6, 7, 8, 9?
5. How many 5-permutations are there of the digits 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 which contain the digit 1?

6. How many 5-permutations are there of the digits 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 which contain the pair 12?

7. How many 5-permutations are there of the digits 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 which do not contain the pair 12?

8. Five friends Alice, Bob, Chuck, Dave, and Eve are going to the movies. How many ways can they sit on one row if Alice and Bob must sit next to each other (in either order)?

9. Five friends Alice, Bob, Chuck, Dave, and Eve are going to the movies. How many ways can they sit on one row if Alice and Bob cannot sit next to each other?

10. There are not enough chairs in the classroom for all 20 discrete students, so 3 will have to stand. How many different ways can we select the 3 who have to stand?

11. There are not enough chairs in the classroom for all 20 discrete students, so 3 will have to stand. How many different ways can we select the 3 who have to stand if Alice or Bob either have to both be sitting or both be standing?

12. There are not enough chairs in the classroom for all 20 discrete students, so 3 will have to stand. How many different ways can we select the 3 who have to stand if exactly one of Alice or Bob must be standing?

13. How many bitstrings of length 10 contain five 1s and five 0s?

14. How many ways can we rearrange the letters TEXAS?

15. How many ways can we rearrange the letters CONNECTICUT?

16. How many ways can we place 10 students into 3 rooms – Room 1, Room 2, and Room 3 – if it does not matter who is in each group, just how many are in each group?

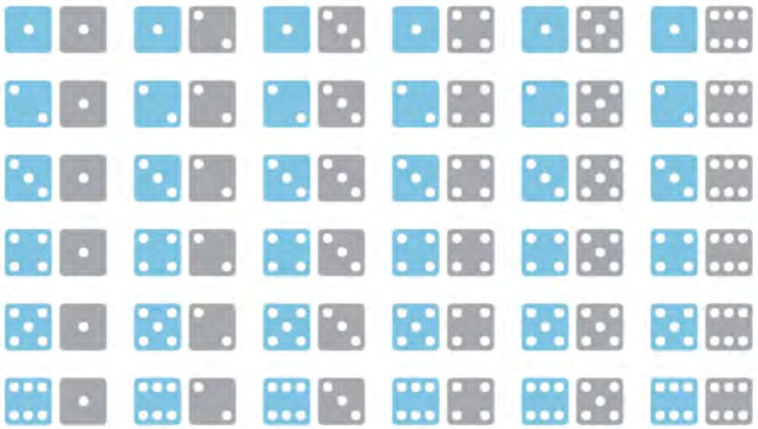
17. How many solutions are there to $x_1 + x_2 + x_3 = 12$ using positive integers?

26 Basic Probability

Events. The *sample space* of a repeatable process, procedure, or experiment is the set of all possible outcomes of the process. An *event* is a subset of the sample space. This is just a set of outcomes of the process. A *simple event* is composed of a single outcome. A *compound event* is a union of multiple simple events.

Example 26.1. Supose that a six-sided die is rolled. The possible outcomes are the numbers 1, 2, 3, 4, 5, and 6. The sample space is $\{1, 2, 3, 4, 5, 6\}$. A compound event is the event that we roll an even number. This is the event $\{2, 4, 6\}$.

Example 26.2. Suppose that two dice are rolled. The possible outcomes are:



Some compound events are:

- The event that the two numbers rolled are equal.
- The event that the two numbers rolled add to 8.
- The event that the first number is less than the second.

Example 26.3. Suppose that a coin is flipped. The possible outcomes are flipping a head (which we denote H) or flipping a tail (which we denote T). The sample space is $\{H, T\}$.

Example 26.4. Suppose that two coins are flipped. The sample space is $\{HH, HT, TH, TT\}$. Some compound events are:

- The event that the flips are the same.
- The event that the flips are different.
- The even that there is at least one H.

Example 26.5. Suppose that the experiment we are doing is that we guess your ATM pin number. There are two outcomes in the sample space. Either we guess correctly, or we guess incorrectly.

Probability Assumptions. Every event can be assigned a *probability*. For any event A in a sample space S, we denote the probability of A as $P(A)$. We make these assumptions about probabilities:

- $0 \leq P(A) \leq 1$.
- $P(\emptyset) = 0$.
- $P(S) = 1$.

- If $A \cap B = \emptyset$ then $P(A \text{ or } B) = P(A \cup B) = P(A) + P(B)$.

We associate the word *likely* with probabilities greater than $1/2$ and the word *unlikely* with probabilities less than $1/2$.

Suppose that S is a sample space with n simple events, $S = \{A_1, A_2, \ldots, A_n\}$, that are all equally likely. This implies that $P(A_1) = P(A_2) = \cdots = P(A_n)$. Since the simple events do not overlap,

$$1 = P(S) = P(A_1 \cup A_2 \cup \cdots \cup A_n) = P(A_1) + P(A_2) + \cdots + P(A_n) = n \cdot P(A_i).$$

It follows that $P(A_i) = 1/n$ for all i. Suppose now that $B = \{A_1, A_2, \ldots, A_k\}$ is a compound event. Then

$$P(B) = (A_1) + P(A_2) + \cdots + P(A_k) = \frac{1}{n} + \frac{1}{n} + \cdots \frac{1}{n} = \frac{k}{n}.$$

That is, the probability of B is the number of simple events in B divided by the total number of simple events.

Theorem 26.6. *Suppose that B is an event in a finite sample space S and that every simple event in S is equally likely. Then*

$$P(B) = \frac{|B|}{|S|}.$$

Example 26.7. Suppose that a six-sided die is rolled. What is the probability of rolling an even number?

Solution: The sample space here, $\{1, 2, 3, 4, 5, 6\}$, contains six simple events. The event of rolling an even number, $\{2, 4, 6\}$ contains three events. Therefore, the probability of rolling an even number is

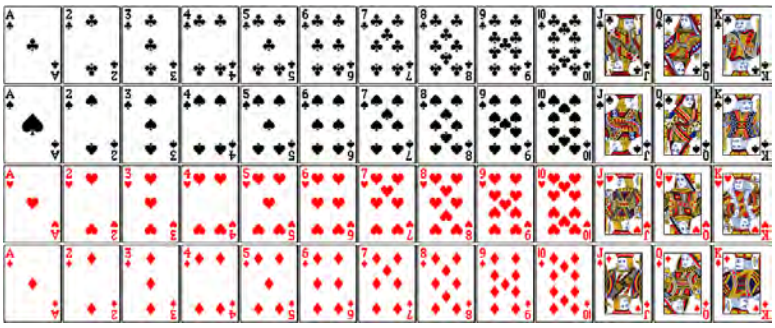
$$P(even) = \frac{3}{6} = \frac{1}{2}.$$

Example 26.8. Suppose that two dice are rolled. What is the probability that the sum of the dice is 5?

Solution: The sample space for this procedure is pictured above. It contains 36 simple events. If we list every event as an ordered pair of numbers, then the events in which the sum is 5 are $(1, 4)$, $(2, 3)$, $(3, 2)$, and $(4, 1)$. There are four of these, so

$$P(sum = 5) = \frac{4}{36} = \frac{1}{9}.$$

Cards. A standard playing deck contains 52 cards divided into two colors (red and black) and four suits - the red suits (hearts and diamonds) and the black suits (clubs and spades). Each suit consists of 13 cards: Ace, 2, 3, 4, 5, 6, 7, 8, 9, 10, Jack, Queen, and King. The Jacks, Queens, and Kings are called face cards.



Example 26.9. A card is drawn randomly from a shuffled deck of cards. What is the probability that it is a red face card?

Solution: The red face cards are the Jack, Queen, and King of hearts and diamonds. There are six of these, so the probability is $\frac{6}{52} = \frac{3}{26}$.

The addition rule for counting can be extended to probabilities. If A and B are events then

$$P(A \cup B) = P(A) + P(B) - P(A \cap B).$$

Example 26.10. Suppose that a card is drawn randomly from a shuffled deck. What is the probability that it is red or a face card?

Solution: There are 52 cards total. There are 12 face cards (three in each suit), and there are six red face cards. Therefore,

$$\begin{aligned} P(\text{red or face card}) &= P(\text{red}) + P(\text{face card}) - P(\text{red face card}) \\ &= \frac{26}{52} + \frac{12}{52} - \frac{6}{52} \\ &= \frac{32}{52} \\ &= \frac{8}{13}. \end{aligned}$$

Definition 26.11. Suppose that A is an event in a sample space S. The *complement* of A is the event that A does not occur. This is denoted as $\bar{A}$.

The event complement of A is precisely the set complement of A in the universe S. Since $A \cup \bar{A} = S$, and since $A \cap \bar{A} = \emptyset$, we have that $1 = P(S) = P(A) + P(\bar{A})$. This implies that $P(\bar{A}) = 1 - P(A)$.

Example 26.12. Suppose five cards are draw at random from a standard deck. What is the probability that at least one of the cards is red?

Solution: Let A be the event that at least one of the five cards is red. Then $\bar{A}$ is the event that none of the cards is red. We will calculate $P(\bar{A})$ and then $P(A) = 1 - P(\bar{A})$. To calculate $P(\bar{A})$, we need to know how many 5-element sets of cards there are, and we need to know how many of them contain no red cards. The number of 5-element sets of cards is $C(52, 5)$ since there are 52 cards in the deck. The number of black cards is 26, so the number of 5-element sets of cards that are only black is $C(26, 5)$. Therefore,

$$P(\bar{A}) = \frac{C(26,5)}{C(52,5)} = \frac{65780}{2598960} \approx 0.0253.$$

It follows that $P(A) = 1 - P(\bar{A}) \approx 0.9747$. It is extremely likely that a set of 5 cards will contain at least one red.

Conditional Probabilities. The probability of an event A given that an event B has already happened is denoted $P(A|B)$, which is read, "the probability of A, given B." Extending the counting rule for simple events, it should seem reasonable that

$$P(A|B) = \frac{P(A \cap B)}{P(B)}.$$

Here, the top of the fraction is the "number" of ways that A can happen if B has happened, and the bottom is the "number" of ways that B can happen. Solving this equation can give a formula for $P(A \cap B)$.

$$P(A \cap B) = P(B) \cdot P(A|B).$$

If the occurrence of B does not affect the probability of the occurrence of A, then A and B are *independent*. In this case, $P(A \cap B) = P(A) \cdot P(B)$.

Example 26.13. An urn contains 3 red marbles and 2 blue marbles. Two marbles are drawn from the urn without replacement. What is the probability they are both red?

Solution: The probability that the first marble is red is $\frac{3}{5}$. If we remove a red marble, then there are 2 red and 2 blue marbles in the urn. Therefore the probability that the second marble is red if the first was red is $\frac{2}{4}$. Then

$$\begin{aligned} P(\text{both red}) &= P(1^{st} \text{ red and } 2^{nd} \text{ red}) \\ &= P(1^{st} \text{ red}) \cdot P(2^{nd} \text{ red} | 1^{st} \text{ red}) \\ &= \frac{3}{5} \cdot \frac{2}{4} \\ &= \frac{3}{10}. \end{aligned}$$

Exercises 26.14. Please answer the questions below.

1. Two dice are rolled.
   (a) What is the probability that both dice show even numbers?
   (b) What is the probability that the second number is a multiple of the first?
   (c) What is the probability that the letter e occurs in the spelling of at least one of the numbers?
2. A card is selected randomly from a standard deck.
   (a) What is the probability that the card is a number card? (Not Ace, King, Queen, or Jack.)
   (b) What is the probability that the card is a face card or a heart?
   (c) What is the probability that the card is not a King?
3. A set of two cards is selected from a standard deck. (*Set* implies that we do not care about order.)
   (a) What is the probability that both cards are red?
   (b) What is the probability that at least one card is black?
   (c) What is the probability that both cards are face cards?
   (d) What is the probability that at least one card is not a face card?
4. A set of five cards is selected from a standard deck.
   (a) What is the probability that the set of five cards contains no face cards?
   (b) What is the probability that the set of five cards contains a face card?
   (c) What is the probability that the set of five cards contains two red cards and three black cards?
5. A set of four cards is selected from a standard deck.
   (a) What is the probability that the set of four cards contains one heart, one club, one spade, and one diamond?
   (b) What is the probability that the set of four cards contains two red and two black cards?
6. A bowl contains 5 red marbles and 7 blue marbles. Two marbles are drawn from the bowl without replacement.
   (a) What is the probability that the first is red and the second is blue?
   (b) What is the probability that the marbles are the same color?
7. A certain type of cheap battery operated alarm clock works 80% of the time. Bob buys two of these clocks.
   (a) On any given morning, what is the probability that both clocks fail?
   (b) On any given morning, what is the probability that at least one clock works?