This tutorial covers concepts from discrete mathematics that you will need.[1]

Topics:

1. strings and recursive definitions
2. sets: set-builder notation $\{x : ...\}$, operations on sets such as $\cup, \cap, \times, \setminus$, and relations between sets such as $\subseteq, \supseteq, =$
3. functions: domain, codomain, image, associative binary operations
4. finite sets and infinite sets
5. graphs and trees
6. asymptotic notation
7. proofs: providing a counterexample to show a claim is false; proof by contrapositive; proof of negation and proof by contradiction; proof by induction

Conventions:

- We write $\mathbb{Z}^+$ for the set of positive integers and $\mathbb{Z}_0^+$ for the set of non-negative integers.
- We will often use `Python`'s slicing notation for indexing, see the documentation.

1 Strings

Since this UoS focuses on strings and sets of strings, I've put that material first. After doing these questions you should be able to:

1. specify sets of strings in English and set-builder notation
2. reason about simple recursive functions that manipulate strings
3. be able to show that strings are/are not in a given set

An *alphabet* is a finite set of characters (aka symbols, aka letters). For instance, the set of ASCII characters is an alphabet. A *string* is a finite sequence of characters from some alphabet. For example, `abracadabra` is a string over the ASCII alphabet. It is also a string over the alphabet of lower-case English letters. A string is like `Python`'s `str` object; we can also think of (the contents of) a text file as being a string. Every string has a length (returned by `len` in `Python`). There is a single string of length 0, it is called the *empty string*, and written ϵ (`''` in `Python`). The *concatenation* of strings u, v is the string uv formed by appending v to the end of u ($u + v$ in `Python`).

We will use the following *exponentiation shorthand* for strings. If u is a string and n is a positive integer then u^n is shorthand for

$$\overbrace{u \cdots u}^{n}$$

We use the convention that if $n = 0$ then $u^n = \epsilon$. For instance, if $u = aab$ then $u^0 = \epsilon, u^1 = u = aab, u^2 = uu = aabaab, u^3 = uuu = aabaabaab$.

[1] Some of the questions are extracted from chapter 0 of "Introduction to the Theory of Computation", by Michael Sipser. I also recommend "Discrete mathematics and its applications" by Kenneth Rosen. Parts of it give a good introduction to discrete mathematics we will use a lot (chapter 2 on sets and functions, chapter 5 on recursion, chapter 9 on relations, the beginning of chapter 10 on graphs and the beginning of chapter 11 on trees), and parts cover material we will also cover (chapter 1 on logic and proofs, and chapter 13 on modeling computation).

In this course we will be interested in strings because inputs to programs, and even programs themselves, can be thought of as strings.

We will also be interested in *sets of strings* because they naturally model computational problems with "yes"/"no" outputs, which are the focus of this UoS.

Problem 1. What does it mean for an integer to be even? Is 0 an even integer? Is the empty string of even length?

Solution 1. An integer is even if it is divisible by 2. More precisely, $n \in \mathbb{Z}$ is even if there is some integer $k \in \mathbb{Z}$ such that $n = 2 \times k$. So 0 is even since $0 = 2 \times 0$ (and so we can take $k = 0$ in the definition).

Problem 2. In this problem, the alphabet is the set $\{a, b\}$.

The set $\{a^n b : n \in \mathbb{Z}_0^+\}$ is the set of strings with just one b, and that b is at the end of the string.

For each of the following sets of strings, give short precise English description of it.

1. $\{a^n b^m : n, m \in \mathbb{Z}_0^+\}$
2. $\{a^n b^n : n \in \mathbb{Z}_0^+\}$
3. $\{u : u$ is a possibly empty string of as and bs such that $|u| = 2n$ for some integer $n\}$

Solution 2.

1. The set of strings with all as coming before all bs
2. The set of strings with the same number of as as bs, and all as coming before all bs
3. The set of strings of even length

Problem 3.

```
1  def myfun(s): # s is a string (using only ASCII characters)
2    if s=='': # empty string
3      return 1
4    if len(s) == 1:
5      return 1
6    if s[0] != s[-1]: # first and last characters are different
7      return 0
8    t = s[1:len(s)-1] # remove the first and last characters
9    return myfun(t) # recursive call
```

myfun is a program that takes as input a string, and returns 0 or 1.

Trace the execution of myfun("abb") and myfun("abba").

What does the function do?

Argue that your answer is correct. [Advanced]

Solution 3. It returns 1 if s reads the same forwards as backwards (such strings are called 'palindromes'), and 0 otherwise.

Reasoning that recursive code is correct only requires 'local' reasoning, i.e., you only have to check (a) the base case is correct, (b) each recursive case is correct assuming that the function works correctly when it is called on 'smaller' (or shorter or simpler) input.

Reason as follows. A string s is either (1) empty, or (2) has one character, or (3) has more than one character.

(1) Suppose s is the empty string. Then it is a palindrome, and myfun(s) correctly returns 1.

(2) Suppose s is a single character. It is a palindrome, and myfun(s) correctly returns 1.

(3) Suppose s has more than one character. We can split this into two cases depending on whether or not the first and last characters are the same.

If the first and last characters are different it is not a palindrome, and myfun correctly returns 0. If the first and last character are the same, say $s = ata$ where a is a single character and t is a string, then s is a palindrome exactly when t is a palindrome. Assuming that myfun(t) give the correct answer (note that t is shorter than s), conclude that myfun(s) gives the correct answer.

Problem 4. [Harder]

```
1 def myfun(s): # s is a string that only uses characters a and b
2     if s=='': # empty string
3         return 1
4     for i in range(1,len(s)):
5         if (s[0] != s[i]) and myfun(s[1:i]) and myfun(s[i+1:]):
6             return 1
7         # recall that s[1:i] is the substring from index 1 to index i-1
8         # recall that s[i+1:] is the substring from index i+1 to the end
9     return 0
```

myfun is a program that takes as input a string that only uses characters a and b, and returns 0 or 1.

What does the function do?

Argue that your answer is correct. [Advanced]

Solution 4. It returns 1 if s has the same number of as as bs, and 0 otherwise.

Reasoning that recursive code is correct only requires 'local' reasoning, i.e., you only have to check (a) the base case is correct, (b) each recursive case is correct

assuming that the function works correctly when it is called on 'smaller' (or shorter or simpler) input.

Reason as follows. A string s is either (1) empty, or (2) not empty.

(1) Suppose s is the empty string. Then it has zero as and zero bs, and $myfun(s)$ correctly returns 1.

(2) Suppose s has one or more characters. Suppose further that $myfun$ works correctly on strings shorter than s.

We need to argue **two** things.

(2.1) if s has the same number of as as bs, then $myfun(s) = 1$.

We will use the following notation. For a string z let $num_a(z)$ be the number of a's in z, and let $D(z) = |num_a(z) - num_b(z)|$. Clearly z has the same number of as as bs means the same thing as $D(z) = 0$. For $i < |z|$ let $f_z(i) = D(z_0 z_1 \cdots z_i)$. If s has the same number of as as bs, then it is of the form $s = axby$ for some strings x, y, each of which themselves have the same number of as as bs — to see this, let j be the smallest position such that $f_s(j) = 0$, and let $x = s_1 \cdots s_{j-1}$ and $y = s_{j+1} \cdots s_{n-1}$ (where $n = len(s)$). Then, in the algorithm, when $i = |x| + 1$ the first condition holds (since $s[0] = a, s[i] = b$), the second condition holds (since $s[1 : i] = x$ is shorter than s), the third condition holds (since $s[i + 1 :] = y$ is shorter than s). Thus, $myfun(s)$ returns 1.

(2.2) if $myfun(s) = 1$ then s has the same number of as as bs.

Now for (2.2). If $myfun(s) = 1$ then there is some $i < len(s)$ such that the three conditions hold, i.e, $s = axby$ (or $s = bxay$) and $myfun(x) = 1$ and $myfun(y) = 1$. Since x, y are shorter than s, by induction we know that x has the same number of as as bs, say k_1, and y has the same number of as as bs, say k_2. So then s has the same number of as as bs, i.e., $k_1 + k_2 + 1$ of each.

Problem 5. In this problem the alphabet is the set of lower-case English letters. Let L be the set of strings of the form ww where w is a string. In other words, these are the strings whose left-half is equal to their right-half.

Now, *aabaab* is in L; to see this, take $w = aab$. On the other hand, *aaaabaaaaaab* is not in L since the left half *aaaaba* is not equal to the right half *aaaaab*.

1. Is the empty string in L?
2. Show that the following strings are *not* in L: $\overbrace{a \cdots a}^{j} b \overbrace{a \cdots a}^{i} b$ where $i, j \in \mathbb{Z}^+$ with $1 \leq i < j$.
3. Show that the following strings are in L: $\overbrace{a \cdots a}^{j} \overbrace{a \cdots a}^{j+2}$ where $j \in \mathbb{Z}^+$.

Solution 5.

1. Yes, take $w = \epsilon$.
2. Say $x = \overbrace{a \cdots a}^{j} b \overbrace{a \cdots a}^{i} b$ for some positive integers i, j with $i < j$. Here we don't know the values of i and j, we only know that $i < j$.

   If $i + j$ were odd, then the x has odd length, and so is not in L since odd length strings are not in L (why?).

   If $i + j$ were even, then the length of the string, which is $2 + i + j$, is also even, and so the left half of x has length $(2 + i + j)/2 = 1 + (i + j)/2 < 1 + (j + j)/2 = 1 + j$. So the left half of x has length at most j (make sure you see where we used the assumption that $i < j$).

   But the first j symbols in x are as, while the right half also has bs, and so the left half of the string is not equal to the right half of the string. So x is not in L.

   A student found the following alternative reasoning: Due to the presence of the two bs, the only way for left and right pieces of x to match would be to split the string after the first b, i.e., $x = s_1 s_2$ where both s_1, s_2 end in b. Since $j > i$, s_1 has more a's than s_2. So, the two pieces are not equal, so x is not in L.
3. Say $x = \overbrace{a \cdots a}^{j} \overbrace{a \cdots a}^{j+2}$ for some positive integer j. Then $x = \overbrace{a \cdots a}^{2j+2}$ which is of the form ww for $w = \overbrace{a \cdots a}^{j+1}$.

Problem 6.[Extra] Recall that the powers of 2 are numbers of the form 2^i for $i \in \mathbb{Z}_0^+$. Is $2^i + 2^j$ a power of two for any $i, j \in \mathbb{Z}_0^+$ with $i < j$.

Solution 6. We don't know the exact value of i, j, but we can still show that $n = 2^i + 2^j$ is not a power of 2 using that $i < j$. Reason as follows: $2^j < 2^j + 2^i < 2^j + 2^j = 2^{j+1}$ (make sure you see where we used the assumption that $i < j$). So n lies strictly between 2^j and 2^{j+1}, i.e., between consecutive powers of 2. So n cannot, itself, be a power of 2.

Problem 7.[Extra]

```
1 def F(n): # input is a non-negative integer, output is an integer
2   if n == 0:
3     return 0
4   if n == 1:
5     return 1
6   return F(n-1)+F(n-2)
```

The number $F(n)$ is often called the nth Fibonacci number, https://www.youtube.com/watch?v=SjSHVDfXHQ4

What does the function return when given $n = 6$ as input?

What happens if we take the same code, but interpret 0 and 1 as strings, and + as string concatenation? Let's write this using as and bs to make it a bit clearer, and rename the function G to distinguish it from F.

```
def G(n): # input is a non-negative integer, output is a string
  if n == 0:
    return a
  if n == 1:
    return b
  return G(n-1)+G(n-2) # + refers to string concatenation
```

The string $G(n)$ is often call the nth Fibonacci string, https://en.wikipedia.org/wiki/Fibonacci_word

What is $G(6)$?

Solution 7. Well, since F calls itself, we can start with the base cases and work our way up to $n = 6$.

$$
\begin{aligned}
F(0) &= 0 \\
F(1) &= 1 \\
F(2) &= F(1) + F(0) = 1 + 0 = 1 \\
F(3) &= F(2) + F(1) = 1 + 1 = 2 \\
F(4) &= F(3) + F(2) = 2 + 1 = 3 \\
F(5) &= F(4) + F(3) = 3 + 2 = 5 \\
F(6) &= F(5) + F(4) = 5 + 3 = 8
\end{aligned}
$$

Similarly for G:

$$
\begin{aligned}
G(0) &= a \\
G(1) &= b \\
G(2) &= G(1) + G(0) = b + a = ba \\
G(3) &= G(2) + G(1) = ba + b = bab \\
G(4) &= G(3) + G(2) = bab + ba = babba \\
G(5) &= G(4) + G(3) = babba + bab = babbabab \\
G(6) &= G(5) + G(4) = babbabab + babba = babbababbabba
\end{aligned}
$$

Problem 8.[Harder, and fundamental] If $u = u_1u_2 \cdots u_n$ is a string over alphabet $\{0, 1\}$ what does the expression $\sum_{i=1}^{n} u_i 2^{i-1}$ represent? Why do 101 and 10100 evaluate to the same number? What is the value of the expression on the empty string?

Solution 8. The expression evaluates to the number represented by the binary string u with the least significant digit first.

The fact that 11 and 1100 both evaluate to 3 is because adding "trailing zeros" to the most significant places of a binary string doesn't change its value (in decimal, with most significant digit first, this is reflected by the fact that 3 and 00003 both represent the number 3). For the empty string, the sum is empty, and the convention is that an empty sum has value 0.

2 Sets

Problem 9. Examine the following formal descriptions of sets so that you understand which members they contain. Write a short informal English description of each set.

1. $\{1, 3, 5, 7, ...\}$
2. $\{..., -4, -2, 0, 2, 4, ...\}$
3. $\{n \mid n = 2m \text{ for some } m \in \mathbb{Z}_0^+\}$
4. $\{n \mid n = 2m \text{ for some } m \in \mathbb{Z}_0^+, \text{ and } n = 3k \text{ for some } k \in \mathbb{Z}_0^+\}$
5. $\{n \mid n \text{ is an integer and } n = n + 1\}$
6. $\{rem(n, 2) : n \in \mathbb{Z}_0^+\}$, where $rem(n, 2)$ is the remainder when dividing n by 2.

Solution 9.

1. The set of odd positive integers.
2. The set of even integers.
3. The set of even non-negative integers.
4. The set of non-negative integers that are multiples of 2 and of 3.
5. The empty set (note that the set of integers that are equal to their successor has no members at all).
6. The set $\{0, 1\}$ (the point is that even though there seem to be infinitely many elements in the set because of the way it is written, once you understand the elements in the set you see there are only two of them).

Problem 10. Let A be the set $\{x, y, z\}$ and B be the set $\{x, y\}$.

1. Is A a subset of B?

2. Is B a subset of A?
3. What is $A \cup B$?
4. What is $A \cap B$?
5. What is $A \setminus B$?
6. What is $B \setminus A$?
7. What is $A \times B$?
8. What is the power set of B?

Solution 10.

1. No
2. Yes
3. $\{x, y, z\}$
4. $\{x, y\}$
5. $\{z\}$
6. $\emptyset$
7. $\{(x, x), (x, y), (y, x), (y, y), (z, x), (z, y)\}$
8. $\{\emptyset, \{x\}, \{y\}, \{x, y\}\}$

Problem 11. Let A, B, Z be sets such that $A, B \subseteq Z$. For each of the following say if it is true or not (give reasons).

1. $Z \setminus (A \cap B) = ((Z \setminus A) \cup (Z \setminus B))$
2. $A \subseteq B$ if and only if $A \cap (Z \setminus B) = \emptyset$.

Solution 11.

1. This is true. To convince yourself if this, you can draw a Venn diagram, or reason as follows: an element of Z is in the LHS means it is not in both A and B; so either it is not in A, or not in B, or not in either; but this is what the RHS says. It is called **De Morgan's Law** for sets.
2. This is true. To see this you can draw a Venn Diagram, or reason as follows: the LHS says that that every element in A is also in B, which is the same as saying there is no element in A that is not in B, which is what the RHS says.

Problem 12. If A has a elements and B has b elements, how many elements are in $A \times B$? Explain.

Solution 12. The number of elements in $A \times B$ is $a \times b$ because for every element a of A there are b pairs in $A \times B$, one for each element in B.

Problem 13. Recall that the power set of a set C is the set of subsets of C. E.g., the powerset of $C = \{a, b, c\}$ has 8 elements, i.e., $\{\emptyset, \{a\}, \{b\}, \{c\}, \{a, b\}, \{a, c\}, \{b, c\}, \{a, b, c\}\}$. If C is a set with c elements, how many elements are in the power set of C? Explain your answer.

Solution 13. 2^c elements. One way to see this is to note that each subset can be identified by a series of c-many binary choices, one for each element of C, to decide whether to include it or exclude it from the subset. To count the number of such series, multiple 2 by itself c times.

3 Functions

Problem 14. Let X be the set $\{1, 2, 3, 4, 5\}$ and Y be the set $\{6, 7, 8, 9, 10\}$. The unary function $f : X \to Y$ and the binary function $g : X \times Y \to Y$ are described in the following tables.

n	$f(n)$
1	6
2	7
3	6
4	7
5	6

g	6	7	8	9	10
1	10	10	10	10	10
2	7	8	9	10	6
3	7	7	8	8	9
4	9	8	7	6	10
5	6	6	6	6	6

1. What is the value of $f(2)$?
2. What are the domain, codomain of f, and image of f?
3. What is the value of $g(2, 10)$?
4. What are the domain, codomain of f, and image of g?
5. What is the value of $g(4, f(4))$?

Solution 14.

1. 7
2. The codomain of f is Y, the domain of f is X, and the image is $\{6, 7\}$.
3. 6

4. The codimain of g is Y, the domain is $X \times Y = \{(x, y) \mid x \in X, y \in Y\}\}$, and the image is Y.
5. 8

Problem 15. Recall that addition is an *associative* binary operation, i.e., $x + (y + z) = (x + y) + z$, and so we may drop the parentheses and simply write $x + y + z$. Which of the following binary operations is associative?

1. Concatenation of strings
2. Intersection on sets
3. Union on sets
4. Multiplication on integers
5. Subtraction on integers
6. Division on real numbers
7. Exponential on real numbers

Solution 15.

1. In `Python`, $('abc' + 'def') + 'ghi'$ evaluates to the same string as $'abc' + ('def' + 'ghi')$, i.e., $'abcdefghi'$. This is not a coincidence: for all strings u, v, w, we have that $(u + v) + w = u + (v + w)$, because these evaluate to the string uvw. We say that concatenation of strings is *associative*. This is important because it allows us to drop parentheses (which makes everyone's life easier). So, we can just write $'abc' + 'def' + 'ghi'$ in `Python` (try it!)
2. Intersection is associative, i.e., $A \cap (B \cap C) = (A \cap B) \cap C$ and so we simply write $A \cap B \cap C$.
3. Union is associative, i.e., $A \cup (B \cup C) = (A \cup B) \cup C$ and so we simply write $A \cup B \cup C$.
4. Addition is associative, i.e., $x + (y + z) = (x + y) + z$ and so we simply write $x + y + z$.
5. Multiplication is associative, and so we simply write $x \times y \times z$.
6. Subtraction is not associative. Here is a counterexample to the claim that subtraction is associative, i.e., to the claim that $x - (y - z) = (x - y) - z$: Let $x = 1, y = 0, z = 1$, and note that $1 - (0 - 1) = 2$ but $(1 - 0) - 1 = 0$.
7. Division is not associative. Here is a counterexample to the claim that division is associative, i.e., to the claim that $x/(y/z) = (x/y)/z$: let $x = 1, y = 1, z = 2$, and note that $1/(1/2) = 2$ but $(1/1)/2 = 1/2$.

8. Exponentiation is not associate. Here is a counterexample to the claim that exponentiation is associative, i.e., to the claim that $x^{(y^z)} = (x^y)^z$: let $x = 2, y = 1, z = 2$, and note that $2^{(1^2)} = 2$ but $(2^1)^2 = 4$.

Problem 16.

The idea of a set being *closed* under an *operation* (aka function) will come up a few times in this course. A set is closed under some operation if, when we apply that operation to elements in that set, we are guaranteed to get another element in that set.

The set $\mathbb{Z}^+$ of positive integers is closed under the addition operation — this means that the sum of any two positive integers is a positive integer. We simply say "$\mathbb{Z}^+$ is closed under addition". On the other hand, $\mathbb{Z}^+$ is not *closed under subtraction*, because if you subtract two positive integers, you are not guaranteed to get back a positive integer. To see this we give a *counterexample*, e.g., $1 - 3 = -2$.

Answer the following questions:

1. Is the set of integers closed under addition? subtraction? multiplication? division?
2. Is the set of even integers closed under addition? subtraction? multiplication? division?
3. Is the set of odd integers closed under addition? subtraction? multiplication? division?
4. Is the set of real numbers closed under addition? subtraction? multiplication? division?
5. Is the set of rational numbers closed under addition? subtraction? multiplication? division?

If you answered "no", you should give a counterexample. If you answered yes, you should give a reason, or even a proof (see the section on 'Proofs').

Solution 16.

1. Yes, yes, yes, no (e.g., $3 \div 2 = 1.5$ is not an integer)
2. Yes, yes, yes, no (e.g., $2 \div 2 = 1$ which is not even).
3. No (e.g., $1 + 1 = 2$ which is not odd), no (e.g., $1 - 1 = 0$ which is not odd), Yes, No (e.g., $5 \div 3$ is not an odd integer).
4. Yes, yes, yes, no (the only issue is division by zero).
5. Yes, yes, yes, no (the only issue is division by zero).

4 Finite sets and infinite sets

Recall that a set S is *infinite* means that for every positive integer n, the set S contains more than n elements.

Infinite sets are a tricky concept — many misconceptions exist. Here are some clarifications:

1. An infinite set does not have to include everything. For example, the set of strings of even length is infinite, but does not include *aaa*.
2. A set can be infinite even though the objects in that set are finite. An individual string is always finite (at least in this unit). A set of strings might be infinite, if it contains infinitely many of these finite strings.
3. Infinite sets don't have the same notion of size as finite sets. If we take a finite set and add a new element, the resulting set will of course be larger. But if we take an infinite set and add a new element, the resulting set can be exactly paired off with the original set. Eg. we can pair off $(0,1),(1,2),(2,3),(3,4),...$ to show that Z_0^+ and Z^+ have the same size.[2]

Problem 17. Suppose A is finite and B is infinite. For each of the following sets, say if it is infinite, finite, or we cannot tell without more information. Briefly explain your answers.

1. $B \cup A$
2. $B \cap A$
3. $B \setminus A$
4. $A \setminus B$

Suppose Z is infinite. For each of the following cases, conclude as much as possible about whether A or B are finite or infinite. Briefly explain your answers.

1. $Z = A \cup B$
2. $Z = A \cap B$
3. $Z = B \setminus A$

Solution 17.

1. $B \cup A$ is infinite because it contains an infinite set B.
2. $B \cap A$ is finite because it is contained in a finite set A
3. $B \setminus A$ is infinite because removing a finite number of elements from an infinite number leaves an infinite number.
4. $A \setminus B$ is finite because it is contained in a finite set A.

[2] Implicit in this is that two sets have the same size if their elements can be paired off. Formally, X, Y have the same size if there is a bijection $f : X \to Y$.

1. At least one of A, B is infinite because otherwise Z, which is contained in both A and B would be finite. However, not necessarily both are infinite. For instance, $\mathbb{Z}^+ = \{1, 2, \} \cup \mathbb{Z}^+$.
2. Both A, B are infinite, for if one were finite, then Z, being contained in both, would be finite.
3. In this case B must be infinite because Z is contained in B. However, A may be finite, e.g., $\mathbb{Z}_0^+ = \mathbb{Z}_0^+ \setminus \{-1\}$, or it may be infinite, e.g., $\mathbb{Z}_0^+ = \mathbb{Z} \setminus \{-n : n > 0\}$.

Problem 18. In this course we will see a few operations on *sets of strings*. For instance, while concatenation is usually an operation on strings, union is an operation on sets of strings.

A collection of sets is *closed* under some *operation* if, when we apply that operation to sets in that collection, we always get another set in that collection.[3] Intuitively, this allows us to "build up" new sets of strings by combining other sets of strings.

Fix an alphabet Σ (recall that an alphabet is a finite set of characters). For concreteness, all strings in this problem only use characters from Σ.

1. Is the collection of finite sets of strings closed under union? Under intersection? Under complement?
2. Is the collection of infinite sets of strings closed under union? Under intersection? Under complement? (Be careful with the last two!)

Solution 18.

1. The class of finite sets is closed under both union and intersection. To see this, let two finite sets have n and m elements. Then the union can have at most $n + m$ elements, so it is clearly finite. Similarly the intersection has at most $min(n, m)$ elements and is clearly finite too.

   The class of finite sets is not closed under complement. In fact any finite set at all is a counterexample, since its complement with respect to the infinite set of all strings must be infinite.
2. The class of infinite sets is closed under union. To see this, notice that the union has at least as many elements as either set individually, so it is at least as large as an infinite set, and thus infinite.

   The class of infinite sets is, perhaps counterintuitively, not closed under intersection. To see this, consider the set of strings of odd length, and the set of strings of even length. Both these sets are infinite, but their intersection is empty, and thus finite.

[3] This is just like the notion of a set being closed under an operation, only now our set consists of sets of strings!

The class of infinite sets is not closed under complement. To see this, take the set of all strings. Its complement is empty, and thus finite. Note though that the complement of an infinite set might be finite or infinite, i.e., we are not guaranteed that the complement of an infinite set is infinite. We say, then, that the infinite sets are not closed under complement.

Problem 19.[Extra, hard] Does there exist a function f from the integers to sets of strings such that $f(x) \subsetneq f(x+1)$ for all integers x? In other words, does there exist a bi-infinite sequence of sets of strings such that $\cdots \subsetneq S_{-2} \subsetneq S_{-1} \subsetneq S_0 \subsetneq S_1 \subsetneq S_2 \subsetneq \cdots$?

Solution 19. Let us attempt to find a solution. What set should we choose for S_0? Notice that there exists an infinite sequence of sets $S_{-1}, S{-}2, \cdots$ that are strict subsets of it. Each subsequent S_{i-1} drops at least one element from S_i. For this to be possible, S_0 must be infinite. By a corresponding argument, since S_{i+1} adds at least one element to S_i, S_0 must have "infinite room to grow" — that is, its complement must be infinite.

A simple example of a set that is infinite and has an infinite complement is the set of strings of even length consisting of just a. So let us take this as S_0, and attempt to construct the other sets. How can we construct the S_{i+1}? Each one needs to add on at least one element, so let us have each one add on the smallest string of a not yet in the set. That is, S_1 will add on a, S_2 will add on aaa, S_3 will ad on $aaaaa$ and in general, S_n will add on a^{2n-1} for positive n. This handles the positive half of the construction.

How about the negative half? Each S_{i-1} needs to remove at least one element. Again, let us remove subsequently larger elements. S_{-1} removes aa, S_{-2} removes $aaaa$, and in general, S_{-n} will remove a^{2n} for positive n.

Hence, our full construction is given by:

For $n < 0$, $S_n = \{a^k : \text{k is even and } k > -2n\}$. For $n = 0$, $S_n = \{a^k : \text{k is even}\}$
For $n > 0$, $S_n = \{a^k : \text{k is even or } k < 2n\}$

5 Graphs and trees

A *directed graph* G is a pair (V, E) where V is a set and $E \subseteq V \times V$. Usually the elements of V are called *vertices* or *nodes*, and elements of E are called *edges*. Aside: this definition of directed graph allows self-loops, i.e., edges of the form (u, u) (another convention you may see is to disallow self-loops).

We usually require that V is not empty.

Note that $|E| \leq |V|^2$.

Problem 20.

1. What restriction should one place on G to call it a "complete graph" (aka

"clique")?

2. What restriction should one place on G to call it an "undirected graph"?
3. What restriction should one place on G to call it a "tree"?

Solution 20.

1. Require that $E = V \times V$.
2. Require that E is symmetric, i.e., $(u, v) \in E$ implies $(v, u) \in E$ for all $u, v \in V$.
3. Require that (a) E has no cycles, i.e., there is no sequence of vertices $u_1, u_2, \cdots, u_k$ such that $u_1 = u_k$ and $(u_i, u_{i+1}) \in E$ for every $i < k$, and (b) there is a single node r, called the "root", from which every other node is reachable by a path, i.e., for every $u \in V$ with $u \neq r$, there is a sequence $u_1, u_2, \cdots, u_k$ such that $(u_i, u_{i+1}) \in E$ for every $i < k$, and $u_i = r$ and $u_k = u$.
   - Another answer is that a tree is an *undirected* connected graph with no cycles, and, optionally, a distinguished vertex called the "root".

6 Asymptotic notation

We will sometimes refer to the growth rate of functions of numeric functions (this notation is very useful in `COMP2123:Data Structures and Algorithms`). For instance, the functions $3n^2 + n$ and $10n^2$ are different, but it is often useful to point out that they are both quadratic functions. This is captured using big-Oh notation.

We say that f is **big-Oh** of g, and write $f(n) = O(g(n))$ or $f = O(g)$, if there are constants M, c such that for all $n > c$ we have that $f(n) \leq Mg(n)$.

For example: $3n^2 + n \leq 4n^2$ for all $n > 0$, and so we conclude that $3n^2 + n = O(n^2)$ by taking $M = 4, c = 0$.

Problem 21. Argue that $f = O(g)$ and $g = O(h)$ implies that $f = O(h)$. Thus big-Oh is transitive.

Solution 21. If $f(n) \leq Mg(n)$ for all $n > c$ and $g(n) \leq Nh(n)$ for all $n > d$, then $f(n) \leq MNh(n)$ for all $n > \max\{c, d\}$.

Problem 22.

Order the following functions according to the big-Oh notation.

- n (linear)
- 2^n (exponential)
- 3^n (exponential)

- 1 (constant)
- $\log n$ (logarithmic)
- $n \log n$ (log linear)
- n^2 (quadratic)
- n^3 (cubic)

Unless otherwise stated, all logs are base 2. You may use the following facts: $\log n < n$ for all positive integers n, and any polynomial f is big-Oh of any exponential g.

Solution 22. The order is $1, \log n, n, n \log n, n^2, n^3, 2^n, 3^n$.

Here are the reasons:

- To see that $1 = O(n)$ just take $M = 1, c = 1$.
- To see that $\log n = O(n)$ note that $\log n < n$ for all positive integers n (so we may take $M = 1, c = 1$.
- To see that $\log n = O(n \log n)$ note that $\log n \leq n \log n$ for all positive n (so we may take $M = 1, c = 1$).
- To see that $n \log n = O(n^2)$ note that $\log n \leq n$ for all positive integers n (so we may take $M = 1, c = 1$).
- To see that $n^2 = O(n^3)$ just take $M = 1, c = 1$.
- To see that $n^3 = O(2^n)$ use the fact that n^3 is a polynomial and 2^n is an exponential.
- To see that $2^n = O(3^n)$ take $M = 1, c = 1$.

By the way, here is how we can justify the facts.

Fact 1. $\log n < n$ for all positive integers n. To see this note that $\log 1 < 1$, and the derivative of $\log n$ is $1/n$, while that of n is 1, so $\log n$ starts below n, and grows slower.

Fact 2. $n^3 = O(2^n)$; and we can replace 2, 3 by any positive constants. First note that if $\lim_n \frac{f(n)}{g(n)}$ converges then $f = O(g)$. Recall that L'Hospital's rule says that if $\lim_n \frac{f(n)}{g(n)} = \frac{\infty}{\infty}$, then $\lim_n \frac{f(n)}{g(n)} = \lim_n \frac{f'(n)}{g'(n)}$. Repeatedly applying this fact we see that $\lim_n \frac{n^3}{2^n} = \lim_n \frac{3n^2}{2^n \ln 2} = \lim_n \frac{6n}{2^n \ln^2 2} = \lim_n \frac{6}{2^n \ln^3 2} = 0$. Actually, because this converges to 0 (rather than some positive number), we have actually shown more, i.e., that n^3 is little-Oh 2^n, written $n^3 = o(2^n)$, which intuitively means that n^3 grows *much* slower than 2^n.

Problem 23. What is the value of the counter c at the end of this program?

```
c = 0
for k = 1 to K:
  for n = 1 to N:
    for m = n to N:
      c = c + 1
```

Also, express this in asymptotic notation.

Solution 23. The value is $KN(N+1)/2 = O(KN^2)$.

7 Proofs

Problem 24.

Negate the following statements:

1. "There is a positive integer smaller than 0"
2. "Every integer is positive"
3. "Some day's are hotter than 30 degrees"
4. "If it is raining then you are getting wet"
5. "If it is not raining then you don't need an umbrella"

Solution 24. The simplest way to negate a statement is to place "It is not the case that ..." in front of the statement. You can also use your knowledge about the world (e.g., that a number is either negative, equal to zero, or positive), to rewrite statements. You can also use logic to simplify the expression, e.g., "If X then Y" is false means that X is true and Y is false.

1. "There is no positive integer smaller than 0". If you answered "Every positive integer is bigger than or equal to 0" that's fine, but this requires that you know something about numbers (i.e., that $<$ is a total order on numbers).
2. "Some integer is not positive". If you answered "Some integer is zero or negative" is ok, but note that this requires that you know something about numbers (again, that every integer is either is positive, negative, or zero).
3. "No day is hotter than 30 degrees"
4. "It is raining and you are not getting wet".
5. "It is not raining and you do need an umbrella".

Problem 25. In this course we will prove things to make sure they are 100% correct. Although there is no recipe/formula for how to prove things, there are a few helpful strategies.

1. Understand what you can assume and what you need to show.
   (a) If you are asked to prove a statement of the form "if P then Q", you should assume P to be true and try show that Q must be true.
   (b) If you are asked to prove a statement of the form "P if and only if Q" (also written "P iff Q") you should prove two *directions*: "if P then Q" as well as "if Q then P".
   (c) If you are asked to prove a statement of the form "all objects that satisfy property X also satisfy property Y", you can assume you have an object with property X, call it x, and try show that x also has property Y (note that the only assumption you are allowed to make about x is that it has property X). If you think the statement is not true, you should try find a concrete object that has property X but fails to have the property Y. Such an object is called a **counterexample**.
   (d) If you are asked to prove a statement of the form "not P" you could assume P and try derive a contradiction (e.g., that $1 = 0$). This is called **proof of negation**.[4]
   (e) If you are asked to prove a statement of the form "P" you could assume "not P" and try derive a contradiction. This is called **proof by contradiction**. It is subtly different from proof of negation.
2. To prove something is true, try get a feeling of why it should be true. Try experimenting with examples.
3. Be patient! Be neat! Come back to it! Be concise!

Let's see some examples.

Recall that an integer n is *even* if it can be expressed as $2m$ for some integer m. Suppose we want to prove that the even integers are closed under addition. Another way of saying this is: for all even integers n_1, n_2, their sum $n_1 + n_2$ is even. Apply template 1(c) above to prove the following:

Theorem 1. The even integers are closed under addition.

Recall that an integer n is *odd* if it can be expressed as $2m + 1$ for some integer m. Give a proof of the following:

Theorem 2. The sum of two odd integers is even.

Here is an example of **proof by cases**.

Theorem 3. If n_1, n_2 are both even or both odd, then $n_1 + n_2$ is even.

Proof. There are two cases to consider.

Case 1: n_1, n_2 are both even. Then their sum is even (by Theorem 1).

Case 2: n_1, n_2 are both odd. Then their sum is even (by Theorem 2). □

[4] We frequently use this type of reasoning in real life: If you enter the classroom from outside and are completely dry, then I know that it is not raining ("not P"). I reason as follows: if it were raining ("P") then you would be wet (which is obviously false since you are completely dry); therefore, it can't be raining.

By the way, this proof was relatively easy because we already did the hard work in the proofs of Theorems 1 and 2! In such cases, we might call Theorem 3 a *corollary* of Theorems 1 and 2.

Finally, use **proof of negation** to show the following:

Theorem 3. There is no smallest integer.

Proof. Assume there is a smallest integer, n. Then $n-1$ is an integer, but it is also smaller than n, which is a contradiction. Therefore there is no smallest integer. □

Solution 25.

Proof of Theorem 1. Assume we have two even integers, let's call them n_1, n_2. We want to show that $n_1 + n_2$ is even.

By the definition of "even", we know that $n_1 = 2m_1$ for some integer m_1, and $n_2 = 2m_2$ for some integer m_2.

So $n_1 + n_2 = 2m_1 + 2m_2 = 2(m_1 + m_2)$.

Since $m_1 + m_2$ is an integer (here we use the fact that the integers are closed under addition!), we conclude (by the definition of "even") that $n_1 + n_2$ is even, which is what we wanted to show. □

Proof of Theorem 2. Let n_1, n_2 be odd integers.

Then, by the definition of "odd", we have $n_1 = 2m_1 + 1$ and $n_2 = 2m_2 + 1$, for some integers m_1 and m_2.

So $n_1 + n_2 = 2m_1 + 1 + 2m_2 + 1 = 2(m_1 + m_2) + 2 = 2(m_1 + m_2 + 1)$.

Since $m_1 + m_2 + 1$ is an integer (because the integers are closed under addition), we conclude (by the definition of "even") that $n_1 + n_2$ is even, which is what we wanted to show. □

Problem 26. We also will use **proof by induction** which shows that all objects in an infinite set have some property.

You've probably seen proofs by induction where the infinite set is $\{1, 2, 3, \cdots\}$. To show that every positive integer has some property P, we do two things:

1. We show that 1 has property P. This is called the **base case**.
2. We show that, for every number n, if n had property P then also $n+1$ has property P. This is called the **inductive case**.

The principles of induction allows us to conclude that every positive integer has property P. Why? Well we know that 1 does (base case), which means we know

that 2 does (just take $n = 1$ and apply the inductive case). But then we know that 3 does (just take $n = 2$ and apply the inductive case), etc.

Prove the following by induction.

Theorem 1. For every positive integer n, we have that $\sum_{i=1}^{n} i = n(n+1)/2$.

Theorem 2. $n! > 2^n$ for $n \geq 4$.

Why does induction work? Mainly because we can generate all positive integers starting from the simplest positive integer, 1, and by applying the operation $n \mapsto n+1$. In this course we will give similar definitions of infinite sets (like the set of formulas, the set of expressions, etc) that have base cases and operations, and then we can use induction to prove things about these sets!

Solution 26.

Proof of Theorem 1. Let $P(n)$ be the statement that $\sum_{i=1}^{n} i = n(n+1)/2$. We want to show that $P(n)$ holds for every positive integer n. We use induction.

1. For the base case we have to show that $P(1)$ holds.
   - Well, $P(1)$ says that $\sum_{i=1}^{1} i = 1(1+1)/2$ which is true since both sides equal 1.
2. For the inductive case we let n be an arbitrary number, and assume that $P(n)$ holds. We have to show that $P(n+1)$ holds.
   - So we have assumed that $\sum_{i=1}^{n} i = n(n+1)/2$, and we want to show that $\sum_{i=1}^{n+1} i = (n+1)(n+2)/2$.
   - Well, $\sum_{i=1}^{n+1} i = (\sum_{i=1}^{n} i) + (n+1) = n(n+1)/2 + (n+1) = (n+1)(n/2+1) = (n+1)(n+2)/2$, so $P(n+1)$ is true.

□

Proof of Theorem 2. Let $P(n)$ be the statement that $n! > 2^n$. We want to show that $P(n)$ holds for every positive integer $n \geq 4$. We use induction.

1. For the base case we have to show that $P(4)$ holds.
   - Well, $P(4)$ says that $4! > 2^n$, which is true since $4! = 4x3x2 = 24 > 2^4 = 16$.
2. For the inductive case we let $n \geq 4$ be an arbitrary number, and assume that $P(n)$ holds. We have to show that $P(n+1)$ holds.
   - So, we have assumed that $n! > 2^n$, and want to show that $(n+1)! > 2^{n+1}$.
   - Well, $(n+1)! = (n+1)n! > (n+1)2^n > 22^n = 2^{n+1}$. The second last inequality holds since $n+1 \geq 5 > 2$.

□