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?

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 $n\}$

Problem 3.

```
def myfun(s): # s is a string (using only ASCII characters)
  if s=='': # empty string
    return 1
  if len(s) == 1:
    return 1
  if s[0] != s[-1]: # first and last characters are different
    return 0
  t = s[1:len(s)-1] # remove the first and last characters
  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]

Problem 4. [Harder]

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

myfun is a program that takes as input a string that only uses characters a and b, 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]

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 $1 \le i < j$.
3. Show that the following strings are in L: $\overbrace{a \cdots a}^{j} \overbrace{a \cdots a}^{j+2}$ where $j \in \mathbb{Z}^+$.

Problem 6.[Extra] In this problem the alphabet is the set of lower-case English letters.

Let L consist of all strings of length a power of 2. E.g., $ab, abaa, aaabaaab$ are strings in L. Show that the following strings are *not* in L:

1. $aaabbbb$
2. $\overbrace{a \cdots a}^{2^j} \overbrace{b \cdots b}^{2^i}$ for some $i, j \in \mathbb{Z}^+$ with $i < j$.

Problem 7.[Extra]

```python
def F(n): # input is a non-negative integer, output is an integer
  if n == 0:
    return 0
  if n == 1:
    return 1
  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 $G(6)$?

Problem 8.[Harder, and fundamental] If $u = u_1 u_2 \cdots u_n$ 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?

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.

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?

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 $A \cap (Z \setminus B) = \emptyset$.

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

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.

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 $g(4, f(4))$?

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

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').

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 $(0,1),(1,2),(2,3),(3,4),\ldots$ 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. $Z = B \setminus A$

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.

[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 $f : X \to Y$.

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

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!)

Problem 19.[Extra] 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 $\cdots \subsetneq S_{-2} \subsetneq S_{-1} \subsetneq S_0 \subsetneq S_1 \subsetneq S_2 \subsetneq \cdots$?

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"?

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.

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.

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.

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"

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. $\square$

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!