The purpose of this tutorial is to familiarise you with some fundamental concepts from discrete mathematics that are useful for designing and analysing data structures and algorithms. For a more complete account, including more detailed explanations and more examples, study Sections 1.2.1, 1.2.2 and 1.2.3 in the recommended reading "Algorithm Design and Applications" by Goodrich and Tamassia. The topics covered there, and which you should revise, are:

1. Summation notation Σ, and formulas for the sum of certain numeric sequences, e.g., $\Sigma_{i=1}^{n} i$ and $\Sigma_{i=1}^{n} 2^i$, etc.

2. Logarithms and exponents

3. Floor and ceiling functions

4. Justification by counterexample

5. Justification by contrapositive

6. Justification by contradiction

7. Justification by induction

8. Basic probability

In addition to that, you should be familiar with basic set notation and basic logic. A set is a group of objects represented as a unit. The objects in a set are called its members or elements. One way to describe a set is to list its elements inside braces, e.g., $S = \{7, 21, 57\}$. The symbols $\in$ and $\notin$ denote set membership and nonmembership. So $7 \in S$ but $8 \notin S$.

Induction:
Induction is a very useful proof technique for proving correctness and bound the running time of an algorithm. Most of the statements about the running times of algorithms that we'll see during this course will hold *for all $n > 0$*, where n is the size of the input. However, this is a claim about an infinite set of numbers and hence we can't prove it by proving it for each element separately.

This is where induction comes in. Induction proves the statement for a *base case*, the smallest element for which the claim must hold ($n = 1$). In some cases, it's helpful to prove an extended base case by explicitly showing that the claim holds for the first few elements, say $n = 1$, $n = 2$, and $n = 3$.

Next, we proceed to the *induction step*, which assumes that the claim holds for all values of n smaller than the current one. This assumption is called the *induction hypothesis*. Using the induction hypothesis, we then show that the claim holds for the next value of n. Typical applications of this are using the induction hypothesis for $n - 1$ to prove the claim for n or using the induction hypothesis for $n/2$ to prove the claim for n when n is even.

As a refresher, work out the problems in this tutorial sheet. If needed, refer to Section 1.2.3 in the class textbook ("Algorithm Design and Applications" by Goodrich and Tamassia) for more detail background on induction and more examples.

Warm-up

Problem 1. Examine the following formal descriptions of sets so that you understand which members they contain. Write a short informal English description of each set.

a) $\{1, 3, 5, 7, ...\}$

b) $\{..., -4, -2, 0, 2, 4, ...\}$

c) $\{n \mid n = 2m \text{ for some } m \in \mathbb{N}\}$

Problem 2. Write formal descriptions of the following sets.

a) The set containing the numbers 1, 10, and 100

b) The set containing all integers that are greater than 5

c) The set containing all natural numbers that are less than 5

d) The set containing nothing at all

**Problem 3.** Let A be the set $\{x, y, z\}$ and B be the set $\{x, y\}$.

a) Is A a subset of B?

b) Is B a subset of A?

c) What is $A \cup B$?

d) What is $A \cap B$?

e) What is $A \times B$?

f) What is the power set of B?

Problem solving

Problem 4. Use induction to show that $\sum_{i=0}^{n} 2^i = 2^{n+1} - 1$.

Problem 5. Recall that one way of defining the Fibonacci sequence is as follows: $F(1) = 1$, $F(2) = 2$, and $F(n) = F(n-2) + F(n-1)$ for $n > 2$.
Use induction to prove that $F(n) < 2^n$.

Problem 6. One line divides the plane into two halves or regions, while two lines divide the plane into four regions. In this problem we study the number of regions in which n lines divide the plane. For simplicity, assume that no three lines intersect at a given point and that no two lines are parallel to one another. Here is a picture of the 7 regions defined by 3 lines.
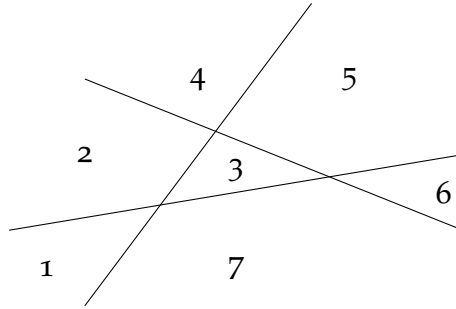
Consider the following algorithm that builds the set of regions by starting from a single region encompassing the whole plane, and iteratively refines the set of regions by processing one line at a time.

```
1:  function BUILD_ARRANGEMENT(lines)
2:      regions ← [new region spanning whole plane]
3:      for line ∈ lines do
4:          old ← []
5:          new ← []
6:          for region ∈ regions do
7:              if line intersects region then
8:                  Append region to old
9:                  left, right ← split region through line
10:                 Append left to new
11:                 Append right to new
12:          for x ∈ old do
13:              Remove x from regions
14:          for x ∈ new do
15:              Append x to regions
16:      return regions
```

a) Prove that at the beginning of the ith iteration of the for loop in Line 3, the line cuts through exactly i regions from the previous iteration.

b) Prove that at the end of the ith iteration of the for loop in Line 3, we have $\frac{i^2+i+2}{2}$ regions.

Remember to use the fact that no three lines intersect in the same point!

Advanced problem solving

Problem 7. Consider the the classical algorithm of Euclid to compute the greatest common divisor of two positive integers a and b

```
1:  function EUCLID(a, b)
2:      a, b ← max(a, b), min(a, b)
3:      while b ≠ 0 do
```

```
4:        a, b ← b, a mod b
5:    return a
```

a) Prove that the algorithm outputs the greatest common divisor of a and b.

b) Prove that $\max(a, b)$ must at least halve in every other iteration.

c) Use the previous fact to argue that the number of iterations is at most $2 \log \max(a, b)$.