Discrete Mathematics - Final

Daniel Rosel

December 6, 2022

Contents

1 Domains **4**

2 Terminology **4**

3 Propositional Logic **4**

3.1 Operators 5
3.1.1 Not 5
3.1.2 Conjunction 5
3.1.3 Disjunction 5
3.1.4 Exclusive OR (XOR) 5
3.1.5 Conditional Statements / Implication 5
3.1.6 Bi-conditionals 6
3.1.7 Order of Operators 6
3.2 DeMorgan's Laws 6
3.3 Fallacies 7
3.4 Converse, Contrapositive, Inverse 7
3.5 Disjunctive Normal Form 7
3.6 Logical Equivalences 7
3.6.1 Identity Laws 7
3.6.2 Denomination Laws 7
3.6.3 Idempotent Laws 8
3.6.4 Double Negation Law 8
3.6.5 Commutative Laws 8
3.6.6 Associative Laws 8
3.6.7 Distributive Laws 8
3.6.8 3.2 8
3.6.9 Absorption Laws 8
3.6.10 Negation Laws 8

3.7 Satisfiability 9

4 Predicate Logic **9**
4.1 Predicates 9
4.2 Quantifiers 9
4.3 Universal Quantifier 9
4.4 Existential Quantifier 10
4.5 Singular Existential Quantifier 10
4.6 Negation 10
4.7 Nesting 11

5 Inference **11**
5.1 Rules of Inference 11
5.1.1 Modus Ponens 11
5.1.2 Modus Tollens 11
5.1.3 Hypothetical Syllogism 12
5.1.4 Disjunctive Syllogism 12
5.1.5 Addition 12
5.1.6 Simplification 12
5.1.7 Conjunction 12
5.1.8 Resolution 12
5.2 Rules of Inference for Quantified Propositions 12
5.2.1 Universal Instantiation 12
5.2.2 Universal Generalization 12
5.2.3 Existential Instantiation 13
5.2.4 Existential Generalization 13

6 Formal Proofs **13**

7 Informal Proofs **13**
7.1 Methods of Proof 13
7.1.1 Direct Proof 14
7.1.2 Indirect: Proof by Contraposition 14
7.1.3 Indirect: Vacuous and Trivial proofs 14
7.1.4 Indirect: Proof by Contradiction 14
7.1.5 Proof by Cases 15
7.1.6 Equivalence Proof 15
7.1.7 Existential Statements Proof 16
7.1.8 Universal Statements Proof 16
7.1.9 Strategy for Proofs 16

8 Set Theory **16**
8.1 Sets . . . 16
8.2 Subsets . . . 17
8.3 Set Cardinality . . . 18
8.4 Power Set . . . 18
8.5 Cartesian Product . . . 18
8.6 Set Operations . . . 19
8.7 Union . . . 19
8.8 Intersection . . . 20
8.9 Difference . . . 20
8.10 Complement . . . 21
8.11 Logical Operations with Sets . . . 22

9 Functions **22**
9.1 Domain, Codomain & Range . . . 23
9.2 One-to-One . . . 23
9.3 Onto . . . 23
9.4 Composite Functions . . . 24
9.5 Inverse Functions . . . 24
9.6 Partial Functions . . . 25
9.7 Practice Questions . . . 25

10 Sequences **26**
10.1 Arithmetic Sequences . . . 26
10.2 Geometric Sequences . . . 26
10.3 Recurrence Relations . . . 27

11 Summations **27**

12 Number Theory **28**

13 Divisibility **28**
13.1 Divisibility Rules . . . 29

14 Modular Arithmetic **29**

15 Integer Representation **29**

16 Binary Representation **30**

17 Hexadecimal Representation **30**

18 Prime Numbers **31**

19 Algorithms **31**
19.1 Complexity of Algorithms 31
19.2 Big O Notation 32

20 Cryptography **33**

21 Relations **33**

22 Computation **33**

1 Domains

Naturals $\mathbb{N}$ $[1, 2, 3, \ldots, \infty)$

Integers $\mathbb{Z}$ $(-\infty, \ldots, -2, 0, 2, \ldots, \infty)$

Rationals $\mathbb{Q}$ Fractions of integers $\{\frac{p}{q} | p \in Z \land q \in Z \land q \neq 0\}$

Reals $\mathbb{R}$ All real numbers

Complex $C All complex numbers

2 Terminology

Theorem A statement we can prove using some axioms

Proof A valid argument/set of arguments that demonstrate a theorem/proposition

Axioms Statements which are assumed to be true, they are in the most basic form

Lemma A small proof that is an element of another larger proof

3 Propositional Logic

A proposition, is a sentence, which can be given some **truth value**. There are two types of propositions:

Tautology A proposition that will always be **true** ($\top$)

Contradiction A proposition that will never be true ($\bot$)

3.1 Operators

There are three main operators which can be used to express any logical expression. These consists of **Not, Conjunction, Disjunction**.

3.1.1 Not

It flips the truth value of a proposition.

$$\neg p, \bar{A}$$

3.1.2 Conjunction

This operators returns a new truth value which is true **if both of the propositions are true**

$$p \wedge q, AB$$

3.1.3 Disjunction

Like the conjunction, it operates on two propositions. It will return true as long as either of the propositions is true.

$$p \vee q, A + B$$

3.1.4 Exclusive OR (XOR)

It is very similar to the **Disjunction** except that it must be exclusively one value that is true.

$$p \oplus q = (p \vee q) \wedge \neg(p \wedge q)$$

3.1.5 Conditional Statements / Implication

A conditional statement consists of two parts, the **premise** and **consequence**. The truth value is **only false when** $T \to F$.

$$p \to q \equiv \neg p \vee q$$

- If p is True, then q must also be True, we say that p is a **sufficient condition** for q. $T \to T$

- If p is False, q could still be True. $F \to T$
- p cannot be true if q is not true. $F \to F$
- p could still be false, even if q is true. $F \to T$

$$((p \to q) \land p) \to q$$

$$((p \to q) \land \neg q) \to \neg p$$

3.1.6 Bi-conditionals

It expressed a **sufficient** and **necessary** conditional relationship between two propositions. It is read as *if and only if p, then q.*

$$p \Leftrightarrow q$$

This operator will return True only if both p and q have equal values. The expression can be represented with the 3 basic operators as:

$$(p \to q) \land (q \to p) \equiv (\neg p \land \neg q) \lor (p \land q)$$

$$p \equiv q = \neg(p \oplus q)$$

3.1.7 Order of Operators

1. $\neg$
2. $\land$
3. $\lor$
4. $\to$
5. $\Leftrightarrow$

3.2 DeMorgan's Laws

$$\neg(p \land q) \equiv \neg p \lor \neg q \tag{1}$$
$$\neg(p \lor q) \equiv \neg p \land q \tag{2}$$

3.3 Fallacies

The fallacies stem from the fact that an implication is true, and the truth value of either p, q.

Affirming the consequent $((p \to q) \land q) \to p$. We falsely assume that the premise is **true** because the consequence is **true**

Nageting the antecedent $((p \to q) \land \neg p) \to \neg q$. We falsely assume that the consequence is **false** because the premise is **false**.

3.4 Converse, Contrapositive, Inverse

Converse $q \to p$

Contrapositive $\neg q \to \neg p$

Inverse $\neg p \to \neg q$

The fallacies are a product of assuming that the **Converse and Inverse** are equivalent to the conditional statement.

3.5 Disjunctive Normal Form

This is a proposition created from the values in the truth table.

3.6 Logical Equivalences

These are various way you can arrive at some conclusion given a few basic propositions.

3.6.1 Identity Laws

$$p \land \top \equiv p$$

$$p \lor \bot \equiv p$$

3.6.2 Denomination Laws

$$p \lor \top \equiv \top$$

$$p \land \bot \equiv \bot$$

3.6.3 Idempotent Laws

$$p \vee p \equiv p$$

$$p \wedge p \equiv p$$

3.6.4 Double Negation Law

$$\neg(\neg p) \equiv p$$

3.6.5 Commutative Laws

$$p \vee q \equiv q \vee p$$

$$p \wedge q \equiv q \wedge p$$

3.6.6 Associative Laws

$$(p \vee q) \vee r \equiv p \vee (q \vee r)$$

$$(p \wedge q) \wedge r \equiv p \wedge (q \wedge r)$$

3.6.7 Distributive Laws

$$p \vee (q \wedge r) \equiv (p \vee q) \wedge (p \vee r)$$

$$p \wedge (q \vee r) \equiv (p \wedge q) \vee (p \wedge r)$$

3.6.8 3.2

3.6.9 Absorption Laws

$$p \vee (p \wedge q) \equiv p$$

$$p \wedge (p \vee q) \equiv p$$

3.6.10 Negation Laws

$$p \vee \neg p \equiv \top$$

$$p \wedge \neg p \equiv \bot$$

3.7 Satisfiability

A proposition is satisfiable if it has at least one case for which it is **true**, it is anything but a **contradiction**.

4 Predicate Logic

With predicate logic, we can break down a complex proposition into predicates and quantifiers.

4.1 Predicates

It is a statement, which is made up of a proposition, with the only difference being that it can take a parameter.

X is faster than me
$=$ Subject + Predicate
$\rightarrow P(x)$

We say that the it is instantiated once it gets evaluated with the passed parameter.

$$P(x_1, x_2, x_3, \ldots, x_n)$$

And once instantiated with variables:

$$P(a_1, a_2, a_3, \ldots, a_n) \equiv p$$

4.2 Quantifiers

With a quantifier, we can specify for which values on a certain domain, a predicate is true of false. There are various ways to express this: (**for all**, **for many**, **for at least one**, **for none**).

4.3 Universal Quantifier

If we have some predicate for which we can say that all the possible values in a certain domain will make it true, we use the universal quantifier.

$P(x)$ for all value of x within its domain
$\forall x P(x)$

- To prove that this quantifier is true, we must demonstrate that it holds true for all the possible values.

- To **dis**-prove the quantifier, it is sufficient to show that there exists just one that does not satisfy the predicate
- If we are considering an empty set, it will always be false.

4.4 Existential Quantifier

$P(x)$ for at least one value of x within its domain

$$\exists x P(x)$$

- To prove this quantifier, we must find at least one value for which the predicate holds true
- To disprove the quantifier, we must show that all the values will not hold for the predicate
- Again, if we consider and empty set, it will always be false

4.5 Singular Existential Quantifier

We can also use the existential quantifier, and with a bit of modification we can use it to specify how many values will satisfy the predict.

There is exactly one x such that $P(x)$

$$\exists! x P(x)$$

The definition for this is as follows:

$$\exists x(P(x) \wedge \forall y(P(y) \Leftrightarrow x = y))$$

We can generalize this to a specific number n for which the predicate is satisfiable.

$$\exists_n x P(x)$$

4.6 Negation

When negating a quantifier, we have to consider both the outside and inside, that being **the quantifier itself** and the **predicate**.

$$\neg \forall x P(x) \equiv \exists x \neg P(x)$$

$$\neg \exists x P(x) \equiv \forall x \neg P(x)$$

4.7 Nesting

If we have more than one variable for which we quantify a predicate, we have to **consider the order**.

$$\exists x \forall y P(x, y) \not\equiv \forall y \exists x P(x, y)$$

5 Inference

A process of arriving at some conclusion through logical steps. This process consists of **arguments** which can either be valid or not. The logical form of an argument is vital, because it is the only way to determine if it is true or not.

$$\begin{array}{r} p \to q \\ \underline{\qquad p} \\ \therefore q \end{array}$$

It is another thing to call an argument **sound**, for this we need to have an argument for which **all premises are true and valid**. **A valid argument implies a tautology**.

5.1 Rules of Inference

An argument form which would generate a valid implication falls under the rules of inference

5.1.1 Modus Ponens

$$\begin{array}{r} p \\ \underline{p \to q} \\ q \end{array}$$

1. If p is true, then q is true
2. p is true
3. Therefore q is true

5.1.2 Modus Tollens

$$\begin{array}{r} \neg q \\ \underline{p \to q} \\ \neg p \end{array}$$

5.1.3 Hypothetical Syllogism

$$\frac{\begin{array}{l} p \to q \\ q \to r \end{array}}{p \to r}$$

5.1.4 Disjunctive Syllogism

$$\frac{\begin{array}{r} p \vee q \\ \neg p \end{array}}{q}$$

5.1.5 Addition

$$\frac{p}{p \vee q}$$

5.1.6 Simplification

$$\frac{p \wedge q}{p}$$

5.1.7 Conjunction

$$\frac{\begin{array}{r} p \\ q \end{array}}{p \wedge q}$$

5.1.8 Resolution

$$\frac{\begin{array}{r} p \vee q \\ \neg p \vee r \end{array}}{\therefore q \vee r}$$

5.2 Rules of Inference for Quantified Propositions

5.2.1 Universal Instantiation

$$\frac{\forall x P(x)}{\therefore P(c)}$$

5.2.2 Universal Generalization

$$\frac{P(c) \text{ for an arbitrary c}}{\therefore \forall x P(x)}$$

5.2.3 Existential Instantiation

$$\frac{\exists x P(x)}{\therefore P(c)}$$

5.2.4 Existential Generalization

$$\frac{P(c) \text{ for an arbitrary c}}{\therefore \exists x P(x)}$$

6 Formal Proofs

This kind of proof is made up of statements which we can infer from previous statements. This we can construct using a table in which we state the logical expression and how we got to it.

$(p \vee q) \to (r \wedge s)$	Premise	(3)
$r \to t$	Premise	(4)
$\neg t$	Premise	(5)
$\neg r$	Modus Tollens from 2 and 3	(6)
$(\neg p \wedge \neg q)$	Modus Tollens from 4 and 1	(7)
$\neg p$	Simplification of 5	(8)

7 Informal Proofs

Because a formal proof can get convoluted, we use informal proofs. The main difference here is that we don't need to provide a step-by-step description of how the proof works; we just show how the conclusion is true by describing a few intermediate steps between the theorem and the corollary. (These are called **premises** and **conclusions** in logic.)

There are various methods of proof we can use. The two primary categories for these proofs are **direct** and **indirect**.

7.1 Methods of Proof

Upon deciding which axioms to use, you then need to use some method of proof. The most common statements we will be proving are implications.

7.1.1 Direct Proof

We want to prove **"If p, then q"**. This is done by assuming that the **premise** is true and then proving that the consequence is also true.

1. We assume p to be true.
2. By applying the laws of inference, we deduce q from p.

7.1.2 Indirect: Proof by Contraposition

If we cannot arrive to the consequence using just the premise, we can use contraposition, which states that $(p \to q) \equiv (\neg q \to \neg p)$.

1. We assume that $\neg q$ is true
2. We find a way to infer that $\neg p$ is true
3. Since this verifies the contrapositive, we have proved that $p \to q$

7.1.3 Indirect: Vacuous and Trivial proofs

If we are saying that $p \to q$ is **True**, we can assume that:

1. p is False (Vacuous proof)
2. q is True (Trivial proof)

It is enough to prove just one of these (as long as we do not accept the premise being True) we can prove the condition to be True.

1. Example If n is an integer and an even number, then $2n$ must be an integer.

   Solution Since an integer multiplied by another integer will also be an integer, we can say that q will be true. This is enough to prove that $p \to q$ is also True.

7.1.4 Indirect: Proof by Contradiction

This can be used to demonstrate any statement p. We say that $\neg p$ implies a contradiction. A nicer way to think about this is by the following:

$$\neg p \to \bot \equiv \neg(\neg p) \vee \bot \equiv p \vee \bot \equiv p$$

If we should use proof by contradiction on an implication, we must prove that $p \wedge \neg q \rightarrow \bot$. This is because in this case, we need to show that the negation of the implication is a contradiction.

$$\neg(p \rightarrow q) \equiv \neg(\neg p \vee q) \equiv p \wedge \neg q \equiv \bot$$

1. Example Show that $\sqrt{2}$ is an irrational number.

   Solution An irrational number is a fraction of 2 integers. We assume that $\sqrt{2}$ is rational. This would mean that $\sqrt{2} = \frac{p}{q}$ where both p, q are integers. The key thing to assume here is that the fraction is in its smallest form. From the previous equation we can get to $p^2 = 2q^2$ which tells us that p is an even number (if the square of a number even, then so is that number$). Further, we can also come to the conclusion that q is even. Since both p, q are even, the fraction is not its simplest form - implying a contradiction.

7.1.5 Proof by Cases

This is a proof which can be used a used as a part of another proof when proving implications. If we have an implication where the premise consists of disjunctions, we can apply the following:

$$(p_1 \vee p_2 \vee \ldots \vee p_n) \rightarrow q \tag{9}$$
$$(p_1 \rightarrow q) \wedge (p_2 \rightarrow q) \wedge \ldots \wedge (p_n \rightarrow q) \tag{10}$$

This proof is very helpful when we want to prove that a condition is satisfiable for a group of elements.

1. Example Prove that if n is not a multiple of 3, $2n$ is not a multiple of 3.

   Solution To define a value as not a multiple of 3, we have two formulas $n = 3k + 1 \quad n = 3k + 2$. We can reformulate the proposition by saying $(n = 3k + 1) \vee (n = 3k') \rightarrow \neg(2n = 3k'')$. We then have to show that both cases imply the consequence.

7.1.6 Equivalence Proof

As previously mentioned $p \leftrightarrow q = p \equiv q$, so if we want to prove some equivalence, we can use $(p \rightarrow q) \wedge (q \rightarrow p)$.

7.1.7 Existential Statements Proof

We have two strategies for this.

Constructive Proof We show an example that verifies a predicate and apply existential generalization to prove the proposition.

Non Constructive Proof We show that there exists an element which validates the predicate but we dont specify which one. This is most commonly proof by contradiction.

7.1.8 Universal Statements Proof

Proving We show that a predicate holds for an arbitrary value and then apply universal generalization.

Counterexamples To disprove a universal statement we need to find a counterexample. This action is a process of negation, making the US an ES.

7.1.9 Strategy for Proofs

So should you use? It comes down to trial and error, but also practice and intuition.

Trial and error If direct proof doesnt work, try indirect lol.

Complex propositions If there are complex propositions in an implication, they quite possibly can be negated, allowing use to use an **indirect method**.

Using proof by contradiction with conditional statements We show that $\neg(p \to q)$ implies $\bot$.

8 Set Theory

8.1 Sets

A set is a collection of objects. We use the notation $\{a, b, c\}$ to denote a set. We can also use the notation $\{x \in S \mid P(x)\}$ to denote a set of elements x in S that satisfy the predicate $P(x)$. A set can be empty, which is denoted by $\emptyset$.

A more concrete definition of a set using quantifiers is as follows:

$$\forall x((x \in S) \leftrightarrow P(x))$$

We also often use **interval definitions** to define sets. These are defined as follows:

$$\{x \in \mathbb{R} \mid a \leq x \leq b\}$$

A common way to define a interval is with brackets and parenthesis. For example, (a, b) is an open interval, while $[a, b]$ is a closed interval.

Two sets are equal if they have the same elements. We denote this as $A = B$. A more rigorous definition of this is as follows:

$$A = B \equiv \forall x(x \in A \leftrightarrow x \in B)$$

The equality of two sets can also be defined using subsets:

$$A = B \equiv A \subseteq B \land B \subseteq A$$

A way to disprove the equality of two sets is by showing that they have at least one element that is not in the other set. This is called a **counterexample**. It can be represented with an existential quantifier and XOR: $\exists x(x \in A \oplus x \in B)$.

Some important elements are the aforementioned empty set $\emptyset$ and the universal set $\mathbb{U}$. The universal set is the set of all elements in a domain. The empty set is the set that contains no elements. The Domains, mentioned at the beginning are also sets.

8.2 Subsets

A subset is a set that is contained in another set. We denote this as $A \subseteq B$. A more rigorous definition of this is as follows:

$$A \subseteq B \equiv \forall x(x \in A \rightarrow x \in B)$$

For all sets, it is true that $A \subseteq A$ and $\emptyset \subseteq A$. Which means that every set is a subset of itself and the empty set is a subset of every set. We can also say that $A \subseteq B$ and $B \subseteq A$ implies $A = B$. This is called the **subset equality**.

What can be confusing is the proper subset. A proper subset is a subset that is not equal to the set. We denote this as $A \subset B$. A more rigorous definition of this is as follows:

$$A \subset B \equiv A \subseteq B \land A \neq B$$

A quantified version of this is as follows:

$$\forall x((x \in A) \rightarrow (x \in B) \land (A \neq B))$$

8.3 Set Cardinality

This is not the absolute value of a set. It is the number of elements in a set. We denote this as $|A|$. A more rigorous definition of this is as follows:

$$|A| \equiv \#\{x \in A \mid x \in A\}$$

8.4 Power Set

The power set is the set of all subsets of a set. We denote this as $P(A)$. What is meant by all subsets is that the power set contains the empty set and the universal set aswell as all the subsets of the set. A more rigorous definition of this is as follows:

$$P(A) \equiv \{B \mid B \subseteq A\}$$

The cardinality of a power set is $2^{|A|}$.

An example of this is the power set of $\{1, 2, 3\}$ is $\{\emptyset, \{1\}, \{2\}, \{3\}, \{1, 2\}, \{1, 3\}, \{2, 3\}, \{1, 2, 3\}\}$.

What is important to remember is that the power set of a power set is the original set. This is because the power set of a set is the set of all subsets of a set. So the power set of the power set is the set of all subsets of all subsets of a set. This is the original set.

8.5 Cartesian Product

The cartesian product is the set of all ordered pairs of elements from two sets. We denote this as $A \times B$. In other words, the cartesian product is the set of all possible combinations of elements from two sets. A more rigorous definition of this is as follows:

$$A \times B \equiv \{(a, b) \mid a \in A \land b \in B\}$$

The cardinality of the cartesian product is $|A| \times |B|$.

An example of this is the cartesian product of $\{1, 2\}$ and $\{3, 4\}$ is $\{(1, 3), (1, 4), (2, 3), (2, 4)\}$.

The relation of the Cartesian product to the power set is that the power set of a cartesian product is the cartesian product of the power sets. This is because the power set of a set is the set of all subsets of a set. So the power set of the cartesian product is the set of all subsets of all ordered pairs of elements from two sets. This is the cartesian product of the power sets. A mathematical proof of this is as follows:

$$P(A\times B) = \{C \mid C \subseteq A\times B\} = \{C \mid C \subseteq \{(a,b) \mid a \in A \wedge b \in B\}\} = \{C \mid C \subseteq \{(a,b) \mid a \in P(A) \wedge b \in P($$

Or simply: $P(A \times B) = P(A) \times P(B)$.

8.6 Set Operations

Set operations are operations that can be performed on sets. The most common set operations are union, intersection, difference, and complement.

It is important to remember that if we have an expression like $AA \cup B$, something that could be tricky is that the order of operations is not the same as in arithmetic. In arithmetic, we would do the multiplication and division first, then addition and subtraction. In set operations, we do the complement first, then the difference, then the intersection, and finally the union. So the expression $AA \cup B$ would be read as $A(A \cup B)$.

Once again, **the order of operations for sets** is:

1. Complement
2. Difference
3. Intersection
4. Union

8.7 Union

Simply explained, the union of two sets is the set of all elements that are in either set. We denote this as $A \cup B$. A more rigorous definition of this is as follows:

$$A \cup B \equiv \{x \mid x \in A \vee x \in B\}$$

And using quantifiers:

$$\forall((x \in A \cup B) \equiv (x \in A \vee x \in B))$$

The cardinality of the union is $|A| + |B| - |A \cap B|$. The reason why we have to subtract the cardinality of the intersection is because the intersection is counted twice. Once in A and once in B.

An example of this is the union of $\{1, 2\}$ and $\{2, 3\}$ is $\{1, 2, 3\}$. The reason why this is the union is because $1 \in A$, $2 \in A$, $2 \in B$, and $3 \in B$. So $1, 2, 3$ are all in the union.

8.8 Intersection

The intersection of two sets is the set of all elements that are in both sets. We denote this as $A \cap B$. A more rigorous definition of this is as follows:

$$A \cap B \equiv \{x \mid x \in A \land x \in B\}$$

The cardinality of the intersection is $|A \cap B|$. What is important to remember is that the intersection of a set with itself is the set itself. This is because the intersection of a set with itself is the set of all elements that are in the set and the set itself. So the intersection of a set with itself is the set itself.

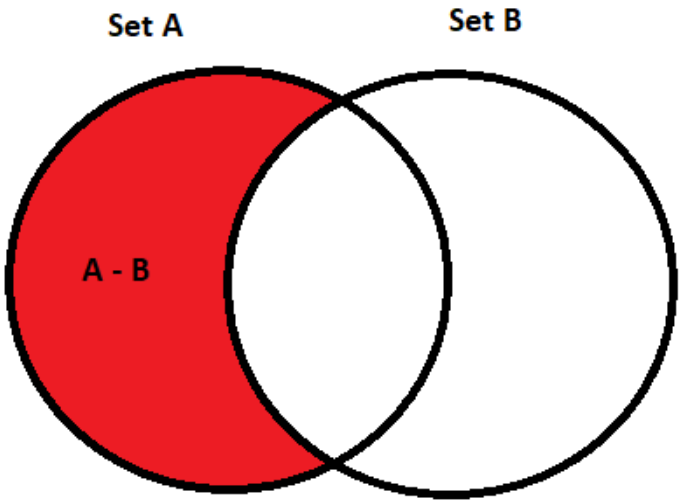
An example of this is the intersection of $\{1, 2\}$ and $\{2, 3\}$ is $\{2\}$. The reason why this is the intersection is because $2 \in A$ and $2 \in B$. So 2 is in the intersection.

8.9 Difference

The difference of two sets is the set of all elements that are in the first set but not the second set. We denote this as $A \setminus B$. A more rigorous definition of this is as follows:

$$A \setminus B \equiv \{x \mid x \in A \land x \notin B\}$$

A visual representation of this is as follows:



The cardinality of the difference is $|A \setminus B|$ or $|A| - |A \cap B|$. The reason why we have to subtract the cardinality of the intersection is because the intersection is counted twice. Once in A and once in B.

An example of this is the difference of $\{1, 2\}$ and $\{2, 3\}$ is $\{1\}$. The reason why this is the difference is because $1 \in A$ and $1 \notin B$. So 1 is in the difference.

8.10 Complement

The complement of a set is the set of all elements that are not in the set. We denote this as A^c. A more rigorous definition of this is as follows:

$$A^c \equiv \{x \mid x \notin A\}$$

In terms of the universal set, the complement of a set is the set of all elements that are not in the set and are in the universal set. It is denoted as $A^c = U \setminus A$. A more rigorous definition of this is as follows:

$$A^c \equiv \{x \mid x \in U \land x \notin A\}$$

The cardinality of the complement is $|U| - |A|$. The reason why we have to subtract the cardinality of the set is because the set is counted twice. Once in U and once in A.

An example of this is the complement of $\{1, 2\}$ is $\{3, 4\}$. The reason why this is the complement is because $3 \in U$ and $3 \notin A$. So 3 is in the complement. This of course depends on the universal set being $\{1, 2, 3, 4\}$.

8.11 Logical Operations with Sets

We can relate sets to logical operations. The most common logical operations are conjunction, disjunction, and negation. We can relate these to sets as follows:

$A \cup B \sim p \vee q$
$A \cap B \sim p \wedge q$
$\bar{A} \sim \neg p$

The reason why we can do this is that the definition of a set, is based on a predicate. So we can relate the set to the predicate and the predicate to the logical operation. For example, the definition of a set is as follows:

$$A \equiv \{x \mid x \in A\}$$

So we can relate the set to the predicate as follows:

$$A \equiv \{x \mid x \in A\} \equiv \{x \mid p(x)\}$$

9 Functions

A function, is the assignment of a single value $b \in B$ to each element a in a set A. We denote this as $f : A \to B$. We can also call a function a mapping, transformation or a black box (in some cases even an oracle). A more rigorous definition of this is as follows:

$$f : A \to B \equiv \{f(a) \mid a \in A\}$$

We are mostly interested in what makes a function a function. So we will define what makes a function a function. A function is a function if the following conditions are met:

1. The domain of the function is a set
2. The range of the function is a set
3. The function is one-to-one
4. The function is onto

A mathematical way to express this is as follows:

$$\forall a \in A \exists! b \in B \wedge f(a) = b$$

The reasons why some assignments might not be functions is that not all elements in A are assigned to an element in B. This is called a partial function. Another reason why some assignments might not be functions is that not all elements in B are assigned to an element in A. This is called a surjection. Another reason why some assignments might not be functions is that not all elements in A are assigned to a unique element in B. This is called a bijection.

How can we tell if a function is a function? We can tell if a function is a function by checking if the function is one-to-one and onto. If the function is one-to-one and onto, then the function is a function. If the function is not one-to-one and onto, then the function is not a function.

9.1 Domain, Codomain & Range

If $f(a) = b$, then we call a the **pre-image** and b the **image** of f. We call A the **domain** of f and B the **codomain** of f. We call $f(A)$ the **range** of f.

The main difference between the **codomain** and the **range** is that the **codomain** is the set of all possible images of f and the **range** is the set of all images of f.

9.2 One-to-One

A function is one-to-one if each element in the range is assigned to a unique element in the domain. We denote this as $f : A \to B$ is one-to-one. A more rigorous definition of this is as follows:

$$f : A \to B \text{ is one-to-one } \equiv \forall a_1, a_2 \in A \wedge f(a_1) = f(a_2) \implies a_1 = a_2$$

or simply: $f(a_1) = f(a_2) \implies a_1 = a_2$.

If we want to check if a functions is one-to-one, we have to test if $f(a_1) = f(a_2) \implies a_1 = a_2$ for all $a_1, a_2 \in A$. If this is true for all $a_1, a_2 \in A$, then the function is one-to-one.

9.3 Onto

A function is onto if each element in the codomain is assigned to an element in the domain. We denote this as $f : A \to B$ is onto. This is to say that for

each element in the codomain, there is an element in the domain that maps to it. A more rigorous definition of this is as follows:

$$f : A \to B \text{ is onto } \equiv \forall b \in B \exists a \in A \wedge f(a) = b$$

or simply: $\forall b \in B \exists a \in A \wedge f(a) = b$.

To see if a function is onto, we have to test if $\forall b \in B \exists a \in A \wedge f(a) = b$. If this is true for all $b \in B$, then the function is onto. For example if $f : \{1, 2\} \to \{1, 2, 3\}$, then $f(1) = 1$, $f(2) = 2$ and $f(3) = 3$. So f is onto. If $f : \{1, 2\} \to \{1, 2\}$, then $f(1) = 1$, $f(2) = 2$ and $f(3) = 3$. So f is not onto. In this case, the function is not onto because 3 is not assigned to an element in the domain.

9.4 Composite Functions

We can combine functions to create composite functions. We can combine functions by applying one function to the output of another function. We denote this as $f \circ g$. We can also call this function composition. A more rigorous definition of this is as follows:

$$f \circ g \equiv \{f(g(a)) \mid a \in A\}$$

If the **codomain** of g is the **domain** of f, then we can combine g and f to create a composite function. We denote this as $f \circ g : A \to C$.

When it comes to functions properties after composition we have the following properties:

1. $f \circ g$ is one-to-one if f is one-to-one and g is one-to-one
2. $f \circ g$ is onto if f is onto and g is onto

If some of the functions are not one-to-one or onto, then the composite function is not one-to-one or onto.

9.5 Inverse Functions

An inverse function is a function that undoes the effects of another function. We denote this as f^{-1}. The definition is as follows:

$$f^{-1} \equiv \{a \mid f(a) = b\}$$

If f is one-to-one and onto, then f^{-1} is also one-to-one and onto.

9.6 Partial Functions

A partial function is a function that is not one-to-one or onto. We denote this as $f : A \to B$ is a partial function. Definition:

$$f : A \to B \text{ is a partial function } \equiv \exists a_1, a_2 \in A \wedge f(a_1) = f(a_2) \wedge a_1 \neq a_2$$

or simply: $\exists a_1, a_2 \in A \wedge f(a_1) = f(a_2) \wedge a_1 \neq a_2$. If this is true for some $a_1, a_2 \in A$, then f is a partial function.

9.7 Practice Questions

1. Let $f : \mathbb{R} \to \mathbb{R}$ be defined by $f(x) = x^2$. Is f one-to-one? Is f onto? Is f a partial function?

   Solution f is one-to-one because $f(x) = f(y) \implies x = y$. f is not onto because $f(x) = x^2$ and x^2 is not in the domain. f is not a partial function because $f(x) = f(y) \implies x = y$.

2. Let $f : \mathbb{R} \to \mathbb{R}$ be defined by $f(x) = x^2$. Is f^{-1} one-to-one? Is f^{-1} onto? Is f^{-1} a partial function?

   Solution f^{-1} is one-to-one because $f^{-1}(x) = f^{-1}(y) \implies x = y$. f^{-1} is onto because $f^{-1}(x) = x^2$ and x^2 is in the domain. f^{-1} is not a partial function because $f^{-1}(x) = f^{-1}(y) \implies x = y$.

3. Let $f : \mathbb{R} \to \mathbb{R}$ be defined by $f(x) = x^2$. Is $f \circ f$ one-to-one? Is $f \circ f$ onto? Is $f \circ f$ a partial function?

   Solution $f \circ f$ is one-to-one because f is one-to-one and f is one-to-one. $f \circ f$ is onto because f is onto and f is onto. $f \circ f$ is not a partial function because $f \circ f$ is one-to-one and $f \circ f$ is onto.

4. Let $f : \mathbb{R} \to \mathbb{R}$ be defined by $f(x) = x^2$. Is $f \circ f^{-1}$ one-to-one? Is $f \circ f^{-1}$ onto? Is $f \circ f^{-1}$ a partial function?

   Solution $f \circ f^{-1}$ is one-to-one because f is one-to-one and f^{-1} is one-to-one. $f \circ f^{-1}$ is onto because f is onto and f^{-1} is onto. $f \circ f^{-1}$ is not a partial function because $f \circ f^{-1}$ is one-to-one and $f \circ f^{-1}$ is onto.

10 Sequences

A sequence is a set of elements, which can be continued by using intuition or mathematical patterns. We denote this as $\{a_n \mid n \in \mathbb{N}\}$. We will mostly focus on **arithmetic** sequences and **geometric** sequences.

10.1 Arithmetic Sequences

An arithmetic sequence is a sequence where the difference between two consecutive terms is constant. We denote this as $\{a_n \mid n \in \mathbb{N}\}$ where $a_n = a_1 + (n-1)d$ for some $a_1, d \in \mathbb{R}$. We can also write this as $a_n = a_1 + nd$, where n is the **nth** term of the sequence and d is the **common difference**.

An arithmetic sequence is defined by the following properties:

1. $a_n = a_1 + (n-1)d$
2. $a_n = a_1 + nd$

This sequence can also be thought of as a linear function. We can write this as $a_n = f(n)$ where $f(n) = a_1 + (n-1)d$. The main elements of a sequence are the **first term** and the **common difference**.

The expansion for the arithmetic sequence is as follows:

$$a_1 = a_1 + 0d = a_1 \tag{11}$$
$$a_2 = a_1 + 1d = a_1 + d \tag{12}$$
$$a_3 = a_1 + 2d = a_1 + 2d \tag{13}$$
$$a_4 = a_1 + 3d = a_1 + 3d \tag{14}$$
$$\vdots = \vdots \tag{15}$$
$$a_n = a_1 + (n-1)d = a_1 + (n-1)d \tag{16}$$

10.2 Geometric Sequences

A geometric sequence is a sequence where the ratio between two consecutive terms is constant. We denote this as $\{a_n \mid n \in \mathbb{N}\}$ where $a_n = a_1 r^{n-1}$ for some $a_1, r \in \mathbb{R}$. We can also write this as $a_n = a_1 r^n$, where n is the **nth** term of the sequence and r is the **common ratio**.

An expansion of such as sequence can look something like this:

$$a_1 = a_1 \tag{17}$$
$$a_2 = a_1 r \tag{18}$$
$$a_3 = a_1 r^2 \tag{19}$$
$$a_4 = a_1 r^3 \tag{20}$$
$$\vdots = \vdots \tag{21}$$
$$a_n = a_1 r^{n-1} \tag{22}$$

10.3 Recurrence Relations

A recurrence relation is a sequence that is defined by a formula. We denote this as $\{a_n \mid n \in \mathbb{N}\}$ where $a_n = f(a_{n-1})$ for some $f : \mathbb{R} \to \mathbb{R}$. It takes the previous term and applies a function to it to get the next term.

We can be told to find the closed formula for a sequence. This is a formula that can be used to find the nth term of a sequence.To do this, we need to use iteration and induction. We can use the following rules to find the closed formula for a sequence:

1. $a_n = a_1 + (n-1)d$ where $a_1, d \in \mathbb{R}$
2. $a_n = a_1 r^{n-1}$ where $a_1, r \in \mathbb{R}$
3. $a_n = f(a_{n-1})$ where $f : \mathbb{R} \to \mathbb{R}$
4. $a_n = f(a_{n-1}, a_{n-2})$ where $f : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$

In short, we need to find the pattern in the sequence and use that to find the closed formula.

11 Summations

A summation is a way to add up a sequence of numbers. We denote this as $\sum_{i=1}^{n} a_i$. There are many ways to define a summation. To find the sum of a sequence, we can use the following rules:

1. $\sum_{i=1}^{n} i = \frac{n(n+1)}{2}$
2. $\sum_{i=1}^{n} i^2 = \frac{n(n+1)(2n+1)}{6}$
3. $\sum_{i=1}^{n} i^3 = \frac{n^2(n+1)^2}{4}$

4. $\sum_{i=1}^{n} C * f(i) = \frac{C}{1-r}$

If we subtract 1 from the upper bound, we can find the sum of the sequence. For example, $\sum_{i=1}^{n} i = \sum_{i=0}^{n-1} i + n = \sum_{i=0}^{n-1} i + \sum_{i=n}^{n} i = \sum_{i=0}^{n-1} i + n = \frac{n(n-1)}{2} + n = \frac{n(n+1)}{2}$.

If we have nested summations, we can expand the far-right summation and include it in each iteration of the left summation.

12 Number Theory

13 Divisibility

A number is divisible by another number if the remainder is 0. We denote this as $a \mid b$ where a is the divisor and b is the dividend. We can also write this as $a = q \cdot d + r$ where r is the remainder and q is the quotient. It is important to note that r is always less than d and always positive.

The elements in the primary equation are as follows:

1. a is the dividend
2. d is the divisor
3. q is the quotient
4. r is the remainder

From this equation we gain two new operations, the **division** operation and the **modulus** operation. We can use these operations to find the remainder and quotient of a division problem.

If we perform division of a negative number, we cannot have a negative remainder. To solve this, we can add the divisor to the dividend until the remainder is positive. This is called the **floor division** operation. In other words, we want to 'overshoot' with the dividend and then compensate with the remainder.

- Division Function: This function is onto but not one-to-one
- Modulus Function: This function neither one-to-one or onto

13.1 Divisibility Rules

We can use divisibility rules to determine if a number is divisible by another number. The rules are as follows:

1. If a divides both b and c, then a divides $b + c$. Expression: $a \mid b \land a \mid c \Rightarrow a \mid (b + c)$
2. If a divides b, then a divides any multiple of b. Expression: $a \mid b \implies a \mid bm$ for some $m \in \mathbb{N}$
3. If a divides b and b divides c, then a divides c. Expression: $a \mid b \land b \mid c \Rightarrow a \mid c$

14 Modular Arithmetic

Modular arithmetic uses the modulus operation to find the remainder of a division problem. We use the notation $a \equiv b \pmod{m}$ to denote that a is congruent to b modulo m.

If we want to identify if two integers are congruent modulo m, we can test this in the following way:

1. $a \equiv b \pmod{m} \land b \equiv c \pmod{m} \Rightarrow a \equiv c \pmod{m}$

In other words, if the remainder of the modulus operation is the same, then the two numbers are congruent modulo m.

15 Integer Representation

Integers can be represented in many ways. In every day life, we use the base 10 system. This means that we use the digits 0-9 to represent numbers. We can also use the base 2 system. This means that we use the digits 0-1 to represent numbers. We can also use the base 16 system. This means that we use the digits 0-9 and A-F to represent numbers. We can also use the base 26 system. This means that we use the digits A-Z to represent numbers.

If we want to convert the value of base b to base b', we can use the following formula:

$$n = a_k b^k + a_{k-1} b^{k-1} + \cdots + a_1 b + a_0$$

The general algorithm for calculating the number representation is as follows:

1. Divide the number by the base
2. Take the remainder and add it to the list of digits
3. Repeat until the number is 0

In mathematics:

$$n = a_0 + bq_0 \tag{23}$$
$$n = a_1 + b(a_1 + bq_1) \tag{24}$$
$$n = a_2 + b(a_1 + b(a_2 + bq_2)) \tag{25}$$

16 Binary Representation

Binary representation is the representation of numbers in base 2. We can use the following rules to convert from base 10 to base 2:

1. Divide the number by 2
2. Take the remainder and add it to the list of digits
3. Repeat until the number is 0

To then put together the binary string we take each bit in reverse order and put it together. For example, the binary representation of 13 is 1101.

To make life a bit easier, we can use a table to convert from base 10 to base 2. The table is made up of the powers of 2 and the digits 0-1.

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
128	64	32	16	8	4	2	1

17 Hexadecimal Representation

Hexadecimal representation is the representation of numbers in base 16. We can use the following rules to convert from base 10 to base 16:

1. Divide the number by 16
2. Take the remainder and add it to the list of digits
3. Repeat until the number is 0

To then put together the hexadecimal string we take each digit in reverse order and put it together. For example, the hexadecimal representation of 15816 is 3DC8. The steps taken to get to this number are as follows:

1. 15816 / 16 = 988 remainder 8
2. 988 / 16 = 61 remainder 12 (12 is C in hexadecimal)
3. 61 / 16 = 3 remainder 13 (13 is represented as D in hexadecimal)
4. 3 / 16 = 0 remainder 3
5. 3DC8

18 Prime Numbers

19 Algorithms

An algorithm is a collection of instructions that are made to solve a problem. To create any algorithm, we need three basic components:

Sequence A sequence is a list of instructions that are executed in order

Selection A selection is a list of instructions that are executed based on a condition

Iteration An iteration is a list of instructions that are executed multiple times

19.1 Complexity of Algorithms

The complexity of an algorithm is given by the relation between the input size and the number of steps required to solve the problem. We can use the following notations to describe the complexity of an algorithm:

$O(n)$ This notation describes the worst case scenario for an algorithm. This means that the algorithm will take at least this amount of time to solve the problem.

$\Omega(n)$ This notation describes the best case scenario for an algorithm. This means that the algorithm will take at most this amount of time to solve the problem.

$\Theta(n)$ This notation describes the average case scenario for an algorithm. This means that the algorithm will take this amount of time to solve the problem.

We will not need omega and theta notation for the exam, but it is good to know.

19.2 Big O Notation

Big O notation is used to describe the worst case scenario for an algorithm. We can use the following rules to determine the complexity of an algorithm:

1. $O(C)$ :: This means that the algorithm will take a constant amount of time to solve the problem. This means that the algorithm will take the same amount of time to solve the problem regardless of the input size.

2. $O(n)$ :: This means that the algorithm will take a linear amount of time to solve the problem. This means that the algorithm will take a linear amount of time to solve the problem. This means that the algorithm will take n amount of time to solve the problem.

3. $O(n^2)$ :: This means that the algorithm will take a quadratic amount of time to solve the problem. This means that the algorithm will take n^2 amount of time to solve the problem.

4. $O(2^n)$ :: This means that the algorithm will take an exponential amount of time to solve the problem. This means that the algorithm will take 2^n amount of time to solve the problem.

5. $O(n!)$ :: This means that the algorithm will take a factorial amount of time to solve the problem. This means that the algorithm will take $n!$ amount of time to solve the problem.

The idea of this notation is that we can use it to compare the complexity of different algorithms. For example, if we have two algorithms that solve the same problem, we can use big O notation to determine which algorithm is better. If one algorithm is $O(n)$ and the other is $O(n^2)$, then the first algorithm is better than the second algorithm.

20 Cryptography

21 Relations

22 Computation