Table of Contents

1 Number Theory:

1.1 Learning Objectives

- Compute the prime factorisation of integers and use it to compute the lowest common multiple and greatest common divisor of two numbers.
- Apply the Euclidean algorithm for finding the greatest common divisor of two numbers.
- Convert between number bases representations of numbers.
- Compute the values of large powers of a number in modular arithmetic.

1.2 Division:

- If a and b are integers, with $a \neq 0$, a divides b if there is an integer k such that $b = ak$
  - a is a factor (or divisor) of b.
  - b is a multiple of a.
  - denoted $a \mid b$.

1.3 Prime and Compositive Numbers:

- Positive integer $p > 1$ is…
- **Prime:** if it is only divisible by itself and 1
- **Compositive:** if it has other positive factors other than itself and 1.
- **Even:** If the number can be divided into two equal parts

1.4 Fundamental Theorem of Arithmetic:

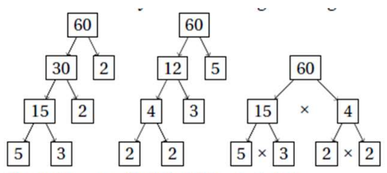
- **Prime Factorization of n**: Every positive integer n greater than 1 can be written uniquely as a product of primes.
  - Example: 60 = 5 x 3 x 2 x 2
- Since the only prime factors of a prime number are 1 and itself, and 1 is not prime, then the prime factors contains itself.
  - Example: 131 = 131

1.5 Prime Factorization:

- **Convert** to prime factorization form
  - Step 1: Find prime numbers < $\sqrt{n}$
  - Step 2: Divide result from greatest to smallest
  - Note: no need to look at prime numbers > $\sqrt{n}$, since these number must need to be multiplied by a smaller number < $\sqrt{n}$.

1.6 Factor Tree:

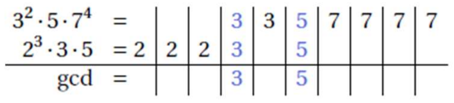
- Graphical representation of prime factorization procedure.



Every time we find that $60 = 5 \cdot 3 \cdot 2 \cdot 2$.

1.7 Greatest Common Divisor (GCD):

- $\mathbf{GCD}(\boldsymbol{a}, \boldsymbol{b})$ is the largest integer d such that d | a and d | b.
  - The numbers in common are those that divide both numbers.
  - Multiplying these will get a number that divides both.
- Example: Find $gcd(3^2 \times 5 \times 7^4, 2^3 \times 3 \times 5) = 3 \times 5$



$3^2 \cdot 5 \cdot 7^4$	=				3	3	5	7	7	7	7
$2^3 \cdot 3 \cdot 5$	=	2	2	2	3		5				
gcd	=				3		5				

- **Find GCD Method:**
  - Step 1: convert numbers to prime factorization form
  - Step 2: take the minimum of the powers.
  - Note: if the prime factorization of any number has factors missing from the other, add as n^0

$$\begin{aligned}&\gcd(3^2\times5\times7^4, 2^3\times3\times5^3)\\&=\gcd(2^0\times3^2\times5^1\times7^4, 2^3\times3^1\times5^3\times7^0)\\&=2^{\min\{0,3\}}\times3^{\min\{2,1\}}\times5^{\min\{1,3\}}\times7^{\min\{4,0\}}\\&=2^0\times3^1\times5^1\times7^0\\&=3\times5\end{aligned}$$

1.8 Lowest Common Multiple (LCM):

- **lcm(a,b)** is the lowest positive interger k such that a | k and b | k.
  - prime factorization of any multiple of the first number must contain all the factors of that number.
  - A factor of both numbers must contain all the factors of both
  - factors in common appears once
- Example: Find $lcm(3^2 \times 5 \times 7^4, 2^3 \times 3 \times 5) = 2\times2\times2\times3\times5\times7\times7\times7\times7$

$$\mathrm{lcm}(3^2\times5\times7^4, 2^3\times3\times5) = \overbrace{2\times2\times2\times\underbrace{3\times5}_{}}^{\textit{First number}}\underbrace{{}\times3\times7\times7\times7\times7}_{\textit{Second number}}$$

- **Find LCM Method:**
  - Step 1: convert numbers to prime factorization form
  - Step 2: take the maximum of the powers.
  - Note: if the prime factorization of any number has factors missing from the other, add as n^0

$$\begin{aligned}&lcm(3^2\times5\times7^4, 2^3\times3\times5^3)\\&=2^{\max\{0,3\}}\times3^{\max\{2,1\}}\times5^{\max\{1,3\}}\times7^{\max\{4,0\}}\\&=2^3\times3^2\times5^3\times7^4.\end{aligned}$$

1.9 Euclidean Algorithm:

- Consider a and b where $a > b$. The Euclidian algorithm states that when we divide b by a, we get a multiplier m_0 and a remainder r_0. Where $\gcd(b,a) = \gcd(a, r_0)$.
- Keep dividing a by the remainder until reaching 0. At this point, the remainder from the previous step will be the GCD.
- From a different angel, the progression below will continue until we reach $r_{k-1} = m_{ak+1} \times r_k + 0$. We return r_k as the GCD of a and b.

$$\begin{array}{rcl} b &=& m_0\times a+r_0\\ a &=& m_1\times r_0+r_1\\ r_0 &=& m_2\times r_1+r_2\\ \vdots &=& \vdots\\ r_j &=& m_{j+2}\times r_{j+1}+r_{j+2}\\ \vdots &=& \vdots \end{array}$$

- Two forms to express a statement which is the arrangement of each other:

$$\boldsymbol{a \div b = cRd}$$
$$\boldsymbol{a - b \times c = d}$$

1.9.1 Recursive Version

- Step 1: Divide b by a.
- Step 2: Divide a by remainder
- Step 3: Continue until the remainder is 0.

```
if $a|b$ then
    $\gcd(b,a) = a$
else
    Solve the division $b = m \times a + r$;
    $\gcd(b,a) = \gcd(a,r)$.
end
```

- Example: gcd(291,201) = 3

$$291 \div 201 = 1R90$$
$$201 \div 90 = 2R21$$
$$90 \div 21 = 4R6$$
$$21 \div 6 = 3R3$$
$$6 \div 3 = 2R0$$

1.9.2 Imperative Version

```
Set $r_0 \leftarrow b$ and $r_1 \leftarrow a$;
repeat
    Solve the division $r_1 = m \times r_0 + r$;
    Set $r_0 \leftarrow r_1$ and $r_1 \leftarrow r$;
until $r_1 | r_0$;
return $r_0$.
```

1.10 Base Systems:

- Our number system is base 10.
- Computers use base 2 (binary), base 8 (octal), and base 16 (hexadecimal).
- Other systems use different bases. E.g. we divide a year into 12 months, and our days into 24 hours, of 60 minutes each.
- The base n representation of a system, for a positive integer $n > 2$, is the place-value representation of a number using n symbols (or digits).
- Base 10 (or decimal) uses 10 symbols (0 to 9).
- Base 2 (or binary) representation uses 2 symbols (0 and 1).
- Base 20 representation uses 20 symbols.

1.10.1 Base 10

- In the **base 10** system, we express numbers based on increasing powers of **10**.
- Example: $2473 = (2 \times 10^3) + (0 \times 10^2) + (1 \times 10^1) + (7 \times 10^0)$.

2	0	1	7
10^3	10^2	10^1	10^0
Thousands	Hundreds	Tens	Ones

1.10.2 Base 5

- In the **base 5** system, we express numbers based on increasing powers of **5**.
- Example: $1432_5 = (2 \times 5^3) + (4 \times 5^2) + (3 \times 5^1) + (2 \times 5^0)$.

1	4	3	2
5^3	5^2	5^1	5^0
125	25	5	1

1.10.3 Base 2

- In the **base 2** system, we express numbers based on increasing powers of 2.
- Example: $100101_2 = (1 \times 2^5) + (0 \times 2^4) + (0 \times 2^3) + (1 \times 2^2) + (0 \times 2^1) + (1 \times 2^0)$.

1	0	0	1	0	1
2^5	2^4	2^3	2^2	2^1	2^0
32	16	8	4	2	1

1.11 Converting x_n to Base 10:

- Multiply out the bases as normal
- Example: $1432_5 = (2 \times 5^3) + (4 \times 5^2) + (3 \times 5^1) + (2 \times 5^0) = (2 \times 125) + (4 \times 25) + (3 \times 5) + (2 \times 1) = 242_{10}$

- Example: $100101_2 = (1 \times 2^5) + (0 \times 2^4) + (0 \times 2^3) + (1 \times 2^2) + (0 \times 2^1) + (1 \times 2^0) = (1 \times 32) + (0 \times 16) + (0 \times 8) + (1 \times 4) + (0 \times 2) + (1 \times 1) = 37_{10}$

1.12 Converting x_{10} to Base n:

1.12.1 Method 1

- Step 1: Divide x by the highest power of n below or equal to it.
- Step 2: Divide the remainder by the next highest power of n.
- Step 3: Repeat Step 2 until there is no more remainder.
- Step 4: The whole number part of the division should be written in the corresponding column.
- Example: 2017

$$\begin{aligned} 2017 \div 10^3 &= \mathbf{2}R17 \\ 17 \div 10^2 &= \mathbf{0}R17 \\ 17 \div 10^1 &= \mathbf{1}R7 \\ 7 \div 10^0 &= \mathbf{7}R0 \end{aligned}$$

1.12.2 Method 2:

- Step 1: Divide x by n. Where $x = kn + r$
- Step 2: Divide k by n
- Step 3: Repeat Step 2 until $k = 0$.
- Step 4: The remainder part of the division should be written in the corresponding column backwards.
- Example: 2473_{10} to 4651_8

$$\left.\begin{aligned} 2473 \div 8 &= 309R1 \\ 309 \div 8 &= 38R5 \\ 38 \div 8 &= 4R6 \\ 4 \div 8 &= 0R4 \end{aligned}\right\uparrow$$

1.13 Modular Arithmetic

- If an integer a, when divided by another interger m, has a remainder $r \geq 0$. Both expressions are equivalent:

$$\boldsymbol{r = a \bmod m}$$
$$\boldsymbol{a = km + r}$$

- Example: Positive Numbers

$$\begin{aligned} 17 \bmod 2 &= 1 \ (\text{as } 17 = 8 \times 2 + 1) \\ 17 \bmod 7 &= 3 \ (\text{as } 17 = 2 \times 7 + 3) \\ 17 \bmod 9 &= 8 \ (\text{as } 17 = 1 \times 9 + 8) \end{aligned}$$

- For negative integer a, $a \bmod m$ can be found by adding a multiple of m to a to obtain a positive number. Divide $\frac{m}{a} \approx z$, round up z, this will be k.

$$\boldsymbol{(-a + m * k) \bmod m}$$

- Example:

$$\begin{aligned} -108 \bmod 13 &= (-108 + 117) \bmod 13 = 9 \\ -253 \bmod 29 &= (-253 + 261) \bmod 29 = 8 \end{aligned}$$

1.14 Moduli Operations

- If $a \bmod m = c$ and $b \bmod m = d$:

$$\boldsymbol{(a + b) \bmod m = (c + d) \bmod m}$$
$$\boldsymbol{(a - b) \bmod m = (c - d) \bmod m}$$
$$\boldsymbol{(a \times b) \bmod m = (c \times d) \bmod m}$$

- Example:

$$\begin{aligned} 37 \bmod 5 &= 2 \\ 29 \bmod 5 &= 4 \end{aligned}$$

$$\begin{aligned} (29 + 37) \bmod 5 &= 66 \bmod 5 &= 1 \\ ((29 \bmod 5) + (37 \bmod 5)) \bmod 5 &= 6 \bmod 5 &= 1 \end{aligned}$$

$$\begin{aligned} (29 \cdot 37) \bmod 5 &= 1073 \bmod 5 &= 3 \\ ((29 \bmod 5) \cdot (37 \bmod 5)) \bmod 5 &= 8 \bmod 5 &= 3 \end{aligned}$$

- Example:

$$\begin{aligned} 2^4 \bmod 10 &= 16 \bmod 10 = 6 \\ 2^5 \bmod 10 &= 32 \bmod 10 = 2 \end{aligned}$$

$$\begin{aligned} \left(2^4 \times 2^5\right) \bmod 10 &= 2^{4+5} \bmod 10 \\ &= 512 \bmod 10 \\ &= 2 \\ \left(2^4 \bmod 10 \times 2^5 \bmod 10\right) \bmod 10 &= 6 \times 2 \bmod 10 \\ &= 12 \bmod 10 \\ &= 2 \end{aligned}$$

- Proof: If $a \bmod m = c$ and $b \bmod m = d$, then:

$$\begin{aligned} a &= km + c \\ b &= lm + d \end{aligned}$$

$$a \times b = (km + c)(lm + d)$$

$$\begin{aligned} a \times b &= klm^2 + clm + kdm + cd \\ (a \times b) \bmod m &= \left(klm^2 + clm + kdm + cd\right) \bmod m \end{aligned}$$

- Removing the first 3 terms, since any multiple of $m \bmod m$ is 0, then:

$$(a \times b) \bmod m = (c \times d) \bmod m$$

1.15 Modular Exponentiation:

- All Moduli values of a numbers with a given base will all have a pattern of repeating results.
- For $2^n \text{mod} 10$, the pattern is as follows as the moduli results repeats, 2, 4 8, 6, 2, 4, 8, 6…

$$\begin{aligned} 2^{4n} \bmod 10 &= 6 \text{ for } n > 1 \\ 2^{4n+1} \bmod 10 &= 2 \text{ for } n > 1 \\ 2^{4n+2} \bmod 10 &= 4 \text{ for } n > 1 \\ 2^{4n+3} \bmod 10 &= 8 \text{ for } n > 1 \end{aligned}$$

1.15.1 Solving:

- Step 1: Figure out pattern.
- Step 2: Arrange pattern to an appropriate recurring expression. The exponents will have an expression of $kn + r$
- Step 3: Apply rules of division and find $e = kn + r$, where e is the desired exponent.
- Example: $4^{112} \bmod 63$

$$\begin{array}{rlr}
4^1 & = 4 & = 4 \\
4^2 & = 16 & = 16 \\
4^3 \bmod 63 & = 64 \bmod 63 & = 1 \\
4^4 \bmod 63 & = 4 \times 4^3 \bmod 63 & = 4 \\
\vdots & = \quad \vdots & = \vdots \\
4^{3n} \bmod 63 & & = 1 \\
4^{3n+1} \bmod 63 & & = 4 \\
4^{3n+2} \bmod 63 & & = 16
\end{array}$$

$$\begin{aligned}
4^{112} \bmod 63 &= 4^{(37\times 3+1)} \bmod 63; \\
&= 4.
\end{aligned}$$

2 Propositional Logic

2.1 Learning Objectives

- Recognise and translate atomic and compound propositions
- construct truth tables
- apply laws of boolean algebra.

2.2 Propositions

- A proposition is a statement that is true or false, but not both. We notate propositions using symbols: p, q, r, s, etc...
- Propositions are used to state or deny facts ($1 < 2, 1 + 1 = 3$, or 'Australia's population is 25 million').
- it must be unambiguous even if we don't know whether it's true or not.
- Example: These are propositions
  - p The 2020 Olympics will be held in Tokyo.
  - q 7 is a prime number
  - r 8 is a prime numbers
- Example: These are not propositions
  - s will you marry me?
  - t 4/9
  - u Go home!
- **Atomic:** (or primitive propositions) as they cannot be broken down into smaller statements
- **Compound (molecular):** combining atomic propositions in different ways via logical operators (or connectives).

2.3 Operations

2.3.1 Logic connectives

- **Equivalence** ($\equiv$)
  - **Words:** "equal to", "equivalent to", or "is".
- **Negation** ($\neg$)
  - **Words:** not, don't, do not, negate, neither, etc.…
- **Conjunction** ($\wedge$)
  - **Words:** but, and, then, etc.…
- **Disjunction** ($\vee$)
  - **Words:** or, both, nor, etc.…
- Use the natural separation in the sentences or commas, semi-colons, etc… to derive precedence and assign brackets.

2.3.2 Conditionals statement or Implications ($\rightarrow$)

- The resulting compound proposition is due to one proposition implying another one.
- **Premise:** p is called the premise or hypothesis.
- **Conclusion:** q is called the conclusion.
- $p \rightarrow q$ means that q is a logical consequence of p or from p, we can logically deduce q.
- Words: $if\ \ldots then\ \ldots\ sentence$, or $if \ldots p \rightarrow q$ means if p then q
- **Contrapositive:**
  - the contrapositive of $p \rightarrow q$ is $\neg q \rightarrow \neg p$.
  - An implication and its contrapositive are logically equivalent when they are either both true or false.

    - User contrapositive when implication is difficult to analyse.
- **Converse**:
    - The converse of $p \rightarrow q$ is $q \rightarrow p$. These are not logically equivalent to the original implication. That is, whether the converse of an implication is true is independent of the truth of the implication.
    - <u>For example</u>, given the propositions: p The number is 4 or 6. q The number is even. Then:
        - Correct: $p \rightarrow q$ – If the number is 4 or 6, then it is even.
        - Not correct: $q \rightarrow p$ – If the number is even, then it is 4 or 6.
- **Direction of the arrow:**
    - Can also be written as $p \rightarrow q$ can also be written $q \leftarrow p$.
    - The direction of the arrow is not important, but from which proposition to which proposition the arrow flows is**.** So, $p \rightarrow q$ means the same thing as $q \leftarrow p$: If *p* then ***q*.**
- **Casual Relationships between proposition:**
    - Implications are logical connectives, but they do not necessarily have a causal relationship (or effect). That is, they may not have any connection such that one causes the other.
    - For example, consider the compound proposition, 'If there are traces of life on a planet, this planet once had water'.
        - p There are traces of life on a planet.
        - q this planet once had water.
        - $p \rightarrow q$, but this does not mean that p caused q (in this case, it is the opposite).
- **Bi-conditional (if and only if):**
    - $p \leftrightarrow q$ is true when both the implication and its converse are true.
    - That is, means each proposition implies the other one: $p \rightarrow q$ and $q \rightarrow p$.
    - You can also express this as
        - p if and only if q
        - p is necessary and sufficient for q.
    - We can write $p \leftrightarrow q$ using the other operators: $(p \rightarrow q) \wedge (q \rightarrow p)$.
    - Determining whether the implication of the converse is the "if" or the "if only" is determined by logical determining which statements are equivalent.
    - <u>Example:</u> "I sing if and only if I'm in the shower".
        - Let p = "I sing", q = "I'm in the shower".
        - Then, $p \rightarrow q$ is "if I sing, then I'm in the shower".
        - $q \rightarrow p$ "if I'm in the shower, then I sing".
        - "if I sing, then I'm in the shower" is equivalent to saying "I sing if I'm in the shower"
        - "if I'm in the shower, then I sing" is equivalent to saying "I sing $only\ if$ I'm in the shower".
        - Then, $only\ if$ is: $p \rightarrow q$ and If is $q \rightarrow p$

2.4 Truth Tables

- Truth tables are represent every possible combination of truth values of the atomic propositions in a compound composition..

2.4.1 Negation ($\neg p$)

- Is the oppositive of p.

p	$\neg p$
T	F
F	T

2.4.2 Conjunction ($p \wedge q$)

- Is only true when both p and q are true.

p	q	$p \wedge q$
T	T	T
T	F	F
F	T	F
F	F	F

2.4.3 Disjunction ($p \vee q$)

- Is false when both p and q are false.

p	q	$p \vee q$
T	T	T
T	F	T

F	T	T
F	F	F

2.4.4 Implication ($p \rightarrow q$)

- Is true when p is false, q is true or both.

p	q	$p \rightarrow q$
T	T	T
T	F	F
F	T	T
F	F	T

2.4.5 Bi-conditionals ($p \leftrightarrow q$):

- Is true when p and q have the same truth values (either T or F).

p	q	$p \leftrightarrow q$
T	T	T
T	F	F
F	T	F
F	F	T

2.4.6 Solving Truth Tables

- Step 1: get all possible combinations for all the individual propositions
- Step 2: decompose logical statements and solve the truth tables for each.
- Step 3: get the final result by combining the decomposed values into the final expression.
- <u>Example:</u> Solve $p \vee (q \wedge \neg p)$

p	q	$\neg p$	$p\ (q \vee \neg p)$
T	T	F	T
T	F	T	T
F	T	T	T
F	F	F	F

2.4.7 Compound Propositions

- **Tautology:** when a compound proposition has all its truth values as T.
- **Contradiction:** when a compound proposition has all its truth values as F.
- **Contingency:** when a compound proposition is neither all T or F.

2.5 Logical Equivalence

- Two propositions are logically equivalent if they have identical truth values (in the final column).
- We can use truth tables to establish whether any two propositions are logically equivalent.
- **The laws of logical equivalence**:

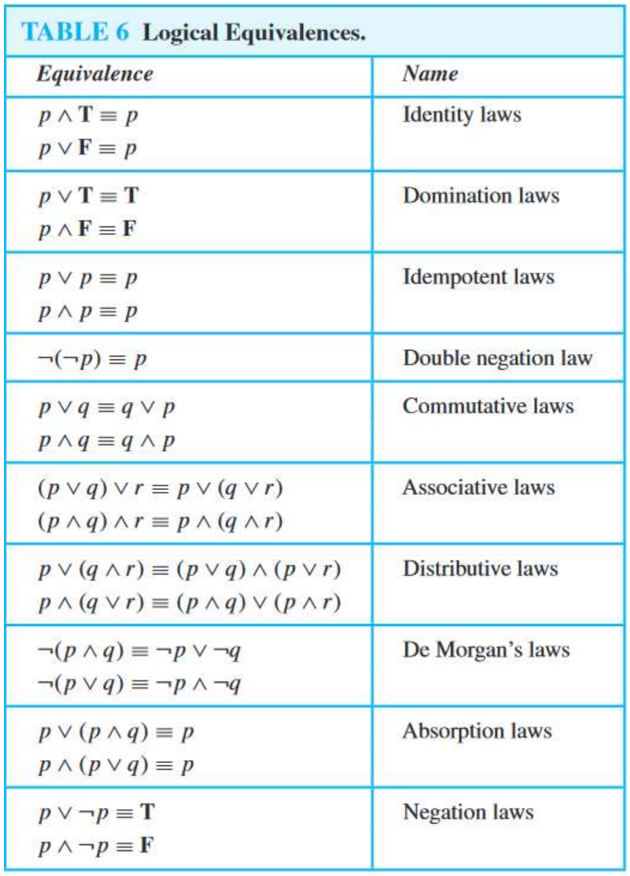


TABLE 6 Logical Equivalences.

Equivalence	*Name*
$p \wedge \mathbf{T} \equiv p$ $p \vee \mathbf{F} \equiv p$	Identity laws
$p \vee \mathbf{T} \equiv \mathbf{T}$ $p \wedge \mathbf{F} \equiv \mathbf{F}$	Domination laws
$p \vee p \equiv p$ $p \wedge p \equiv p$	Idempotent laws
$\neg(\neg p) \equiv p$	Double negation law
$p \vee q \equiv q \vee p$ $p \wedge q \equiv q \wedge p$	Commutative laws
$(p \vee q) \vee r \equiv p \vee (q \vee r)$ $(p \wedge q) \wedge r \equiv p \wedge (q \wedge r)$	Associative laws
$p \vee (q \wedge r) \equiv (p \vee q) \wedge (p \vee r)$ $p \wedge (q \vee r) \equiv (p \wedge q) \vee (p \wedge r)$	Distributive laws
$\neg(p \wedge q) \equiv \neg p \vee \neg q$ $\neg(p \vee q) \equiv \neg p \wedge \neg q$	De Morgan's laws
$p \vee (p \wedge q) \equiv p$ $p \wedge (p \vee q) \equiv p$	Absorption laws
$p \vee \neg p \equiv \mathbf{T}$ $p \wedge \neg p \equiv \mathbf{F}$	Negation laws

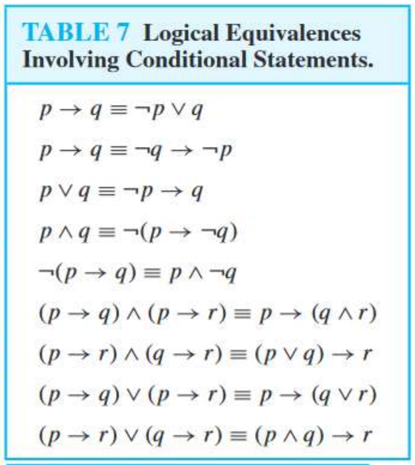


TABLE 7 Logical Equivalences Involving Conditional Statements.

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

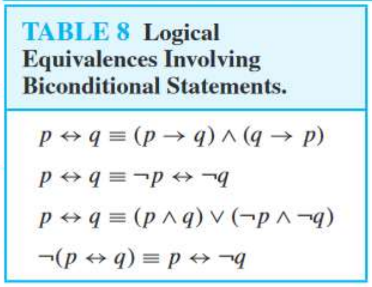


TABLE 8 Logical Equivalences Involving Biconditional Statements.

$p \leftrightarrow q \equiv (p \to q) \wedge (q \to p)$
$p \leftrightarrow q \equiv \neg p \leftrightarrow \neg q$
$p \leftrightarrow q \equiv (p \wedge q) \vee (\neg p \wedge \neg q)$
$\neg(p \leftrightarrow q) \equiv p \leftrightarrow \neg q$

- Example: Solve $(p \wedge \neg(p \ \vee q) \equiv F)$

$$\begin{aligned} p \wedge \neg(p \vee q) &\equiv p \wedge (\neg p \wedge \neg q) && \textit{(De Morgan's law)} \\ &\equiv (p \wedge \neg p) \wedge \neg q && \textit{(Associative law)} \\ &\equiv F \wedge \neg q && \textit{(Negation Law)} \\ &\equiv F && \textit{(Domination Law)} \end{aligned}$$

3 Predicate Logic

3.1 Learning Objectives

- formulate complex statements using quantifiers and predicates
- nest and compound quantified statements
- prove or disprove quantified statements.

3.2 Predicate Logic:

- Predicate logic, or quantified logic is a formal language in which propositions are expressed in terms of predicates, variables and quantifiers.
- It is an extension to propositional logic, in which the notions of truth values, logical connectives, etc… still apply but atomic statements, will be replaced by propositions involving predicates and quantifiers.
- Predicate logic is superior to propositional logic in the sense that it is able to capture the structure of several arguments in a formal sense which propositional logic cannot.
    - Finite: to express that every element in the domain satisfy the predicate, we need lots of and statements.
    - Finite: to express that one or more elements in the domain satisfy the predicate, we need lots of or statements.
    - Infinite: to express that infinite numbers of elements satisfy the predicate when the domain is infinite, we need an infinite number of atomic statements.

Compound proposition	$(atomic\ statement)\ connective\ (atomic\ statement) \ldots$
Quantified Proposition	$[Quantifier] : [Predicate]$

- Example: All numbers in the set $\{2,4,8,16\}$ are even
    - Compound proposition: $(2\ is\ even) \wedge (4\ is\ even) \wedge (8\ is\ even) \wedge (16\ is\ even)$

   - Quantified proposition: $\forall x \in \{2,4,8,16\} : x \text{ is even}$
- Example: There are real numbers such that $x > 13$
   - Compound proposition: ... $(2 > 13) \vee (3 > 13) \vee (4 > 13) \vee (5 \vee 13)$ ...
   - Quantified proposition: $\exists x \in \mathrm{R} : x > 13$

3.3 Predicates

- A predicate is the expression of the logical statement it must satisfy.
- Denoted by $P(x)$, where $P(x)$ is the propositional function P at x.
- In any given predicate $P(x)$:
   - **Subject:** (x) is the subject of the statement.
   - **Predicate:** (P) is a property that the subject of the statement can have.
- Examples
   - $P(s) \equiv s$ is enrolled in SIT192.
      - Subject: s.
      - Predicate: 'is enrolled in SIT192.'
      - Domain: could be a set of people: The people sitting in the room or watching the video, the population of Australia, but not a number ('12 is enrolled in SIT192.').
      - Evaluate $P(You)$: 'You are enrolled in SIT192.' (true).
      - Evaluate $P(\text{prime minister})$: 'The prime minister is enrolled in SIT192.' (false).
   - Let *P*(*x*) denote *x* > 13.
      - Subject: is x.
      - Predicate: 'greater than 13.'
      - Domain: is a set of numbers (e.g. natural numbers):
      - Evaluate: $P(4) = 4 > 13$ (false).
      - Evaluate: $P(14) = 14 > 13$ (true).
      - Evaluate: $P(prime\ minister) = 'The\ prime\ minister > 14'$ is meaningless.

3.4 Quantifiers

- Quantifiers give us the power to express propositions involving entire sets of objects, some of them, enumerate them, etc.
- The quantifier declares the domain that values can take in the statement. These can then be used to analyse whether the predicate is true or false and hence whether the entire statement is true or false.
- **Domain:** is the ensemble of values that the subject can take. Note that according to the predicate function, it can either be just numbers or subjects (people, etc…), but not both.

3.4.1 Universal quantifier $(\forall)$

- is used to construct statements where the predicate is satisfied by all elements in the domain.
- Words: for every, for all, any, etc…
- '$\forall x \in \mathrm{R}$...' means 'for all x in the real number set.
- Example: if the domain consists of the natural numbers:
   - If $P(\mathrm{x}) = x < x + 1$, $\forall x P(x)$ means that 'every $\boldsymbol{x}$ is less than $x + 1$.'
   - If $P(x) = x > 12$, $\forall x P(x)$ means that 'all numbers are greater than 12.'

3.4.2 Existential quantifier $(\exists)$

- is used to construct statements where the predicate is satisfied by at last one element in the domain.
- Words: there is, at least, no number, etc…
- '$\exists x \in \mathrm{R}$...' means 'There is x in the real number set'.
- Example:
   - If $P(x) = x > 12$, $\exists x P(x)$ means that 'there is a number greater than 12.'
   - If $P(x) = \mathrm{x} > \mathrm{x} + 1$, $\exists x P(x)$ means that 'there is a x which is greater than $x + 1$.'

3.5 Proving Quantifiers:

- According to the quantified statement, its compound proposition equivalent will be based on the domain requirements.
- For $\forall$ there needs to be a big sequence of disjunctions for its compound equivalent. e.g. $(2 < 10) \wedge (3 < 10) \wedge$ ...
   - True when all are true. Hence, a general algebraic expression is required.
   - False when at least one is false. Hence, one example to show this is required.
- For $\exists$ there needs to be a big sequence of disjunctions for its compound equivalent. e.g. $(2 < 10) \vee (3 < 10) \vee$ ...
   - True when at least one is true. Hence, one example to show this is required.
   - False when all are false. Hence, a general algebraic expression is required.

- Never give an example to prove a statement when a general argument is required. You can give an example to illustrate your proof, but not as the main argument.

Proof	∀	∃
True	*General*	*Example*
False	*Example*	*General*

- **Solving Quantifiers**:
  - Step 1: Determine the quantifier and the predicates.
  - Step 2: determine whether the statement can be proved by a single example or a general example. Iof single, solve equation, otherwise thin of values x can take.
  - Step 3: Solve the equation by substituting the value(s) and show whether the statement is true or false.
  - Step 4: derive the result of the quantified statement based on step 2.
- Example: $\forall x \in R: x^2 - 1 \geq 1$
  - A single example must be used since $\forall x$
  - Since at $x = 0$, we get $0^2 - 1 < 1$, then this shows that the statement is false.
- Example: $\exists x \in R: x^2 = -4$
  - A general example must be used since $\exists x$
  - Since $x^2 \geq 0$ for all $x \in R$, we get $x^2 \geq 0 > -4$, then this shows that the statement is true.

3.6 Nested Quantifiers

- A statement involving two or more quantifiers (∀, ∃).
- This allows us to construct more complex propositions
- Example:
  - $\forall x \exists y : y = x$ means 'Any number has a number equal to it.'
  - $\forall x \forall y : (x < 0) \wedge (y < 0) \rightarrow (x + y < 0)$ where ***x*** and ***y*** are real numbers means 'for all negative real numbers *x* and *y*, *x* + *y* is negative.'

3.6.1 Parsing or Translating:

- The act of making sense of nested propositions. That is, translating them to enligsh.
- Propositions with multiple quantifiers are not fundamentally different from other quantified propositions. They are composed of one quantifier and one predicate, but the predicate is itself quantified.

$$[Quantifier]([Quantifier]: [Predicate])$$

- Example: $\forall x \exists y : y = x$
  - Quantifiers: $\forall$ and $\exists$
  - Predicates: $P(x) = \exists y : y = x$ and $P(x, y) = y = x$
- **Order of the quantifier:**
  - The order of the quantifiers makes a difference and can change the meaning of the proposition.
  - For example: If $P(x, y)$ is '*x* loves *y*', where the domains of *x* and *y* are people.
    - Meaning of $\forall x \exists y : P(x, y)$ is "Everybody loves somebody".
    - Meaning of $\exists y \forall x : P(x, y)$ is "There is a person that everybody loves".
- Note that to parse nested quantifiers, it helps to break things down.
- **Translating:**
  - Step 1: Determine the quantifier and the predicates.
  - Step 2: start translating them into English one by one, each time removing more mathematical symbols.
  - Step 3: Bring it all together into a statement.
- Example: $\forall x \exists y: y = x^3$
  - $P(x, y) = y = x^3$ ≡number y equals the cube of another number with variables x and y.
  - $Q(y)$ ≡ there is a number $y: P(x, y)$ with domain $y \in R$.
  - $R(x)$ ≡ for all real numbers $x: Q(x)$ with domain $x \in R$.
  - Quantifiers are ∀ and ∃.
  - $\forall x \exists y: y = x^3$ ≡ for all real numbers x, there is a number y such that $y = x^3$

3.6.2 Proving

- Step 1: translate the statement
- Step 2: decide whether the statement is true or false, based on your intuition.
- Step 3: determine the quantifier and the predicates.
- Step 4: analyse the external quantifier by applying the steps for "proving quantifiers"
- Step 5: analyse the internal quantifier by applying the steps for "proving quantifiers"
- Step 6: draw a conclusion.

- <u>Example:</u> $\forall x \exists y: x = y$ is true
    - Translation: There is a real number x for all numbers y such that their product equals number y.
    - Let:

$$Predicate: P(x,y) \equiv y = x^3 \text{ with variable } x \text{ and } y$$
$$Predicate: Q(x) \equiv \exists y: P(x,y) \text{ with domain } x \in R$$
$$Statement: R(x) \equiv \forall x Q(x) \text{ with domain } y \in R$$

    - To proof $R(x)$ general example must be used since $\forall x$
    - To proof $Q(x)$ a single example must be used since $\exists y$
    - Let $y = \sqrt[3]{x}$ hence:

$$P(x,y) = x = \left(\sqrt[3]{x}\right)^3 = x$$

    - Since this is true, then the state is true.
- <u>Example:</u> $\exists x \forall y: y \neq 0 \rightarrow xy = 0$ is true
    - Translation: there is a number x for all numbers y that are not zero, such that their products equals 0
    - Let:

$$Predicate: P(x,y) \equiv y \neq 0 \rightarrow xy = 0 \text{ with variable } x \text{ and } y$$
$$\text{Let } p \equiv y \neq 0 \text{ and } q \equiv xy = 0 \text{ then}$$
$$P(x,y) \equiv p \rightarrow q$$
$$Predicate: Q(y) \equiv \forall y: P(x,y) \text{ with domain } y \in R$$
$$Statement: R(x) \equiv \exists x Q(y) \text{ with domain } x \in R$$

    - To proof $R(x)$ a single example must be used since $\exists x$
    - To proof $Q(x)$ a general example must be used since $\forall y$
    - Analyse possibilities:
        - Option 1: If $y \neq 0$ then p is true and q must also be true so that $p \rightarrow q$ is true.
        - Option 2: If $y = 0$ then p is false and $p \rightarrow q$ is true regardless of the value of q
    - Since a general example must be used for y then take option 2.
    - Let $x = 0$ and $y \neq 0$, $P(x,y) \equiv y \neq 0 \rightarrow 0y = 0$ which is true and thus showing the statement to be true.

3.7 Negative Quantifiers:

- The negation of $\forall x P(x)$ *is* $\neg \forall x P(x)$. This means 'not for all *x*, *P*(*x*)'; '*P*(*x*) is not true for every single *x*.'
- $\neg \forall x P(x)$ can also be expressed as $\exists x(\neg P(x))$. i.e. 'there exists at least one real number *x* such that *x* is not greater than 3.'
- <u>Example:</u> $\neg \forall x : x > 3$
    - $\neg \forall x P(x)$ means 'it is not true that all real numbers are greater than 3.'
    - $\exists x(\neg P(x)) = x < 3$ means 'there exists at least one real number *x* such that *x* is not greater than 3.'
- <u>Example:</u> $\neg \exists x \forall y : x \leq y$:
    - $\neg \exists x \forall y P(x)$ means "it is not true that x such that for all numbers y, y is less than or equal to y
    - $\forall x \neg P(x) = y < x$ for every number x there is a number y which is smaller than x.

$$\neg \forall \boldsymbol{x} \boldsymbol{P}(\boldsymbol{x}) = \exists \boldsymbol{x}(\neg \boldsymbol{P}(\boldsymbol{x}))$$
$$\neg \exists \boldsymbol{x} \boldsymbol{P}(\boldsymbol{x}) = \forall \boldsymbol{x}(\neg \boldsymbol{P}(\boldsymbol{x}))$$

4 Sets and Functions

4.1 Learning Objectives

- express sets using enumeration or set builder notation, and go back and forth between the two representations
- apply common set operations
- identify common properties of functions.

4.2 Defining Sets

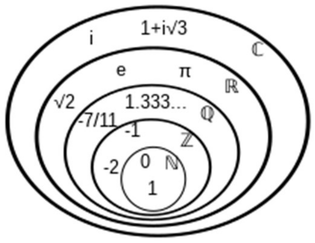
- A set is a collection of objects.
- The objects are called elements of the set.
- A set is indicated by curly brackets/braces: { }
- ∈: 'is an element of'
- ∉: 'is not an element of'
- Uppercase letters denote sets
- Lowercase letters denotes elements of sets.
- <u>Example:</u>
    - The set of students in this class
    - The inventory of a shop
    - Let P be the set of prime numbers less than 20. $P = \{2, 3, 5, 7, 11, 13, 17, 19\}$
    - $A = \{3, 4, 8\}$ (We can see that 3 ∈ *A*, and 6 ∉ *A*)

- $B = \{1,2,3\} = \{x \in \mathbb{N} : x < 4\}$
- $C = \{22, 24, 26, \ldots\} = \{x \in \mathbb{N} : x > 20 \text{ and even}\}$
- $D = \{\{1,2,3\},\{1\},\{4,5,2\}\}$

- A set is not ordered. For example, $\{1,2,3\} = \{3,2,1\} = \{1,3,2\}$.
- Elements are either in or out of the set, there is no count on how many times they appear: $\{1,2,3\} = \{1,2,3,2,1\}$.
- **Explicit notation:** When sets are small: $\{1,2,3\}$, $\{\{1,2,3\},\{1\},\{4,5,2\}\}$
- **Enumeration notation:** when pattern is clear $\{2,4,6,8,\ldots\}$.
- **Set builder notation:** uses propositions and predicates to define sets. $\{x \in \mathbb{N} : x < 4\}, \{x \in \mathbb{N} : x > 20 \text{ and even}\}, \ldots$

4.2.1 Numbers Sets

- Natural Numbers ($\mathbb{N}$): $\{1,2,3,4,\ldots\}$
- Integers ($\mathbb{Z}$): $\{0, \pm1, \pm2, \ldots\}$
- Rational numbers ($\mathbb{Q}$): *a*/*b*, for *a*, *b* integers, $b \neq 0$
- Real numbers ($\mathbb{R}$): which includes $\mathbb{N}$, $\mathbb{Z}$, $\mathbb{Q}$, and irrational numbers
- Complex Numbers ($\mathbb{C}$): $x + iy$, for x, y real, $i^2 = -1$



4.2.2 Properties of Sets

- **Cardinality:** is the size of set A, denoted by |*A*|, is the number of elements in *A*. e.g. |*A*| = 1 means that *A* has one element.
- **Null set:** (empty set) is denoted by ∅ (or {}), and is the set containing no elements, i.e. |∅| = 0.
- **Universal set:** denoted by *U*, is the set that contains all elements under consideration.
- **Equality:** Two sets *A* and *B* are equal if they contain exactly the same elements, i.e., *A* = *B* means $x \in A \leftrightarrow x \in B$.

4.3 Subsets and Proper Subsets

- Let A and B be sets.
- **Subset:** *A* is a subset of *B* (denoted as $A \subseteq B$ or $A \subset B$) if every element of *A* is in *B*.
- **Superset:** *B* is a superset of *A* (denoted $B \supseteq A$) when every element of B is in A.
- **Proper set:** *A* is a proper subset of *B* ($A \subsetneq B$) when *A* is a subset of *B* but is not equal to *B*. Useful when we want to emphasise that *A* and *B* should not be equal.
- Example:
  - The set of all even natural numbers is a proper subset of all natural numbers.
  - $\{1, 2, 3\} \subset \mathbb{N}$
  - $\mathbb{R} \supset \mathbb{N}$
- We use $A \not\subset B$ to denote that *A* is not a subset of *B*.
- **∈ versus ⊆:**
  - It is important not to confuse the use of the symbols ∈ and ⊆ (or ⊂).
  - The symbol ∈ refers to elements, e.g. $2 \in A$.
  - The symbols ⊂ and ⊆ refer to sets, e.g. $B \subset A$ or $B \subseteq A$.

4.4 Set Operations

Property	Definition	Expression
Union	The union of *A* and *B* is the set of all elements in *A* or *B* (or both), and is denoted by $A \cup B$	$A \cup B = \{x : x \in A \text{ or } x \in B\}$
Intersection	The intersection of *A* and *B* is the set of all elements in both *A* and *B* , and is denoted by $A \cap B$	$A \cap B = \{x : x \in A \text{ and } x \in B\}$
Compliment	The compliment *A* is the set of all elements a universal set *U*, but not in *A*, and is denoted by **Ā** (or *A'*, or A^C)	$\bar{A} = \{x : x \in U \text{ and } x \notin B\}$
Difference	The set difference of *A* and *B* is the set of all elements in *A* excluding those in *B*, and is denoted by *A*\B or *A* − *B*	$A \setminus B = \{x : x \in A \text{ and } x \cap B^c\}$

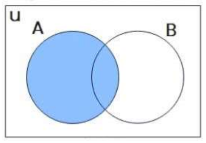
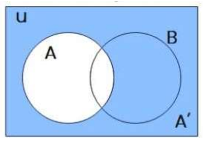
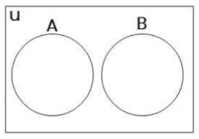
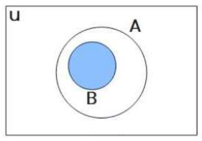
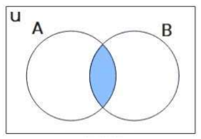
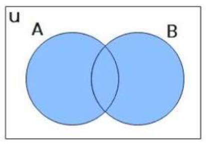
4.5 Laws of Set Operations

- There are many properties which constitute the laws of set algebra. The laws are analogous to the laws of logical equivalence.
- $\vee$ corresponds to $\cup$
- $\wedge$ corresponds to $\cap$
- $\neg$ corresponds to $\cdot^{C}$ (the complement set)
- T corresponds to U
- F corresponds to $\emptyset$.

$A \cap U = A$ $A \cup \emptyset = A$	Identity Laws
$A \cup U = U$ $A \cap \emptyset = \emptyset$	Domination Laws
$A \cap A = A$ $A \cup A = A$	Idempotent Laws
$\overline{\overline{A}} = A$	Double Complement Laws
$A \cap B = B \cap A$ $A \cup B = B \cup A$	Commutative Laws
$(A \cap B) \cap C = A \cap (B \cap C)$ $(A \cup B) \cup C = A \cup (B \cup C)$	Associative Laws
$A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$ $A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$	Distributive Laws
$\overline{A \cap B} = \overline{A} \cup \overline{B}$ $\overline{A \cup B} = \overline{A} \cap \overline{B}$	De Morgan's Laws
$A \cup \overline{A} = U$ $A \cap \overline{A} = \emptyset$	Complement Law
$A \cap (A \cup B) = A$ $A \cup (A \cap B) = A$	Absorbtion Law

4.6 Venn Diagrams:

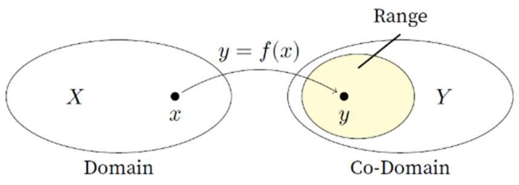
- Venn diagrams help us illustrate relations of sets. They show all of the possible mathematical or logical relationships between groups of sets.
- Venn diagrams are not accepted as a formal mathematical proof and should only be used to assist in understanding relationships between sets.

A		A'	
A and B		$B \subset A$	
$A \cap B$		$A \cup B$	

4.7 Functions

- **Definition:** Let X and Y be sets. A function f from X to Y assigns each element of X to exactly one element in Y.
- Functions are also called mappings or transformations.

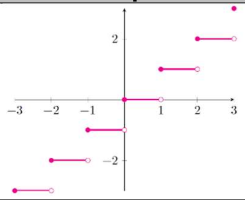
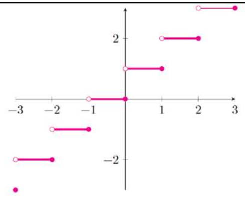
- $\boldsymbol{f : X \to Y}$ means the function f is a mapping that transforms elements from Set X to Set Y (i.e. f maps X to Y).
- $\boldsymbol{f(x) = y}$ means the function f maps Element x (in X) to Element y (in Y).
- **Domain:** X is the domain of f.
- **Co-Domain:** Y is the co-domain of f.
- **Image:** y is the image of x.
- **Pre-Image:** x is the pre-image of y.
- **Range of $\boldsymbol{f}$:** is the set of all images of elements in X, i.e., $f(X)$.



- Examples:
  - $f : \mathbb{R} \to \mathbb{R}\ f(x) = 3x + 5$
  - $f : \mathbb{R} \to \mathbb{R} : f(x) = x2$
  - Let $A = \{1, 2, 3\}$, $B = \{3, 4, 5\}$. Then, $g : A \to B$ where $g(a) = a + 2$ for $a \in A$.
  - Let $A = \{1, 2, 3\}$, $B = \{3, 4, 5, 6\}$. Then, $h : A \to B$ where $h(a) = a + 2$ for $a \in A$. However, the range is not equal to the co-domain.
  - ¬ : {logical propositions} → {logical propositions}
  - Hash functions map data (e.g. strings) of any size to data of fixed size
  - Cryptographic function map strings to strings

4.7.1 Floor and Ceiling Functions

- **Floor function:** takes an input x and outputs the greatest integer less than or equal to x. In looser terms, we round x down to the closest integer.
- **Ceiling function:** takes an input x and outputs the smallest integer greater than or equal to x. In looser terms, we round x up to the closest integer.

Name	Expression	Example	Graph
Floor Function	$f : \mathbb{R} \to \mathbb{Z}\ f(x) = \lfloor x \rfloor$	$f(2.3) = \lfloor 2.3 \rfloor = 2$	
Ceiling Function	$f : \mathbb{R} \to \mathbb{Z}\ f(x) = \lceil x \rceil$	$f(2.3) = \lceil 2.3 \rceil = 3$	

4.7.2 Injective Functions

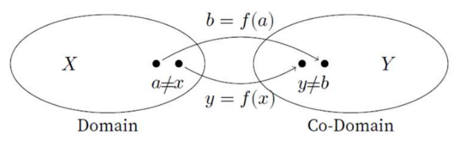
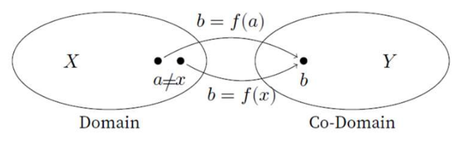
- A function $f : X \to Y$ is injective if for all $a, b \in X$, if $a \neq b$ then $f(a) \neq f(b)$.

$$\forall \boldsymbol{a}, \boldsymbol{b}, \in \boldsymbol{X}, \boldsymbol{a} \neq \boldsymbol{b} \Rightarrow \boldsymbol{f}(\boldsymbol{a}) \neq \boldsymbol{f}(\boldsymbol{b})$$

- For distinct inputs, injective functions produce distinct outputs. This is useful when we want to keep distinctness – for example if we want to 'go back', or if we want to assign unique identifiers.
- Injectivity is unwanted if we want to reduce the size of our input (or data).
- **Counter positive**: a function is not injective if two distinct values *a* and b have the same output. Such that $f(a) = f(b)$ such that $a = b$

$$\forall \boldsymbol{a}, \boldsymbol{b}, \in \boldsymbol{X}, \boldsymbol{f}(\boldsymbol{a}) = \boldsymbol{f}(\boldsymbol{b}) \Rightarrow \boldsymbol{a} = \boldsymbol{b}$$

Name	Examples	Graph

Injective Function	• $f\ (car)\ \rightarrow\ number\ plate$ • $f\ (phone\ contract)\ \rightarrow\ telephone\ number$ • cryptographic functions • $f : \mathbb{R} \rightarrow \mathbb{R}, f(x) = x + 1$ • $f : \mathbb{R} \rightarrow \mathbb{R},\ (x) = e^x$	 $b = f(a)$ X $a \neq x$ $y = f(x)$ $y \neq b$ Y Domain Co-Domain
Counter Positive	• most hash functions • f (student) → date of birth • $f : \mathbb{R} \rightarrow \mathbb{R} : f(x) = x^2$, (e.g., 3 and −3 are both mapped to 9).	

- **Proving:**
  - Step 1: Assume $f(a) = f(b)$ is true for arbitrary $a,\ b \in X$.
  - Step 2: Show that $a = b$ is the only solution to step 1.
  - Example: prove $f: \mathbb{Z} \rightarrow \mathbb{Z}$ with $f(x) = x + 1$ is injective:
    - Let $f(a) = a + 1$ and $f(b) = b + 1$
    - Assume $f(a) = f(b)$, then:

$$f(a) = f(b)$$
$$a + 1 = b + 1$$
$$a = b$$

    - This shows the function is injective.
- **Disproving:**
  - Step 1: Find two values a and b that are different but $f(a) = f(b)$
  - Example: disprove $f: \mathbb{R} \rightarrow \mathbb{R}$ with $f(x) = x^2$ is injective:
    - Let $f(a) = a + 1$ and $f(b) = b + 1$
    - Assume $f(a) = f(b)$, then:

$$f(a) = f(b)$$
$$a^2 = b^2$$
$$\pm a = \pm b$$

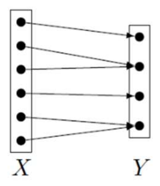
    - For a = -2 and b = 2 this is not true. Hence, this shows the function is not injective.

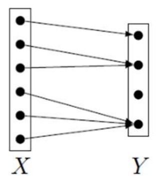
4.7.3 Surjective Functions

- A function $f\ :\ X\ \rightarrow\ Y$ is surjective if for all $y\ \in\ Y$, there is a x value such that $f\ (x)\ =\ y$. That is, when the range of f, (i.e. $f\ (x)$), is identical to the co-domain of f, (i.e. Y).
- There must be at least one x for any y.

$$\forall \boldsymbol{y} \in \boldsymbol{Y}, \exists \boldsymbol{x} \in \boldsymbol{X} : \boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{y}$$

- In the diagram below, the left side represents a surjective function and the right side represents a function that is not surjective.





- **Proving:**
  - Step 1: Assume some arbitrary $y\ \in\ Y$, and find x in terms of y, i.e. $f(x)\ =\ y\ \rightarrow\ x\ =\ f^{-1}(y)$
  - Step 2: If $x\ \in\ X$ and you can justify this for all $y\ \in\ Y$, then your proof is complete. You only need to find one x for every y. that is, we need to show that the equation $f(x)\ =\ y$ always has some solutions (it can have more than one, but must have at least one).
  - Example: $f\ :\ \mathbb{R}\ \rightarrow\ \mathbb{R}$ defined by $f(x) = 5x - 9$ is surjective.
    - let any number $y \in \mathbb{R}$, then solving for x:

$$5x - 9 = y$$
$$5x = y + 9$$
$$x = \frac{(y + 9)}{5}$$

        - since there is a solution for x, then f is surjective.
- **Disproving:**
    - Step 1: Find a counter example.
    - Example: $f : \mathbb{Z} \to \mathbb{Z}$ defined by $f(x) = 5x - 9$ is not surjective.
        - let any number $y \in \mathbb{Z}$, then solving for x:

$$5x - 9 = y$$
$$5x = y + 9$$
$$x = \frac{(y+9)}{5}$$

        - However, let $y = 0$ then $x = \frac{5}{9} \notin \mathbb{Z}$, then this shows that f is not surjective.

4.7.4 Bijection

- Surjectivity is not the opposite of injectivity. The two notions are unrelated. A function is classified as a surjective independent whether it is injective function.
- A function $f : X \to Y$ is bijective if it is both injective and surjective. That is, $f : X \to Y$ is bijective if for all $y \in Y$, there is exactly one $x \in X$ such that $f(x) = y$.
- **Proving:**
    - Step 1: Prove injectivity.
    - Step 2: Prove surjectivity.
- **Disproving:**
    - Step 1: Find a counter example to either injectivity or surjectivity.

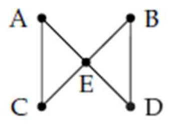
5 Graph Theory

5.1 Learning Objectives

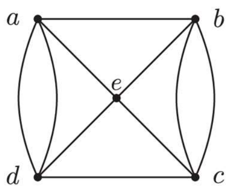
- Identify trees and obtain spanning trees of graphs
- recognise planar graphs, and apply Euler's formula to determine planarity
- find Euler and Hamiltonian paths and circuits in a graph.

5.2 Graph Theory

- Graph theory was inspired by an 18th century problem, now referred to as the Seven Bridges of Königsberg.
- Graphs are made up vertices and edges which are often be represented as diagrams with dots joined by lines.
- **Vertices:** the collection of objects (dots).
- **Edges**: the relationship (lines) between the vertices.
- Example: Al, Bob, Cam, Dan, and Euler are all members of *Facebook*. The site allows members to be “friends” with each other. Al and Cam are friends, Bob and Dan are friends. Euler is friends with everyone. Represent this situation with a graph.
    - Vertices: each person
    - Edges: their friendship.

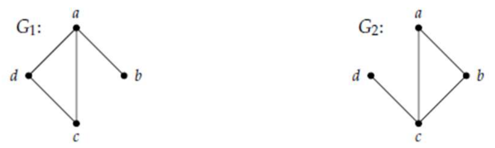


- **Graph (Simple) definition:** A **graph** is an ordered pair $G = (V, E)$ consisting of a nonempty set V (called the **vertices**) and a set E (called the **edges**) of two-element subsets of V.
    - No pair of vertices can be connected by an edge more than once.
    - Cannot have a single vertex connected to itself by an edge.
- **Adjacent:** Two vertices are **adjacent** if they are connected by an edge. Two edges are **adjacent** if they share a vertex.
- **Multigraph:** A **multigraph** is just like a graph but can contain multiple edges between two vertices as well as single edge loops (that is an edge from a vertex to itself).



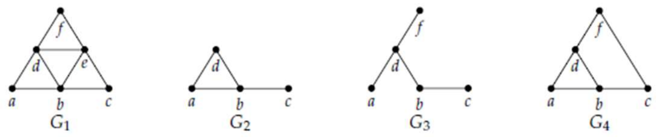
5.2.1 Equality and Isomorphism

- **Equality:** Two sets are said to be equal if an only if they have the same V and E independent of their graphical representation
- **Isomorphism ($G1 \cong G2$):**
  - Two graphs are isomorphic if there is an isomorphism between them. That is if they are "basically the same".
  - A quick check for possible isomorphism is two graphs have the same degree sequence in different order.
  - An isomorphism between two graphs $G1$ and $G2$ is a bijection $f : V1 \to V2$ between the vertices of the graphs such that $\{a, b\}$ is an edge in $G1$ if and only if $\{ f(a), f(b)\}$ is an edge in $G2$.
    - Bijection: an isomorphism is simply a function which renames the vertices. Every vertex gets a new name (i.e. it creates a new function from itself).
    - Edge relation: these newly named vertices must be connected by edges precisely when they were connected by edges with their old names.
- **Isomorphism class:** collection of isomorphic graphs.
- Example: Determine whether $G1 = \{V1, E1\}$ and $G2 = \{V2, E2\}$ are equal or isomorphic.
  - $V1 = \{a, b, c, d\}$, $E1 = \{\{a, b\}, \{a, c\}, \{a, d\}, \{c, d\}\}$
  - $V2 = \{a, b, c, d\}$, $E2 = \{\{a, b\}, \{a, c\}, \{b, c\}, \{c, d\}\}$
  - It is not equal since $\{a, d\} \in E1$ but $\{a, d\} \notin E2$
  - It is isomorphic. Let $f : V_1 \to V_2$. Let $f(a) = b, f(b) = c, f(c) = d$ and $f(d) = a$
    - Bijection: there is bijection (injectivity and surjectivity).
    - Edge Relation:
      - from the graph below it appears to be f is not an isomorphism: the edge $\{a, b\}$ corresponds to $\{ f(a), f(b)\} = \{b, c\}$, $\{a, c\}$ corresponds to $\{f(a), f(c)\} = \{b, d\}$, but there is no edge between b and d in $G2$.
      - Propose a new function. Let $g: V1 \to V2$, $g(a) = c$, $g(b) = d$, $g(c) = b$ and $g(d) = a$. Then:
        $\{c, d\}, \{c, b\}, \{c, a\}, \{b, a\} = \{g(a), g(b)\}, \{g(a), g(c)\}, \{g(a), g(d)\}, \{g(c), g(d)\}$



5.2.2 Subgraphs and Induced Subgraphs

- **Subgraph (*G′* ⊆ *G*):**
  - We say that *G′* = (*V′*, *E′*) is a subgraph of *G* = (*V*, *E*), provided *V′* ⊆ *V* and *E′* ⊆ *E*.
  - That is G' is a **subgraph** of *G* if every vertex and edge of *G'* is also a vertex or edge of *G*.
  - Think of a subgraph as the result of deleting some vertices and edges from the larger graph.
- **Induced Subgraph:**
  - We say that *G′* = (*V′*, *E′*) is an induced subgraph of *G* = (*V*, *E*) provided *V′* ⊆ *V* and every edge in *E* whose vertices are still in *V′* is also an edge in *E′*.
  - That is, *G'* is an **induced** subgraph of *G* if every vertex of G' is a vertex of *G* and each pair of vertices in *G′* are adjacent in *G'* if and only if they are adjacent in *G*.
  - For the subgraph to be an induced subgraph, only delete edges of adjacent deleted vertices from the larger graph.
- Notice that every induced subgraph is also an ordinary subgraph, but not conversely.
- Example: let graphs be G1, G2, G3 and G4. Determine the graphs that are subgraphs and/or induced subgraphs in relation to G1.
  - *G*2 is a subgraph and an induced subgraph: every edge in *G*1 that connects vertices in *G*2 is also an edge in *G*2.
  - *G*3 is a subgraph: the edge {*a*, *b*} is in *E*1 but not *E*3, even though vertices ***a*** and *b* are in *V*3.
  - *G*4 is not a subgraph: even though it looks like all we did is remove vertex ***e***. The reason is that in *E*4 we have the edge {*c*, *f*} but this is not an element of *E*1, so we don't have the required *E*4 ⊆ *E*1.



5.2.3 Handshake Lemma

- **Degree of Vertex ($d(v)$):** the number of edges emanating from a given vertex.
- In any graph, the sum of the degrees of vertices in the graph is always twice the number of edges.
- The handshake lemma is sometimes called the degree sum formula.
- Degree sequence: the list of every degree of every vertex in the graph.

- In any graph, the number of vertices with odd degree must be even.

$$\sum_{v \in V} \boldsymbol{d(v) = 2e}$$

$$d(v): degree\ of\ vertex$$
$$e: edges$$

- Example: How many vertices and edges for a graph with the degree sequence of: $(4, 4, 3, 3, 3, 2, 1)$
  - Vertices: 7
  - Edges: $\frac{4+4+3+3+3+2+1}{2} = 10\text{e}$

5.2.4 Name Graphs

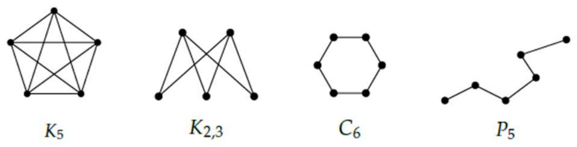
- **Complete Graph ($\boldsymbol{k_n}$):**
  - A graph in which every pair of vertices n is adjacent.
  - Complete graphs have twice the number of edges than they do vertices:

$$e = 2v$$

- **Bipartite graphs:** A graph for which it is possible to divide the vertices into two disjoint sets A and B such that there are no edges between any two vertices in the same set.
- **Complete bipartite graph ($\boldsymbol{k_{m,n}}$):**
  - A bipartite graph for which every vertex in the first set m is adjacent to every vertex in the second set n.
  - The total number of edges is given by the degree of 1 vertex in one set multiplied by the number of vertices in that same set. This is the same as the handshake Lemma.

$$\boldsymbol{e = d(v) * m}$$

- **Cycle ($\boldsymbol{C_n}$):** A path that starts and stops at the same vertex but contains no other repeated vertices. That is, a big loop on *n* vertices.
- **Path $\boldsymbol{P_n}$:** A **path** is a walk that doesn't repeat any vertices (or edges) except perhaps the first and last. That is, just one long path on *n* + 1 vertices (so *n* edges). If a path starts and ends at the same vertex, it is called a cycle**.**
- **Connected:** A graph is connected if there is a path from any vertex to any other vertex.
- **Walk:** A sequence of vertices such that consecutive vertices (in the sequence) are adjacent (in the graph).
- **Trail:** A walk in which no edge is repeated.



5.3 Trees

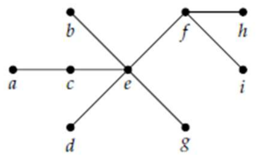
- A tree is defined as a connected graph containing no cycles.
- Every tree is also a bipartite graph.
- **Forest:** is a graph containing no cycles. Note that this means that a connected forest is a tree.
- **Leaves**: vertices of degree one
- **Properties of trees:**
  - A graph T is a tree if and only if between every pair of distinct vertices of T there is a unique path.
  - A graph F is a forest if and only if between any pair of vertices in F there is at most one path.
  - Any tree with at least two vertices has at least two vertices of degree one.
  - If T is a tree with v vertices and e edges, then:

$$\boldsymbol{e = v - 1}$$

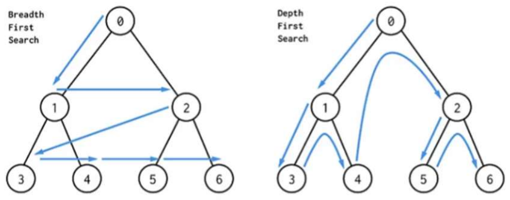
5.3.1 Rooted

- **Root:** a vertex of a tree from which the tree emanates from. Ancestry will be different dependant on which vertex is designated the root in a graph.
- **Parent and Child:** if two vertices are adjacent one of them is the parent of the other and the other is the child. Of the two, the parent is the vertex closer to the root.
- **Siblings:** vertices that have the same parent but are not adjacent.
- **Cousins:** vertices that have a common ancestor (grandparent).
- **Grandparent and grandchild:** the child of a child is the grandchild of the vertex.

- **Ancestor and Descendant:** v is a descendant of vertex u provided u is a vertex on the path from v to the root. u then is an ancestor of v.
- Example: Consider the tree below where vertex *f* is the root,
  - *e*, *h*, and *i* are the children of *f* , and are siblings of each other.
  - *a* is a child of *c*, and a descendant of *f* .
  - The vertex *g* is a descendant of *f* , in fact, is a grandchild of *f* .
  - Vertices *g* and *d* are siblings, since they have the common parent *e*.

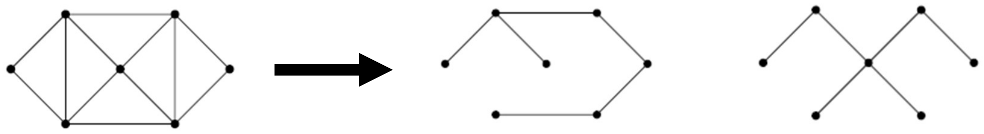


- **Breath first search:** visiting all vertices in the same generation before any vertices of the next generation.
- **Depth first search:** travelling as far from the root as fast as possible, then backtrack until we can move forward again.



5.3.2 Spanning

- Given a connected graph *G*, a spanning tree of *G* is a subgraph of *G* which is a tree and includes all the vertices of *G*.
- That is, you can only remove edges and not vertices.
- Every connected graph has a spanning tree.
- A graph which can be drawn (in the plane) without any edges crossing.
- **Solving:** you can add edges or remove edges.
- Example: the two spanning trees on the right can be derived from the graph on the left.

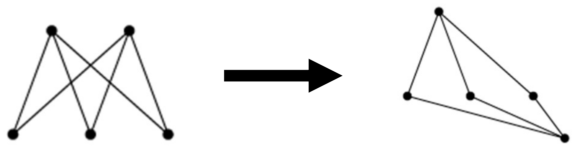


5.4 Planar Graphs

- **Planar Graph:**
  - A graph which can be drawn (in the plane) without any edges crossing.
  - Even though a graph does not look planar, it still might be.
- **Face:** a plane region of the graph formed by the related edges and vertices of a graph.
- **Boundary:**
  - Is a shared edge between faces in the graph.
  - In a planar graph each edge is used as a boundary exactly twice.
  - The total number of boundaries around all the faces in a graph is given by:

$$B = 2e$$

- Example:
  - The complete bipartite graph on the left can be redrawn into a planar graph on the right
  - The planar has 3 faces including the outside face.
  - There are $B = 2(6) = 12$ boundaries below.



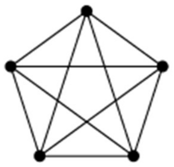
- **Euler's Formula:** determines the number of faces required to have a planar graph in relation to the vertices, edges and faces.

$$\boldsymbol{v - e + f = 2}$$

$$v: vertices$$
$$e: edges$$
$$f: faces$$

5.4.1 Non-Planar:

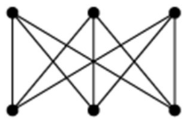
- Graphs which contain edges crossing.
- Trying to redraw the graph below with no edges crossing cannot be possible. Since there are too many edges and too few vertices. The edges will need to intersect at least once if redrawn.



- **Girth (g):** the number of edges in the smallest cycle (which is a face) in a graph.
- Multiplying g and the f gives us the smallest total number of edges to surround all faces in a graph.
- For a planar graphs, gf must be smaller than or equal to the total number of boundaries:

$$\boldsymbol{g\,f \leq 2e}$$

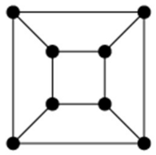
- Nonplanar graph exists when the above expression is false.
- **Proving nonplanar graphs**:
  - Step 1: Apply Euler's formula to find faces.
  - Step 2: Find the value of g.
  - Step 3: Substitute the values into the expression.
  - Step 4: Determine if the graph is nonplanar if expression is false.
- <u>Example:</u> K5 is not planar
  - Assume that K_5 is planar then $e = 2v$. Since $v = 5$ and $e = 10$
  - By Euler's formular we have $f = -5 + 10 + 2 = 7$.
  - Each face must be surrounded by at least $g = 3$ edges.
  - Since $B = 2e$, $e = 10$ and $f = 7$ then: $3f \leq 2e \equiv 3(7) \leq 2(10) \equiv 21 \nleq 20$
  - This is a contradiction so in fact K_5 is not planar.
- <u>Example:</u> it is not possible to connect each of three houses to each of three utilities without the lines crossing.
  - $K_{3,3}$ is a complete bipartite and assume to be planar, then $v = 6$ and $e = 3 * 3 = 9$.
  - By Euler's formula $f = -6 + 9 + 2 = 5$.
  - By inspection, each face must be surrounded by at least $g = 4$ edges.
  - Since $B = 2e$ for a planar graph, $e = 9$ and $f = 5$ then: $4f \leq 2e \equiv 4(5) \leq 2(9) \equiv 20 \nleq 18$
  - Which is clearly false. Thus $K_{3,3}$ is not planar.



5.4.2 Polyhedra:

- **Polyhedron:** is a geometric solid made up of flat polygonal faces joined at edges and vertices.
- **Convex polyhedral:**
  - All convex polyhdra are planar graphs
  - which means that any line segment connecting two points on the interior of the polyhedron must be entirely contained inside the polyhedron.
  - *every* convex polyhedron can be projected onto the plane without edges crossing.
- **Projecting a convex polyhedra onto the plane**
  - Think of placing the polyhedron inside a sphere, with a light at the centre of the sphere.
  - The edges and vertices of the polyhedron cast a shadow onto the interior of the sphere.
  - You can then cut a hole in the sphere in the middle of one of the projected faces and "stretch" the sphere to lie down flat on the plane.

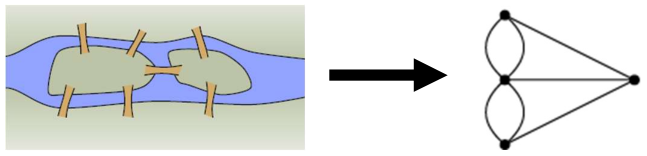
- The face that was punctured becomes the "outside" face of the planar graph.
- Example: a cube



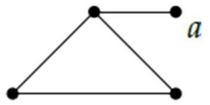
- **Applying Euler's formula to polyhedra**
  - Since every convex polyhedron can be represented as a planar graph, we see that Euler's formula for planar graphs holds for all convex polyhedra as well.
- <u>Example:</u> Is there a convex polyhedron consisting of three triangles and six pentagons?
  - There are 3 edges per triangle then, $3 * 3 = 9$ edges. There are 5 edges per pentagon, then, $6 * 5 = 30$.
  - Since $B = 2e$, then $39/2 = e$, is an impossibility.
- <u>Example:</u> Is there a convex polyhedron consisting of three triangles, six pentagons and five heptagons?
  - There are 3 edges per triangle then, $3 * 3 = 9$ edges, There are 6 edges per triangle then, $6 * 5 = 30$ edges, There are 5 edges per heptagon then, $5 * 7 = 30$ edges.
  - Since $B = 2e$, then $e = 74/2 = 37$.
  - By Euler's formula: $v = 37 - 14 + 2 = 25$ and use v to count the edges again.
  - Each vertex must have degree *at least* three (that is, each vertex joins at least three faces since the interior angle of all the polygons must be less that 180◦), so the sum of the degrees of vertices is at least 75.
  - Since the sum of the degrees must be exactly twice the number of edges given by the handshake lemma, this says that there are strictly more than 37 edges.
  - As a result, there is no such polyhedron.

5.5 Euler Trails and Circuits

- **Euler path (or walk)**: is a walk through the graph or multigraph which uses every edge exactly once.
- **Euler circuit (or tour):** is an Euler path which starts and stops at the same vertex. If a graph is not connected, there is no hope of finding such the circuit.
- **Proving:**
  - **Euler path:** if and only if there are at most two vertices with an odd degree.
  - **Euler circuit:** if and only if the degree of every vertex is even.
- <u>Example:</u> the graphical representation of the Konigsberg bridges is shown below:
  - Since the bridges of Konigsberg graph has all four vertices with odd degree, there is no Euler path through the graph. Thus, there is no way for the townspeople to cross every bridge exactly once.



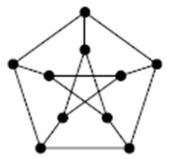
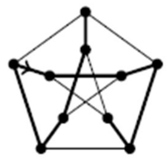
- **Proving Euler Circuits:**
  - There is no Euler circuit when there is vertices with odd degrees.
  - Consider the reasoning for a graph with $v = 3$ and $e = 3$. As such there are both an Euler path and circuit.
  - Now introduce a spike which is a vertex *of* degree 1.
  - if you try to make an Euler circuit, you see that you will get stuck at the vertex. It is a dead end.
  - That is, unless you start there. But then there is no way to return, so there is no hope of finding an Euler circuit.
  - There is however an Euler path. It starts at the vertex *a*, then loops around the triangle. You will end at the vertex of degree 3.



5.5.1 Hamilton's Graphs:

- **Hamilton path:** is a path which visits every vertex exactly once.
- **Hamilton cycles:** is a Hamiltonian path which starts and stops at the same vertex.
- **Proving**
  - **Hamilton circuit:**

  - **Hamilton path:** There is no known simple test for whether a graph has a Hamilton path.
- Example**:** The graph on the left has a Hamilton path as shown on the right.

5.5.2 Fleury's Algorithm

- **Bridge:** is an edge in a connected graph that disconnects the graph.
- Fleury's algorithm can be used to find an Euler trail or circuit. The algorithm functions as follows:
  - Step 1: the Input is an Eulerian graph G.
  - Step 2: Choose a starting vertex of odd degree if one exists.
  - Step 3: Go to any available edge, choosing a bridge only if there is no alternative. Then record that edge, erase the edge and any isolated vertex.
  - Step 4: Repeat step 2 until there are no more edges.

6 Proofs

6.1 Learning Objectives

- Analyse proofs
- Construct simple proofs for mathematical statements
- Select and explain appropriate methods of proofs.

6.2 Methods Of Proof

- A proof is a sequence of logical arguments, starting from a proposition p (the premise) and ending at a proposition q (the conclusion).
- Writing proofs is a bit of an art. Like any art, to be truly great at it, you need some sort of inspiration, as well as some foundational technique.
- There are a relatively small number of standard proof styles that are often used in mathematics.

6.3 Direct Proof

- Proving the conclusion as a result of the proposition.
- Direct proofs are especially useful when proving implications. That is, 'if *p*, then *q*' ($p \rightarrow q$).
- **Solving:**
  - Step 1: Identify p and q
  - Step 2: assume p is true.
  - Step 3: use p to show that q is true.
- Example: if n is odd then $n^2 + n$ is even
  - let $p = n$ is odd
  - let $q = n^2 + n$ is even.
  - Assume p to be true then let $n = 2k + 1$

$$n^2 + n = (2k+1)^2 + 2k + 1 = 4k^2 + 4k + 2$$
$$= 2(2k^2 + 2k + 1)$$

  - Since q is true, then p is true and $p \rightarrow q$ is true which shows the statement to be true.

6.4 Indirect proof

- Proving based on the contrapositive of the implication.
- Proving $p \rightarrow q$ is equivalent to proving $\neg q \rightarrow \neg p$, as these two statements are logically equivalent.
- **Solving:**
  - Step 1: Identify p and q and their countrapositives $\neg p$ and $\neg q$
  - Step 2: assume $\neg q$ is true.
  - Step 3: use $\neg q$ to show that $\neg p$ is true.
- Example: n is even if $-6n^2 + 5n - 9$ is odd
  - let $p = n$ is even and $q = -6n^2 + 5n - 9$ is odd
  - let $\neg p = n$ is odd and $\neg q = -6n^2 + 5n - 9$ is even
  - Assume $\neg q$ to be true then let $n = 2k + 1$

$$-6n^2 + 5n - 9 = 6(2k+1)^2 + 5(2k+1) - 9 = -24k^2 - 14k + 10$$
$$= 2(-12k^2 - 7k + 5)$$

- Since $\neg q$ is true, then $\neg p$ is true and $\neg q \to \neg p$ is true which shows the statement to be true.

6.5 Proofs of Equivalence

- Since $p \leftrightarrow q \equiv (p \to q) \land (q \to p)$, hence, to prove $p \leftrightarrow q$, we must prove $p \to q$ and $q \to p$.
- Note that we do not have to use the same method of proof to prove $p \to q$ and $q \to p$.
- **Solving:**
  - Step 1: Identify p and q and their counter positives $\neg p$ and $\neg q$
  - Step 2: Proof $p \to q$ using direct proof method.
  - Step 3: Proof $\neg q \to \neg p$ using indirect proof method.
- <u>Example:</u> prove that n is odd I and only if $3n^2 - 3n + 5$ is even
  - let $p = n$ is odd and $q = -6n^2 + 5n - 9$ is even
  - let $\neg p = n$ is even and $\neg q = -6n^2 + 5n - 9$ is odd
  - Direct proof: assume that p is true, then let $n = 2k + 1$

$$3n^2 - 3n + 5 = 3(2k+1)^2 - 3(2k+1) + 5 = 12k^2 + 8k + 10$$
$$2(6k^2 + 4k + 5)$$

  - Since q is true, then p is true and $p \to q$ is true which shows the statement to be true.
  - Indirect proof: assume that $\neg q$ is true, then let $n = 2k$

$$3n^2 - 3n + 5 = 3(2k)^2 - 3(2k) + 5 = 12k^2 - 6k + 5$$
$$2(6k^2 - 3k) + 5$$

  - Since $\neg q$ is true, then $\neg p$ is true and $\neg q \to \neg p$ is true which shows the statement to be true.
  - Since both $p \to q$ and $\neg q \to \neg p$ are both trye, then it shows the statement to be true.

6.6 Proof by contradiction

- To prove that a proposition p is true, prove that $\neg p$ is false ($\neg p \to F$).
- That is, assume that the conclusion p is false, and then proof that it leads to a contradiction.
- As a result, the only conclusion is that $\neg p$ is false, so p is true.
- **Solving:**
  - Step 1: Identify p and $\neg p$
  - Step 2: assume $\neg p$ is true.
  - Step 3: use $\neg p$ to show that p is true.
- <u>Example:</u> proof that there is no largest integer
  - Let $p =$ there is no largest integer
  - Let $\neg p =$ there is a largest integer
  - Assume that $\neg p$ then the largest integer is n. However, since $\mathrm{n} + 1$ is also an integer where $\mathrm{n} + 1 > \mathrm{n}$, it is not possible that n is the largest integer. Hence, this shows the statement to be true,

6.7 Proof by Cases

- Need to proof a universal quantifier where a break down into cases is needed to consider all scenarios.
- That is, rather than prove that $p \to q$, let p_1, p_2, etc. be the cases to consider. Then prove that $(p_1 \to q) \land (p_2 \to q) \land (p_3 \to q) \land \ldots$, where $p_1 \lor p_2 \lor p_3 \lor \ldots \equiv p$.
- **Solving:**
  - Step 1: Identify the cases to consider
    Step 2: use cases to show that statement is true.
- Example: prove that for all $n \in \mathbb{N}$, $n^2 + n$ is even
  - Case 1: when n is even, then let $n = 2k$

$$n^2 + n = (2k)^2 + 2k = 4k^2 + 2k$$
$$2(2k^2 + k)$$

  - Case 2: when n is odd, then let $n = 2k + 1$

$$n^2 + n = (2k+1)^2 + 2k + 1 = 4k^2 + 6k + 2$$
$$2(2k^2 + 3k + 1)$$

  - Since both cases show that the result is even, then this shows that the statement is true.

6.8 Proof by Counter Example

- To proof a universal statement false, there is no need for a list of logical statements but just a counter example.
- That is, using an example that disproves the statement.
- Example: for all numbers $x \in \mathbb{R}$, $x^2 > x$
  - Let $n = \frac{1}{2}$, then

$$\left(\frac{1}{2}\right)^2 \geq \frac{1}{2}$$

$$\frac{1}{4} \ngeq \frac{1}{2}$$

  - Since $\frac{1}{4}$ is not greater than or equal to $\frac{1}{2}$, it shows that the statement is false.

7 Recurrence Relations and Induction

7.1 Learning Objectives

- find the elements of a sequence given in closed form or in recurrence form
- evaluate partial sums of sequences
- solve linear recurrence of order 1 and of order 2.

7.2 Sequences:

- A sequence is an ongoing list of numbers, for example:
  - List of natural numbers: $1, 2, 3, 4, \ldots$
  - List of odd numbers: $1, 3, 5, 7, 9, \ldots$
  - List of powers of 2: $2, 4, 8, 16, 32, \ldots$
  - Fibonacci sequence: $0, 1, 1, 2, 3, 5, 8, 13, 21, 34, \ldots$
- A sequence is a function defined on the domain of natural numbers $\mathbb{N} \cup \{0\}$
- The numbers appearing in the sequence are called its elements. $3,7,11$ and 415 are all elements of the sequence of odd numbers.
- The index of an element indicates its position in the sequence. Usually (by convention) indices start from 0. The index of 5 in the sequence of odd numbers is 2.
- Elements of a sequence are given using a letter (corresponding to the sequence), and the index is given as a subscript:
- In the sequence of odd numbers, $a_0 = 1$, $a_4 = 9$, $a_{231} = 463$.
- The sequence itself is denoted as $(a_n)_{n \geq 0}$, $(a_n)_{n \in \mathbb{N}}$ or simply (a_n)

7.2.1 Closed Form

- A closed formula for a sequence $(a_n)_{n \in N}$ is a formula for a_n using a fixed finite number of operations on n. That is, function in terms of n. The values of the elements are given as the function on the domain:

$$\mathbb{N} \cup \{\mathbf{0}\}\text{: } \boldsymbol{a_n = f(n)}$$

- <u>Example</u>: The sequence defined by the function $a_n = n + 2$. Then,

$$f(n) = n + 2$$
$$a_0 = f(0) = 0 + 2 = 2$$
$$a_1 = f(1) = 1 + 2 = 3$$
$$a_2 = f(2) = 2 + 2 = 4$$
$$a_3 = f(3) = 3 + 2 = 5$$

7.2.2 Recurrence Form

- **Recursive definition:** (also inductive definition) for a sequence $(a_n)_{n \in N}$ consists of a recurrence relation and initial condition.
- **Recurrence relation:** is an equation relating a term of the sequence to previous terms (terms with smaller index). That is, when each element is given as a function of the elements preceding it: $a_n = f(a_{n-1}, a_{n-2}, \ldots)$.
- **Initial condition:** a list of a few initial terms of the sequence a_0, $a_1, \ldots$

$$\boldsymbol{f(a_n) = f(a_{n-1}) + expression}$$

- **Recurrence order**: The number of preceding elements in the function.

- Example: $a_n = a_{n-1} + 2$ where $a_0 = 1$
  - Recurrence form: $a_n = a_{n-1} + 2$,
  - Initial elements: $a_0 = 1$

$$\begin{gathered} f(a_n) = f(a_{n-1}) + 2 \\ a_1 = f(a_0) + 2 = 1 + 2 = 3 \\ a_2 = f(a_1) + 2 = 3 + 2 = 5 \\ a_3 = f(a_2) + 2 = 5 + 2 = 7 \\ a_4 = f(a_3) + 2 = 7 + 2 = 9 \end{gathered}$$

7.2.3 Partial Sums

- **Partial sum:** Given any sequence $(a_n)_{n\in N}$, we can always form a new sequence from this by changing the values of n in $(b_n)_{n\in N}$. Thus, a partial sum is the sum of the elements of a sequence $(a_n)_{n \geq 0}$ all the way up to index k.

$$b_n = a_0 + a_1 + a_2 + \cdots + a_n = \sum_{\boldsymbol{i=0}}^{\boldsymbol{k}} \boldsymbol{a_i}$$

- Example: Solve $\sum_{i=0}^{3} a_i$ where $a_i = 10 + 3i + 2i^2$

$$\begin{gathered} a_0 = 10 + 3(0) + 2(0)^2 = 10 \\ a_0 = 10 + 3(1) + 2(1)^2 = 15 \\ a_0 = 10 + 3(2) + 2(3)^2 = 24 \\ a_0 = 10 + 3(3) + 2(3)^2 = 37 \\ \sum_{i=0}^{3} a_i = a_0 + a_1 + a_2 + a_3 = 10 + 15 + 24 + 37 = 86 \end{gathered}$$

7.2.4 Difference Equations

- **Difference equations** are properties of a sequence that allows us to determine what degree the $f(n)$ is.
- **First difference of a sequence** $(\boldsymbol{a_n})$:
  - the sequence formed from the difference between each element in a sequence.
  - b_n is constant for a linear sequence.

$$\boldsymbol{b_n = a_{n+1} - a_n}$$

- **Second difference of a sequence** $(\boldsymbol{a_n})$:
  - the sequence formed from the difference between each element in the first difference sequence.
  - c_n is constant for a quadratic sequence.

$$\boldsymbol{c_n = b_{n+1} - b_n}$$

- **Further Difference of a sequence:** In the same way, we can define the third, fourth, fifth, … differences.
- Example: calculate the first order and second order difference for the sequence 6,8,10,20,36,58 ...

$$\begin{gathered} b_0 = a_1 - a_0 = 6 - 8 = -2 \\ b_1 = a_2 - a_1 = 10 - 6 = 4 \\ b_2 = a_3 - a_2 = 20 - 10 = 10 \\ b_3 = a_4 - a_3 = 36 - 20 = 16 \\ b_4 = a_5 - a_4 = 58 - 36 = 22 \end{gathered}$$

$$\begin{gathered} c_0 = b_1 - b_0 = 4 - (-2) = 8 \\ c_1 = b_2 - b_1 = 10 - 4 = 8 \\ c_2 = b_3 - b_2 = 16 - 10 = 8 \\ c_3 = b_4 - b_3 = 22 - 16 = 6 \end{gathered}$$

7.3 Arithmetic Sequences

- Also called linear sequence can be of two forms linear and quadratic.

7.3.1 Linear sequence:

- Defined by the recurrence relation $a_n = a_{n-1} + d$ for some number d.
- The value of d can be a multiple, constant, etc… That is, we move from one element of the sequence to the next element by always adding the same number.
- Closed form of an arithmetic sequence is given by:

$$\boldsymbol{a_n = a_0 + nd}$$

- **Find Closed form of linear Sequence**
  - Step 1: calculate the first order difference to find d
  - Step 2: find the first element of the sequence to be a_0
  - Step 3: substitute these values into the final form $a_n = a_0 + nd$
- Example: find closed form for $2,5,8,11,14,\ldots$
  - Calculate first order difference

$$\begin{gathered}
b_0 = a_1 - a_0 = 5 - 2 = 3 \\
b_1 = a_2 - a_1 = 8 - 5 = 3 \\
b_2 = a_3 - a_2 = 11 - 8 = 3 \\
b_3 = a_4 - a_3 = 14 - 11 = 3
\end{gathered}$$

  - The first element is $a_0 = 2$
  - The closed form is $a_n = a_0 + nd = 2 + 3n$

7.3.2 Quadratic Sequence:

- Closed form Formula is given by:

$$\boldsymbol{a_n = an^2 + bn + c}$$

- **Find Closed form of quadratic Sequence**
  - Step 1: calculate the first and second order difference.
  - Step 2: find c by substituting the first element of the sequence a_0 into $a_0 = a(0)^2 + b(0) + c$
  - Step 3: solving simultaneous equations by using and substituting the values of a_1 and a_2
  - Step 4: substitute the values of c, a and b into the final form $a_n = an^2 + bn + c$
- Example: find the closed form of $-3,0,-1,-6,-15,-28,-45,\ldots$
  - calculate the first and second order difference.

$$\begin{gathered}
b_0 = a_1 - a_0 = 0 - (-3) = 3 \\
b_1 = a_2 - a_1 = -1 - 0 = -1 \\
b_2 = a_3 - a_2 = -6 - (-1) = -5 \\
b_3 = a_4 - a_3 = -15 - (-6) = -9 \\
b_4 = a_5 - a_4 = -28 - (-15) = -13
\end{gathered}$$

$$\begin{gathered}
c_0 = b_1 - b_0 = -1 - 3 = -4 \\
c_1 = b_2 - b_1 = -5 - (-1) = -4 \\
c_2 = b_3 - b_2 = -9 - (-5) = -4 \\
c_3 = b_4 - b_3 = -13 - (-9) = -4
\end{gathered}$$

  - $a(0)^2 + b(0) + c = -3$
  - solving simultaneous equations by using and substituting the values of a_1 and a_2

$$\begin{gathered}
a_1 = a(1)^2 + b(1) - 3 \\
3 = a + b \\
a_2 = a(2)^2 + b(2) - 2 \\
1 = 2a + b \\
a = -2 \; and \; b = 5
\end{gathered}$$

  - substitute the values of c, a and b into the final form $a_n = an^2 + bn + c = -2n^2 + 5n - 3$

7.3.3 Closed Form of partial Sums

- If a_n is an arithmetic sequence with recurrence formula $a_n = a_{n-1} + \ d$ or the closed formula of $a_n = a_0 + nd$, its partial sum can be calculated using the formula

$$\boldsymbol{\sum_{i=0}^{k} a_i = \frac{(k+1)(2a_0 + kd)}{2}}$$

- Example: find $\sum_{i=0}^{5} a_i$ where $a_0 = 2$ and $a_n = a_{n-1} + 3$

$$\sum_{i=0}^{5} a_i = \frac{\mathbf{(5+1)(2 \times 2 + 5 \times 3)}}{\mathbf{2}} = \mathbf{57}$$

7.4 Geometric Sequences

- In a geometric sequence, we multiply to find the next element, rather than adding.
- A geometric sequence is defined by the recurrence relation $a_n = ka_{n-1}$, for some number k.
- That is, each element is obtained by multiplying the previous element by the same number k.
- Closed form of an arithmetic sequence is given by:

$$\boldsymbol{a_n = k^n a_0}$$

7.4.1 Closed Form of partial Sums

- If $\boldsymbol{a_n}$ is a geometric sequence with recurrence formula $a_n = ka_{n-1}$ or the closed formula $a_n = k^n a_0$ its partial sum can be calculated using the formula:

$$\boldsymbol{\sum_{i=0}^{n} a_i = a_0\left(\frac{k^{n+1}-1}{k-1}\right)}$$

- Example: find $\sum_{i=0}^{5} a_i$ where $a_0 = 2$ and $a_n = 3a_{n-1}$

$$\sum_{i=0}^{5} a_i = 2 \times \left(\frac{3^{12+1}-1}{3-1}\right) = 728$$

7.5 Linear Recurrence

- Finding the closed form for a recurrence relation is known as solving that recurrence relation.

7.5.1 Order 1

- Linear recurrence sequences have the values k and d as constants.
- All sequences of order 1 have each element only referring back to the first previous element.
- If $\boldsymbol{k = 1}$, then it is an arithmetic sequence.
- If $\boldsymbol{d = 0}$, then it is a geometric sequence with form $a_n k^n$ called exponential growth.

$$\boldsymbol{a_n = ka_{n-1} + d}$$

- The first difference of a linear recurrences is a geometric sequence.
- **Solving Order 1 linear recurrence:**
  - Step 1: Find the recurrence formula for the first difference
  - Step 2: Find the closed form of the first difference.
  - Step 3: Use this with the recurrence formula for the sequence to solve for a_n.
- <u>Example:</u> Find the closed form for a_n where a_0 and recurrence form $a_n = 3a_{n-1} + 4$
  - Find the recurrence formula for the first difference

$$b_0 = a_1 - a_0$$
$$b_0 = (3 \times 2 + 4) - 2 = 8$$
$$b_n = a_{n+1} - a_n$$
$$b_n = (3a_n + 4) - (3a_{n-1} + 4)$$
$$b_n = 3b_{n-1}$$

  - The closed form of the first difference is in the form $a_n = k^n a_0$ hence $b_n = 8 \times 3^n$
  - Use this with the recurrence formula for the sequence to solve for a_n

$$b_n = a_{n+1} - a_n$$
$$b_n = 3a_n + 4 - a_n$$
$$a_n = 4(3)^n - 2$$

7.5.2 Order 2

- Linear recurrence sequences have the values k_1, k_2 and d as constants.
- It has order 2 as each element refers to the two previous elements.
- Recursions of higher order will refer to a greater number of previous elements.
- Homogeneous case is when $d = 0$.

$$a_n = k_1 a_{n-1} + k_2 a_{n-2} + d$$

- **Solving Order 2 linear recurrence:**

- Step 1: Rewrite the relation to $a_n - k_1 a_{n-1} - k_2 a_{n-2} = 0$ and solve the polynomial $x^2 - k_1 x + k_2 = 0$
- Step 2: If the solutions are x_1 and x_2, then the closed form of the sequence will be $a_n = s_1 x_1^n + s_2 x_2^n$
- Step 3: Use the initial conditions $a_0 = s_1 x_1^0 + s_2 x_2^0$ and $a_1 = s_1 x_1^1 + s_2 x_2^1 = a_1$ to find the values of s_1 and s_2
- Step 4: substitute the values of s_1, s_2, x_1 and x_2 into the final form $a_n = s_1 x_1^n + s_2 x_2^n$

- Example: Find the closed form for b_n where $b_0 = 1, b_1 = 2$ and $b_n = -5b_{n-1} - 4b_{n-2}$
  - Rewrite the relation to $b_n + 5b_{n-1} + 4b_{n-2} = 0$ and solve $x^2 - 5x + 4 = 0$
  - $x_1 = -1$ and $x_2 = -4$ then the closed form of the sequence will be $a_n = s_1 x_1^n + s_2 x_2^n$
  - Use the initial conditions $a_0 = s_1 x_1^0 + s_2 x_2^0$ and $a_1 = s_1 x_1^1 + s_2 x_2^1 = a_1$ to find the values of s_1 and s_2

$$\begin{gathered} b_0 = s_1(-1)^0 + s_2(-4)^0 \\ 1 = s_1 + s_2 \\ b_1 = s_1(-1)^1 + s_2(-4)^1 \\ 2 = -s_1 - 4s_2 \\ s_1 = 2 \ and \ s_2 = -1 \end{gathered}$$

  - The closed form is then given by: $b_n = 2(-1)^n - (-4)^n$

8 Counting

8.1 Learning Objectives

- apply basic principles of counting
- select the appropriate principle when solving real-world problems
- identify and count the different ways items can be combined.

8.2 Product rule

- Suppose that we have objects composed of m parts, $A_1, A_2, \ldots, A_m$. If there are n_1 possible values for A_1, n_2 possible values for $A_2, \ldots, n_m$ possible values for A_m, then there are N possible values for objects composed of A_1 and A_2 ... and A_m.

$$\boldsymbol{N = n_1 \times n_2 \times \ldots \times n_m}$$

- "And" is a key word in determining that product applies.
- **Solving:**
  - Step 1: Determine all the parts that compose the object.
  - Step 2: For each part, determine how many options there are.
  - Step 3: Calculate the product.
- Example: with 26 letters and 0-9 digits. How many options are there in picking a letter and number
  - m = parts = 2
  - n = possibilities = 26 and 10
  - Therefore 1 object has 2 parts of 26 and 10 possibilities each respectively. Then, $N = 26 \times 10 = 260$

8.3 Sum rule

- Suppose that we have objects composed of m parts, $A_1, A_2, \ldots, A_m$. If there are n_1 possible values for A_1, n_2 possible values for $A_2, \ldots, n_m$ possible values for A_m, and if the objects have only one part then there are N possible values for objects composed of A_1 or A_2 ... or A_m.
- Note that the sum rule only works when the sets $A_1, \ldots A_2, \ldots A_m$ do not overlap.

$$\boldsymbol{N = n_1 + n_2 + \cdots + n_m}$$

- "Or" is a key word in determining that sum applies.
- **Solving:**
  - Step 1: Determine all the parts that compose the objects.
  - Step 2: For each part, determine how many options there are.
  - Step 3: Calculate the sum.
- Example: with 26 letters and 0-9 digits. How many options are there in picking a letter or number
  - m = parts = 2
  - n = possibilities = 26 and 10
  - $N = 26 + 10 = 36$

8.4 Counting number of functions

- Let X and Y be sets containing m and n elements respectively. That is, $|X| = m$ and $|Y| = n$.
- Consider all functions $f : X \to Y$. Then, each element of X can be mapped to any of the elements in Y.
- That is, in n ways, and we have m elements in X.

- Hence, by the product rule, the total number of functions that maps elements from X to Y:

$$\boldsymbol{N = n_1 \times n_2 \times \ldots \times n_m = n^m}$$

- That is, there are n^m distinct such functions $f : X \rightarrow Y$.
- **Solving:**
  - Step 1: Apply steps outline in product rule.
- Example: let $A = \{a, b, c, d\}$ and $B = \{1,3,5,7,9\}$ how many functions $f: A \rightarrow B$ are there?
  - m = parts = 4
  - n = possibilities = 5
  - Therefore 1 object has 4 parts of 5 possibilities each. Then, $N = 5 \times 5 \times 5 \times 5 = 5^4$

8.5 Counting injective functions

- If there are $N = n^m$ functions that maps elements from X to Y. Then, we have that
  - The $1st$ element of X can be mapped to one of n elements.
  - The $2nd$ element of X can be mapped to one of $(n - 1)$ elements.
  - The $3rd$ element of X can be mapped to one of $(n - 2)$ elements.
  - The mth element of X can be mapped to one of $(n - m + 1)$ elements.
- Therefore, the total number of one-to-one functions is:

$$\boldsymbol{N = n(n-1)(n-2)\ldots(n-m+1)}$$

- **Solving:**
  - Step 1: Apply steps outline in product rule.
- Example: let $A = \{a, b, c, d\}$ and $B = \{1,3,5,7,9\}$ how many injective functions $f: A \rightarrow B$ are there?
  - m = parts = 4
  - n = possibilities = 5,4,3,2 since one x can map only one y
  - Therefore 1 object has 4 parts of 5,4,3 and 2 possibilities each respectively. Then, $N = 5 \times 4 \times 3 \times 2 = 120$

8.6 Counting divisible numbers

- The number of natural numbers $m \leq N$ that are divisible by n is

$$\boldsymbol{m = \lfloor \frac{N}{n} \rfloor}$$

- Natural numbers m between M and N, inclusive that are divisible by a n is:

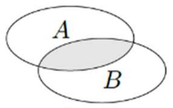
$$\boldsymbol{m = \lfloor \frac{N}{n} \rfloor - \lfloor \frac{M-1}{n} \rfloor}$$

- In both cases, divide and round down.
- **Solving:**
  - Step 1: determine the values of M, N and n
  - Step 3: apply formula
- Example: How many ≤ 13 are divisible by 3
  - $N = 13$ and $n = 3$. Then, $m = \left\lfloor \frac{13}{3} \right\rfloor = 4.33 \ldots = 4$

8.7 Inclusion exclusion principle

- Counting two sets that are overlapping, means that there are repeating elements. Hence, there needs to be a way to remove these repeating elements.
- **Solving:**
  - Step 1: compute the cardinality of $|A|$ and $|B|$
  - Step 2: compute the cardinality of $|A \cap B|$
  - Step 3: remove (once) the elements that were counted twice (which are in both A and B). That is, subtract the value $|A \cap B|$.

$$\boldsymbol{|A \cup B| = |A| + |B| - |A \cap B|}$$



$A \cap B$

- Example: Compute the number of positive integers less than 100 that are divisible by 2 and 5.
  - Calculate $|A|$ where $N = 99$ and $n = 2$. Hence, $m = \left\lfloor \frac{99}{2} \right\rfloor = 49$
  - Calculate $|B|$ where $N = 99$ and $n = 5$. Hence, $m = \left\lfloor \frac{99}{5} \right\rfloor = 19$
  - Calculate $|A \cap B|$ where $N = 99$ and $n = 5 \times 2$. Hence, $m = \left\lfloor \frac{99}{10} \right\rfloor = 9$
  - Calculate $|A \cup B| = 49 + 19 - 9 = 59$

8.8 Pigeonhole Principle

- It states that if m pigeons occupy n pigeonholes with $m > n$, then there must be at least one pigeonhole with at least two pigeons.
- If there are more pigeons than pigeonholes, there must be a hole with at least two pigeons.
- **Solving:**
  - Step 1: determine the pigeons
  - Step 2: determine the pigeonholes.
  - Step 3: apply the pigeonhole principle.
- Example: How many people in Melbourne share the same birthday.
  - Pigeons: 4.5 million people
  - Pigeonholes: 366 days in a year
  - Principle: $\frac{4.5 \times 10^6}{366} = 12295$

8.9 Permutations

- A permutation of a set of n distinct objects is an ordered arrangement of the objects.
- An r-permutation of a set is an ordered arrangement of r elements.
- Example:
  - SIT192 and 2T9I1S are permutations of $\{S, I, T, 1,9,2\}$.
  - 2-permutations of SIT192: SI, ST, S1, S9, S2, IS, IT, I1 etc.
  - 5-permutations of SIT192: SIT91, SI19T, S91IT, etc.
- **Factorial**: The factorial of $\boldsymbol{n}$, is the product of the first $\boldsymbol{n}$ natural numbers, that is:

$$\boldsymbol{n! = n(n-1)(n-2) \dots (2)(1)}$$

8.9.1 Counting permutation

- The number of r-permutations of n elements is

$$\boldsymbol{P(n, r) = n(n-1)(n-2) \dots (n-r+1) = \frac{n!}{(n-r)!}}$$

- Example: the number of ways to choose 3 balls out of 12 in a bag:

$$\frac{12!}{(12-3)!} = \frac{12 \times 11 \times 10 \times 9!}{9!} = 12 \times 11 \times 10 = 1320$$

8.10 Combinations

- A combination is an un-ordered collection of unique elements.
- An r-combination of a set of n elements is an unordered selection of a subset with r elements.
- Unlike in permutations, in combinations the order does not matter.
- Example:
  - SIT and TIS are different 3-permutations of SIT192, but they correspond to the same 3-combination.
  - Permutation of horse racing: picking first three finishers in exact order.
  - Combination of lotto: 6 winning numbers out of 45.

8.10.1 Counting combinations

- We can determine the number of r-permutations of n elements) as follows.
- This can be read as 'n choose r' and is referred to as the binomial coefficient.

$$\boldsymbol{C(n, r) = nCr = \binom{n}{r} = \frac{n!}{r!\,(n-r)!}}$$

- There are $r!$ times as many permutations as there are *r*-combinations
- Example: How 4-combinations of $\{S, I, T, 1,9,2\}$ are there

$$C(6,4) = \frac{6!}{4!\,2!} = \frac{6 \times 5}{2 \times 1} = 15$$

8.10.2 Counting combinations with repetition

- There are $C(n + r - 1, r)$ r-combinations from a set of n elements when repetition is allowed.
- We can prove this by applying the same technique to the general case:
  - n is number of types of objects available, each type has multiple copies.
  - r is number of objects you are taking/selecting.
- Selecting r objects from n types of objects whilst repetition is allowed, is like having $n - 1$ separators, and you are to place r objects out of the $n - 1 + r$ possible positions. Hence, it is $C(n - 1 + r, r)$.
- Therefore, there are $C(n - 1 + r, r)$ r-combinations from a set of n elements when repetition is allowed.

$$\boldsymbol{C(n-1+r,r) = \frac{(n-1+r)!}{r!\,(n-1)!}}$$

- Example: an ice-cream shops has 5 flavours, how many 3-combinations are possible?

$$C(5-1+3,4) = \frac{(5-1+3)!}{3!\,(5-1)!} = \frac{7!}{3!\,4!} = \frac{7 \times 6 \times 5}{3 \times 2 \times 1} = 35$$

- ***| Representation:**
  - To count these, we will use an alternative representation of these combinations, using "*" and "|" to represent a combination.
  - Since the order doesn't matter with combinations, we get to pick an order that is useful to us. Let's take every combination in the list and put them in alphabetical order.
  - Example: How many 4 letter combinations composed of the letters "ABC" are there
    - We put a '|' to symbolise a change of letters (between 'A' and 'B' and between 'B' and 'C').
    - If one of the letters is not in our string, we put the '|' before the next one. That is, 'A|BB|C', '|B|CCC', 'AAA|B|' or 'AA||CC'.
    - We replace every letter with a '*'*. So, 'ABBC' will become* '*|**|*' and '|***|*' corresponds to "BBBC".
- **Integer equations:**
  - Let x_A, x_B, ... be the number of times we pick an object A, B, ... respectively.
  - We want to select N letters in total where $x \geq 0$ for $i \in \{A, B, C\}$, so we are solving:

$$\boldsymbol{x_A + x_B + \cdots = N}$$

- Example: how many solutions are possible for $\boldsymbol{x_1 + x_2 + x_3 = 4}$?

$$C(3-1+4,4) = \frac{(3-1+4)!}{4!\,(3-1)!} = \frac{6!}{4!\,2!} = \frac{6 \times 5}{2 \times 1} = 15$$

9 Complexity (Advanced)

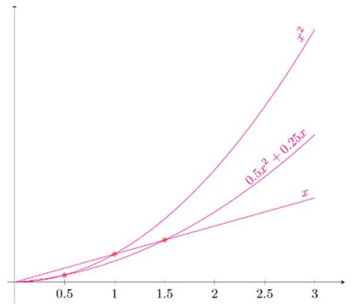
9.1 Learning outcomes

- Compare two functions using big O notation
- Calculate witness pairs for big O computations.

9.2 Computational Complexity

- When designing an algorithm, we may need to include many loops and conditionals (if... then... instructions). This will impact on the running time of the algorithm, particularly as the size of the input increases.
- It is important to have a sense of how efficient an algorithm is, particularly when the size of the input is large since increasing the input has a significant effect on them.
- The **complexity** of an algorithm is a measure of how long (or, how many steps) it will take an algorithm to run, with respect to the size of its input. The lower the complexity, the more efficient an algorithm is.
- Usually, we are only interested in the **order of magnitude** of the run time. If we have two loops, we only look at the one that takes longest to complete.
- If we measure the running time of an algorithm based on the input size, we obtain a function. So, complexity theory is about comparing functions.
- **Comparing runtime:**
  - Function complexity can be classified based on how they behave – linearly, quadratically, etc.

- We study only the asymptotic behaviour (what the curves look like as the input grows towards infinity) since there are many other instructions in the algorithm that would prevent the run time from being a pure linear function of the input and just measuring the time for some input might be misleading.
- This makes sense, as computer programs seldom contain a loop and nothing else. Hence, as the input grows towards infinity some of the functions classify as having 'linear complexity' are not linear.
- For example, if we compare the curves below, $\boldsymbol{x^2}$, $\mathbf{1/2}\boldsymbol{x^2} + \mathbf{1/4}\boldsymbol{x}$ and $\boldsymbol{x}$, each of the curves is 'better' (below the other two) for *some* input (values of $\boldsymbol{x}$). However, the choice changes as x increases.

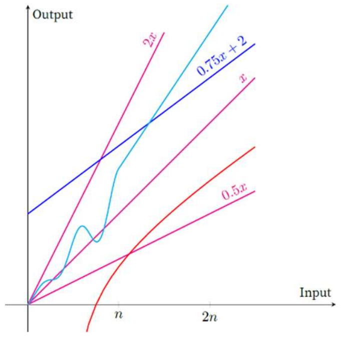


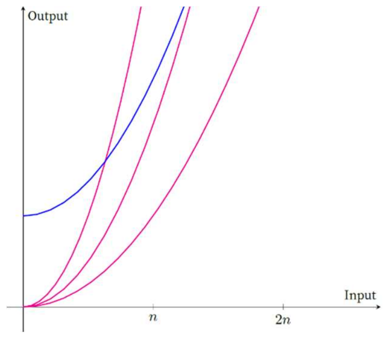
9.3 Big-O Notation

- Notation used to provide an idea of the worst possible runtime of an algorithm.
- A function $f(x)$ is said to be of order $g(x)$, denoted $O(g(x))$, if:

$$\exists \boldsymbol{C}, \boldsymbol{k} \in \mathbb{R} : |\boldsymbol{f}(\boldsymbol{x})| \leq \boldsymbol{Cg}(\boldsymbol{x})\ \forall \boldsymbol{x} \geq \boldsymbol{k}$$

- C and k are referred to as witnesses that $\boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{O}(\boldsymbol{g}(\boldsymbol{x}))$.
- That is, the values of C and k are 'witnessing' the fact that $f(x)$ is order $g(x)$.
- **Linear complexity** – $\boldsymbol{O}(x)$**:**
  - The number of operations grows proportionally (or less) to the input.
  - For example: doubling the input doubles the running time.
  - The functions on the graph below tend to look more and more like straight lines as the input grows larger.
  - Some have a wiggly start, and some never actually turn into a *real* straight line, but just resembles one more and more.
  - All these curves are grouped into the same category of 'linear' complexity – O**(x)**.
  - As for linear complexity, functions don't have to be actual parabolas, they just need to look more and more like one as the input grows.
- **Quadratic complexity** – $\boldsymbol{O}(x^2)$**:**
  - the number of operations grows proportionally (or less) to the square of the input.
  - For example: doubling the input multiplies the running time by **4**.
  - The four curves on the graph below are all parabolas. All these curves are grouped into the same category of 'quadratic' complexity – $\boldsymbol{O}(x^2)$.

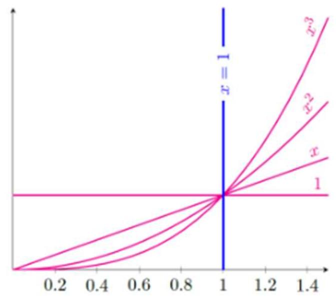




- **Calculating witness pairs:** To show that $f(x) = O(g(x))$, we need to find witnesses C and k.
  - Step 1: let $k = 1$ and analyse every term to be true for values of $k = 1$
  - Step 2: Get the absolute version of the function.
  - Step 3: Solve the equation to find C.
  - For example:

- **The value of $\boldsymbol{k}$**
  - There are no hard-and-fast rules to choose k. However, $k = 1$ is usually a safe first choice.

- In the graph below, you can see that the curves for x^k all meet at $x = 1$. Then, the curves with higher values of k are above curves with lower values. Since we analyse at asymptotic states, then $k = 1$ is the value that we should be starting our analysis from.



- **Generalisation: powers of x**
  - For any polynomial of the powers of with the general form of $f(x) = a_n x^n + a_{n-1}x^{n-1} + \cdots + a_1 x + a_0$, where $a_0, \dots, a_m$ are all real numbers and that $a_n \neq 0, f(x)$ is $O(x^n)$.
  - That is, the order is that of the algorithm is that of the first term without the coefficient.
- **Other functions**
  - We can plot a lot of 'common' functions, like we did with **1** and ***x*** and ***x*²**, etc. and compare them. Through analysis we can determine how these behave at difference values of k and thus allows us to develop a hierarchy that we can reuse.
  - Note that here we are using the natural log. Sources using other log functions (e.g. the decimal log) use different witnesses.

$f(x) \leq Cg(x)$	$\forall x \geq k$
$1 \leq x$	$\forall x \geq 3$
$\log x \leq x$	$\forall x \geq 1$
$x \leq x\log x$	$\forall x \geq 3$
$x \leq x^2$	$\forall x \geq 1$
$x^2 \leq 2^x$	$\forall x \geq 4$
$2^x \leq x!$	$\forall x \geq 4$
$x! \leq x^x$	$\forall x \geq 1$

- **Calculating witness pairs for other functions**
- Step 1: Analyse every term to be true for different values of k
- Step 2: Get absolute version of the function.
- Step 3: Solve the equation to find C and k.
- For Example: show that $6\log(x) + x^2 + 3x^x = O(x^x)$
  - $6\log(x) \leq 6x^x \qquad \forall x \geq 4$
  - $x^2 \leq x^x \qquad \forall x \geq 4$
  - $3x^x \leq 3x^x \qquad \forall x \in \mathbb{R}$

$$\begin{aligned}|6\log(x) + x^2 + 3x^x| &= 6\log(x) + x^2 + 3x^x && \forall x \geq 0\\ &= 6x^x + x^x + 3x^x && \forall x \geq 4\\ &= 10x^x && \forall x \geq 4\end{aligned}$$

  - So, $C = 10$ and $k = 4$ are witnesses that $f(x) = O(x^x)$

10 Number Theory (Advanced)

10.1 Learning Objectives

- apply the extended Euclidean algorithm for computing the modular inverse
- apply Fermat's and Euler's theorems to compute high powers in modular arithmetic
- solve linear equations in modular arithmetic.

10.2 Congruency:

- **The division algorithm:**
  - Given two integers a and b, we can always find an integer a such that

$$\boldsymbol{a = qb + r}$$

  - Where r is an integer satisfying $0 \leq r \leq |b|$
- **Modulo classes:**
  - The division algorithm implies that there are only b possible remainders when dividing by b.
  - These can be grouped into groups of classes.
  - Each group is called remainder class modulo b.
- **Congruency:**
  - a and b belong to the same remainder class if they both have the same remainder when divided by n. This is then expressed as below

$$\boldsymbol{a \equiv b\ (mod\ n)}$$

  - Note that is not the same as $a = b$.
  - $a \equiv b$ is not enough to denote that they belong to the same class since there is no way to know what n is. Hence the $mod\ n$ part of the expression.
  - For example: $26 \equiv 11 mod 5$. They both belong to the same remainder class.
    - $26(mod 5) = 1$
    - $11(mod 5) = 1$
- **Equivalence Relation**
  - Given any integers a, b and c and any positive integer n, then:

$$\boldsymbol{a \equiv a\ (mod\ n)}$$
$$\boldsymbol{a \equiv b\ (mod\ n) \quad \text{then} \quad b \equiv a\ (mod\ n)}$$
$$\boldsymbol{a \equiv b\ (mod\ n) \quad \text{and} \quad b \equiv c\ (mod\ n) \quad \text{then} \quad a \equiv c\ mod\ n}$$

- **Congruence Arithmetic:**
  - Suppose $a \equiv b\ mod\ n$ and $c \equiv d\ mod\ n$. Then:

$$\boldsymbol{a + c \equiv b + d\ (mod\ n)}$$
$$\boldsymbol{a - c \equiv b - d\ (mod\ n)}$$
$$\boldsymbol{ac \equiv bd\ (mod\ n)}$$

- **For example:** let $12 \equiv 3(mod 9)$ and $22 \equiv 4(mod 9)$

$$12 + 22 \equiv 3 + 4(mod 9) = \ 34 \equiv 7(mod 9)$$
$$12 \times 22 \equiv 3 \times 4(mod 9) = \ 264 \equiv 12(mod 9)$$

10.3 Bezout's Lemma

- For any integers a and b, if $gcd(a,\ b) = c$, there exist numbers m and n such that $m \cdot a + n \cdot b\ =\ c$.
- In particular, a and b are relatively prime if (and only if) there are numbers m and n such that $m \cdot a + n \cdot b = 1$.
- Limited to two-step solution.
- **Solving:**
  - Step 1: Apply Euclidean algorithm
  - Step 2: Re-arrange equations.
  - Step 3: Sub-in the first equation into the second one if required.
- For example: find Bezout's coefficients of 18 and 48

$$48 - 2 \times 18 = 12 \ldots (1)$$
$$18 - 1 \times 12 = 6 \ldots (2)$$
$$12 - 2 \times 6 = 0$$
$$gcd(18{,}48) = 6$$

  - Sub 1 into 2

$$18 - 1 \times (48 - 2 \times 18) = 6$$
$$3 \times 18 - 1 \times 48 = 6$$

  - The Bezout's coefficients are -1 and 3

10.4 Extended Eucledian Algorithm:

- The extended Euclidean algorithm is a version of the Euclidean algorithm, which can be used to find Bézout's coefficients.
- In the classical algorithm, we only need to keep track of the remainders of the division which eventually gives us the gcd.
- In the extended algorithm, we add two more parameters, the Bézout coefficients ***s*** and ***t*** and execute the algorithm for them. Note that, initially ***s*** corresponds to ***a***, and ***t*** corresponds to ***b***.

$$\begin{array}{ccc} b - m_0 \cdot a = r_2 & 1 - m_0 \cdot 0 = s_2 & 0 - m_0 \cdot 1 = t_2 \\ a - m_1 \cdot r_2 = r_3 & 0 - m_1 \cdot s_2 = s_3 & 1 - m_1 \cdot t_2 = t_3 \\ \vdots & \vdots & \vdots \\ r_{n-1} - m \cdot r_n = r_{n+1} & s_{n-1} - m \cdot s_n = s_{n+1} & t_{n-1} - m \cdot t_n = t_{n+1} \end{array}$$

- S**olving:**
  - Step 1: find the gdc between a and b
  - Step 2: solve for s and t
  - Step 3: Identify the gcd(a,b) and the coefficients.
- For example: Find Bezout's coefficients of 18 and 48

$$\begin{array}{ccc} 48 - 2 \times 18 = 12 & 1 - 2 \times 0 = 1 & 0 - 2 \times 1 = -2 \\ 18 - 1 \times 12 = 6 & 0 - 1 \times 1 = -1 & 1 - 1 \times -2 = 3 \\ 12 - 2 \times 6 = 0 & 1 - 2 \times -1 = 3 & -2 - 2 \times 3 = -8 \end{array}$$

  - Therefore, the Bezout's coefficients are -1 and 3.

10.5 Modular Inverse

- The modular inverse of a number *x* mod *p* is a number *y* such that

$$\boldsymbol{xy \equiv 1 \ (mod\ p)}$$

- For example:
  - 7 is the inverse of 4 mod 9. That is, $7 \times 4 = 28 \equiv 1(mod\ 9)$, since $28\ (mod\ 9) = 1$.
  - 6 is the inverse of 3 mod 17. That is, $6 \times 3 = 18 \equiv 1(mod\ 17)$, since $18\ (mod\ 17) = 1$.
  - 3 does not have an inverse mod 6.
- A number ***m*** has an inverse mod ***p*** if and only if ***m*** and ***p*** are relatively prime (**gcd(*m*, *p*) = 1**).
- Finding the inverse:
  - Step 1: Apply the extended Euclidean algorithm.
  - Step 2: Find the Bézout coefficient for ***m***.
- For example: Find the inverse of 3mod17

$$\begin{array}{ccc} 17 - 5 \times 3 = 2 & 1 - 5 \times 0 = 1 & 0 - 2 \times 1 = -5 \\ 3 - 1 \times 2 = 1 & 0 - 1 \times 1 = -1 & 1 - 1 \times -5 = 6 \\ 2 - 2 \times 1 = 0 & 1 - 2 \times -1 = 3 & -5 - 2 \times 6 = -17 \end{array}$$

  - The inverse if $3mod17$

10.6 Solving Linear Equations

- If **gcd(*m*, *p*) = 1**, then for any number ***x*** the equation below has a solution.

$$\boldsymbol{mx \equiv a \ (mod\ p)}$$

- Hence, If $\boldsymbol{p}$ is a prime number, then any linear equation $\mathbf{m}\boldsymbol{x} \equiv \mathbf{a}\ (\mathbf{mod}\ \boldsymbol{p})$ has a solution, except if $\mathbf{m} \equiv \mathbf{0}\ \mathbf{mod}\ \boldsymbol{p}$.
- **Proof**: If $\boldsymbol{m} \not\equiv \mathbf{0}\ \mathbf{mod}\ \boldsymbol{p}$, then m is not a multiple of ***p***, and so $\boldsymbol{gcd(a,\ p)} = \mathbf{1}$.
- Finding the inverse: let $a(mod\ b)$
  - Step 1: find numbers Bezout's coefficients m and n such that $\gcd(a, b) = 1$.
  - Step 2: multiple both sides by m.
  - Step 3: simply so that a is $0 \leq a \leq b$.
- For example: solve $7x \equiv 5mod9$
  - Find m

$$7 \times 4 - 1 \times 9 = 1$$

  - Then,

$$7x \times 4 \equiv 5 \times 4 mod 9$$
$$28x \equiv 20 mod 9$$
$$x \equiv 20 mod 9$$
$$x \equiv (18+2) mod 9$$
$$x \equiv 18 mod 9 + 2 mod 9$$
$$x \equiv 2 mod 9$$

10.7 Using Euler's and Fermat's Theorems:

- Euler's and Fermat's theorems can be used to:
  - quickly compute powers of ***a*** mod ***p***
  - compute the inverse of ***a*** mod ***p***.

10.7.1 Fermat's Little Theorem:

- **Fermat's little theorem** states that for a prime number ***p***, and a number ***a*** not divisible by ***p***:

$$\boldsymbol{a^p \equiv a(mod\ p)}$$

10.7.2 Euler's Theorem:

- Euler's theorem is a generalisation of Fermat's theorem for when $\boldsymbol{p}$ is not prime.
- In Euler's theorem, we only count the numbers that are relatively prime to $\boldsymbol{p}$.
- Applied for when the exponent is large.
- **Euler's totient function** $(\boldsymbol{\phi})$**:** That count, the number of integers between $\mathbf{1}$ and $\boldsymbol{p}-\mathbf{1}$ which are relatively prime to $\boldsymbol{p}$.
- We define Euler's totient function as:

$$\boldsymbol{\phi(p) = |\{m \le p-1 : m\ and\ p\ are\ relatively\ prime\}|}$$

- For any $\boldsymbol{a}$ such that $\boldsymbol{gcd(a,\ b) = 1}$

$$\boldsymbol{a^{\phi(p)} \equiv 1\ (mod\ p)}$$

- **Properties of $\boldsymbol{\phi}$:** Euler's totient function has the following properties:

$$\boldsymbol{\phi(p_{prime}) = p - 1}$$

$$\boldsymbol{\phi(a_{even}) = \phi(n_{prime} \times m_{prime}) = \phi(n_{prime}) \times \phi(m_{prime})}$$

$$\boldsymbol{\phi(a_{even}) = \phi(c^b_{prime}) = c^b_{prime} - c^{b-1}_{prime}}$$

10.7.3 Calculating Powers:

- Suppose that $\boldsymbol{gcd(a,\ p) = 1}$. Then, to calculate $\boldsymbol{a^m(mod\ p)}$, follow:
  - Step 1: Determine the value of $\phi(p)$.
  - Step 2: solve for r the equation $m = \phi(p) \times k + r$
  - Step 3: substitute values into $a^m \equiv a^r(mod\ p)$.
- For Example: Find $11^{27699} mod 13$
  - 13 is prime, then

$$\phi(13) = 12$$

  - Then:

$$m - \phi(p)k = r$$
$$27699 - 12 \times 2308 = 3$$
$$r = 3$$

  - Therefore:

$$11^{27699} mod 13 \equiv 11^3 mod 13$$
$$\equiv 1331 mod 13$$
$$\equiv (1326 + 5) mod 13$$
$$11^{27699} mod 13 \equiv 5 mod 13$$

10.7.4 Calculating Inverses:

- Re-arranging Euler's formula**:**

$$\boldsymbol{a^{\phi(p)} \equiv 1\ (mod\ p)}$$
$$\boldsymbol{a^{\phi(p)-1} \times a^{1} \equiv 1\ (mod\ p)}$$

- Then the inverse of $\boldsymbol{a}$ $(\boldsymbol{mod\ p})$ is given by:

$$\boldsymbol{a^{\phi(p)-1}}$$

- For Example: find the inverse of $3mod10$
  - 10 can be factorised into 2×5, then:

$$\phi(10) = \phi(2) \times \phi(5)$$
$$\phi(10) = 1 \times 4 = 4$$

  - Then the inverse is given by:

$$\phi(p) - 1 = 4 - 1 = 3$$

  - The inverse is given by $3^3 mod10$ and applying Euler's theorem, we get

$$3^3 mod10 = 27mod10$$
$$= (20 + 7)mod10$$
$$\equiv 7mod10$$

11 LaTeX (Advanced)

11.1 Learning Outcomes:

- typeset a document using the latex typesetting system and compile it into a pdf
- typeset simple mathematical equations in a LaTeX document
- include highlighted code in a LaTeX document.
- use the automatic referencing and citation mechanisms in LaTeX.

11.2 Introduction

- LaTeX is a high-quality typesetting system which includes features designed for the production of technical and scientific documentation.
- LaTeX is the standard for the communication and publication of scientific documents.
- LaTeX encourages authors not to worry too much about the appearance of documents and rather to concentrate on the content.
- MikTex:
  - MiKTeX is a modern TeX distribution for Windows, Linux and macOS.
  - MiKTeX's integrated package manager installs missing components to the project from the internet as required.
- Documentation on how to create documents can be found in the LaTeX website.

11.3 Equations

- There are many TeX libraries that support mathematical symbols and equations.
- Amsmath: is an extension package for LaTeX that provides various features to facilitate writing math formulas and to improve the typographical quality of their output.
- Amssymb: provides an extended symbol collection to amsmath.
- Equations are declared using "$" symbol.
- Example: $a = b$

11.4 Text Editing

- Geometry: One of the most popular libraries or physical configurations of the document. It allows you to configure things like the page dimensions.
- TeX's inbuilt commands such "\textbf{}" and "\begin{center}" will allow the user to embolden and centre text respectively.

11.5 Referencing and citations:

- Biblatex is a modern program to process bibliography information. It provides an easier and more flexible interface for managing and using referencing in your document.
- The library involves created a directory outside the LaTeX document and bring it into the project. This allows for separation of the referencing resources which decrease the length of the document.

- Documentation on how to use Biblatex can be found in the Overleaf website.
- To create caption, LaTeX has the \caption{} command which allows the user to specify whether it is a figure or table. The user can then use the “\ref” tag to reference.

12 Study Your Own Topic

12.1 Learning Outcomes

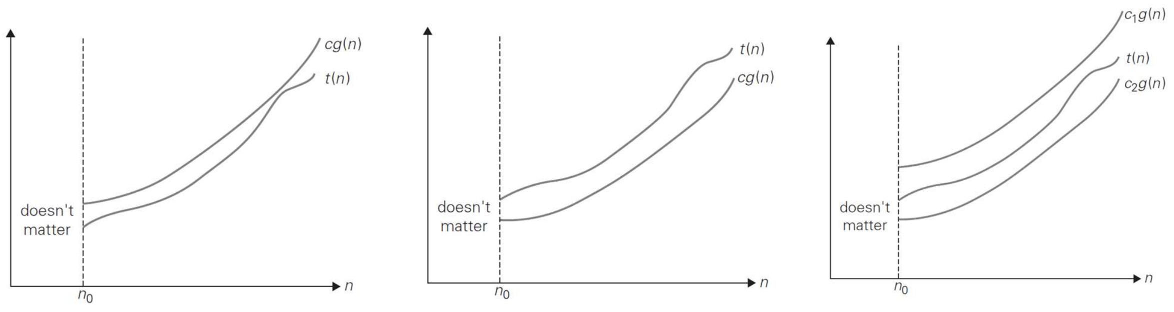
- Define the Big-Ω and Big-θ notations and explain how they are used in the analysis of algorithms.
- Calculate witness pairs for Big-Ω computations.
- Calculate witness pairs for Big-Θ computations.
- Use limits to analyse the orders of growth.

12.2 Preliminaries:

- Both time and space efficiencies are measured as functions of the algorithm’s input size.
- ***Time efficiency***:
  - also called ***time complexity***, indicates how fast an algorithm in question runs.
  - Time efficiency is measured by counting the number of times the algorithm’s basic operation is executed.
- ***Space efficiency***:
  - also called ***space complexity***, refers to the amount of memory units required by the algorithm in addition to the space needed for its input and output.
  - Space efficiency is measured by counting the number of extra memory units consumed by the algorithm
- **Input size:**
  - all algorithms run longer on larger inputs.
  - Therefore, it is logical to investigate an algorithm’s efficiency as a function of some parameter *n* indicating the algorithm’s input size.
- **Basic Operation:**
  - The thing to do is to identify the most important operation of the algorithm, called the ***basic operation***, the operation contributing the most to the total running time, and compute the number of times the basic operation is executed.
  - As a rule, it is not difficult to identify the basic operation of an algorithm: it is usually the most time-consuming operation in the algorithm’s innermost loop. For example, most sorting algorithms work by comparing elements (keys) of a list being sorted with each other; for such algorithms, the basic operation is a key comparison.
- **Analysis Framework**
  - the established framework for the analysis of an algorithm’s time efficiency suggests measuring it by counting the number of times the algorithm’s basic operation is executed on inputs of size *n*.
  - Let c_{op} be the execution time of an algorithm’s basic operation on a particular computer. Let $C(n)$ be the number of times this operation needs to be executed for this algorithm. We can thus estimate the running time $T(n)$ of a program implementing this algorithm on the computer by:

$$T(n) \approx c_{op}C(n)$$

  - A difference in running times on small inputs is not what really distinguishes efficient algorithms from inefficient ones. We need to analyse its order of growth.
  - the efficiency analysis framework concentrates on the order of growth of an algorithm’s basic operation count as the principal indicator of the algorithm’s efficiency.
- **Orders of growth:**
  - Order of growth of an algorithm is how the running time changes based on the <u>input size</u> going to infinity.
  - For large values of *n*, it is the function’s order of growth that counts.
  - To compare and rank such orders of growth, computer scientists use three asymptotic notations: *Big-O, Big-*Ω and Big-Θ.
  - Let $t(n)$ be an algorithm’s running time (indicated by its basic operation count $C(n)$), and $g(n)$ will be some simple function to compare the count with. Where n is input size. We can see how the Big-O, Big$-\Omega$, Big-Θ



- The asymptotic notations are used in the analyses of these cases are not the cases themselves.
- The efficiencies of some algorithms may differ significantly for inputs of the same size. For example, bubble sort algorithm is a native $O(n^2)$ sorting algorithm, but when the list is sorted it takes $O(n)$.
- The asymptotic notations tell nothing about input type for constant input size.

- **The *worst-case efficiency***
  - of an algorithm is its efficiency for the worst-case input of size *n*, which is an input (or inputs) of size *n* for which the algorithm runs the longest among all possible inputs of that size.
  - The way to determine the worst-case efficiency of an algorithm is: analyze the algorithm to see what kind of inputs yield the largest value of the basic operation's count *C(n)* among all possible inputs of size *n* and then compute this worst-case value *Cworst(n)*. it guarantees that for any instance of size *n*, the running time will not exceed *Cworst(n),* its running time on the worst-case inputs.
- **The *best-case efficiency***
  - ***O***f an algorithm is its efficiency for the best-case input of size *n*, which is an input (or inputs) of size *n* for which the algorithm runs the fastest among all possible inputs of that size.
  - we determine the kind of inputs for which the count *C(n)* will be the smallest among all possible inputs of size *n.*
  - Note that the best case does not mean the smallest input; it means the input of size *n* for which the algorithm runs the fastest.
  - Then we ascertain the value of *C(n)* on these most convenient inputs.
- ***The average-case efficiency:***
  - algorithm's behaviour on a "typical" or "random" input.
  - To analyse the algorithm's average case efficiency, we must make some assumptions about possible inputs of size *n*.
  - investigation of the average-case efficiency is considerably more difficult than investigation of the worst-case and best-case efficiencies.
  - there are many important algorithms for which the average case efficiency is much better than the overly pessimistic worst-case efficiency would lead us to believe. So, without the average-case analysis, computer scientists could have missed many important algorithms.

12.3 Big-Ω Notation

- is used to provide an idea of the best possible runtime of an algorithm.
- **Definition:** Let f and g be functions from the set of integers or the set of real numbers to the set of real numbers. We say that $f(x)$ is $\Omega(g(x))$ if there are constant C and k, with C positive and $x \geq k$ such that

$$\exists C, k \in \mathbb{R} : |f(x)| \geq C|g(x)| \;\forall\; x \;\geq k$$

- That is, a function $t(n)$ is said to be in $\Omega(g(n))$, if $t(n)$ is bounded below by some positive constant multiple of $g(n)$ for all large $n.$
- **Calculating witness pairs:**
  - There to ways to approach this.
  - **Find by Big-O:** when you are asked to propose a lower bound.
    - Step 1: Find the Big-O and its witnesses for reference.
    - Step 2: Use the table backwards to get an appropriate function.
    - Step 2: Choose constants for the inequality to be true. usually the constant from the big-O
  - **Find directly from the definition:** When asked to proof that $f(x) \in \Omega(g(x))$,
    - Step 1: Find the Big-O for reference.
    - Step 2: assert the inequality $f(x) \geq g(x)$ (its definition)
    - Step 2: Choose constants C and k for the inequality to be true.
- For example: since $6\log(x) + x^2 + 3x^x = O(x^x)$, show that $\Omega(x!)\; \forall x \geq 4$
  - From $O(x^x)$, then using the table in reverse $x! \leq x^x$. Let $c = 2$ and $k = 4$.

$$\begin{aligned} 6\log(x) + x^2 + 3x^x &\geq c \times x! \;\forall x \geq k \\ &\geq 10x! \;\forall x \geq 4 \end{aligned}$$

12.4 Big-Θ Notation

- is used to provide an idea of the average possible runtime of an algorithm
- **Definition:** Let f and g be functions from the set of integers or the set of real numbers to the set of real numbers. We say that $f(x)$ is $\Theta(g(x))$ if $f(x)$ is $O(g(x))$ and $f(x)$ is $\Omega(g(x))$. That is, $f(x)$ is $\Theta(g(x))$ if an only if there are positive numbers C_1 and C_2 and with a positive real number k, where $x \geq k$, such that

$$\exists C_1, C_2, k \in \mathbb{R} : C_1|g(x)| \leq |f(x)| \leq C_2|g(x)| \; \forall \, x \; \geq k$$

- A function $t\ (n)$ is said to be in $\Theta(g(n))$, if *t (n)* is bounded both above and below by some positive constant multiples of $g(n)$ for all large *n.*
- **Calculating witness pairs:**
  - There to ways to approach this.
  - **Find by Big-O and Big-Θ**:
    - Step 1: proof that that $f(x) \in O(g)$
    - Step 2: Proof that $f(x) \in \Omega(g)$
    - Step 3: Find witnesses for each if required.
  - **Find directly from the definition:**
    - Step 1: Proof the left side of the inequality. The upper bound.
    - Step 2: Proof the right side of the inequality. The lower bound.
    - Step 3: Find witnesses if required.
- <u>For example:</u> proof that $\frac{1}{2}(n^2 - n) \in \Theta(x^2)$
  - Proving the lefthand side

$$\frac{n^2}{2} - \frac{n}{2} \leq \frac{n^2}{2} \forall n \geq 0$$

  - Proving the righthand side

$$\frac{n^2}{2} - \frac{n}{2} \geq \frac{n^2}{2} - \frac{n}{4} \forall n \geq 2$$

  - Hence, the witnesses are $c_1 = \frac{1}{2}, c_2 = \frac{1}{4}$ and $k = 2$.

12.5 Limits Property:

- Limits provide a much more convenient method in analysing the order of growth of an algorithm.
- For doing so is based on computing the limit of the ratio of two functions in question.
- Suppose that the limit $L = \lim_{n \to \infty} \frac{f(n)}{g(n)}$ exists. Then

if $L = 0$, then $f \in O(g)$ and $f \notin \Omega(g)$
if $L = c$ then $f \in O(g)$ and $f \in \Omega(g)$ thus $f \in \Theta(g)$
if $L = \infty$ then $f \in \Omega(g)$ and $f \notin O(g)$

- To compute the limit if it exists, the standard *L'Hˆopital's rule* of calculus is useful:

$$\lim_{n \to \infty} \frac{f(n)}{g(n)} = \lim_{n \to \infty} \frac{f'(n)}{g'(n)}$$

- **Calculating witness pairs:**
  - Step 1: determine the values of $f(x)$ and $g(x)$
  - Step 2: Choose constants for the inequality to be true.
- For example: show that $f(x) = \frac{1}{2}n(n-1)$ has order $g(x) = x^2$

$$\lim_{n \to \infty} \frac{f(n)}{g(n)} = \lim_{n \to \infty} \frac{\frac{1}{2}x(x-1)}{x^2}$$

$$= \frac{1}{2} \lim_{n \to \infty} \frac{x^2 - x}{x^2}$$

$$= \frac{1}{2} \lim_{n \to \infty} \left(1 - \frac{1}{x}\right)$$

$$= \frac{1}{2}$$

- Since the limit is equal to a positive constant, the functions have the same order of growth. That is, $f(x) \in \Theta(g(x))$