Discrete Mathematics

Max Kasperowski

January 3, 2018

Contents

1 Logic

1.1 Propositional Logic—Zeroth Order Logic

In propositional logic, propositions are denoted by letters (p, q) and are formed by connecting other propositions using logical connectives. Propositions can either be true (T) or false (F).

Logical Connectives The logical connectives listed below are the basic connectives available in propositional logic in order of their precedence. Below are the truthtables corresponding to each of the connectives.

1. $\neg$, not
2. $\wedge$, and, $\bigwedge_{i=1}^{n} p_i$
3. $\vee$, or, $\bigvee_{i=1}^{n} p_i$
4. $\rightarrow$, $\Rightarrow$, implies (only if) defined as: $p \rightarrow q \equiv \neg p \vee q$
5. $\leftrightarrow$, $\Leftrightarrow$, is equivalent to (if and only if, iff) defined as: $p \leftrightarrow q \equiv (p \rightarrow q) \wedge (q \rightarrow p)$

Negation

p	¬ p
T	F T
F	T F

Logical Or

p q	p ∨ q
T T	T T T
T F	T T F
F T	F T T
F F	F F F

Equivalence

p q	p ↔ q
T T	T T T
T F	T F F
F T	F F T
F F	F T F

Logical And

p q	p ∧ q
T T	T T T
T F	T F F
F T	F F T
F F	F F F

Implication

p q	p → q
T T	T T T
T F	T F F
F T	F T T
F F	F T F

1.1.1 Definitions

Converse, Contrapositive, Inverse When given the proposition $p \rightarrow q$, $q \rightarrow p$ is its converse, $\neg q \rightarrow \neg p$ is its contrapositive and $\neg p \rightarrow \neg q$ is its inverse. The contrapositive is equivalent to the original proposition and the converse and inverse are also equivalent.

Tautology A proposition that is always true ($p \vee \neg p$).

Contradiction A proposition that is always false ($p \wedge \neg p$).

Contingency A proposition that is neither a tautology nor a contradiction.

Logical Equivalence p and q are logically equivalent if $p \leftrightarrow q$ is a tautology. The notation for equivalence is typically $\equiv$.

1.1.2 Properties

De Morgan's Laws

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

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

Identity Laws

$$p \vee F \equiv p$$

$$p \wedge T \equiv p$$

Domination Laws

$$p \vee T \equiv T$$

$$p \wedge F \equiv F$$

Idempotent Laws

$$p \vee p \equiv p$$

$$p \wedge p \equiv p$$

Negation Laws

$$p \vee \neg p \equiv T$$

$$p \wedge \neg p \equiv F$$

Commutative Laws

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

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

Associative Laws

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

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

Distributive Laws

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

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

Absorption Laws

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

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

1.1.3 Equivalence Proof

This is an example of how to perform an equivalence proof. The aim is to show that $\neg(p \vee (\neg p \wedge q))$ is logically equivalent to $\neg p \wedge \neg q$. We can prove this by forming a series of logical equivalences.

$$\begin{aligned}
\neg(p \vee (\neg p \wedge q)) &\equiv \neg p \wedge \neg(\neg p \wedge q) && \text{2nd De Morgan's law} \\
\neg p \wedge \neg(\neg p \wedge q) &\equiv \neg p \wedge (p \vee \neg q) && \text{1st De Morgan's law} \\
\neg p \wedge (p \vee \neg q) &\equiv (\neg p \wedge p) \vee (\neg p \wedge \neg q) && \text{Associative law} \\
(\neg p \wedge p) \vee (\neg p \wedge \neg q) &\equiv F \vee (\neg p \wedge \neg q) && \text{Negation law} \\
F \vee (\neg p \wedge \neg q) &\equiv \neg p \wedge \neg q && \text{Identity law}
\end{aligned}$$

1.2 Predicate Logic —First Order Logic

Predicate logic uses quantified variables over non-logical objects and allows the use of sentences that contain variables. This allows a generalisation of propositions for a set of variables from a domain.

1.2.1 Definitions

Predicates A predicate is a generalisation of propositions when the variable x is replaced by a specific element from its domain. $P(x)$ becomes a proposition. When no other domain is specified the domain is U.

Quantifiers Quantifiers are used to express that a proposition is true for all elements of the domain and that there exists some element in the domain for which it is true. They also have the highest precedence among the logical operators.

Universal quantifier	$\forall x P(x)$	$P(x)$ is true for every x in U
Existential quantifier	$\exists x P(x)$	$P(x)$ is true for some x in U

1.2.2 Properties

Uniqueness Quantifier The uniqueness quantifier is a commonly used quantifier to express that there is only one x for which $P(x)$ is true. It is usually written as $\exists!$ or $\exists_1$.

$$\exists_1 x P(x) \equiv \exists x (P(x) \wedge \forall y (P(y) \rightarrow y = x))$$

De Morgan's Laws De Morgan's laws for quantifiers state that $P(x)$ is not true for all x if and only if there exists an x for which $P(x)$ is false and furthermore that if $P(x)$ is false for all x if and only if there does not exist an x for which $P(x)$ is true.

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

1.3 Logical Proofs

Using logical inference it is possible to build an argument given a set of premises to reach a logical conclusion. An argument is valid if truth of all premises p_i implies that the conclusion q is also true.

$$\left(\bigwedge_{i=1}^{n} p_i\right) \to q \equiv T$$

1.3.1 Rules of Inference

Modus Ponens

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

Modus Tollens

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

Hypothetical Syllogism

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

Disjunctive Syllogism

$$\begin{array}{l} p \vee q \\ \underline{\neg p} \\ \therefore q \end{array}$$

Addition

$$\begin{array}{l} p \\ \hline \therefore p \vee q \end{array}$$

Simplification

$$\begin{array}{ll} \underline{p \wedge q} & \underline{p \wedge q} \\ \therefore p & \therefore q \end{array}$$

Conjunction

$$\begin{array}{l} p \\ \underline{q} \\ \therefore p \wedge q \end{array}$$

Resolution

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

Universal Instantiation

$$\begin{array}{l} \underline{\forall x P(x)} \\ \therefore P(c) \end{array}$$

Universal Generalisation

$$\begin{array}{l} \underline{P(c) \text{ for an arbitrary } c} \\ \therefore \forall x P(x) \end{array}$$

Existential Instantiation

$$\begin{array}{l} \underline{\exists x P(x)} \\ \therefore P(c) \text{ for some element } c \end{array}$$

Existential Generalisation

$$\begin{array}{l} \underline{P(c) \text{ for some element } c} \\ \therefore \exists x P(x) \end{array}$$

Universal Modus Ponens

$$\begin{array}{l} \forall x(P(x) \to Q(x)) \\ \underline{P(a) \text{ for a particular element } a} \\ \therefore Q(a) \end{array}$$

1.3.2 Inference Proof

All men are mortal. Socrates is a man. Prove that Socrates is mortal.

$P(x) \equiv T$ means that x is a person. $M(x) \equiv T$ means that x is mortal. s is the particular element Socrates.

1.	$\forall x P(x) \to M(x)$	premise
2.	$P(s)$	premise
3.	$P(s) \to M(s)$	UI (1)
4.	$M(s)$	Modus Ponens, (2) & (3)

2 Set Theory

A set is a collection of distinct elements without duplicates and the order of elements is unimportant.

2.1 Definitions

Containment $a \in A$ denotes that a is an element of the set A, whereas $a \notin A$ denotes that a is not contained in A. A set may contain other sets.

2.1.1 Describing Sets

Roster Method List all elements contained in the set. $S = \{a, b, c, d\}$ is equivalent to both $S = \{d, c, a, b\}$ and also $S = \{a, a, b, c, d, d\}$. When the pattern is clear, (...) may be used as in $S = \{a, b, c, \ldots, z\}$.

Set-Builder Method Specify the proposition that all members must satisfy. $S = \{x | P(x)\}$ denotes that all elements x in S must satisfy the proposition $P(x)$.

Interval Notation The elements in a set can be constrained by an open or closed interval. $[a, b] = \{x | a \leq x \leq b\}$ represents the closed interval between a and b. $(a, b) = \{x | a < x < b\}$ represents the open interval between a and b. Additionally combinations like $[a, b)$ and $(a, b]$ are possible.

2.1.2 Special Sets

Universal Set The universal set, denoted by U, contains everything currently under consideration.

Empty Set The empty set is the set that contains no elements. It is denoted by $\varnothing$. It is important to note that $\varnothing = \{\}$, but $\varnothing \neq \{\varnothing\}$. A set containing the empty set is not the equal to the empty set.

Singleton A singleton is a set containing exactly one element.

2.1.3 Set Equality

Two sets are equal if they contain the same elements. If A and B are sets then their equality is denoted as $A = B$.

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

.

2.1.4 Subsets

Set A is a subset of B if all elements in A are also in B. This relation is denoted as $A \subseteq B$.

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

Every set is always a subset of itself and the empty set $\varnothing$ is a subset of every other set.

Proper Subset When $A \subseteq B$, but $A \neq B$ then A is a proper subset of B, which is denoted as $A \subset B$.

$$\forall x(x \in A \rightarrow x \in B) \wedge \exists x(x \in B \wedge x \notin A) \equiv A \subset B$$

.

2.1.5 Set Cardinality

For finite sets the set cardinality describes the number of distinct elements in a set. It is denoted as $|S|$. The cardinality of the empty set is zero ($|\varnothing| = 0$). If A is the set containing all the letters of the english alphabet, then $|A| = 26$.

2.1.6 Power Sets

The set of all subsets of a set A is called the power set of A and is denoted as $\mathcal{P}(A)$. If $|A| = n$, then $|\mathcal{P}(A)| = 2^n$. For example, given the set $A = a, b$ we get the power set $\mathcal{P}(A) = \{\varnothing, \{a\}, \{b\}, \{a, b\}\}$.

2.2 Tuples

Tuples are a collection of ordered elements. An n-tuple is written as $(a_1, a_2, a_3, \ldots, a_n)$. Two n-tuples are equal if and only if all of their corresponding elements are equal.

$$(a, b) = (c, d) \equiv (a = c) \wedge (b = d)$$

A subset R of $A \times B$ is called a relation from the set A to the set B.

2.2.1 Cartesian Product

The cartesian product $A \times B$ is the set of ordered pairs (a, b) where $a \in A$ and $b \in B$.

$$A \times B = \{(a, b) | a \in A \wedge b \in B\}$$

The cartesian product of n sets is the set of tuples containing all possible combinations of elements as tuples.

$$A_1 \times A_2 \times \ldots \times A_n = \{(a_1, a_2, \ldots, a_n) | a_i \in A_i \text{ for } i = 1, 2, \ldots, n\}$$

The cardinality of the cartesian product is equal to the product of the cardinalities of the individual sets.

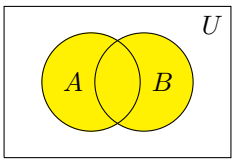
$$|A_1 \times A_2 \times \ldots \times A_n| = \prod_{i=1}^{n} |A_i|$$

2.3 Operations

The set operators are based on boolean algebra and are analogous to the operators in propositional calculus. There must always be a universal set U and all sets are assumed to be a subset of U.

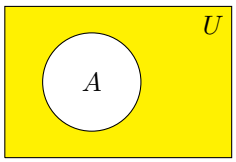
Union
The union of A and B is the set that contains those elements that are contained in A, B or in both.

$$A \cup B = \{x | x \in A \vee x \in B\}$$



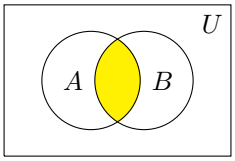
Complement
The complement of the set A contains all the elements in U that are not in A. It is equivalent to $U - A$.

$$\bar{A} = \{x \in U | x \notin A\}$$



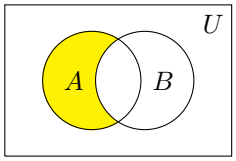
Intersection
The intersection of A and B is the set that contains those elements that are contained in both A and B.

$$A \cap B = \{x | x \in A \wedge x \in B\}$$



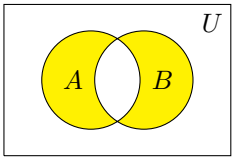
Difference
The difference $A - B$ is the set that contains all the elements in A but not in B. It is the union of A with the complement of B.

$$A - B = \{x | x \in A \wedge x \notin B\} = A \cap \bar{B}$$



Symmetric Difference
The symmetric difference is equivalent to a logical exclusive or meaning that the symmetric difference between A and B contains the elements that are in A or in B, but not those that are in both A and B.

$$A \oplus B = (A - B) \cup (B - A)$$



2.4 Properties

Inclusion-Exclusion Principle The inclusion-exclusion principle helps to calculate the cardinality of unions of sets. The principle states that the cardinality of the union of A and B is equal to the cardinality of A plus the cardinality of B minus the cardinality of the intersection of A and B.

$$|A \cup B| = |A| + |B| - |A \cap B|$$

De Morgan's Laws

$$\overline{A \cup B} = \bar{A} \cap \bar{B}$$

$$\overline{A \cap B} = \bar{A} \cup \bar{B}$$

Identity Laws

$$A \cup \varnothing = A$$

$$A \cap U = A$$

Domination Laws

$$A \cup U = U$$

$$A \cap \varnothing = \varnothing$$

Idempotent Laws

$$A \cup A = A$$

$$A \cap A = A$$

Commutative Laws

$$A \cup B = B \cup A$$

$$A \cap B = B \cap A$$

Associative Laws

$$A \cup (B \cup C) = (A \cup B) \cup C$$

$$A \cap (B \cap C) = (A \cap B) \cap C$$

Distributive Laws

$$A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$$

$$A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$$

Absorption Laws

$$A \cup (A \cap B) = A$$

$$A \cap (A \cup B) = A$$

Complement Laws

$$A \cup \bar{A} = U$$

$$A \cap \bar{A} = \varnothing$$

Double Negation Law

$$\overline{(\bar{A})} = A$$

3 Relations

3.1 Definitions

A relation is the subset of the cartesian product of a number of sets. The simplest relation is a binary relation such as R from the set A to the set B.

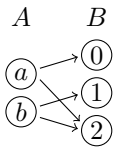
$$R \subseteq A \times B$$

Given are the following sets A and B and the relation R from A to B.

$$A = \{a, b\}, B = \{0, 1, 2\}, R = \{(a, 0), (a, 2), (b, 1), (b, 2)\}$$

This relation can be visualised in the following ways.

R	a	b
0	X	
1		X
2	X	X



The following notation can be used to simply show whether or not the pair (a, b) is in R.

$$a \ R \ b = (a, b) \in R$$

$$a \not R \ b = (a, b) \notin R$$

The number of relations on a set A is equal to 2 to the power of of the cardinality of its cartesian product.

$$2^{|A \times A|} = 2^{|A|^2}$$

3.2 Properties

Relations are sets so the regular set operations can be used on them. Additionally it is possible to create a composition of two relations. If R_1 is a relation from A to B and R_2 is relation from B to C then $R_2 \circ R_1$ is a relation from A to C.

Power of a relation

$$R^n = \left\{ \begin{array}{ll} R, & \text{for } n = 1 \\ R^{n-1} \circ R, & \text{for } n > 1 \end{array} \right\}$$

3.2.1 Reflexivity

A relation is reflexive if every element is related to itself.

$$\forall a[a \in U \rightarrow (a, a) \in R]$$

This property can be seen in a matrix if there is a diagonal line through the matrix.

R	a	b	c
a	1		
b		1	
c			1

If this property is negated then a relation is irreflexive.

3.2.2 Symmetry

In a symmetric relation for every pair (a, b) in R there must also be a pair b, a.

$$\forall a \forall b[(a, b) \in R \rightarrow (b, a) \in R]$$

The resulting matrix will also be a symmetric matrix.

R	a	b	c
a	0	1	0
b	1	0	1
c	0	1	0

In an antisymmetric relation for every pair which has a symmetric counterpart the elements must be related to themselves.

$$\forall a \forall y[(a, b) \in R \land (b, a) \in R \rightarrow x = y]$$

3.2.3 Transitivity

In a transitive relation if $(a, b) \in R$ and $(b, c \in R$ then $a, c \in R$ must also be true.

$$\forall a \forall b \forall c[(a, b) \in R \land (b, c) \in R \rightarrow (a, c) \in R]$$

3.3 Equivalence Classes and Partitions

3.3.1 Equivalence Relations

If a relation is reflexive, symmetric and transitive then it is an equivalence relation meaning that for a related pair (a, b), a and b are equivalent.

$$a \sim b$$

Examples of equivalence relations are $=, \equiv$.

3.3.2 Equivalence Classes

When elements of a set have some notion of equivalence they can be partitioned into a set of equivalence classes where all elements in the same equivalence class have an equivalence relation. The equivalent class of the element a from the set S, which has an equivalence relation $\sim$, is the set in which every element is equivalent to a.

$$[a] = \{x \in S | x \sim a\}$$

In set partitioned into equivalent classes x is equivalent to y if and only if x and y are in the same equivalence class.

$$x \sim y \leftrightarrow [x] = [y] \leftrightarrow [x] \cap [y] \neq \varnothing$$

3.3.3 Partition of a set

A set can be partitioned into a many disjoint non-empty subsets. The union of these subsets is the original set.

$$S = \bigcup_i A_i$$

Properties None of the subsets can be an empty set and no two subsets have any common elements.

$$A_i \neq \varnothing$$

$$i \neq j \rightarrow A_i \cap A_j = \varnothing$$

Partitioning a set into equivalent classes follows the same principle.

$$\bigcup_{a \in A} [a] = A$$

3.3.4 Partially Ordered Sets

A partially ordered set, also known as a poset, consists of a set together with a binary relation indicating that, for certain pairs of the set, one element precedes the other. Not every pair must be comparable in a partial order. In order to be a partial order, the relation must be reflexive, antisymmetric and transitive. A relation which is a partial order is denoted by $\preceq$. Given the poset $(S, \preceq)$ and the elements $a \in S$ and $b \in S$, then $a \preceq b$ means that a precedes b.

Totally Ordered Set If all elements in a set are comparable, then it is known as a totally ordered set.

Well Ordered Set If the set contains a least element then it is known as a well ordered set.

3.4 Functions

A special kind of relation is a function. A function is a subset of $A \times B$ with the restriction that no two elements may have the same first element.

$$\forall x \left[x \in A \rightarrow \exists y \left(y \in B \wedge (x, y) \in f\right)\right] \wedge \forall x, y_1, y_2 \left[\left((x, y_1) \in f \wedge (x, y_2) \in f\right) \rightarrow y_1 = y_2\right]$$

3.4.1 Definitions

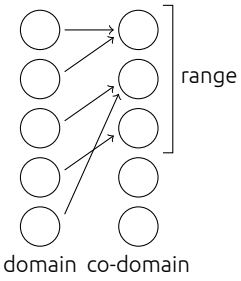
A function is a mapping from one set to another. A and B are two non-empty sets and the function f is a mapping from A to B.

$$f : A \rightarrow B$$

Each element of A is assigned to exactly to exactly one element B.

$$f(a) = b$$

In this case the set A is the domain of the function and the set B is the co-domain. The range is a subset of the co-domain and contains all the elements of B that f actually maps to.



3.4.2 Properties

Addition & Multiplication

$$(f_1 + f_2)(x) = f_1(x) + f_2(x)$$

$$(f_1 f_2)(x) = f_1(x) f_2(x)$$

Increasing & Decreasing

Increasing

$$\forall a, b \in A \left(a < b \rightarrow f(a) \leq f(b)\right)$$

Strictly Increasing

$$\forall a, b \in A \left(a < b \rightarrow f(a) < f(b)\right)$$

Decreasing

$$\forall a, b \in A \left(a < b \rightarrow f(a) \geq f(b)\right)$$

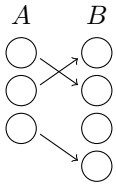
Strictly Decreasing

$$\forall a, b \in A \left(a < b \rightarrow f(a) > f(b)\right)$$

3.4.3 Types of Functions

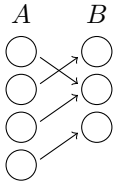
Injection Functions that are injective have a one-to-one mapping from A to B. That means that every a is assigned to one b and every b is the image of at most one pre-image a.

$$\forall a, b(f(a) = f(b) \rightarrow a = b)$$

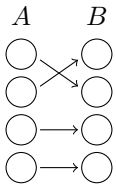


Surjection A surjective function maps A onto B, which means that for every element in B has a pre-image.

$$\forall b \in B\,[\exists a \in A(f(a) = b)]$$

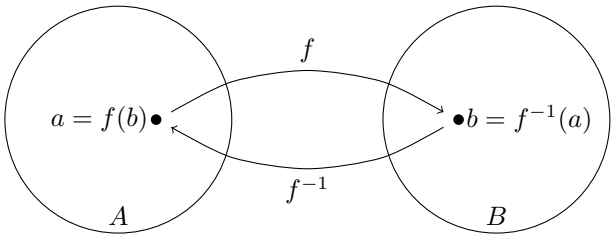


Bijection A bijection is a function, which is both injective and surjective.



3.4.4 Inverse Function

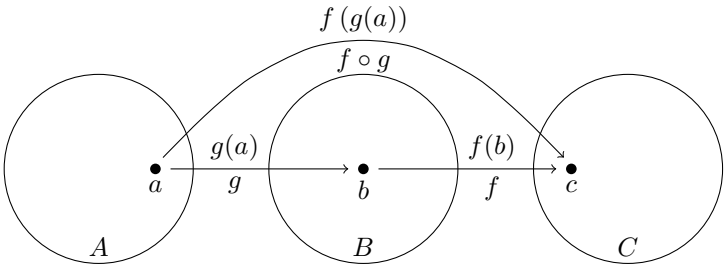
If the function f is bijective then it has an inverse. The inverse of f is denoted as f^{-1}.



3.4.5 Composition

Given the functions $f : B \to C$ and $g : A \to B$, the composition $f \circ g$ is the function $A \to C$.

$$f \circ g(x) = f\left(g(x)\right)$$



3.5 Relational Alegebra

Relational algebra defines additional operators for operations between two relations. This is used in relational databases, where a relation is a table and the tables attributes are the relation's sets. Each row in the below tables is a tuple of the relations.

Employee

Name	EmployeeId	DepartmentName
Harry	3415	Finance
Sally	2241	Sales
George	3401	Finance
Harriet	2202	Sales

Department

DepartmentName	Manager
Finance	George
Sales	Harriet
Production	Charles

3.5.1 Projection

$$\Pi_{a_1,\ldots,a_n}(R)$$

$a_1, \ldots, a_n$ are names of attributes in the relation R. The result of the projection is a new relation containing only the attributes specified in the projection.

3.5.2 Selection

$$\sigma_{\varphi}(R)$$

φ is a propositional formula. The selection returns a relation containing only those tuples for which φ is true.

3.5.3 Rename

$$\rho_{a/b}(R)$$

The result of renaming is identical to the original relation, but the the attribute a is renamed to b.

3.5.4 Join

$$R \bowtie S$$

The natural join merges two relations on a common attribute. If there is no common attribute then the natural join becomes the cartesian product combining all possible attributes. Joining the two relations from earlier on the common attribute DepartmentName, would yield the below relation.

Employee $\bowtie$ Department

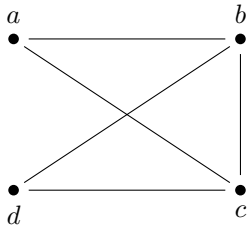
Name	EmployeeId	DepartmentName	Manager
Harry	3415	Finance	George
Sally	2241	Sales	Harriet
George	3401	Finance	George
Harriet	2202	Sales	Harriet

4 Graphs

4.1 Definitions

A graph G is defined by its vertices V and edges E. The set of vertices must be non-empty. Every edge in E has one or two endpoints.

$$
\begin{aligned}
&G = (V, E)\\
&V = \{a, b, c, d\}\\
&E = \{(a, b), (a, c), (b, c), (b, d), (c, d)\}
\end{aligned}
$$



Two vertices are adjacent or neighbouring if they have a connecting edge. An edge is incident with the vertices it connects.

4.1.1 Types of Graphs

Simple Graph In a simple graph there are no loops and no multiple edges allowed.

Multigraph In a multigraph it is allowed to have multiple edges between two vertices.

Pseudograph In a pseudograph there can be multiple edges as well as loops.

Digraph A directed graph has directed edges also known as arcs, so every edge is an ordered pair of vertices.

4.1.2 Neighbourhood

The neighbourhood of a vertex $N(v)$ is the set of all neighbours of that vertex. It is a subset of V.

If A is a subset of V, then $N(A)$ is the set of all vertices that are adjacent to at least one vertice in A.

$$N(A) = \bigcup_{v \in A} N(v)$$

4.1.3 Degree of a Vertex

Undirected Graphs In an undirected graph, the degree of a vertex $\deg(v)$ denotes the number of edges incident with the vertice. Loops contribute 2 to the degree.

Handshaking Theorem If $G = (V, E)$ is an undirected graph with m edges, then every new edge increases the degree of two vertices by one.

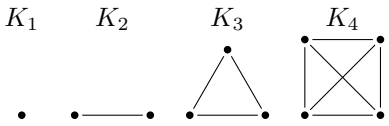
$$2m = \sum_{v \in V} \deg(v)$$

Directed Graphs A directed graph's vertices have an in-degree $\deg^-(v)$, which corresponds to the number of edges terminating at v and an out-degree $\deg^+(v)$, which corresponds to the number of edges originating from v. The sum of in-degrees equals the number of out-degrees and the number of edges.

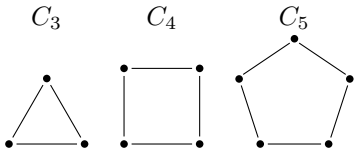
$$|E| = \sum_{v \in V} \deg^-(v) = \sum_{v \in V} \deg^+(v)$$

4.2 Special Types of Graphs

Complete Graphs A complete graph K_n has n vertices, where every vertice is connected to every other vertice.

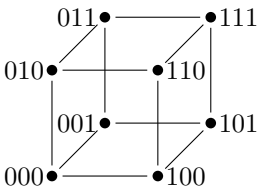


Cycles A cycle C_n is a graph with all vertices connected so that there is exactly one cycle. There must be at least three vertices.



Wheels If you add an addtional vertice to a cycle and connect all vertices of the cycle to it, then you get a wheel.

n-Cubes An n-Cube Q_n is a graph which has 2^n vertices. With Q_1 being a line, Q_2 a square, Q_3 a cube, Q_4 a hypercube and so on. They can be used to represent bitstrings of length n with each dimension of Q_n corresponding to a different index in the bitstring. For example, to represent a bitstring of length three we need to use Q_3, where each of the eight vertices of the graph correspond to a possible state of the bitstring. Flipping a single bit is equivalent to moving along one of the edges of the graph. The axes x, y, z could correspond to the indices $0, 1, 2$, therefore changing the bitstring from 010 to 011 moving along the edge on the z-axis.



4.2.1 Computer Network Architecture

Star Topology In a star topology there is one central control device to which all other devices are connected. All messages between devices are sent over the central device.

Ring Topology The devices are connected in a cycle with each device connected to two neighbours. Messages are passed along until they reach their destination.

Wheel Topology A wheel topology combines the star and ring topology.

n-Cube Topology An n-Cube based topology is used for mesh networks and connecting processors for parallel processing.

4.2.2 Bipartite Graphs

A bipartite graph is a simple graph where V can be partitioned into two disjoint sets V_1 and V_2 such that every edge connects a vertex from V_1 to a vertex from V_2 and there are no edges within V_1 or within V_2. A complete bipartite graph $K_{m,n}$ has m vertices in V_1 and n vertices in V_2 and each vertex in V_1 is connected to each vertex in V_2. These graphs can be used to match elements of one set to elements of another.

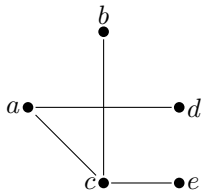
4.3 Subgraphs

Given the two graphs $G = (V, E)$ and $H = (W, F)$, H is a proper subgraph of F if $W \subset V$ and $F \subset E$. The union of two graphs is performed by taking the the union of the sets of vertices and the union of the sets of edges. Given the graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ then $G_1 \cup G_2 = (V_1 \cup V_2, E_1 \cup E_2)$.

4.4 Representation of Graphs

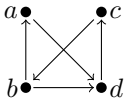
4.4.1 Adjacency Lists

An adjacency list consists of a list of all vertices, which each have an associated list of adjacent vertices.



vertex	adjacent vertices
a	c, d
b	c
c	a, b, e
d	a
e	c

In the above example we are limited to undirected graphs and we cannot represent multiple edges. In order to represent a digraph, the first column contains the initial vertices and the second column contains the connected terminal vertices.



initial vertex	terminal vertex
a	d
b	a, d
c	b
d	c

4.4.2 Adjacency Matrix

Another representation, more suitable for dense graphs, is an adjacency matrix.

$$A_G = [a_{ij}], a_{ij} = \left\{ \begin{array}{ll} 1, & \text{if } (v_i, v_j) \text{ is an edge of G} \\ 0, & \text{otherwise} \end{array} \right\}$$

To represent multigraphs the number of edges between i and j is used instead of one. A simple graph will produce a symmetrical matrix.

4.4.3 Incidence Matrix

An incidence matrix is a matrix of size $|V| \times |E|$. If an edge e_j is incident with v_i then the corresponding position in the matrix is filled with a one and a zero otherwise.

$$M = [m_{ij}], m_{ij} = \left\{ \begin{array}{ll} 1, & e_j \text{ is incident with } v_i \\ 0, & \text{otherwise} \end{array} \right\}$$

4.5 Isomorphism of Graphs

A function mapping a one-to-one correspondence from $G_1 = (V_1, E_1)$ to $G_2 = (V_2, E_2)$ is an isomorphism. The relationship between two simple graphs is called isomorphic or non-isomorphic.

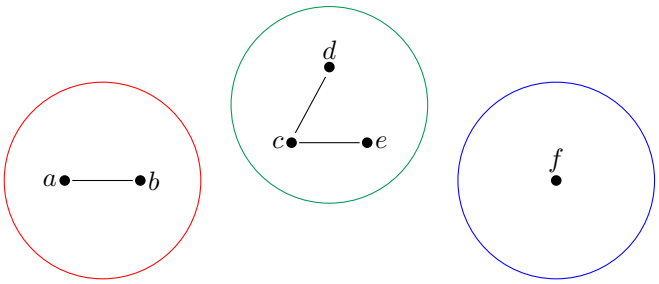
4.6 Paths

A path is a seqence of connected edges between two vertices. A path is known as a circuit when the initial and terminal vertices are the same. A path is said to pass through vertices and traverse edges. A simple path contains no edge more than once. The degree of seperation between two vertices is equal to the length of the shortest path between them.

4.6.1 Connectedness

All vertices in an undirected graph are connected if there is a path between any pair of vertices.

Connected Components A connected component is a connected subgraph that is not a proper subgraph of another connected component.



Cut Vertex Removal of a cut vertex creates a subgraph that is not connected. There will be more connected components than before. V' of V of $G = (V, E)$ is a vertex cut. It is the set of vertices that when removed will disconnect the graph. $G - V'$ is a disconnected graph. Graphs without cut vertices are called non-seperable graphs.

Vertex Connectivity $K(G)$ denotes the minimum number of vertices in a vertex cut. A graph is k-connected if $K(G) \geq k$.

Edge Connectivity E' denotes an edge cut analogous to a vertex cut. $\lambda(G)$ denotes the minimum number of edges in an edge cut.

$$K(G) \leq \lambda(G) \leq \min_{v \in V} \deg(v)$$

Directed Graphs A directed graph is strongly connected if all vertices are connected to all other vertices with regard the directions of the edges. It is weakly connected if it is connected when treated as a simple graph ignoring the directions.

4.6.2 Euler Paths and Circuits

Euler Circuit A simple circuit containing every edge in G. The necessary conditions for an Euler circuit are a starting vertex and every vertex must have an even degree.

Euler Path A simple path containing every edge in G. The necessary conditions for an Euler path are that the initial and terminal vertices have an odd degree and all other vertices have an even degree.

4.6.3 Hamiltonian Paths and Circuits

Hamiltonian Circuit A simple circuit containing every vertex in G exactly once.

Hamiltonian Path A simple path containing every vertex in G exactly once.

4.7 Planar Graphs

A planar graph is a graph that can be drawn in a plane without any edges crossing. That is called the graph's planar representation. The number of regions in a connected planar graph is given by Euler's formula.

$$r = e - v + 2$$

A graph is non-planar if and only if it contains a subgraph that is homeomorphic to $K_{3,3}$ or K_5.

4.8 Trees

A tree is a connected undirecte graph with no simple circuits. There exists exactly one path between every vertex of the tree.

4.8.1 Definitions

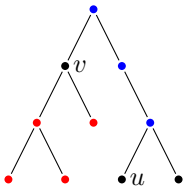
Forest An unconnected graph where every component is a tree is called a forest.

Rooted Trees In a tree one vertex may be designated as the root and all edges are directed away from it. Usually rooted trees are used in practice.

Parents & Children

In the figure on the left u is the parent of v and v is the child of u.

Ancestors and Descendants



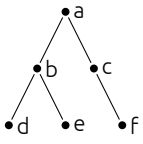
The ancestors of vertex u are all vertices in the path from the root to u, but excluding u itself. The descendants of v are all the vertices that have v as an ancestor.

Leaves & Internal Vertices A vertex that has no children is a leaf node. Vertices that have children are known as internal vertices.

Subtrees A subtree with vertex v as its root contains v and its descendants.

m-ary Trees Every internal vertex of an m-ary tree has no more than m children. A full m-ary tree is a tree where every internal vertex has m children. A balanced m-ary tree with a height h has all leaves at height h or $h-1$. There will be at most m^h leaves. The height of a full tree is $h = \lceil \log_m l \rceil$, if it is not full it is $h \geq \lceil \log_m l \rceil$.

Ordered Rooted Trees The elements in an ordered rooted tree are ordered from top to bottom and from left to right. Such as the following binary tree where each vertex represents a letter of the alphabet.



4.8.2 Applications of Trees

Binary Search Trees A binary search tree is a tree in which every value can be reached by making a comparison at each node and moving to either the left child if the desired value is smaller than the current node or to the right child if it is larger. Constructing a binary search tree is a form of sorting values. The number of permutations in a list of length n is $n!$. The minimum number of binary comparisons that must be made is therefore $\log_2(n!)$. The minimum time complexity for sorting algorithms based on binary comparisons is therefore $\Omega(n \log n)$.

Huffman Coding Huffman Coding is a method of encoding data into unique bitstrings for data compression. The frequency of each character is counted and stored. Then, a each character becomes a single node tree, so that there is a forest of trees for all characters. The two trees with the lowest frequency become the children of a new node which becomes their root. The frequency of this new node is the sum of frequencies of the children. This process repeats until there is only one root left. The bitstring encoding for each letter corresponds to the path taken through the tree.

Game Trees Trees can be used to store possible game states. The root node represents the start of the game when nothing has happened yet and every edge represents a possible action. The child nodes correspond to the new state of the game and leaves would represent final positions. For simple games like tic-tac-toe it is easy to construct an entire game tree to let a computer play optimally, but more complex games like chess can make it unviable to construct the entire tree. Partial game trees, which only look a few turns ahead can be used for complex games instead.

4.8.3 Tree Traversal

A traversal is the procedure used for visiting every vertex in an orderd tree. The three methods described below are all recursively defined and differ only in when they visit the root.

Pre-order Traversal First visit the root, then traverse the left child and finally traverse the right child.

In-order Traversal First traverse the left child, then visit the root and finally traverse the right child.

Post-order Traversal First traverse the left child, then traverse the right child and finally visit the root.

Expression Trees An example of the application of tree traversals is for expression trees for evaluating mathematical expressions. In an expression tree each internal vertex corresponds to an operator and the leaves correspond to the operands. When this tree is traversed in-order the expression is returned in infix notation. If pre-order traversal is used we get the expression in prefix notation or Polish notation, where parentheses are not necessary to remove ambiguity. Similarly post-order traversal yields postfix notation or reverse Polish notation.

4.8.4 Spanning Trees

A spanning tree of a graph G is a subgraph of G, which is a tree and contains every vertex in G. Every connected graph has a spanning tree.

Depth-first Search When using a depth-first search to construct a spanning tree by adding vertices and edges for as long as possible and then backtracking to add more edges. This is repeated until all vertices have been added to the spanning tree.

Breadth-first Search In a breadth-first search vertices are added to the tree level by level. All adjacent vertices are added recursively until there are no more vertices left to add.