Discrete Mathematics

Contents

1 Elementary Set Theory **3**
- 1.1 Relations Among Sets 4
- 1.2 Basic Operations of Sets 5
- 1.3 Index Sets 6
- 1.4 Some More Set Operations 7
- 1.5 Relations 9
  - 1.5.1 Equivalence Relations and Partitions 10
  - 1.5.2 Partial Order Relations 12
- 1.6 Functions 13
- 1.7 Function Composition 15
- 1.8 Isomorphisms of Sets 17
- 1.9 Cardinality of Sets 19
  - 1.9.1 Uncountable Sets 22

2 Finite Automata and Regular Expressions **24**
- 2.1 Deterministic Finite Automata 24
- 2.2 Nondeterministic Finite Automata 27
  - 2.2.1 Conversion of NFA to DFA 28
- 2.3 Regular Expressions 33
  - 2.3.1 Conversion of NFA to RE 35

3 Graph Theory **39**
3.1 Subgraphs and Complements 40
3.2 Walks, Paths, and Cycles 42
3.3 Distances 43
3.4 Connectedness 44
3.5 Graph Isomorphism 45
3.6 Self-Complementary Graphs 46
3.7 Bipartite Graphs 48
3.8 Trees 49
3.9 Adjacency Matrices 53
3.10 Dijkstra's Algorithm 57

1 Elementary Set Theory

A set is (informally) an unordered collection of elements. More formally, any set is defined by the membership relation $\in$ (read "belongs to", "is an element of", "is a member of", or "is in"), where $s \in S$ if and only if s is an element of the set S. If x is not a member of S, then we write $x \notin S$. We may define a set either as a list of all its members enclosed by curly braces – { and } – or in the form $Y = \{\, x \in X \mid P(x) \,\}$, where X is a previously defined set, and $P(x)$ is a predicate (a function with a true/false value depending on the value of x) – then Y is the set consisting of all members of X that satisfy the predicate $P(x)$ (i.e. those $x \in X$ for which $P(x)$ is true). The latter form of defining a set is called the set-builder notation. For example

$$S = \{1, \text{'a'}, 2.5, +, 9, -3\}$$

defines S to be the set consisting of the six elements 1, 'a', 2.5, $+$, 9, and -3, and

$$T = \{\, s \in S \mid s \text{ is a number} \,\}$$

defines T to be the set consisting of the four elements 1, 2.5, 9, and -3. The symbol $\mid$ is read as "such that" (and can be replaced by a $:$ as well).

The universal set (usually denoted by U) is the set consisting of all elements currently under consideration. We may write $\{\, x \mid P(x) \,\}$ to mean $\{\, x \in U \mid P(x) \,\}$.

The empty set (or null set) is the set $\varnothing$ that has no elements. That is, for every element a of the universal set, $a \notin \varnothing$.

Note. A variant of the set-builder notation replaces the element on the left side of $\mid$ by an expression involving one or more elements, with the specifications of memberships of these elements appearing on the right side of $\mid$, along with other predicates, if any. For example

$$S = \{\, 2n \mid n \in \mathbb{Z} \,\}$$

defines S to be the set of all elements obtained by doubling an integer – in other words, S is the set of even integers.

1.1 Relations Among Sets

A set A is a subset of a set B, denoted by $A \subseteq B$, if every element of A is an element of B. That is, for any element a (of the universal set), $a \in A \implies a \in B$. Then B is a superset of A, denoted by $B \supseteq A$. We may also say that B contains A, or that A is contained in B.

Two sets A and B are equal, written $A = B$, if each contains the other – i.e. $A \subseteq B$ and $B \subseteq A$. This is equivalent to the statement that A and B have exactly the same elements. Otherwise, A is not equal to B (written $A \neq B$). A is a proper subset of A (or is properly contained in B) if $A \subseteq B$ and $A \neq B$. Then we write $A \subsetneq B$. Similarly, B is a proper superset of A, denoted by $B \supsetneq A$, if $B \supseteq A$ and $B \neq A$.

Note. It is also common to use $\subset$ and $\supset$ instead of $\subseteq$ and $\supseteq$, respectively. They are usually *not* alternatives to $\subsetneq$ and $\supsetneq$, except when explicitly stated to be so.

The set of all subsets of a set A is called the power set of A, is denoted by 2^A or $\mathcal{P}(A)$. That is,

$$2^A = \{ S \mid S \subseteq A \}.$$

Exercise 1.1. Let A, B, and C be arbitrary sets.

1. Show that $A \subseteq A$.
2. Show that if $A \subseteq B$ and $B \subseteq C$, then $A \subseteq C$.
3. Show that the $\supseteq$ also satisfies these properties.
4. Show that $\varnothing \subseteq A$.
5. Show that if A is a set consisting of n elements, for some non-negative integer n, then 2^A contains 2^n elements.

1.2 Basic Operations of Sets

The union of two sets A and B is the set $A \cup B$ consisting of all elements that belong to A or B:

$$A \cup B = \{\, x \mid x \in A \text{ or } x \in B \,\}.$$

The intersection of two sets A and B is the set $A \cap B$ consisting of all elements that belong to A and B:

$$A \cap B = \{\, x \mid x \in A \text{ and } x \in B \,\}.$$

The complement of a set A is the set $\overline{A}$ of all elements (of the universal set) that do not belong to A:

$$\overline{A} = \{\, x \mid x \notin A \,\}.$$

We may also denote the complement of A by A', A^c, or $U \setminus A$.

Exercise 1.2. Let A, B, and C be arbitrary sets.

1. Show that $A \cap B \subseteq A \subseteq A \cup B$.
2. Show that $\cup$ is
   (i) associative: $(A \cup B) \cup C = A \cup (B \cup C)$,
   (ii) commutative: $A \cup B = B \cup A$, and
   (iii) idempotent: $A \cup A = A$.
3. Prove that $\cap$ is also associative, commutative, and idempotent.
4. Prove that $\cup$ distributes over $\cap$ and vice-versa:

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

5. Prove that $\cup$ and $\cap$ satisfy the law of absorption:

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

6. Show that the following are equivalent:
   (a) $A \subseteq B$.
   (b) $A \cup B = B$.
   (c) $A \cap B = A$.
7. Prove that $\cup$, $\cap$ and - satisfy De Morgan's laws. That is:

$$\overline{(A \cup B)} = \overline{A} \cap \overline{B}$$
$$\overline{(A \cap B)} = \overline{A} \cup \overline{B}.$$

8. Show that $A \cap \overline{A} = \varnothing$.

Two sets A and B are disjoint if their intersection is empty – i.e. $A \cap B = \varnothing$. Note that A and $\overline{A}$ are always disjoint. Also note that $\varnothing$ is disjoint with all sets, and is the unique set that is disjoint with itself.

Since $\cup$ and $\cap$ are associative, expressions of the form $A_1 \cup A_2 \cup \cdots \cup A_n$ and $A_1 \cap A_2 \cap \cdots \cap A_n$ are well-defined and unambiguous. These are called the n-ary (or finite) union and intersection, and denote them as $\bigcup_{i=1}^{n} A_i$ and $\bigcap_{i=1}^{n} A_i$, respectively. But it is possible to define unions and intersections of collections of sets even more generally. For this, we will discuss the concept of an indexing set.

1.3 Index Sets

A set I is an index set (or indexing set) of a set S if we can write S as

$$S = \{s_i \mid i \in I\}.$$

That is, each element of $s_i \in S$ corresponds to a unique element $i \in I$. Then S is indexed by I, and it is common to write $S = \{s_i\}_{i \in I}$.

Let $\mathcal{A} = \{A_i\}_{i \in I}$ be a collection of sets (for some index set I). Thus, for each $i \in I$, A_i itself is a set in the collection $\mathcal{A}$. Then we can define

the union and intersection of the sets in $\mathcal{A}$, as given below.

$$\bigcup_{i \in I} A_i = \{\, a \mid a \in A_i, \text{ for some } i \in I \,\}$$
$$\bigcap_{i \in I} A_i = \{\, a \mid a \in A_i, \text{ for all } i \in I \,\}$$

We refer to these operations as arbitrary unions and intersections. In the particular case where I is a set containing finitely many elements, these reduce to the finite union and intersection defined earlier.

1.4 Some More Set Operations

We say that (a, b) is an ordered pair[1] where the first element is a and the second element is b.

The Cartesian product of two sets A and B, denoted by $A \times B$, is the set of all ordered pairs of elements with the first element from A and the second from B. That is,

$$A \times B = \{\, (a, b) \mid a \in A,\ b \in B \,\}.$$

The Cartesian product of A with itself is often written as A^2. Similarly, the Cartesian product $A \times \cdots A$ with n terms (defined as an iterated Cartesian product of two sets at a time) is denoted by A^n.

The disjoint union (or coproduct) of two sets A and B, denoted by $A \sqcup B$, consists of all the ordered pairs of the form (x, i) where $i = 1$ when $x \in A$ and $i = 2$ when $x \in B$. That is,

$$A \sqcup B = \{\, (a, 1) \mid a \in A \,\} \cup \{\, (b, 2) \mid b \in B \,\}.$$

The disjoint union is also denoted by $A \cup B$, or $A \uplus B$.

[1] Formally, we may define the ordered pair in terms of sets as $(a, b) = \{a, \{a, b\}\}$. Note that this is only one possible "encoding" of the concept of an ordered pair, and in practice, we do not think of (a, b) as the (unordered) set $\{a, \{a, b\}\}$.

Note. The second element in each ordered pair (i.e. 1 or 2) only serves to distinguish the elements that are originally from A, from those that are originally from B. Thus, for example, if x is an element common to both A and B, then $A \sqcup B$ contains two "copies" of x, namely $(x, 1)$ and $(x, 2)$. When A and B are disjoint, $A \sqcup B$ is equivalent to $A \cup B$ (where the meaning of "equivalent" will be formalised later).

We can also define Cartesian products and disjoint unions of a collection of sets. Let $\mathcal{A} = \{A_i\}_{i \in I}$ be a collection of sets indexed by I. Then the Cartesian product of the collection $\mathcal{A}$ is

$$\prod_{i \in I} A_i = \{ (a_i)_{i \in I} \mid a_i \in A_i,\ i \in I \}$$

where $(a_i)_{i \in I}$ is a sequence of elements indexed by I, with $a_i \in A_i$ for each $i \in I$. The disjoint union of the collection $\mathcal{A}$ is

$$\bigsqcup_{i \in I} A_i = \bigcup_{i \in I} \{ (a_i, i) \mid a_i \in A_i,\ i \in I \} = \bigcup_{i \in I} A_i \times \{i\}.$$

Note. We will formally define sequences later, in Section 1.6.

The difference of two sets A and B is the set $A \setminus B$ consisting all elements of A that are not elements of B. That is,

$$A \setminus B = \{a \in A \mid a \notin B\}.$$

The difference of A and B is also denoted by $A - B$. Note that $A \setminus B = A \cap \overline{B} = A \setminus (A \cap B)$.

The symmetric difference of two sets A and B is the set $A \triangle B$ consisting of all the elements that are present in exactly one of A and B. That is,

$$A \triangle B = (A \setminus B) \cup (B \setminus A).$$

The symmetric difference of A and B is also denoted by $A \ominus B$.

Exercise 1.3. Let A and B be arbitrary sets.

1. Show that $(A \cup B) \setminus B = A \setminus B$.
2. Show that $A \triangle B = (A \cup B) \setminus (A \cap B)$.
3. Show that $(2^A, \triangle)$ is an Abelian group. That is:

   (i) $\triangle$ is associative.
   (ii) $\triangle$ is commutative.
   (iii) There exists $E \in 2^A$ such that for all $S \in 2^A$, $S \triangle E = S$.
   (iv) For all $S \in 2^A$, there exists $T \in 2^A$, such that $S \triangle T = E$.

   What is the order of any non-identity element of this group?

1.5 Relations

A relation R from a set A to a set B, denoted $R : A \to B$, is a subset of $A \times B$, i.e. $R \subseteq A \times B$. If $(a, b) \in R$, then we write aRb, and if $(a, b) \notin R$, then we write $a \not R b$. The set A is the domain and B the codomain of R. Note that $\varnothing$ is also a relation, called the empty or void relation, from A to B.

Note. A relation from a set A to a set B is a binary relation. More generally, if $A_1, \ldots, A_n$ are n sets, then a subset of $A_1 \times \cdots \times A_n$ is an n-ary relation.

A relation $R : A \to B$ is said to be

1. left-total if for each $a \in A$, aRb for some $b \in B$.
2. right-total if for each $b \in B$, aRb for some $a \in A$.
3. left-unique if for each $b \in B$, if $a, a' \in A$ are such that aRb and $a'Rb$, then $a = a'$.
4. right-unique if for each $a \in A$, if $b, b' \in B$ are such that aRb and aRb', then $b = b'$.

A relation from A to itself is said to be a relation on (or over) A (also called a homogeneous relation on A). Such relations can have a number of properties. In the following, let $\sim$ be a relation on a set A.

1. Reflexivity: For all $a \in A$, $a \sim a$.
2. Symmetry: For all $a, b \in A$, if $a \sim b$, then $b \sim a$.
3. Anti-symmetry: For all $a, b \in A$, if $a \sim b$ and $b \sim a$, then $a = b$.
4. Transitivity: For all $a, b, c \in A$, if $a \sim b$ and $b \sim c$, then $a \sim c$.
5. Irreflexivity: For all $a \in A$, $a \nsim a$.
6. Asymmetry: For all $a \in A$, if $a \sim b$, then $b \nsim a$.

Example 1.1. The following list gives examples of familiar relations satisfying one or more of the above properties:

1. The relations $=$ (on any set of elements where equality is defined), $\leq$ and $\geq$ (on any set of real numbers), $|$ (*divides*, see Exercise 1.7), $\subseteq$ and $\supseteq$ (on any collection of sets), $\cong$ and $\sim$ (on any set of triangles), and $\parallel$ (on any set of lines) are reflexive.
2. The relations $=$, $\neq$, $\cong$, $\sim$, $\parallel$, $\perp$ are symmetric.
3. The relations $|$ (on any set of non-negative integers), $\leq$, $\geq$, $\subseteq$, $\supseteq$ are anti-symmetric.
4. The relations $=$, $\leq$, $\geq$, $<$, $>$, $|$, $\subseteq$, $\supseteq$, $\subsetneq$, $\supseteq$, $\cong$, $\sim$, $\parallel$ are transitive.
5. The relations $\neq$ and $\perp$ are irreflexive.
6. The relations $<$, $>$, $\subsetneq$, $\supsetneq$ are asymmetric.

Exercise 1.4.

1. Let $\sim$ be the relation of the set $A = \{a, b\}$ defined by $a \sim a$, $a \sim b$. Is $\sim$ transitive?
2. Show that every asymmetric relation is irreflexive.
3. Prove or disprove: Any transitive, irreflexive relation is asymmetric.
4. Prove or disprove: Any symmetric, transitive relation is reflexive.

1.5.1 Equivalence Relations and Partitions

A reflexive, symmetric, and transitive relation is called an equivalence relation. For example, $=$, $\cong$, $\sim$, and $\parallel$ are equivalence relations. If $\sim$ is an

equivalence relation on a set A, and $a \in A$, then the equivalence class of a, denoted as $[a]$ or $\overline{a}$ is the set of all elements of A that a is related to by $\sim$. That is,

$$[a] = \{\, b \in A \mid a \sim b \,\}.$$

Example 1.2. Let $A = \{1, 2, 3, 4, 5\}$, and define a relation $\sim$ on A as follows: For any $a, b \in A$, $a \sim b$ if and only if $a - b$ is even. Then, for example, the equivalence class of 1 is $[1] = \{1, 3, 5\}$, and the equivalence class of 2 is $[2] = \{2, 4, 6\}$. Note that $[1] = [3] = [5]$ and $[2] = [4] = [6]$. Also observe that $[1]$ and $[2]$ are disjoint, and $[1] \cup [2] = A$.

An equivalence relation on a set is essentially the same as a partition of the set, as you will show in Exercises 1.5 and 1.6. A partition of a set S is a collection of non-empty and pairwise disjoint subsets of S whose union is equal to S. That is, a partition of S is a collection $\{P_i\}_{i \in I}$ of sets $P_i \subseteq S$, $i \in I$, such that

1. $P_i \neq \emptyset$, for each $i \in I$,
2. $P_i \cap P_j = \emptyset$, for all $i, j \in I$, $i \neq j$, and
3. $\bigcup_{i \in I} P_i = S$.

The subsets P_i, $i \in I$, are called the parts of the partition P.

Example 1.3. Let $S = \{1, 2, 3, 4, 5\}$. Then $P = \{\{1, 3, 5\}, \{2, 4, 6\}\}$ and $Q = \{\{1, 4\}, \{2\}, \{3, 5\}\}\}$ are two different partitions of S.

Exercise 1.5. Let $\sim$ be an equivalence relation on a set A. Show that the following hold:

1. For all $a \in A$, $a \in [a]$, and hence, each equivalence class is non-empty and $A = \bigcup_{a \in A} [a]$.
2. For all $a, b \in A$, if $a \neq b$, then $[a] \cap [b] = \emptyset$ (i.e. any two equivalence classes are either disjoint or identical).
3. The set of all equivalence classes of $\sim$ is a partition of A.

Exercise 1.6. Let $P = \{P_i\}_{i \in I}$ be a partition of a set A. Define a relation $\sim$ on A as follows: For any $a, b \in A$, $a \sim b$ if and only if a and b belong to the same part of the partition P (i.e. $a, b \in P_i$, $\exists i \in I$). Show that the following hold:

1. The relation $\sim$ is an equivalence relation on A.
2. The equivalence classes of $\sim$ are exactly the parts of the partition of P.

1.5.2 Partial Order Relations

A reflexive, anti-symmetric, and transitive relation is called a partial order relation (or simply a partial order). For example, $\leq$ and $\geq$ on any set of real numbers, $|$ on any set of non-negative integers, and $\subseteq$ and $\supseteq$ on any set of sets are partial order relations. A set A together with a partial order $\preccurlyeq$ on it forms a partially ordered set or poset $(A, \preccurlyeq)$.

Note. The term *partial* refers to the fact that two particular elements in a partially ordered set may be incomparable – i.e. neither may be related to the other in the partial order. For instance, consider the subsets of $S = \{x, y, z\}$, which are partially ordered by the subset relation $\subseteq$ – i.e. consider the poset $(2^S, \subseteq)$. Then $A = \{x, y\}$ and $B = \{y, z\}$ are incomparable, as neither is A a subset of B, nor is B a subset of A. On the other hand, A and $C = \{x\}$ are comparable (as $C \subseteq A$), and A and S itself are also comparable (as $A \subseteq S$). A poset in which every pair of elements is comparable (i.e. in which there are no incomparable pairs of elements) is called a total order.

Exercise 1.7. Let $|$ denote the divides relation on any set of integers. That is, for any two integers m and n, define $m \mid n$ if and only if $n = km$ for some integer k.

1. Prove that $(\mathbb{N}, |)$ is a poset. Is $(\mathbb{N}_0, |)$ also a poset?
2. Is $(\mathbb{Z}, |)$ a poset?

3. Let $n \in \mathbb{N}$, and let P be the set of all positive divisors of n. Then show that $(P, |)$ is a poset. What are all the natural numbers n such that $(P, |)$ is a total order?

1.6 Functions

A function is a left-total, right-unique binary relation. In other words, a function $f : A \to B$ is a relation from A to B such that each element of A is related to exactly one element of B under f. If $a \in A$ is related to $b \in B$ in f, then we say that f maps a to b, and write $b = f(a)$, or $a \mapsto b$. A function is also called a mapping. Recall that A is the domain of f and B the codomain. We may also write $\operatorname{dom} f$ and $\operatorname{cod} f$ to denote the domain and codomain of f, respectively. The image of f, denoted as $\operatorname{im} f$ or $f(A)$, is the set of all elements b of the codomain such that $b = f(a)$ for some $a \in A$. We can write this in the following two ways:

$$\operatorname{im} f = \{\, b \in B \mid b = f(a), \text{ for some } a \in A \,\}$$
$$\operatorname{im} f = \{\, f(a) \mid a \in A \,\}.$$

The preimage of any element of the codomain is the set of all elements of the domain that map to it. That is, for $b \in B$, the preimage of b, denoted $f^{-1}(b)$, is defined as

$$f^{-1}(b) = \{\, a \in A \mid f(a) = b \,\}.$$

Note that the image of the function is a subset of the codomain, while the preimage of an element is the subset of the domain.

Note. Intuitively, we think of a function as a rule that assigns, to each element of the domain, a unique element of the codomain. For instance, it is common in calculus to define a function using a formula – e.g. $f : \mathbb{R} \to \mathbb{R}$, $f(x) = x^2 - 1$. However, the formula or the expression itself is not the function. The function $g : \mathbb{Z} \to \mathbb{Q}$, defined by the formula $g(x) = x^2 - 1$, is different from the previously defined function f, although they are

both defined using the same formula. Moreover, it may not be possible to define a function using any closed-form formula.

A function is injective (or 1-1) if it is left-unique. That is, $f : A \to B$ is injective if, for any $a_1, a_2 \in A$, $f(a_1) = f(a_2)$ implies that $a_1 = a_2$. An injective function is also called an injection. A surjective (or onto) function is one in which every element of the codomain has a non-empty preimage. That is, $f : A \to B$ is surjective if, for each $b \in B$, $b = f(a)$ for some $a \in A$. Note that f is surjective if and only if $\operatorname{im} f = \operatorname{cod} f$. A surjective function is also called a surjection. A function that is both injective and surjective is bijective. A bijective function is also called a bijection or a one-to-one correspondence.

Example 1.4. Let $A = \{1, 2, 3, 4, 5\}$ and $B = \{a, b, c\}$.

1. Define a function $f : A \to B$, $f(1) = f(2) = a$, $f(3) = b$, $f(4) = f(5) = c$. Then f is a surjection (a has preimage $\{1, 2\}$, b has preimage $\{3\}$, and c has preimage $\{4, 5\}$). It is clearly not an injection, since, for example, $f(1) = f(2)$.
2. Define $g : B \to A$, $g(a) = 1$, $g(b) = 5$, $g(c) = 4$. Then g is an injection from B to A, as $g(a) \neq g(b), g(c)$ and $g(b) \neq g(c)$. The image of g is $\operatorname{im} g = \{1, 4, 5\} \neq A = \operatorname{cod} g$, and hence g is not a surjection.
3. Define a $h : B \to B$, $h(a) = b$, $h(b) = c$, $h(c) = a$. Note that h is a bijection from B to itself.

Example 1.5. Define $f : \mathbb{Z} \to \mathbb{R}$ as $f(n) = n$, for all $n \in \mathbb{Z}$. Then f is an injection, but not a surjection e.g. $1.5 \in \mathbb{R}$ has no preimage under f). Define $g : \mathbb{R} \to \mathbb{Z}$ as $g(x) = \lceil x \rceil$, the ceiling of x (i.e. the smallest integer greater than or equal to x – e.g. $\lceil 3.2 \rceil = 4$, and $\lceil -1.5 \rceil = 0$). Then g is a surjection (for any $n \in \mathbb{Z}$, $n \in \mathbb{R}$ as well, and $g(n) = \lceil n \rceil = n$), but not an injection e.g. $g(1.8) = g(2) = 2$). Similarly, define $h : \mathbb{R} \to \mathbb{Z}$ as $h(x) = \lfloor x \rfloor$, the floor of x (i.e. the greatest integer less than or equal to x). Then h is a surjection (but not an injection) from $\mathbb{R}$ to $\mathbb{Z}$, different from g.

Example 1.6. Define $f : \mathbb{R} \to (0, 1)$ (the set of all real numbers strictly between 0 and 1) as

$$f(x) = \frac{1}{1 + e^x}.$$

Firstly, note that this is indeed a well-defined function from $\mathbb{R}$ to $(0, 1)$, since $e^x \geq 0$ for all $x \in \mathbb{R}$. Now, if $f(x) = f(y)$, then observe that $e^x = e^y$, or $e^{x-y} = 1$, which implies that $x = y$. Hence, f is injective. Next, let y be any element of the codomain, $(0, 1)$. Then observe that $\frac{1}{y} > 1$, and hence $\frac{1}{y} - 1$ is a positive real number. Take $x = \log\left(\frac{1}{y} - 1\right)$, so that $\frac{1}{1+e^x} = y$, i.e. $f(x) = y$. Thus, for every $y \in (0, 1)$, there exists $x \in \mathbb{R}$ such that $f(x) = y$, which shows that f is surjective. Therefore, f is a bijection from $\mathbb{R}$ to $(0, 1)$.

Example 1.7. Let $A = \{a_1, a_2, \ldots, a_n\}$, and let $P = 2^A$, the power set of A. Let X be the set of all binary strings of length n – i.e. the set of all sequences of the form $x_1x_2 \cdots x_n$, where $x_i \in \{0, 1\}$, $i = 1, \ldots, n$. Define a function $\chi : P \to X$ as follows: For each $S \in P$, $\chi(S) = b_1b_2 \cdots b_n$ such that $b_i = 1$ if $a_i \in S$ and $b_i = 0$ if $a_i \notin S$. Then χ is a bijection, as shown below.

First, suppose that for $S, T \in P$, $\chi(S) = \chi(T) = b_1b_2 \cdots b_n$ (say). Then, for each $i = 1, \ldots, n$, $a_i \in S$ if and only if $b_i = 1$, which is equivalent to $a_i \in T$. Thus, $S = T$. This shows that χ is injective.

Next, let $b_1b_2 \cdots b_n \in X$. Define $S \subseteq A$ as

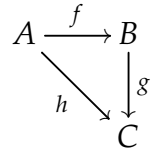
$$S = \{\, a_i \in A \mid b_i = 1 \,\}.$$

Then clearly, $\chi(S) = b_1b_2 \cdots b_n$. Hence, χ is surjective.

1.7 Function Composition

If $f : A \to B$ and $g : B \to C$ are two functions, then the composition of f and g is the function $h : A \to C$ defined by $h(a) = g(f(a))$, for all $a \in A$.

We denote this function h by $g \circ f$, which is read as "g circle f", "g after f", or "g composed with f". We can also write this definition in terms of a commutative diagram. Consider the diagram given below.



We say that such a diagram commutes if the result of following any two directed paths from the same starting point to the same end point is the same. In the above diagram, A, f, B, g, C and A, h, C are two directed paths from A to C (and these are the only two directed paths with the same starting points and the same end points). Thus, the diagram commutes if and only if applying g after f equals h.

Theorem 1.8. *Composition of functions is an associative operation.*

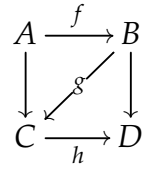
Proof. To prove that two functions are equal, we need to show that they both map each element of the domain to the same element of the codomain. Consider three functions $f : A \to B$, $g : B \to C$, and $h : C \to D$. Let $a \in A$. We will show that $h \circ (g \circ f)$ and $(h \circ g) \circ f$ map a to the same element of C.

$$\begin{aligned}(h \circ (g \circ f))(a) &= h((g \circ f)(a)) \\ &= h(g(f(a))) \\ &= (h \circ g)(f(a)) \\ &= ((h \circ g) \circ f)(a).\end{aligned}$$

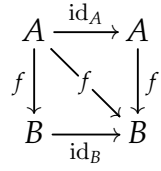
Hence, $h \circ (g \circ f) = (h \circ g) \circ f$. □

Exercise 1.8. Show that in the diagram below, the square commutes if

both the triangles commute.



The identity function on a set A, denoted as id_A, is the function from A to itself that maps every element of A to itself. That is, $\mathrm{id}_A : A \to A$ is defined as $\mathrm{id}_A(a) = a$, for all $a \in A$. Note that id_A is a bijection from A to itself. The identity function is so named because it is the identity element (or neutral element) for the operation of function composition. That is, if $f : A \to B$ is any function, then $f \circ \mathrm{id}_A = f$, and $\mathrm{id}_B \circ f = f$. Alternatively, we can say that the diagram given below commutes.



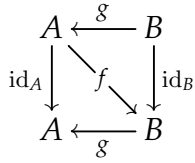
Exercise 1.9. Let A be a set.

1. Show that $\mathrm{id}_A \circ \mathrm{id}_A = \mathrm{id}_A$.
2. Show that if $i : A \to A$ is any function satisfying the same property as the identity function, i.e. for every set B and every function $f : A \to B$, $f \circ i = f$, and for every set C and every function $g : C \to A$, $i \circ g = g$, then $i = \mathrm{id}_A$.

1.8 Isomorphisms of Sets

If $f : A \to B$ is a function, then a function $g : B \to A$ is an inverse of f if $g \circ f = \mathrm{id}_A$ and $f \circ g = \mathrm{id}_B$. Equivalently, g is an inverse of f if the

diagram given below commutes.



Exercise 1.10. Prove that any function has at most one inverse.
Hint.

A function $f : A \to B$ is an isomorphism (of sets) if it has an inverse. Then the unique inverse (see Exercise 1.10) of f is denoted by f^{-1}. Note that f^{-1} is an isomorphism from B to A. For any two sets A and B, we say that A is isomorphic to B if there is an isomorphism from A to B. We write this as $A \cong B$.

Exercise 1.11.

1. Show that the identity function on a set is an isomorphism from the set to itself.
2. Prove that if f and g are isomorphisms, from A to B and from B to C, respectively, then $g \circ f$ is an isomorphism from A to C.
3. Show that $\cong$ is an equivalence relation on any collection of sets.

Now, we will prove that an isomorphism of sets is exactly the same thing as a bijection.

Theorem 1.9. *A function $f : A \to B$ is an isomorphism if and only if it is a bijection.*

Proof. First, suppose that f is a bijection. We need to show that f is an isomorphism from A to B – i.e. that it has an inverse. We construct the inverse as follows. Define $g : B \to A$ as $g(b) = a$, where $a \in A$ is such that $f(a) = b$, for each $b \in B$.

To see that g is well-defined, observe that as f is bijective, for every $b \in B$, there exists an $a \in A$ such that $f(a) = b$, and as f is injective, this element a is unique (i.e. if $f(a') = b = f(a)$, then $a' = a$).

To see that g is the inverse of f, first consider any $a \in A$. If $f(a) = b$, then $g(b) = a$, by definition of g. That is, $g(f(a)) = a$, which shows that $g \circ f = \mathrm{id}_A$. Next, consider any $b \in B$. Then, by definition of g, $g(b) = a \in A$ such that $f(a) = b$. That is, $f(g(b)) = b$, which shows that $f \circ g = \mathrm{id}_B$. Thus, g is the inverse of f, and therefore f is an isomorphism.

Conversely, suppose that $f : A \to B$ is an isomorphism. We will show that f is a bijection. For any $b \in B$, if we take $a = f^{-1}(b)$, then $f(a) = b$, which shows that f is surjective. To see that f is injective, suppose that $a_1, a_2 \in A$ and $f(a_1) = f(a_2)$. Then $a_1 = f^{-1}(f(a_1)) = f^{-1}(f(a_2)) = a_2$. Hence, f is a bijection. □

1.9 Cardinality of Sets

The cardinality of a set is the number of elements in it. As sets can have infinitely many elements, we will define cardinality more rigorously using the concept of set isomorphisms.

Definition 1.10. Two sets A and B have the same cardinality if $A \cong B$. If $A \cong \{1, \ldots, n\}$, then A is finite of cardinality n. If A is not finite, then it is infinite. If $A \cong \mathbb{N}$, then A is countably infinite. If A is either finite or countably infinite, it is countable. Otherwise, A is uncountable or uncountably infinite.

Exercise 1.12. Prove that $\mathbb{Z}$ is countably infinite.

Solution. Define a function $f : \mathbb{Z} \to \mathbb{N}$ by

$$f(n) = \begin{cases} 2(n+1), & n \geq 0 \\ 2|n| - 1, & n < 0. \end{cases}$$

Then f is a bijection, which can be shown as follows. Suppose $f(m) = f(n)$, for some $m, n \in \mathbb{Z}$. Then, either $m, n \geq 0$ and $2(n+1) = 2(m+1)$, which implies that $m = n$, or else $m, n < 0$, and $2|m| - 1 = 2|n| - 1$, which implies $|m| = |n$, and hence $m = n$ (since both are negative). Therefore, f is injective. Now, for any $n \in \mathbb{N}$, n is either even or odd. If it is even, then it is of the form $2k + 2$ for some integer $k \geq 0$, i.e. $n = f(k)$. If it is odd, then it is of the form $2k - 1$, for some $k \geq 1$, i.e. $n = 2|-k| - 1 = f(-k)$. Therefore, f is surjective.

Since there is a bijection from $\mathbb{Z}$ to $\mathbb{N}$, we have $\mathbb{Z} \cong \mathbb{N}$. Therefore, $\mathbb{Z}$ is countably infinite.

Exercise 1.13. Let A and B be two sets. Prove the following.

1. If A and B are finite, then $A \cup B$ is finite.
2. If A is finite and B is countably infinite, then $A \cup B$ is countably infinite.
3. If A and B are countably infinite, then $A \cup B$ is countably infinite.
4. If A and B are countable, $A \cup B$ is countable.

Hence show that if $A_1, \ldots, A_n$ are n countable sets, and $A = A_1 \cup \cdots \cup A_n$, then

1. A is countable, and
2. A is countably infinite if and only if at least one A_i is countably infinite.

Remark. A bijection from $\mathbb{N}$ to a set S defines an enumeration of all the distinct elements of S. That is, suppose that $f : \mathbb{N} \to S$ is a bijection. Now, $f(1)$ is an element of S, say s_1. Then, $f(2)$ is another element of S, say s_2, and $s_1 \neq s_2$ (since f is injective). Similarly, $f(3), f(4), \ldots$, are distinct elements $s_3, s_4, \ldots$ of S. Since f is surjective, each element of S is $s_n = f(n)$, for some $n \in \mathbb{N}$. Thus, in order to show that a set is countably infinite, it is sufficient to give an enumeration of all its distinct elements as a sequence. But note that in this case, one must prove that

every element of the set is indeed an n^{th} term of the sequence for some positive integer n.

Lemma 1.11. *If A and B are countably infinite, then $A \times B$ is countably infinite.*

Proof. Since A and B are countably infinite, there exist bijections $a : \mathbb{N} \to A$ and $b : \mathbb{N} \to B$. For each $n \in \mathbb{N}$, denote $a(n)$ by a_n, and $b(n)$ by b_n (see Section 1.9). Then, $A = \{a_1, a_2, \ldots\}$ and $B = \{b_1, b_2, \ldots\}$. Now,

$$A \times B = \{(a_1, b_1), (a_1, b_2), (a_1, b_3), \ldots, (a_2, b_1), (a_2, b_2), (a_2, b_3), \ldots\}.$$

Consider the elements of $A \times B$ as being arranged on an infinite grid as shown below.

$$\begin{array}{cccc} (a_1, b_1) & (a_1, b_2) & (a_1, b_3) & \cdots \\ (a_2, b_1) & (a_2, b_2) & (a_2, b_3) & \cdots \\ (a_3, b_1) & (a_3, b_2) & (a_3, b_3) & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{array}$$

Now, we can define an enumeration of the elements of $A \times B$ by reading the elements along the antidiagonals, starting from (a_1, b_1). Explicitly, the enumeration is

$$(a_1, b_1),\ (a_1, b_2),\ (a_2, b_1),\ (a_1, b_3),\ (a_2, b_2),\ (a_3, b_1),\ \ldots.$$

Since each antidiagonal is finite, every element (a_m, b_n) appears in the above enumeration after some finite number of terms. Indeed, it is not hard to verify that this enumeration defines a bijection $f : A \times B \to \mathbb{N}$, given by $f(a_m, b_n) = \binom{m+n-1}{2} + m$. □

From Lemma 1.11, it is clear that $\mathbb{N}\times\mathbb{N}$, $\mathbb{Z}\times\mathbb{Z}$, and $\mathbb{Z}\times\mathbb{N}$ are countably infinite. Since each rational number can be written in the form $\frac{a}{b}$ where $a \in \mathbb{Z}$ and $b \in \mathbb{N}$, the set of rational numbers, it follows that $\mathbb{Q}$, is countable. Clearly, it is not finite. So, in fact, it is countably infinite.

Theorem 1.12. *The set of rational numbers, $\mathbb{Q}$, is countably infinite.* □

1.9.1 Uncountable Sets

All the examples of infinite sets that we have seen so far were countable. Are there any uncountable sets at all? We shall show that the set of real numbers is uncountably infinite. Recall from Example 1.6 that $\mathbb{R} \cong (0,1)$. Thus, in order to show that $\mathbb{R}$ is uncountable, it is enough to show that $(0,1)$ is uncountable.

Theorem 1.13. *The open interval $(0,1)$ consisting of all real numbers strictly between 0 and 1 is uncountably infinite.*

Proof. Let $f : \mathbb{N} \to (0,1))$ be any injection from $\mathbb{N}$ to $(0,1)$. Since each real number in the interval $(0,1)$ has a unique infinite decimal expansion, for each $n \in \mathbb{N}$, $f(n)$ can be written in the form $f(n) = 0.a_{n1}a_{n2}a_{n3}\cdots$, where $0 \leq a_{ni} \leq 9$, $i = 1, 2, \ldots$.

Let $x = 0.x_1x_2x_3\ldots$, where

$$x_n = \begin{cases} 2, & a_{nn} \neq 2 \\ 3, & a_{nn} = 2. \end{cases}$$

Then clearly, $x \in (0,1)$. But $x \neq f(n)$ for any $n \in \mathbb{N}$, as can be seen by comparing the n^{th} digits of x and $f(n)$. If the n^{th} digit of $f(n)$ is 2, then the n^{th} digit of x is $x_n = 3$, whereas if the n^{th} digit of $f(n)$ is not 2, then that of x is $x_n = 2$. Thus, $x \notin \operatorname{im} f$, and hence f is not surjective.

The above argument shows that no injection from $\mathbb{N}$ to $(0,1)$ can be surjective. That is, there is no bijection (surjective injection) from $\mathbb{N}$ to $(0,1)$, and therefore the latter is not countably infinite. Since $(0,1)$ is infinite, but not countably infinite, it is uncountably infinite. □

Exercise 1.14. Show that the cardinality of any set is different from that of its power set.

Solution.

2 Finite Automata and Regular Expressions

2.1 Deterministic Finite Automata

A deterministic finite automaton[2] (DFA) is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$ where

- Q is a finite set whose elements are states.
- Σ is a finite input alphabet, whose elements are input symbols.
- $\delta : Q \times \Sigma \to Q$ is the transition function.
- $q_0 \in Q$ is the start state (or initial state).
- $F \subseteq Q$ is the set of accept states (or final states).

The transition function may be specified using a transition table, which has rows indexed by the states and columns indexed by the input symbols, with the entry in the row of q and column of s being the value of $\delta(q, s)$). See Example 2.1.

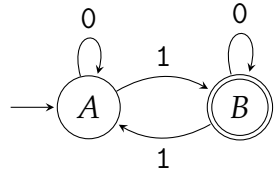
A DFA has a graphical representation as a directed graph in which each vertex is a state and each edge is a transition. That is, if $\delta(q_1, s) = q_2$ (where q_1 and q_2 are states and s is an input symbol), then the graph has an edge labelled s from the vertex q_1 to the vertex q_2. It is usual to depict the vertices by circles, with the vertex label (name of the state) written inside the circle, if necessary. The start state is indicated by an arrow without a source entering it, and accept states are indicated by using double-circles. See Example 2.1.

Example 2.1. Consider a DFA $M = (Q, \Sigma, \delta, A, F)$, where $Q = \{A, B, C\}$, $\Sigma = \{0, 1\}$, $F = \{B\}$, and δ is as defined by the transition table given below.

δ	0	1
A	A	B
B	B	A

The graphical representation of M is shown below.

[2] The plural of *automaton* is *automata*



An input to a DFA $M = (Q, \Sigma, \delta, q_0, F)$ is a string over Σ – i.e. a finite sequence of symbols of Σ. If $w = s_1 s_2 \cdots s_k$ is an input string, and $q_1, q_2, \ldots, q_k \in Q$ such that

$$\delta(q_0, s_1) = q_1,\ \delta(q_1, s_2) = q_2,\ \ldots,\ \delta(q_{k-1}, s_k) = q_k,$$

then M is in the state q_k after reading the input w. If $q_k \in F$ (i.e. q_k is an accept state), then M accepts w, otherwise it rejects w. For instance, the DFA M given in Example 2.1 is in the state A after reading the input `0110`, and is in the state B after reading `0100`. Since $A \notin \{F\}$ and $B \in \{F\}$, it rejects `0110` and accepts `0100`.

Given the graphical representation of a DFA M, to check which state M is in after reading an input $w = s_1 \cdots s_k$, simply begin at the start state of M and follow the arrows labelled $s_1, \ldots, s_k$. For instance, in Example 2.1, the input `0110` corresponds to the sequence of states A, A, B, A, A, and the input `0100` corresponds to the sequence of states A, A, B, B, B (note that in both cases, the first A is the start state, and only the next four states are the results of transitions).

A language over an alphabet Σ is a set of strings over Σ. In particular, the language Σ^* is the set of all strings over Σ. Thus, a language over Σ is any subset of Σ^*. If M is a DFA with input alphabet Σ, then M accepts the language L if L is the set of all strings accepted by M. For example, the DFA M in Example 2.1 accepts the language of all binary strings containing an odd number of `1`s.

Note. Although we say simply that M accepts L, this means not only that every string in L is accepted by M, but also that every string *not* in L is *rejected* by M.

Exercise 2.1. Construct a DFA that accepts all binary strings with an even number of `0`s.

Solution.

Exercise 2.2. Construct a DFA that accepts all strings over the alphabet {`a`, `b`} that begin in `aab`.

Solution.

Observe that two of the states of the DFA given in the solution of Exercise 2.2 are states that, once entered, cannot be left – i.e. every transition out of such each of these states is into that state itself. Such a state is a trap state. One of these two trap states is an accept states – which indicates that any input that leads to this state is accepted regardless of what the rest of the input contains – while the other trap state is a non-accept state – which indicates that any input leading to this state is *permanently* rejected and that no further "correction" can be made to get the input accepted.

2.2 Nondeterministic Finite Automata

A nondeterministic finite automaton (NFA) is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$ where

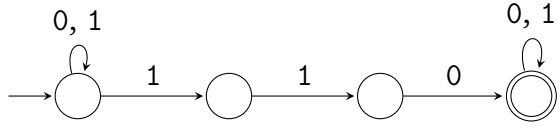
- Q is a finite set of states.
- Σ is a finite input alphabet.
- $\delta\colon Q \times (\Sigma \cup \{\varepsilon\}) \to 2^Q$ is the transition function.
- $q_0 \in Q$ is the start state.
- $F \subseteq Q$ is the set of accept states.

Note that the transition function has codomain 2^Q, the power set of Q. That is, each transition of a nondeterministic finite automaton is from one state into possibly several states or even to no state. The interpretation of this is that at each step, for a given input, the machine may have many options of the state to transition to, and an input string is accepted if there is at least one path from the start state to an accept state with arrows labelled by the input. More formally, an input $s_1 \cdots s_n$ is accepted by an NFA $M = (Q, \Sigma, \delta, q_0, F)$, if and only if there exist states $q_1, \ldots, q_k \in Q$, with $q_k \in F$, such that

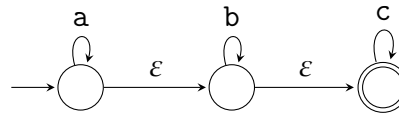
$$q_1 \in \delta(q_0, s_1),\ q_2 \in \delta(q_1, s_2),\ \ldots,\ q_k \in \delta(q_{k-1}, s_n).$$

A transition labelled by ε, i.e. a transition of the form $\delta(q, \varepsilon)$, is an epsilon transition, and does not consume an input symbol. An NFA also has a graphical representation similar to that of a DFA.

Example 2.2. The NFA shown below accepts all binary strings that contain `110` as a (contiguous) substring.



Example 2.3. The NFA shown below accepts all strings over the alphabet $\{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$ in which the symbols appear in alphabetical order.



2.2.1 Conversion of NFA to DFA

Trivially, a deterministic finite automaton can be considered as a nondeterministic finite automaton. That is, NFAs are at least as powerful as DFAs, in the sense that any language accepted by a DFA is also accepted by an NFA. Now we will see that conversely, NFAs are not strictly more powerful than DFAs. Any NFA can be converted to an equivalent DFA[3], by the subset construction. Each transition of an NFA is from one state to a set of states, whereas any transition of a DFA is from one state to a single state. The main idea of the subset construction is to consider each set of states of an NFA as a single state of a DFA. The conversion procedure is described below.

Let $M = (Q, \Sigma, \delta, q_0, F)$ be an NFA (without epsilon transitions). Define a DFA $D_M = (Q', \Sigma, \delta', q_0', F')$ as follows:

1. Every set of states of the NFA M becomes a single state of the DFA D_M. That is, $Q' = 2^Q$.
2. The start state of D_M is the singleton set containing the start state of M. That is, $q_0' = \{q_0\}$.
3. The transition function δ' of D_M is given by

$$\delta'\big(\{q_1, \ldots, q_k\}, s\big) = \bigcup_{i=1}^{k} \delta(q_i, s).$$

[3]Two finite automata are equivalent if the language accepted by them is the same

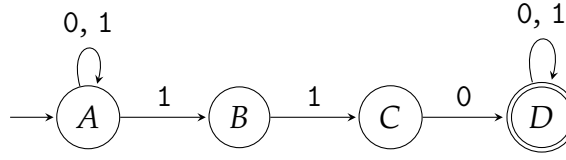
That is, given a state q' of D_M, which is a set of states of the NFA M, and an input symbol $s \in \Sigma$, for each state $q \in q'$ $\delta(q, s)$ is some set of states of M. Combine all these sets $\delta(q, s)$ for $q \in q'$ (via union) to get $\delta'(q', s)$.

4. Each state of D_M that contains at least one accept state of M becomes an accept state. That is,

$$F' = \{ q' \in Q' \mid q' \cap F \neq \varnothing \}.$$

If M has epsilon transitions, then $\delta'(\{q_1, \ldots, q_k\}, s)$ will contain, in addition to the states described above, the states of M that can be reached from each state in $\delta(q_i, s)$ via any sequence epsilon transitions. Similarly, q_0' will be the set of all states of M that can be reached from q_0 via (a sequence of) epsilon transitions.

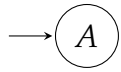
Example 2.4. Consider the NFA given in Example 2.2.



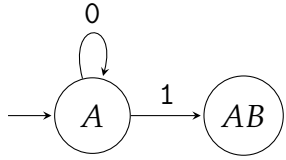
Here, the initial state is $q_0 = A$, and the set of accept states is $F = \{D\}$. Now, the equivalent DFA D_M will $2^4 = 16$ states, namely $\varnothing, \{A\}, \{B\}, \{C\}, \{D\}, \{A, B\}, \{A, C\}, \ldots, \{A, B, C, D\}$. For the sake of simplicity, we will denote these as $\varnothing, A, B, C, D, AB, AC, \ldots, ABCD$.

Note that many of these states may be unreachable – i.e. there will be no path from the start state to them. We will not include them in the graphical representation of D_M, even though, formally, they are part of D_M. To find all the reachable (i.e. not unreachable) states, we will begin with the start state, and iteratively add the other states while computing the transitions from the current states corresponding to each input symbol.

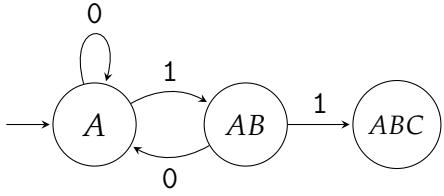
1. Initially, we have only the start state, A.



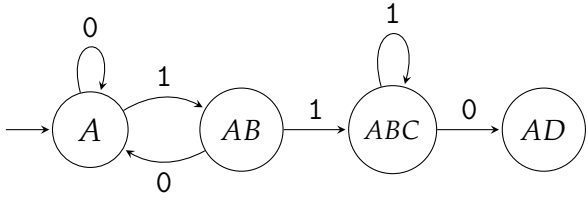
2. Next, $\delta'(A, 0) = \delta(A, 0) = \{A\}$ (written as just A) and $\delta'(A, 1) = \delta(A, 1) = \{A, B\}$ (written as just AB).



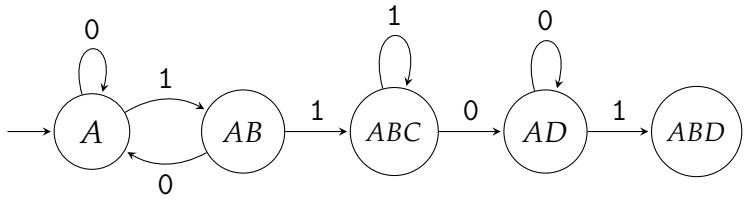
3. Now, $\delta'(AB, 0) = \delta(A, 0) \cup \delta(B, 0) = \{A\} \cup \varnothing = \{A\}$ (i.e. A) and $\delta'(AB, 1) = ABC$.



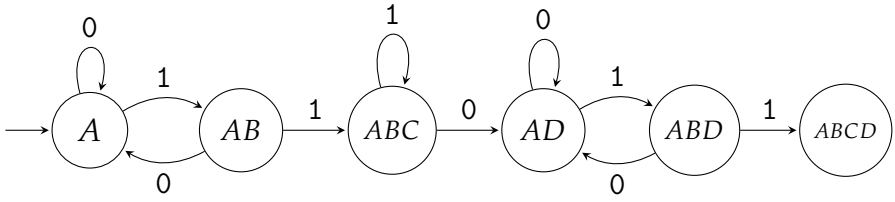
4. Similarly, $\delta'(ABC, 0) = AD$ and $\delta'(ABC, 1) = ABC$.



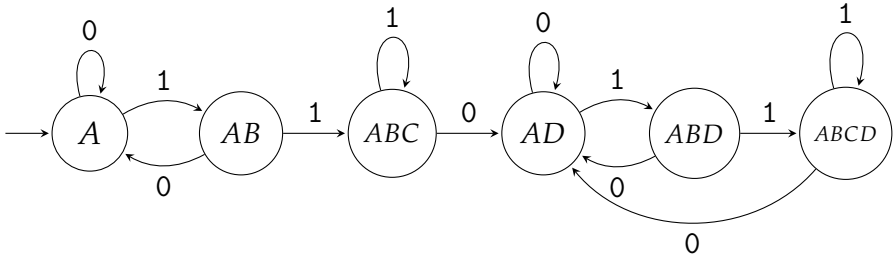
5. Similarly, $\delta'(AD, 0) = AD$ and $\delta'(AD, 1) = ABD$.



6. Similarly, $\delta'(ABD, 0) = AD$ and $\delta'(ABD, 1) = ABCD$.

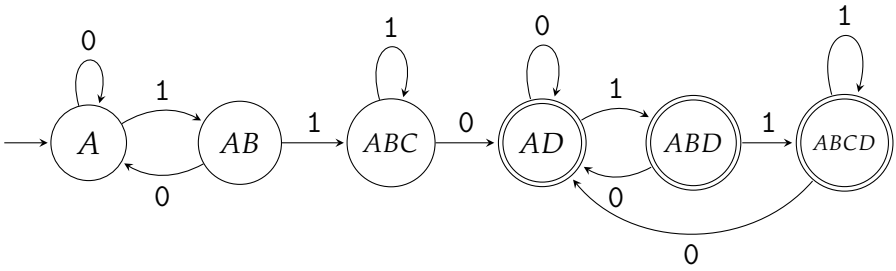


7. Finally, $\delta'(ABCD, 0) = AD$ and $\delta'(ABCD, 1) = ABCD$.

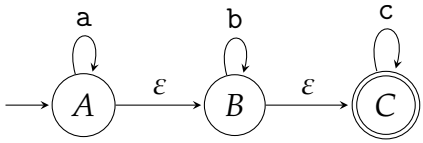


   As no new state is added in this step, all the other states (not shown here) are unreachable.

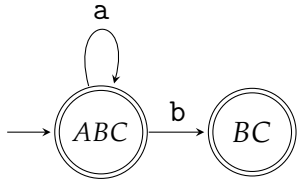
Now, the states AD, ABD, and $ABCD$ all contain D, which is the only accept state of the original NFA. Hence, these states are accept state of the DFA.



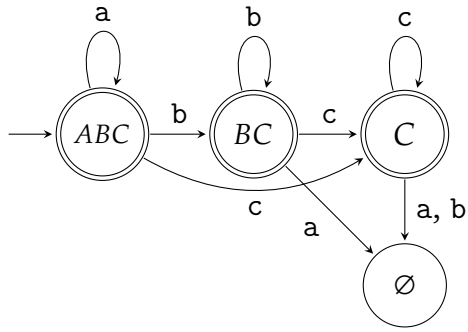
Example 2.5. Consider the NFA given in Example 2.2.



Here, the start state is A, but B can be reached from A via an epsilon transition, and C can be reached from B via an epsilon transition. Hence, the start state of the converted DFA will be ABC, which will also be an accept state, since it contains C, an accept state of the original NFA. Next, $\delta(A, \mathtt{a}) = A$, $\delta(B, \mathtt{a}) = \delta(C, \mathtt{a}) = \varnothing$. But B and C can be reached from A by epsilon transitions. Therefore, $\delta'(ABC, \mathtt{a}) = ABC$. Similarly, $\delta'(ABC, \mathtt{b}) = BC$, which will also be an accept state. Similarly, $\delta'(ABC, \mathtt{c}) = C$, also an accept state.



But now, $\delta(B, \mathtt{a}) = \delta(C, \mathtt{a}) = \varnothing$. Hence, $\delta'(BC, \mathtt{a}) = \varnothing$. Note that $\varnothing$ will be state from which there are no transitions into any other states – i.e. it is a trap state. Hence, $\delta'(\varnothing, \mathtt{a}) = \delta'(\varnothing, \mathtt{b}) = \varnothing$. Proceeding thus, we find that $\delta'(BC, \mathtt{b}) = BC$, $\delta'(BC, \mathtt{c}) = C$, $\delta'(C, \mathtt{a}) = \delta'(C, \mathtt{b}) = \varnothing$, and $\delta'(C, \mathtt{c}) = C$.



2.3 Regular Expressions

A regular expression (RE) over an alphabet Σ is an expression consisting of one or more of the symbols in Σ or ε or $\emptyset$, one or both of the operators $+$ and *, and parentheses (and), whose syntax is defined recursively as follows:

1. For each $s \in \Sigma$, s is a regular expression, and ε and $\emptyset$ are regular expressions.
2. If r is a regular expression, then so is $(r)^*$.
3. If r_1 and r_2 are regular expressions, then so are $r_1 + r_2$ and $(r_1)(r_2)$.

Note. If r is a single symbol (i.e. $r \in \Sigma$), then $(r)^*$ can also be written as r^*; $(r)(r')$ can also be written as $r(r')$; and $(r')(r)$ can also be written as $(r')r$.

Example 2.6. The following are some regular expressions over $\Sigma = \{0, 1\}$:

$$0 + 1, \quad 0(0 + 1), \quad (0 + 1)^*110(0 + 1)^*, \quad 1^*0^* + 0^*1^*,$$
$$0^*10^*(0^*10^*1)^*0^*, \quad 0(0 + 1)^*0 + 1(0 + 1)^*1$$

Each RE represents or generates a particular language. To rigorously define the language generated by an RE, we will first define some operations on languages.

Let L_1 and L_2 be two languages over the same alphabet Σ. Recall that this means L_1 and L_2 are sets of strings of symbols in Σ, i.e. $L_1, L_2 \subseteq \Sigma^*$.

1. The union of L_1 and L_2 is $L_1 \cup L_2$, defined as their union as sets.

$$L_1 \cup L_2 = \{ w \in \Sigma^* \mid w \in L_1 \text{ or } w \in L_2 \}$$

2. The concatenation of L_1 and L_2 is $L_1 L_2$, the language of all strings obtained by concatenating a string of L_1 with a string of L_2.

$$L_1 L_2 = \{ w_1 w_2 \mid w_1 \in L_1,\ w_2 \in L_2 \}$$

3. The Kleene star[4] of L_1 is L_1^*, the language of all strings obtained by concatenating any finite number of strings of L_1 together.

$$\begin{aligned} L_1^* &= \{ w_1 w_2 \ldots w_n \mid w_i \in L_i,\ i = 1, \ldots, n,\ n \in \mathbb{N}_0 \} \\ &= \{\varepsilon\} \cup L_1 \cup L_1 L_1 \cup L_1 L_1 L_1 \cup \cdots \end{aligned}$$

The language generated by an RE can now be recursively defined as follows:

1. The language generated by the RE s, where $s \in \Sigma$ (a single symbol), is $\{s\}$ and the language generated by the RE ε is ε.
2. The language generated by the RE $\varnothing$ is the empty language $\varnothing = \{\}$. Note that this language does *not* contain the empty string!
3. If r_1 and r_2 are two REs over the same alphabet Σ, that generate languages L_1 and L_2, respectively, then
   (i) the RE $r_1 + r_2$ generates the language $L_1 + L_2$
   (ii) the RE $(r_1)(r_2)$ generates the language $L_1 L_2$
   (iii) the RE $(r_1)^*$ generates the language L_1^*.

Example 2.7. Let $\Sigma = \{0, 1\}$.

[4] Named after the mathematician Stephen Cole Kleene, pronounced `klay'nee`.

1. The RE $0 + 1$ generates the language $\{0, 1\}$.
2. The RE $0(0 + 1)$ generates the language $\{00, 01\}$.
3. The RE $(0+1)^*110(0+1)^*$ generates the language of all binary strings containing 110 as a substring.
4. The RE $0^*1^* + 1^*0^*$ generates the language of all binary strings that are either a string of (zero or more) 0s followed by a string of 1s, or a string of 1s followed by a string of 0s.

Exercise 2.3. Let $\Sigma = \{0, 1\}$.

1. What is the language generated by $0(0 + 1)^*0 + 1(0 + 1)^*1$?
2. What is the language generated by $0^*10^*(0^*10^*1)^*0^*$?

Solution.

2.3.1 Conversion of NFA to RE

Any nondeterministic finite automaton can be converted to an equivalent[5] regular expression. The conversion procedure involves first reducing the given NFA, by eliminating one state at a time, to intermediate finite automata of a new kind, until just one start state and one accept state remain. A generalised (nondeterministic) finite automaton (GFA or GNFA) is a finite automaton in which the arrows are labelled by REs, and a transition along this arrow is made if the next few characters of the input match the RE. That is, if the arrow from state q_1 to state q_2 is labelled by the regular expression r, and the unread part of the input when q_1 is reached is $s_1 s_2 \cdots s_n$, then a transition can be made to q_2 is

[5] An RE is equivalent to an NFA is one that generates the language accepted by the NFA.

$s_1 s_2 \cdots s_i$ is a string generated by r for some $i \leq k$. Trivially, every NFA is a GFA, since any single symbol or ε is an RE, and a transition of an NFA consists of reading one symbol and moving along one arrow, or reading no symbol (i.e. an empty string) and making an epsilon transition.

The procedure for converting an NFA to RE is as follows. Let $M = (Q, \Sigma, \delta, q_0, F)$ be an NFA. Furthermore, assume that q_0 has no arrows entering it and F contains a single state q_f with no arrows leaving it – in both cases, even self-loops are not allowed[6]. Now, iteratively reduce M by eliminating one state other than q_0 and q_f at a time as given below:

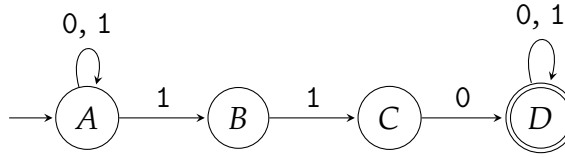
1. Let q be any state different from q_0 and q_f, and let r be the label on the self-loop on q.
2. For each state $q_1 \neq q$ with a transition into q labelled r_1, and each state $q_2 \neq q$ into which there is a transition from q labelled r_2, let $r_{12} = r_1 r^* r_2$ (suitably parenthesised), where r is the regular expression on the self-loop on q (if q has no self-loop, then this is just $r_1 r_2$).
3. If there is no arrow from q_1 to q_2, add an arrow labelled r_{12}. If an arrow already exists from q_1 to q_2, replace its label, say r', by $r' + r_{12}$.
4. Delete q.

When q_0 and q_f are the only states remaining, the label on the arrow from q_0 to q_f is the RE equivalent to the NFA M.

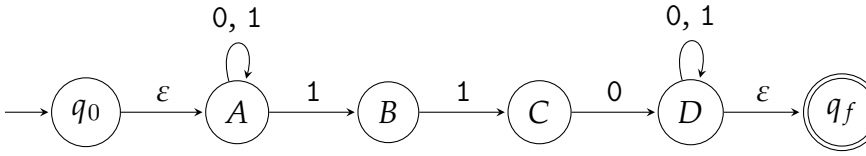
Note. The order in which the states are eliminated does not matter, although the REs obtained may differ (but be equivalent) according to the order followed.

[6] It is easy to see that any NFA can be converted to an equivalent NFA satisfying this condition by using epsilon transitions. If q_0 has an arrow entering it, then make a new start state q_0' with an epsilon transition into q_0, which will no longer be the start state. Similarly, if q_f has an arrow leaving it, make it a non-accept state and add a new accept state q_f' with an epsilon transition into it from q_f. If M has multiple accept states, make all of them non-accept states, and add a new accept state q_f with an epsilon transition into it from each of the old accept states.

Example 2.8. Consider the NFA given in Example 2.2.

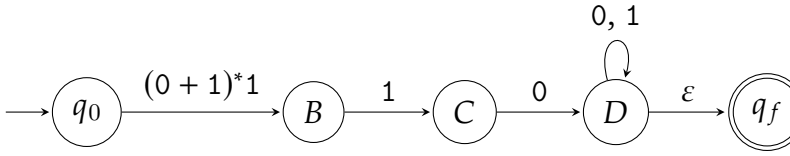


The start state A has two arrows entering it and the accept state D has two arrows leaving it (in each case, self-loops labelled 0 and 1). Therefore, we first add a new start state q_0 with an epsilon transition into A (no longer a start state), and a new accept state q_f with an epsilon transition into it from D, which will be made a non-accept state.

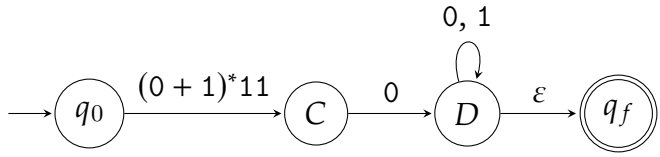


Now we eliminate the non-initial and non-accept states one by one.

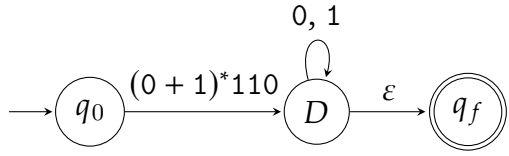
1. If we choose to eliminate A first, we first note that there is one arrow, labelled ε, coming into A from q_0, and one arrow, labelled q, going from A into B. Also, there are self-loops on A labelled 0 and 1. Therefore, we add an arrow from q_0 to B labelled $\varepsilon(0+1)^*1 = (0+1)^*1$, and then delete A.



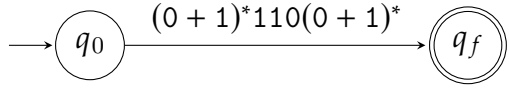
2. Next, if we wish to eliminate B, we replace the arrow from q_0 to B labelled $(0+1)^*1$ and the arrow labelled 1 from B to C by the arrow $(0+1)^*11$ from q_0 to C.



3. Similarly, we may eliminate C next.



4. Finally, we eliminate D.



Thus, an RE equivalent to the given NFA is $(0+1)^*110(0+1)^*$. We easily verify that indeed, this is an equivalent RE. The original NFA accepts all binary strings containing the substring 110. Every such string can be written as some binary string (i.e. a string generated by $(0+1)^*$), followed by 110, (generated by 110) followed by any binary string (generated by $(0+1)^*$)).

3 Graph Theory

A graph is an ordered pair $G = (V, E)$ where V is a non-empty set called the vertex set, whose elements are the vertices of G, and E is a set of unordered pairs of distinct vertices of G, whose elements are the edges of G. The order of G is its number of vertices and the size of G is its number of edges.

Two vertices u and v of G are adjacent, denoted by $u \sim v$, if $uv \in E$ (note that uv here denotes an unordered pair of vertices, and $uv = vu$). If there is no edge between u and v, then they are non-adjacent, denoted by $u \nsim v$. The vertex v and the edge uv are said to be incident with each other. The degree of a vertex v is the number of edges incident with it, or equivalently, the number of vertices adjacent with it, and is denoted by $\deg(v)$.

Lemma 3.1 (Handshaking Lemma)**.** *The sum of the degrees of all vertices of a graph is twice its size.*

Proof. Let G be a graph, and consider the sum of its degrees, $\sum_{v \in V(G)} \deg v$. As each edge is incident with exactly two vertices, each edge is counted once by two terms in this summation. Thus, the sum is equal to twice the number of edges. □

Corollary 3.2. *The number of vertices of odd degree in a graph is always even.*

Proof. Let G be a graph. We know that

$$\sum_{v \in V(G)} \deg v = 2|E(G)|. \tag{1}$$

Considering the vertices of odd and even degrees, we can write (1) as

$$\sum_{v \in V(G) \,:\, \deg v \text{ odd}} \deg v + \sum_{v \in V(G) \,:\, \deg v \text{ even}} \deg v = 2|E(G)|. \tag{2}$$

Both the RHS and the second term on the LHS are even integers, and therefore so is the first term on the LHS. But a sum of odd integers is even only if the number of terms is even. Thus, G has an even number of vertices of odd degree. □

A regular graph is a graph in which all vertices have the same degree. This common value of the degree is the regularity of the graph. A regular graph of regularity k is a k-regular graph. A cubic graph is a 3-regular graph.

Exercise 3.1. Prove that any cubic graph has even order. For which other regular graphs will this statement hold?

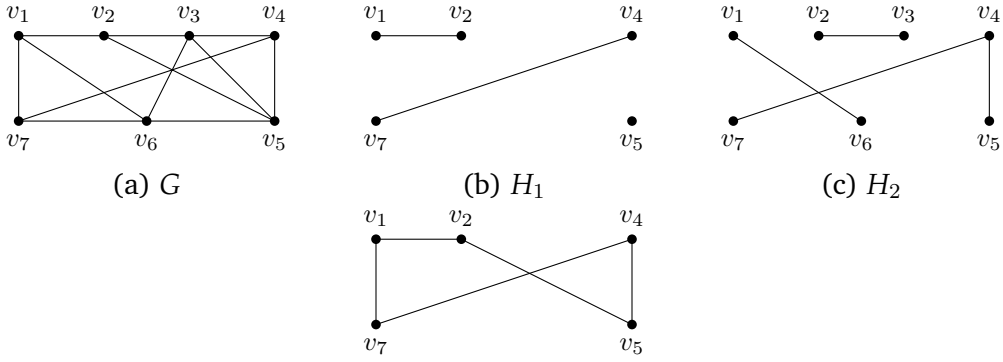
Theorem 3.3. *In any graph on two or more vertices, at least two vertices have the same degree.*

Proof. Let G be a graph of order $n \geq 2$. The minimum possible degree of a vertex of G is 0, and the maximum possible degree is $n-1$ (as there are only n vertices in G). Thus, in order for the n vertices of G to have different degrees, the n different degrees must be exactly $0, 1, \ldots, n-1$. But if some vertex of G has degree $n-1$, then it is adjacent to all other vertices, and therefore G cannot have a vertex of degree 0. Thus, at least two vertices of G have the same degree. □

3.1 Subgraphs and Complements

A subgraph of a graph G is a graph H whose vertex and edge sets are, respectively, subsets of the vertex and edge sets of G – i.e. $V(H) \subseteq V(G)$ and $E(H) \subseteq E(G)$. A spanning subgraph of G is a subgraph containing all the vertices of G. A subgraph H of a graph G is an induced subgraph if for all vertices $u, v \in V(H)$, $uv \in E(H)$ whenever $uv \in E(G)$ – i.e. every edge of G that joins two vertices of H is present in H.

Example 3.4. Fig. 1 shows a graph G and three subgraphs H_1, H_2, and H_3 of G. The subgraph H_1 is neither a spanning subgraph e.g. it does not contain v_3), nor an induced subgraph e.g. it contains vertices v_2 and v_5 of G but not the edge v_2v_5). H_2 is a spanning subgraph and H_3 is an induced subgraph of G.



(a) G (b) H_1 (c) H_2

(d) H_3

Figure 1: A graph with three of its subgraphs

A graph is complete if all its vertices are adjacent to one another. The complete graph of order n is denoted by K_n. As there is an edge between every pair of distinct vertices of K_n, the size of K_n is $\binom{n}{2}$. The totally disconnected graph of order n, denoted by $\overline{K_n}$ is the graph of order n that has no edges.

The complement of a graph G is the graph $\overline{G}$ with the same vertices as G, in which any two vertices u and v are adjacent if and only if they are non-adjacent in G. Thus, if G is an (n, m)-graph, then $\overline{G}$ is an $\left(n, \binom{n}{2} - m\right)$-graph. Observe that $\overline{K_n}$ is (as the notation suggests) the complement of K_n.

Exercise 3.2. If G is a graph of order n, and $\deg_G v = k$, what is $\deg_{\overline{G}} v$?

Exercise 3.3. Show that if G and $\overline{G}$ are k-regular graphs of order n then k is an even number and $n = 4t + 1$ for some integer t.

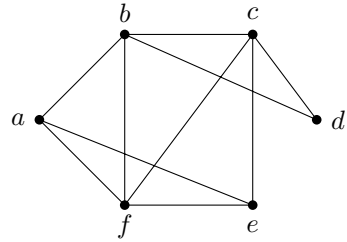
3.2 Walks, Paths, and Cycles

A walk in a simple graph G is a sequence of vertices $v_1, v_2, \ldots, v_k$ such that $v_i \sim v_{i+1}$ for $i = 1, 2, \ldots, k-1$. This is a walk from v_1 to v_k or between v_1 and v_k, having length $k-1$. We may also say that this is a walk of length $k-1$ starting at v_1 and ending at v_k.

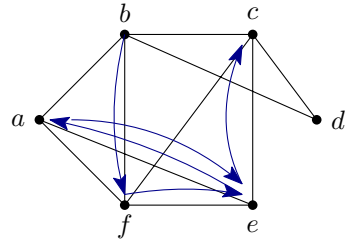
A walk in which no vertex is repeated is a path. Note that the length of a path is the number of edges in it.

A closed walk in G is a walk starting and ending at the same vertex. A closed walk in which no vertex is repeated (except for the start and end vertices being the same) is a cycle. The length of a cycle is also equal to the number of vertices in it. A cycle of the form $v_1, \ldots, v_{k-1}, v_k$ is usually referred to as "the cycle $v_1, \ldots, v_{k-1}$".

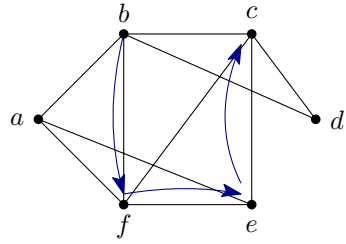
Example 3.5. Consider the graph G shown below.



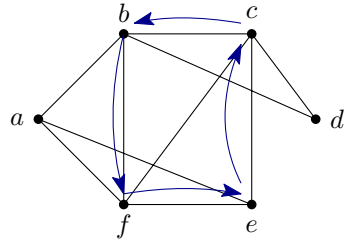
A walk of length 5 from b to c in G is b, f, e, a, e, c.



This walk is not a path, since the vertex e is repeated. An example of a path from b to c in the same graph G is b, f, e, c. The length of this path is 3.



A cycle of length 4 in G is b, f, e, c, b.



3.3 Distances

The distance between two vertices u and v of a graph G is the length of a shortest path between u and v – such a shortest path between u and v is a geodesic from u to v. Thus, the distance between u and v in G, denoted by $d_G(u, v)$ is the length of a geodesic from u to v in G. When the graph G is clear from the context, we will write $d_G(u, v)$ as $d(u, v)$.

The eccentricity of a vertex v, denoted by $\mathrm{ecc}(v)$, is the maximum of the distances from v to all the other vertices. That is,

$$\mathrm{ecc}(v) = \max_{u \in V(G)} d(u, v).$$

The diameter of G is the maximum of the eccentricities of vertices of G, and the radius of G is the minimum of the eccentricities of vertices of G. They are denoted by $\operatorname{diam}(G)$ and $\operatorname{rad}(G)$ respectively. Thus,

$$\operatorname{diam}(G) = \max_{v \in V(G)} \operatorname{ecc}(v) = \max_{u,v \in V(G)} d(u,v)$$
$$\operatorname{rad}(G) = \min_{v \in V(G)} \operatorname{ecc}(v) = \min_{v \in V(G)} \max_{u \in V(G)} d(u,v).$$

The set of all vertices of G having minimum eccentricity, i.e. the set of all $v \in V(G)$ such that $\operatorname{ecc} v = \operatorname{rad} G$, is the centre of G.

3.4 Connectedness

A graph is connected if there is a path between every two of its vertices. Otherwise, it is disconnected. A (connected) component of a graph G is a maximal connected subgraph of G. Thus, a graph G is connected if and only if it has exactly one component.

Theorem 3.6. *For any graph G, either G or $\overline{G}$ is connected.*

Proof. Consider a graph G, and suppose that G is disconnected. We shall show that $\overline{G}$ is connected, by showing that there is a path between every two vertices of $\overline{G}$.

Let u and v be any two vertices of $\overline{G}$. If they are not adjacent in G, then they are adjacent in $\overline{G}$, and hence there is a path uv from u to v in $\overline{G}$. If u and v are adjacent in G, then they belong to the same component of G. As G is disconnected, it has at least one more component containing at least one vertex, say w, which is necessarily non-adjacent to both u and v. Hence, in $\overline{G}$, w is adjacent to both u and v. Thus, there is a path uwv from u to v in $\overline{G}$. Therefore, $\overline{G}$ is connected. □

Theorem 3.7. *For any connected graph G, if $\operatorname{diam} G \geq 3$, then $\operatorname{diam} \overline{G} \leq 3$.*

Proof. Consider a graph G of diameter at least 3. Then there exist vertices u and v in G such that $d_G(u,v) = 3$. This implies that u and v are non-adjacent in G, and hence adjacent in $\overline{G}$.

Now, consider any two vertices x and y other than u and v. In G, x can be adjacent to at most one of u and v, for otherwise, $d_G(u,v) = 2$. Hence, x is adjacent to at least one of u and v in $\overline{G}$. Similarly, y is adjacent to at least one of u and v in $\overline{G}$. Thus, there is a path of length at most 3 from x to y in $\overline{G}$ (namely xuy, xvy, $xuvy$, or $xvuy$), which implies that $d_{\overline{G}}(x,y) \leq 3$. Therefore, $\operatorname{diam} \overline{G} \leq 3$. □

3.5 Graph Isomorphism

An isomorphism from a graph G to a graph H is a bijective[7] function $f : V(G) \to V(H)$ such for all vertices u and v of G, $u \sim_G v$ if and only if $f(u) \sim_H f(v)$. In other words, a graph isomorphism is a bijection from the vertex set of the first graph to that of the second, that preserves both edges and non-edges. If there exists an isomorphism from G to H, then G and H are isomorphic. Then we write $G \cong H$.

Isomorphic graphs have exactly the same structure – i.e. all properties of the graph that do not depend on the labelling or drawing will be shared by isomorphic graphs. For example, isomorphic graphs have the same order, size, degree sequence, diameter, and radius. Indeed, if f is an isomorphism from G to H, and v is a vertex of G, then the neighbours of $f(v)$ in H are the images of the neighbours of v in G, and $\deg_G v = \deg_H f(v)$. Similarly, if u and v are two vertices of G, then $d_G(u,v) = d_H(f(u), f(v))$.

Theorem 3.8. *Graph isomorphism is an equivalence relation on the class of all graphs.*

Proof. To show that graph isomorphism is an equivalence relation, we need to show that $\cong$ is a reflexive, symmetric, transitive relation between

[7] injective and surjective; i.e. one-to-one and onto

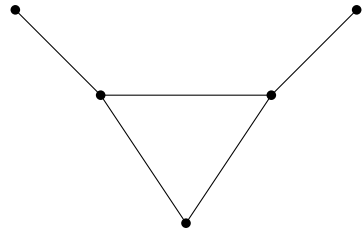
graphs. First, observe that the identity map on the vertex set of a graph is an isomorphism from the graph to itself – for, if G is a graph and id denotes the identity map on its vertex set then for any two vertices u and v of G, $u \sim v$ if and only if $\mathrm{id}(u) \sim \mathrm{id}(v)$, as $\mathrm{id}(u) = u$ and $\mathrm{id}(v) = v$. Hence, $\cong$ is reflexive.

Next, suppose that $G \cong H$. Then there exists an isomorphism f from G to H. Since $f : V(G) \to V(H)$ is bijective, it has an inverse $f^{-1} : V(H) \to V(G)$. We claim that f^{-1} is an isomorphism from H to G. We know that f^{-1} is bijective. To see that it preserves edges and non-edges, observe that if x and y are two vertices of H, then $x = f(f^{-1}(x))$ and $y = f(f^{-1}(y))$, which implies that $x \sim_H y$ if and only if $f^{-1}(x) \sim_G f^{-1}(y)$ (since f is an isomorphism from G to H). Therefore, f^{-1} is an isomorphism from H to G. This shows that $H \cong G$. Thus, $\cong$ is symmetric.

Finally, suppose that $G \cong H$ and $H \cong K$. Then there exist isomorphisms f from G to H and g from H to K. We claim that $g \circ f$ is an isomorphism from G to K. Indeed, $g \circ f$ is a function from $V(G)$ to $V(K)$, and being a composition of bijections, is itself a bijection. Now, suppose u and v are two vertices of G. Then $u \sim_G v$ if and only if $f(u) \sim_H f(v)$ (since f is an isomorphism from G to H) if and only if $g(f(u)) \sim_K g(f(v))$ (since g is an isomorphism from H to K). Thus, $g \circ f$ is an isomorphism from G to K. Therefore, $\cong$ is transitive. □

3.6 Self-Complementary Graphs

A graph is self-complementary if it is isomorphic to its complement – i.e. G is self-complementary if $G \cong \overline{G}$. For example, K_1, P_4 and C_5 are self-complementary graphs of orders 1, 4, and 5 respectively. There is one more self-complementary graph of order 5, namely the bull graph, which can be constructed by adding one new vertex to P_4 and making it adjacent to the two non-pendant vertices of the path. This graph is shown below.



Theorem 3.9. *The order of any self-complementary graph is* $4k$ *or* $4k+1$*, for some non-negative integer* k*.*

Proof. Let G be a self-complementary graph of order n and size m. Then the size of $\overline{G}$ is $\binom{n}{2} - m$. Since $G \cong \overline{G}$, $m = \binom{n}{2} - m$, which implies that $m = \frac{1}{2}\binom{n}{2} = \frac{n(n-1)}{4}$. As m is an integer, this implies that 4 divides $n(n-1)$, which in turn implies that either one of n and $n-1$ is divisible by 4, or both n and $n-1$ are even. Since the latter is not possible, it follows that $n = 4k$ or $n - 1 = 4k$, i.e. $n = 4k$ or $4k+1$ for some integer k. □

From Theorem 3.6, we know that a graph and its complement cannot both be disconnected. Thus, if G is a disconnected graph, then $\overline{G}$ must be connected, and therefore it cannot be isomorphic to G. This implies that a self-complementary graph is necessarily connected.

Corollary 3.10. *Every self-complementary graph is connected.* □

Similarly, from Theorem 3.7, we obtain the following corollary about the diameter of self-complementary graphs.

Corollary 3.11. *Every non-trivial self-complementary graph has diameter* 2 *or* 3*.*

Proof. Consider a non-trivial, self-complementary graph G, and suppose that it has diameter strictly greater than 3. Then, by Theorem 3.7, $\operatorname{diam} \overline{G} \leq 3$, which implies that $\operatorname{diam} \overline{G} \neq \operatorname{diam} G$, a contradiction, since $G \cong \overline{G}$. Hence, $\operatorname{diam} G \leq 3$. On the other hand, a non-trivial graph cannot have diameter 0, hence $\operatorname{diam} G \geq 1$. But if $\operatorname{diam} G = 1$, then G is a

non-trivial complete graph, whose complement is a totally disconnected graph, which is again impossible. Hence, $\operatorname{diam} G = 2$ or 3. □

Exercise 3.4. If G is a regular self-complementary graph of order n, show that $n = 4k + 1$, for some integer k, and $\operatorname{diam} G = 2$.

Exercise 3.5. Let G be a self-complementary graph, and let H be the graph obtained by taking the disjoint union of H and P_4, and making every vertex of H adjacent to the two non-pendant vertices of this P_4. Then show that H is also self-complementary. What is the graph H obtained in this manner if $G = K_1$?

Exercise 3.6. Construct a self-complementary graph of order 9.

Exercise 3.7. Show that for each positive integer n of the form $4k$ or $4k+1$, where k is an integer, there exists at least one self-complementary graph of order n.

3.7 Bipartite Graphs

A graph $G = (V, E)$ is bipartite if its vertex set V can be partitioned into two subsets V_1 and V_2 such that no two vertices in V_i are adjacent, for $i = 1, 2$. That is, there exist two non-empty, disjoint subsets $V_1, V_2 \subseteq V$ such that $V_1 \cup V_2 = V$, and every edge of G (if any) joins a vertex in V_1 with a vertex in V_2. Then we say that (V_1, V_2) is a bipartition of G.

The following theorem characterises bipartite graphs in terms of its cycles. A cycle is even if its length is even and odd if its length is odd.

Theorem 3.12. *A graph is bipartite if and only if it contains no odd cycles.*

Proof. First, suppose that G is a bipartite graph with bipartition (V_1, V_2), and let $v_1, v_2, \ldots, v_k$ be a cycle of length k in G. Without loss of generality, suppose that $v_1 \in V_1$. Then, since $v_2 \sim v_1$, $v_2 \in V_2$, which in turn implies that $v_3 \in V_1$, as $v_2 \sim v_3$. Proceeding similarly, we see $v_i \in V_1$ if

i is odd and $v_i \in V_2$ if i is even. But $v_k \sim v_1$ and $v_1 \in V_1$ implies that $v_k \in V_2$. Therefore, k (the length of the cycle) is even.

Conversely, suppose G is a graph that has no odd cycles. Without loss of generality, assume that G is connected – otherwise, apply the argument to each component. Let v be any vertex of G. Define subsets V_1 and V_2 of $V(G)$ as follows:

$$V_1 = \{\, u \in V \mid d(u, v) \text{ is even} \,\}$$
$$V_2 = \{\, u \in V \mid d(u, v) \text{ is odd} \,\}.$$

We claim that (V_1, V_2) is a bipartition of G.

Suppose not, and let x and y be adjacent vertices of G, both belonging to V_i for $i = 1$ or 2. Let P and Q be shortest paths from v to x and y respectively. Let w be the last vertex common to both P and Q when traversing P from v to x. Then the portion of P from w to x, the edge (x, y), and the portion of Q from y to w forms a cycle in G, having length, say l, given by

$$\begin{aligned} l &= d(w, x) + d(w, y) + 1 \\ &= [d(v, x) - d(v, w)] + [d(v, y) - d(v, w)] + 1 \\ &= d(v, x) + d(v, y) - 2d(v, w) + 1. \end{aligned}$$

If $x, y \in V_1$, then both $d(v, x)$ and $d(v, y)$ are even, and if $x, y \in V_2$, then both these numbers are odd. In either case, $d(v, x) + d(v, y)$ is even, and hence l is odd, which is a contradiction. Hence, (V_1, V_2) is a bipartition of G as claimed. □

3.8 Trees

A tree is a connected, acyclic graph. There are several well known characterisations or alternative definitions of trees. We take the given definition as the basic one and prove its equivalence to some others.

Theorem 3.13. *A graph T is a tree if and only if there is a unique path joining every two vertices of T.*

Proof. First, suppose that T is a tree, and let u and v be vertices of T. Since T is connected, there is a path, say P_1, joining u and v. Now we must show that this path is unique. Assume to the contrary that there exists another path P_2 from u to v. When traversing P_1 from u to v, let w be the first vertex that is present on P_1 but not P_2. Let x be the vertex on P_1 preceding w, and note that x is on P_2 as well. Let y be the next vertex common to both P_1 and P_2 when traversing P_1 from x to v. Then the portion of P_1 from x to y together with the portion of P_2 from y to x forms a cycle in the tree T, which is a contradiction. Thus, P_1 is the unique path joining u and v.

Conversely, suppose that T is a graph in which there is a unique path joining any two vertices. Clearly, T is connected. To show that T is acyclic, suppose that $v_1, v_2, \ldots, v_n$ is a cycle in T. Then we get two different paths joining v_1 and v_n, namely the path $v_1, v_2, \ldots, v_n$ and the path v_1, v_n (since $v_1 \sim v_n$ in the cycle). This contradicts our assumption. Thus, T must be acyclic and hence, is a tree. □

The next two results show that the size of a tree is always one less than its order, and that conversely, this property together with either connectedness or acyclicity implies that the graph is a tree.

Theorem 3.14. *A (p, q)-graph T is a tree if and only if it is connected and $p = q + 1$.*

Proof. Let T be a tree with p vertices and q edges. Then T is connected. We prove that $p = q + 1$ by induction. This is clearly true when $p = 1$. Assume it to be true for all trees of order less than p. Now in T, we know that every two vertices are joined by a unique path. Thus, if e is any edge of T, then the graph $T - \{e\}$ obtained by deleting e has exactly two components, say T_1 and T_2. Each one is a tree, since it is connected and acyclic. Let T_i have p_i vertices and q_i edges, $i = 1, 2$.

Then by the hypothesis, $p_i = q_i + 1$ (since $p_i < p$). But $p = p_1 + p_2$ and $q = q_1 + q_2 + 1$ (since the size of $T - \{e\}$ is one less than that of T). Thus, $p = q_1 + q_2 + 2 = q + 1$.

For the converse, suppose that T is a connected (p, q)-graph with $p = q + 1$. We must show that is acyclic. Suppose to the contrary that T has a cycle C with k vertices. Then C has k edges as well. Since T is connected, there is a path from every vertex not on C to some vertex of C. The shortest path from each vertex v not on C to a vertex on C has a unique edge incident with v, which is not part of C. Since there are $p - k$ vertices in T not on C, there are $p - k$ such edges. Thus $q \geq (p - k) + k = p$, which contradicts our assumption that $p = q + 1$. Thus, T must be acyclic. □

In the following theorem, the proof of the direct part is identical to that of Theorem 3.14, except for the assertion being about acyclicity rather than connectedness. The proof of the converse part is entirely different.

Theorem 3.15. *A (p, q)-graph T is a tree if and only if it is acyclic and $p = q + 1$.*

Proof. Let T be a tree with p vertices and q edges. Then T is acyclic. We prove that $p = q + 1$ by induction. This is clearly true when $p = 1$. Assume it to be true for all trees of order less than p. Now in T, we know that every two vertices are joined by a unique path. Thus, if e is any edge of T, then the graph $T - \{e\}$ obtained by deleting e has exactly two components, say T_1 and T_2. Each one is a tree, since it is connected and acyclic. Let T_i have p_i vertices and q_i edges, $i = 1, 2$. Then by the hypothesis, $p_i = q_i + 1$ (since $p_i < p$). But $p = p_1 + p_2$ and $q = q_1 + q_2 + 1$ (since the size of $T - \{e\}$ is one less than that of T). Thus, $p = q_1 + q_2 + 2 = q + 1$.

Conversely, suppose that T is an acyclic (p, q)-graph with $p = q + 1$. To show that T is connected, we need to prove that it is connected – i.e. it

has only one component. Let T have k components $T_1, \ldots, T_k$. Each one is acyclic, and being connected, is a tree. Thus from the first part of the theorem, we know that if p_i and q_i are respectively the order and size of the component T_i, $p_i = q_i + 1$. Now $p = p_1 + \cdots p_k = (q_1 + 1) + \cdots + (q_k + 1) = q + k$. But we know that $p = q + 1$. Therefore, $k = 1$. Thus, T is a tree. □

Exercise 3.8. A pendant vertex of a graph is a vertex of degree 1. Prove that every non-trivial tree contains at least two pendant vertices.
Hint.

Exercise 3.9. The centre of a graph G is the set of all vertices of G with minimum eccentricity – i.e. the set of all vertices v of G with $\operatorname{ecc} v = \operatorname{rad} v$. Show that every tree has a centre consisting of either exactly one vertex or exactly two adjacent vertices.
Hint.

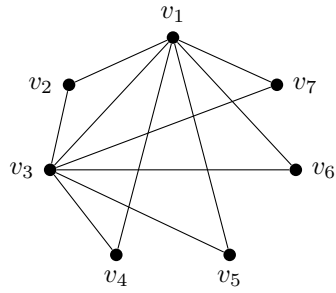
Solution.

3.9 Adjacency Matrices

The adjacency matrix of a graph G of order n, with vertex set $V = \{v_1, \ldots, v_n\}$, is the $n \times n$ matrix $A = \mathcal{A}(G)$ whose (i, j)-entry is

$$a_{ij} = \begin{cases} 1, & v_i \sim v_j \\ 0, & v_i \nsim v_j. \end{cases}$$

Example 3.16. The adjacency matrix of the graph



is

$$A = \begin{bmatrix} 0 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

Observe that, as the graphs we discuss are simple graphs and therefore have no self-loops on vertices, no vertex is adjacent to itself – i.e. $a_{ii} = 0$ for all $i = 1, \ldots, n$. Also, since the graphs are undirected, $v_i \sim v_j$ if and only if $v_j \sim v_i$ – i.e. $a_{ij} = a_{ji}$. Thus, we have the following observation.

Observation 3.17. *The adjacency matrix of a (simple, undirected) graph is a symmetric, zero-diagonal,* $(0,1)$*-matrix.*

In the i^{th} row of the adjacency matrix, for each j, the j^{th} entry is 1 if v_j is adjacent to v_i, and 0 otherwise. That is, the number of 1s in the i^{th} row is the number of vertices adjacent to v_i, or in other words, the degree of v_i. Thus, the row sums of A are the vertex degrees. Observe that $A\mathbb{1}$ is the vector of row sums, where $\mathbb{1}$ is the vector (of suitable size) with all entries equal to 1. For instance, with the matrix A given in Example 3.16,

$$A = \begin{bmatrix} 0 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 6 \\ 2 \\ 6 \\ 2 \\ 2 \\ 2 \\ 2 \end{bmatrix}.$$

From the Handshaking Lemma and the preceding observation, it follows that the sum of all entries of A is twice the number of edges of the graph.

Exercise 3.10. Show that for any graph G of order n, $\mathcal{A}(\overline{G}) = J - I - \mathcal{A}(G)$ where I is the identity matrix and J is the square all-ones matrix (i.e. a matrix with all entries 1), both of order n.

Exercise 3.11. Suppose G is a disconnected graph with components $G_1, G_2, \ldots, G_k$, and furthermore, assume that the vertices are ordered in such a way that the first n_1 vertices are in G_1, the next n_2 vertices are in G_2, and so on, the last n_k vertices are in G_k. Show that the adjacency matrix of G is given by the partitioned matrix

$$\mathcal{A}(G) = \begin{bmatrix} \mathcal{A}(G_1) & 0_{n_2} & \cdots & 0_{n_k} \\ 0_{n_1} & \mathcal{A}(G_2) & \cdots & 0_{n_k} \\ \vdots & \vdots & \ddots & \vdots \\ 0_{n_1} & 0_{n_k} & \cdots & \mathcal{A}(G_k) \end{bmatrix}$$

where 0_{n_i} denotes the $n_i \times n_i$ zero-matrix.

Exercise 3.12. Show that the (i, j)-entry of A^2 is the number of walks of length 2 from v_i to v_j. Hence show that $\mathrm{tr}(A^2) = 2|E(G)|$.
Hint.

The following result (which generalises the statement in Exercise 3.12) shows that the adjacency matrix can be used to obtain certain information about walks in the graph.

Theorem 3.18. *Let A be the adjacency matrix of a graph G with vertex set $\{v_1, \ldots, v_n\}$. Then the (i, j)-entry of A^m, for any positive integer m, is the number of walks of length m from v_i to v_j.*

Proof. We prove the result by induction on m. For $m = 1$, the (i, j)-entry of $A^1 = A$ is a_{ij}, which is 1 if and only if v_i is adjacent to v_j, i.e. if and only if there is a walk of length 1 (namely, an edge) from v_i to v_j. Thus, the result holds for $m = 1$.

Now suppose, for the sake of induction, that the result holds for some $m \geq 1$, and consider A^{m+1}. The (i, j)-entry of A^{m+1} is

$$(A^{m+1})_{ij} = \sum_{k=1}^{n} (A^m)_{ik} a_{kj}.$$

First, note that $a_{kj} = 1$ if and only if $v_k \sim v_j$. Therefore, the above sum is equal to the sum of all $(A^m)_{ik}$ where $v_k \sim v_j$. Now, by the induction hypothesis, $(A^m)_{ik}$ is the number of walks of length m from v_i to v_k. If v_j is adjacent to v_j, then each walk of length m from v_i to v_k, together with the edge from v_j to v_j, forms a walk of length $m + 1$ from v_i to v_j. Thus, for each k such that $v_k \sim v_j$, $(A^m)_{ik} a_{kj} = (A^m)_{ik}$ is the number of

walks of length $m+1$ from v_i to v_j that pass through k. Summing over all k, this gives the total number of walks of length $m+1$ from v_i to v_j. Hence the result follows by induction. □

Alternatively, note that

$$(A^m)_{ij} = \sum_{k_1,k_2,\ldots,k_{m-1}} a_{ik_1}a_{k_1k_2}\cdots a_{k_{m-1}j}$$

and each term in this summation will be 1 if and only if $a_{ik_1} = a_{k_1k_2} = \cdots = a_{k_{m-1}j} = 1$, i.e. $v_i \sim v_{k_1} \sim v_{k_2} \sim \cdots \sim v_{k_{m-1}} \sim v_j$, which corresponds to a walk of length m from v_i to v_j. Hence $(A^m)_{ij}$ is the number of walks of length m from v_i to v_j.

Theorem 3.19. *Let A be the adjacency matrix of a graph G with vertex set $\{v_1, \ldots, v_n\}$. Then*

(i) $\operatorname{tr}(A^2) = 2|E(G)|$
(ii) $\operatorname{tr}(A^3) = 6c_3(G)$

where $c_3(G)$ denotes the number of triangles in G.

Proof. We know that $(A^m)_{ij}$ is the number of walks of length m from v_i to v_j.

(i) Hence, the i^{th} diagonal entry of A^2, viz. $(A^2)_{ii}$, is the number of walks of length 2 from v_i to itself. Any walk of length 2 from v_i to itself is of the form $v_iv_jv_i$, where v_j is a vertex adjacent to v_i. Thus, the number of such walks is equal to the number of vertices adjacent to v_i, i.e. $\deg v_i$. Hence, $\operatorname{tr}(A^2) = \sum_{i=1}^{n} \deg v_i = 2|E(G)|$, by Handshaking Lemma.

(ii) Similarly, $(A^3)_{ii}$ is the number of walks of length 3 from v_i to itself. Any such walk is of the form $v_iv_jv_kv_i$, which implies that v_i, v_j, and v_k form a triangle in G. Moreover, each such triangle corresponds to two distinct walks from v_i to itself, when traversed in the

two opposite directions. Thus, $(A^3)_{ii}$ is twice the number of triangles having v_i as one of its vertices. Therefore, $\operatorname{tr}(A^3)$ is $6c_3(G)$, since each triangle contains three vertices, each of which counts the triangle twice in the sum $\operatorname{tr}(A^3)$. □

Exercise 3.13. Prove that the i^{th} diagonal entry of A^4 is

$$(A^4)_{ii} = 2c_4(v_i) + \sum_{j:\, v_j \sim v_i} \deg v_j + 2\binom{\deg v_i}{2}$$

where $c_4(v_i)$ denotes the number of cycles of length 4 containing v_i, and

$$\begin{aligned} c_4(G) &= \frac{1}{8}\operatorname{tr} A^4 - \frac{1}{4}\sum_{i=1}^{n}(\deg v_i)^2 + \frac{|E(G)|}{4} \\ &= \frac{1}{8}\operatorname{tr} A^4 - \frac{1}{4}\sum_{uv\in E(G)}(\deg u + \deg v - 1) \\ &= \frac{1}{8}\operatorname{tr} A^4 - \frac{|E(G)|}{4} - \frac{1}{2}\sum_{i=1}^{n}\binom{\deg v_i}{2}. \end{aligned}$$

3.10 Dijkstra's Algorithm

Let G be a directed graph on n vertices $v_1, \ldots, v_n$, in which each (directed) edge has a positive weight attached to it (say, representing the cost of traversing that edge). Let W be the weighted adjacency matrix of G, defined as follows. W is an $n \times n$ matrix with rows and columns indexed by the vertices of G, and having (i, j)-entry

$$w_{ij} = \begin{cases} 0, & i = j \\ \text{Weight of the edge } (v_i, v_j), & v_i \sim v_j \\ \infty, & v_i \nsim v_j. \end{cases}$$

Dijkstra's algorithm is a procedure to determine the shortest paths (and hence distances) from a given source vertex s of G to all the other vertices.

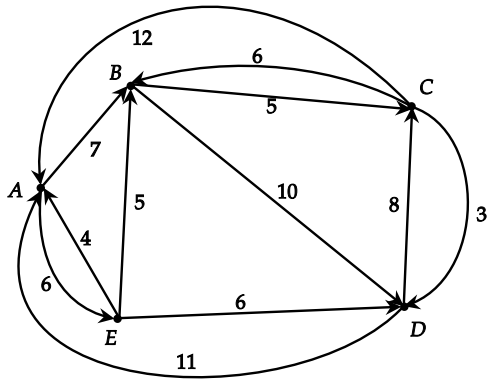
Algorithm 1 Dijkstra's algorithm

```
1:  K ← s, U ← V(G) − {s}
2:  bestDTo(u) ← w_su, ∀u ∈ U
3:  tree(u) ← s, ∀u ∈ U
4:  while |U| > 1 do
5:      t ← u ∈ U such that bestDTo(u) is minimum
6:      U ← U − {t}                          ▷ Remove t from U and
7:      K ← K ∩ {t}                          ▷ add it to K
8:      for u ∈ U do
9:          du_t ← bestDTo(t) + w_tu         ▷ Distance to u through t
10:         if du_t < bestDTo(u) then
11:             bestDTo(u) ← dsu_t
12:             tree(u) ← t
13:         end if
14:     end for
15: end while
```

The input to the algorithm is the set of vertices V and the weighted adjacency matrix W. Initially, the algorithm takes the weights of the edges from s to the other vertices as the tentative best distances to those vertices. It also maintains a set K of vertices to which the shortest paths from s have been found (so that no further improvement is possible), and a set U of vertices to which shorter paths may yet be found, passing through some vertex in K. In each step, the vertex $t \in U$ with minimum tentative best distance is selected – it is guaranteed that no shorter path exists from s to this vertex (since any such path would have to pass through some other vertex of U, but the distances to such a vertex is larger than the distance to t). This vertex t is removed from U and

added to K, and then for each vertex u remaining in U, the distance from s to u *through* t is compared with the current best distance to u. If the former is found to be smaller, the best distance to u is updated to that value and t is marked as the vertex through which u is to be reached in the shortest path – this is indicated by $\mathrm{tree}(u)$ in the algorithm. This procedure is repeated until there is only vertex remaining in U (i.e. until $|U| = 1$).

Example 3.20. In the graph shown below, find the shortest paths from B to all the other vertices.



The weighted adjacency matrix of the graph is

$$W = \begin{array}{c} \\ A \\ B \\ C \\ D \\ E \end{array} \begin{array}{c} \begin{array}{ccccc} A & B & C & D & E \end{array} \\ \begin{bmatrix} 0 & 7 & \infty & \infty & 6 \\ \infty & 0 & 5 & 10 & \infty \\ 12 & 6 & 0 & 3 & \infty \\ 11 & \infty & 8 & 0 & \infty \\ 4 & 5 & \infty & 6 & 0 \end{bmatrix} \end{array}.$$

The results of applying the algorithm are tabulated below and the tree of shortest paths is also shown.

	A	C	D	E
$B(0)$	∞_B	$\boxed{5_B}$	10_B	∞_B
$C(5)$	17_C	–	$\boxed{8_C}$	∞_B
$D(8)$	$\boxed{17_C}$	–	–	∞_B
$A(17)$	–	–	–	23_A
B	17_C	5_B	8_C	23_A