Discrete Mathematics

MAT313

Prof. Dr. Chandrani Nag

Shahjalal University of Science and Technology

Edited by
Mehedi Hasan

Preface

This is a compilation of lecture notes with some books and my own thoughts. This document is not a holy text. So, if there is a mistake, solve it by your own judgement.

Contents

Part I

Lecture Note/Sheet

Chapter 1

Logic and Proof

1.1 Proposition

Definition 1 (Proposition)**.** A proposition is a declarative sentence (that is a sentence that declares a fact) that is either true or false but not both.

Example.

1. Toronto is the capital of Canada.

2. $1 + 1 = 2$

3. $2 + 2 = 3$

Proposition 1 and 3 are false whereas 2 is true.
Consider the following sentences:

1. What time it is?

2. $x + y = z$

are not propositions.

Definition 2 (Negation of p)**.** Let p be a proposition. The negation of p denoted by $\neg\, p$ (also denoted by $\bar{p}$) is the statement "It is not the case that p". The proposition $\neg\, p$ is read "not p".

"Today is Friday."
The negation is, "It is not the case that today is Friday." Simply, "Today is not Friday."

p	$\neg\, p$
T	F
F	T

Table 1.1: Truth table for negation of a proposition

Definition 3 (Conjunction of p and q)**.** Let p and q be propositions. The conjunction of p and q, denoted by $(p \wedge q)$ is the proposition "p and q". The conjunction $p \wedge q$ is true when both p and q are true and is false otherwise.

"Today is Friday and it is raining."

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

Table 1.2: Truth table for $p \wedge q$

Definition 4 (Disjunction of p and q)**.** Let p and q be propositions. The disjunction of p and q denoted by $(p \vee q)$ is the proposition "p or q." The disjunction $p \vee q$ is false when both p and q are false and is true otherwise.

"Students who have taken calculus or computer science can take this class."

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

Table 1.3: Truth table for $p \vee q$

Definition 5 (Exclusive or of p and q)**.** Let p and q be propositions. The exclusive or of p and q denoted by $(p \oplus q)$ is the proposition that is true when exactly one of p and q is true and is false otherwise.

"Students who have taken calculus or computer science, but not both, can take this class."

p	q	$p \oplus q$
T	T	F
T	F	T
F	T	T
F	F	F

Table 1.4: Truth table for $p \oplus q$

Let p and q be propositions. The conditional statement $p \to q$ is the proposition "If p then q." The conditional statement $p \to q$ is false when p is true and q is false and true otherwise. In the conditional statement $p \to q$, p is called hypothesis and q is called conclusion. "If I am elected, then I will lower taxes."

p	q	$p \to q$
T	T	T
T	F	F
F	T	T
F	F	T

Table 1.5: Truth table for $p \to q$

Definition 6 (Converse, Contrapositive and Inverse)**.** The proposition $q \to p$ is called converse of $p \to q$. The contrapositive of $p \to q$ is the proposition $\neg\, q \to \neg\, p$. The proposition $\neg\, p \to \neg\, q$ is called inverse of $p \to q$.

Example. "If it is raining, then the home team wins."
Contrapositive: If the home team does not win then it is not raining.
Converse: If the home team wins, then it is raining.
Inverse: If it is not raining, then the home team does not win.

Definition 7 (Biconditional)**.** Let p and q be propositions. The biconditional statement $p \leftrightarrow q$ is the proposition "p if and only if q." The biconditional statement $p \leftrightarrow q$ is true when p and q have the same truth values and is false otherwise. "You can take the flight if and only if you buy a ticket."

$$(p \to q) \wedge (q \to p)$$

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

Table 1.6: Truth table for $p \leftrightarrow q$

Problem 1.1.1. Construct the truth table of the compound proposition.

$$(p \vee \neg\, q) \to (p \wedge q)$$

Solution. Let us construct a table

p	q	$\neg\, q$	$p \vee \neg\, q$	$p \wedge q$	$(p \vee \neg\, q) \to (p \wedge q)$
T	T	F	T	T	T
T	F	T	T	F	F
F	T	F	F	F	T
F	F	T	T	F	F

Problem 1.1.2. How can this English sentence be translated into a logical expression?
"You can access the internet from campus only if you are a computer science major or you are not a freshman."

Solution.
$a \to$ You can access the internet from campus
$b \to$ You are a computer science major
$c \to$ You are a freshman
$a \to (b \vee \neg\, c)$

- Computer represent information using bits.
- A bit is a symbol with two possible values, namely, 0 and 1, 1 represents true, 0 represents false.
- A variable is called a Boolean variable if its value is either true or false.
- A bit string is a sequence of zero or more bits. The length of this string is the number of bits in this string.

$$\begin{array}{l} 01\,1011\,0110 \\ 11\,0001\,1101 \\ \hline 11\,1011\,1111 \longleftarrow \text{bitwise OR} \\ 01\,0001\,0100 \longleftarrow \text{bitwise AND} \\ 10\,1010\,1011 \longleftarrow \text{bitwise XOR} \end{array}$$

Definition 8 (Tautology and Contradiction)**.** A compound proposition that is always true, no matter what the truth values of the propositions that occur in it is called tautology. A compound proposition that is always false is called a contradiction. A compound proposition that is neither a tautology nor a contradiction is called a contingency.

p	$\neg p$	$p \wedge \neg p$	$p \vee \neg p$
T	F	F	T
F	T	F	T

Table 1.7: Truth table for tautology and contradiction

Definition 9 (Logically Equivalent)**.** The compound proposition p and q are called logically equivalent if $p \leftrightarrow q$ is a tautology. It is denoted as $p \equiv q$

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

Table 1.8: Truth table for logically equivalent

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

Problem 1.1.3. Prove that $p \vee (q \wedge r) = (p \vee q) \wedge (p \vee r)$

Problem 1.1.4. Show that $\neg (p \vee (\neg p \wedge q)) \equiv \neg p \wedge \neg q$

Solution.

$$\begin{aligned}\neg (p \vee (\neg p \wedge q)) &\equiv \neg p \wedge \neg (\neg p \wedge q)\\ &\equiv \neg p \wedge (\neg \neg p \vee \neg q)\\ &\equiv \neg p \wedge (p \vee \neg q)\\ &\equiv (\neg p \wedge p) \vee (\neg p \vee \neg q)\\ &\equiv \neg p \vee \neg q\end{aligned}$$

Definition 10 (Predicates)**.** Suppose a statement involving variables, such as $x > 3$
The statement "x is greater than 3" has two parts. The first part, the variable x, is the subject of the statement. The second part - the predicate, "is greater than 3" refers to a property that the subject of the statement can have.

We can denote the statement by $P(x)$, where p denotes the predicates and x is the variable. Propositional function p at x. Once a value has been assigned to the variable x the statement $P(x)$ becomes a proposition and has a truth value.

Example. Let $P(x)$ denotes the statement "$x > 3$". What are the values of $P(4)$ and $P(2)$?
$P(4)$, which is the statement "$4 > 3$" is true and $P(2)$, which is the statement "$2 > 3$" is false.

A statement of the form $P(x_1, x_2, \dot{;} x_n)$ is the value of the propositional function P at the n-tuple $(x_1, x_2, \dot{;} x_n)$ and P is called an n-place predicate or a n-ary predicate.

Problem 1.1.5. Consider the following program:

```
temp:=x
x:=y
y:=temp
```

Find predicates that we can use as the precondition and the post-condition to verify the correctness of this program. Then explain how to use them to verify that for all valid input the program does what is intended?

TODO:: Solve this.

Definition 11 (Quantifiers)**.** The universal quantification of $P(x)$ is the statement "$P(x)$ for all the values of x in the domain". The notation $\forall x P(x)$ denotes the universal quantification of $P(x)$ and $\forall$ is called the universal quantifier.

An element for which $P(x)$ is false is called a counterexample of $\forall x P(x)$.

Example. Let $P(x)$ be the statement "$x + 1 > x$". What is the truth value of the quantification $\forall x P(x)$. Where domain consists of all real numbers?

Because $P(x)$ is true for all real numbers x, the quantification, $\forall x P(x)$ is true.

Definition 12 (Existential Quantifier)**.** The existential quantification of $P(x)$ is the proposition "There exists an element x in the domain such that $P(x)$."

We use the notation $\exists x P(x)$ for the existential quantification of $P(x)$. Here $\exists$ is called the existential quantifier. Let $P(x)$ denote the statement "$x > 3$". What is the truth value of the quantification $\exists x P(x)$, where the domain consists of all real numbers?

$\#$ $\forall x < 0\,(x^2 > 0)$, $\forall y \neq 0(y^3 \neq 0)$
$\forall x(x < 0 \to x^2 > 0)$ $\forall y(y \neq 0 \to y^3 \neq 0)$
$\exists z > 0(z^2 = 2)$
$\exists z(z > 0 \wedge z^2 = 2)$

$\#$ Statements involving predicate and quantifiers are logically equivalent if and only if they have the same truth value no matter which predicates are substituted into these statements and which domain of discourse is used for the variables in these propositional function.

Problem 1.1.6. Show that $\forall x\,(P(x) \wedge Q(x))$ and $\forall x\,P(x) \wedge \forall x\,Q(x)$ are logically equivalent.

Solution. Let $\forall x\,(P(x) \wedge Q(x))$ is true. This means if a is in the domain then $(P(a) \wedge Q(a))$ is true. Hence, $P(a)$ and $Q(a)$ is true for every element in the domain, we can conclude that $\forall x P(x)$ and $\forall x Q(x)$ are both true. This means that $\forall x P(x) \wedge \forall x Q(x)$ is true.

Next suppose that $\forall x P(x) \wedge \forall x Q(x)$ is true. It follows that $\forall x P(x)$ is true and $\forall x Q(x)$ is true. Hence, if a is in the domain, $P(a)$ is true and $Q(a)$ is true. It follows that for all a, $P(a) \wedge Q(a)$ is true. Hence, $\forall x(P(x) \wedge Q(x))$ is true.

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

1. $\neg\ \exists x P(x) \equiv \forall x \neg\ P(x)$ [For every x, $P(x)$ is false]

2. $\neg\ \forall x P(x) \equiv \exists x \neg P(x)$ [There is an x for which $P(x)$ is false]

Problem 1.1.7. Show that, $\neg\ \forall x(P(x) \to Q(x)) \equiv \exists x(P(x) \wedge \neg\ Q(x))$

TODO: solve Problem

Problem 1.1.8. Express the statements “some student in this class has visited Mexico” and “Every student in this class has visited either Canada or Mexico” using predicates and quantifiers.

Solution. Let,
$S(x) \to x$ is a student in this class.
$M(x) \to x$ has visited Mexico.
$C(x) \to x$ has visited Canada.
The solution for first statement is $\exists x(S(x) \wedge M(x))$
The solution for second statement is $\forall x(S(x) \to (C(x) \vee M(x)))$

Definition 13 (Theorem)**.** A theorem is a statement that can be shown to be true. Less important theorems are sometimes called *proposition.*

Definition 14 (Proof)**.** A proof is a valid argument that establish the truth of a theorem.

Definition 15 (Lemma)**.** A less important theorem that is helpful in proof of other results is called a lemma.

Definition 16 (Corollary)**.** A corollary is a theorem that can be established directly from a theorem that has been proved.

Definition 17 (Conjucture)**.** A conjecture is a statement that is being proposed to be true statement, usually on the basis of some partial evidence.

Definition 18 (Direct Proof)**.** A direct proof of a conditional statement $p \to q$ is constructed when the first step is the assumption that P is true; subsequent steps are constructed using rules of interference, with the final step showing that q must also be true.

Definition 19 (Indirect Proof)**.** The proof that do not start with the hypothesis and end with the conclusion are called indirect proof.

Definition 20 (Proof by Contraposition)**.** Proof by contraposition means that the conditional statement $p \to q$ can be proved by showing that the contrapositive $\neg\, q \to \neg\, p$ is true.

Problem 1.1.9. Prove that if n is an integer and $3n + 2$ is odd, then n is odd.

Proof. Direct proof:
Let, $3n + 2$ is odd and $3n + 2 = 2k + 1$, for some integer k
This implies, $3n + 1 = 2k$
There does not seem to be any direct way to conclude that n is odd.
Our attempt at direct proof failed.
Proof by contraposition:
Assume that n is even.
So, $n = 2k$ for some integer k.
So, we have,

$$3n + 2 = 3(2k) + 2 = 2(3k + 1)$$

This tells us that $3n + 2$ is even.
Because the negation of the conclusion of the conditional statement implies that the hypothesis is false, the original conditional statement is true. □

Definition 21 (Vacuous Proof)**.** $p \to q$ is true when P is false $p \to q \equiv (\neg\, p \vee q)$.
If we can show that p is false, then we have a proof called a vacuous proof of the conditional statement $p \to q$.

Definition 22 (Trivial Proof)**.** A proof of $p \to q$ that uses the fact that q is true is called a trivial proof.

Definition 23 (Proofs by Contradiction)**.** Because the statement $r \wedge \neg r$ is a contradiction whenever r is a proposition, we can prove that p is true if we can show that $\neg p \rightarrow (r \wedge \neg r)$ is true for some proposition r. Proofs of this type are called proofs by contradiction.

Problem 1.1.10. Prove that $\sqrt{2}$ is irrational by giving a proof by contradiction.

Proof. If $\sqrt{2}$ is rational, there exists integer a and b with, $\sqrt{2} = \frac{a}{b}$ where a and b have no common factors.(Every rational number can be written in the lowest terms.) So, $2 = \frac{a^2}{b^2} \Rightarrow 2b^2 = a^2$
So, a^2 is even and a is also even.
So, $a = 2c$. Then, $2b^2 = 4c^2 \Rightarrow b^2 = 2c^2$
Similarly b is also even.
Now we have a and b have no common factors, a ans b are even, that is 2 divides both a and b.

Because our assumption of $\neg p$ leads to the contradiction that 2 divides both a and b and 2 does not divide both a and b, $\neg p$ must be false.
So $\sqrt{2}$ must be irrational. □

A statement of the form $\forall x P(x)$ is false we need only find a counterexample that is an example for which $P(x)$ is false.

Mistakes is proofs:

$$\begin{aligned}
&a = b\\
\Rightarrow &a^2 = ab\\
\Rightarrow &a^2 - b^2 = ab - b^2\\
\Rightarrow &(a+b)(a-b) = b(a-b)\\
\Rightarrow &a + b = b\\
\Rightarrow &2b = b\\
\Rightarrow &2 = 1
\end{aligned}$$

Definition 24 (Exhaustive Proof)**.** Some theorems can be proved by examining a relatively small number of examples. Such proofs are called exhaustive proof.

Example. Prove that, $(n+1)^3 \geq 3^n$ if n is a positive integer and $n \leq 4$.
If $n = 1$, $(1+1)^3 \geq 3^1 \Rightarrow 8 \geq 3$
$n = 2$, $(2+1)^3 \geq 3^2 \Rightarrow 27 \geq 9$
$n = 3$, $(3+1)^3 \geq 3^3 \Rightarrow 64 \geq 27$
$n = 4$, $(4+1)^3 \geq 4^3 \Rightarrow 125 \geq 81$

A proof by cases must cover all possible cases that arise in a theorem.
When the phrase "without loss of generality" is used in a proof, we assert that by proving one case of a theorem, no additional argument is required to prove other specified cases.

Definition 25 (Uniqueness Proofs)**.** The two parts of a uniqueness proof are:

- **Existence:** We show that an element x with desired property exists.
- **Uniqueness:** We show that if $y \neq x$, then y does not have the desired property.
  Equivalently, we can show that if x and y both has the desired property, then $x = y$.

Definition 26 (Argument)**.** By an argument, we mean a sequence of statements that end with a conclusion.
By *valid*, we mean that the conclusion or final statement of the argument must follow form the truth of the preceding statements or premises of the argument.

That is an argument is valid if and only if it is impossible for all the premises to be true and the conclusion to be false.

Definition 27 (Premises & Conclusion)**.** An argument in propositional logic is a sequence of propositions. All but the final proposition in the argument are called premises and the final proposition is called the conclusion.

An argument is valid if the truth of all its premises implies that the conclusion is true.

# We can first establish the validity of some relatively simple argument forms called rules of interference.
$p \rightarrow$ It is below freezing now
$q \rightarrow$ It is raining now
"It is below freezing now. Therefore, it is either below freezing or raining now."

$$\begin{array}{l} p \\ \hline \therefore\ p \vee q \end{array} \qquad p \rightarrow (p \vee q) \quad \text{addition}$$

Rule of interference	Tautology	Name
$\begin{array}{l} p \\ p \rightarrow q \\ \hline \therefore q \end{array}$	$[p \wedge (p \rightarrow q)] \rightarrow q$	Modus pones
$\begin{array}{l} \neg q \\ p \rightarrow q \\ \hline \therefore \neg p \end{array}$	$[\neg q \wedge (p \rightarrow q)] \rightarrow \neg p$	Modus tollens
$\begin{array}{l} p \rightarrow q \\ q \rightarrow r \\ \hline \therefore p \rightarrow r \end{array}$	$[(p \rightarrow q) \wedge (q \rightarrow q)] \rightarrow (p \rightarrow r)$	Hypothetical Syllogism
$\begin{array}{l} p \vee q \\ \neg p \\ \hline \therefore q \end{array}$	$[(p \vee q) \wedge \neg p] \rightarrow q$	Disjunctive Syllogism
$\begin{array}{l} p \\ \hline \therefore p \vee q \end{array}$	$p \rightarrow (p \vee q)$	Addition
$\begin{array}{l} p \wedge q \\ \hline \therefore p \end{array}$	$(p \wedge q) \rightarrow p$	Simplification
$\begin{array}{l} p \\ q \\ \hline \therefore p \wedge q \end{array}$	$[(p) \wedge (q)] \rightarrow (p \wedge q)$	Conjunction
$\begin{array}{l} p \vee q \\ \neg p \vee r \\ \hline \therefore q \vee r \end{array}$	$[(p \vee q) \wedge (\neg p \vee r)] \rightarrow (q \vee r)$	Resolution

Table 1.9: Rules of Interference

Problem 1.1.11. Show that the hypothesis "It is not sunny this afternoon and it is colder than yesterday", "We will go swimming only if it is sunny", "If we do not go swimming, then we will take a canoe trip", and "If we take a canoe trip, then we will be home by sunset" lead to the conclusion "We will be home by sunset."

Solution. Let,
$p \rightarrow$ It is sunny this afternoon
$q \rightarrow$ It is colder than yesterday
$r \rightarrow$ We will go swimming
$s \rightarrow$ We will take a canoe trip
$t \rightarrow$ We will be home by sunset
We construct an argument to show that our hypothesis lead to the desired conclusion as follows:

Steps	Reason
1. $\neg\, p \wedge\ q$	Hypothesis
2. $\neg\, p$	Simplification using 1
3. $r \rightarrow\ p$	Hypothesis
4. $\neg\, r$	Modus tollens using 2 and 3
5. $\neg\, r \rightarrow s$	Hypothesis
6. s	Modus pones using 4 and 5
7. $s \rightarrow t$	Hypothesis
8. t	Modus pones using 6 and 7

Problem 1.1.12. Prove that, if n is an integer then $n^2 \geq\ n$.

Proof. We split the proof into three cases.
Case i: When $n = 0$, because $0^2 = 0$. We see that, $0^2 = 0$. It follows that $n^2 \geq\ n$ is true in this case.
Case ii: When $n \geq\ 1$. We obtain $n^2 \geq\ n$ for positive integer n.
Case iii: When $n \leq\ -1$. We obtain $n^2 \geq\ 0 \geq\ n$.

Because the inequality $n^2 \geq\ n$ holds in all three cases, we can conclude that if n is an integer then $n^2 \geq\ n$. □

# **Division Algorithm** Let a be an integer and d be a positive integer. Then there are unique integers q and r with $0 \geq\ r <\ d$. Such that,

$$a =\ dq + r$$

$$q =\ \text{div}\ d, \qquad r =\ a \mod\ d$$

Problem 1.1.13. What are the quotients when -11 is divided by 3?

Solution. $11 = 3(-4) + 1$ remainder cannot be negative.

If a and b are integers and m is a positive integer. Then a is congruent to b modulo m if m divides $(a - b)$

$$a \equiv\ b \mod\ m$$

$$17 \equiv\ 5 \mod\ 6$$

# Fundamental Theorem of Arithmetic Every positive integer greater than 1 can be written uniquely as a prime or as the product of two or more primes where the prime factors are written in order of non-decreasing size.

$$100 = 2 \cdot 2 \cdot 5 \cdot 5 = 2^2 \cdot 5^2$$

$$999 = 3 \cdot 3 \cdot 3 \cdot 37 = 3^3 \cdot 37$$

# Representations of Integers Let b be a positive integer greater than 1. Then if n is a positive integer, it can be uniquely expressed in the form,

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

Where k is a non-negative integer, $a_0, a_1, \ldots, a_k$ are non-negative integers less than b and $a_k \neq 0$

Problem 1.1.14. What is the decimal expansion of the integer that has $(1\,0101\,1111)_2$ as its binary expansion?

Solution. We have,

$$(1\,0101\,1111)_2 = 1 \cdot 2^8 + 0 \cdot 2^7 + 1 \cdot 2^6 + 0 \cdot 2^5 + 1 \cdot 2^4 + 1 \cdot 2^3 + 1 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 = 351$$

# Hexadecimal Expansions The base 16 expansion of an integer is called its hexadecimal expansion.

$$(2AE0B)_{16} = 2 \cdot 16^4 + 10 \cdot 16^3 + 14 \cdot 16^2 + 0 \cdot 16^1 + 11 \cdot 16^0 = (175627)_{10}$$

$$(1110\,0101)_2 = (E5)_{16}$$

# Base Conversion integer n, base b

$$\begin{aligned} n =& \; bq_0 + a_0, \quad 0 \le a_0 < b \\ q_0 =& \; bq_1 + a_1, \quad 0 \le a_1 < b \\ \cdots & \end{aligned}$$

This process terminates when we obtain a quotient equal to zero.

Problem 1.1.15. Find the base 8 or octal expansion of $(12345)_{10}$.

Solution. First divide 12345 by 8, we obtain

$$\begin{aligned} 12345 =& \; 8 \cdot 1543 + 1 \\ 1543 =& \; 8 \cdot 192 + 7 \\ 192 =& \; 8 \cdot 24 + 0 \\ 24 =& \; 8 \cdot 3 + 0 \\ 3 =& \; 8 \cdot 0 + 3 \end{aligned}$$

So $(12345)_{10} = (30071)_8$

Problem 1.1.16. Find the hexadecimal expansion of $(177130)_{10}$.

Solution. First divide 177130 by 16 to obtain,

$$\begin{aligned} 177130 =& \; 16 \cdot 11070 + 10 \\ 11070 =& \; 16 \cdot 691 + 14 \\ 691 =& \; 16 \cdot 43 + 3 \\ 43 =& \; 16 \cdot 2 + 11 \\ 2 =& \; 16 \cdot 0 + 2 \end{aligned}$$

So $(177130)_{10} = (2B3EA)_{16}$

Problem 1.1.17. Find the binary expansion of $(241)_{10}$.

Solution. First divide 241 by 2 to obtain,

$$\begin{aligned}
241 &= 2 \cdot 120 + 1 \\
120 &= 2 \cdot 60 + 0 \\
60 &= 2 \cdot 30 + 0 \\
30 &= 2 \cdot 15 + 0 \\
15 &= 2 \cdot 7 + 1 \\
7 &= 2 \cdot 3 + 1 \\
3 &= 2 \cdot 1 + 1 \\
1 &= 2 \cdot 0 + 1
\end{aligned}$$

So $(241)_{10} = (1111\,0001)_2$

# **Binary Addition** Add $(1110)_2$ and $(1011)_2$
Following the procedure specified in the algorithm
$a_0 + b_0 = 0 + 1 = 0 \cdot 2 + 1$
$c_0 = 0$ and $s_0 = 1$
Then $a_1 + b_1 + c_0 = 1 + 1 + 0 = 1 \cdot 2 + 0$
$c_1 = 1$ and $s_1 = 0$
Continuing,
$a_2 + b_2 + c_1 = 1 + 0 + 1 = 1 \cdot 2 + 0$
$c_2 = 1$ and $s_2 = 0$
$a_3 + b_3 + c_2 = 1 + 1 + 1 = 1 \cdot 2 + 1$
$c_3 = 1$ and $s_3 = 1$
Then $s_4 = c_3 = 1$
so, $(1110)_2 + (1011)_2 = (1\,1001)_2$

# **Binary Subtraction**

1.
$$\begin{array}{c} 0 \\ \text{-}0 \\ \hline 0 \end{array} \qquad \begin{array}{c} 1 \\ \text{-}0 \\ \hline 1 \end{array} \qquad \begin{array}{c} 1 \\ \text{-}1 \\ \hline 0 \end{array} \qquad \begin{array}{c} 0 \\ \text{-}1 \\ \hline 11 \end{array}$$

$$\begin{array}{r|r} 1\ 0101 & 21 \\ \text{-}1\ 0100 & 20 \\ \hline 0\ 0001 & 1 \end{array} \qquad\qquad \begin{array}{r|r} 1101 & 13 \\ \text{-}0011 & 3 \\ \hline 1010 & 10 \end{array}$$

2. (i) 1's complement method:

   To subtract a smaller number form a larger number, the 1's complement method is as follows:

   Example. Subtract $(1010)_2$ from $(1111)_2$

$$\begin{array}{rrl} & 1111 & \\ & +\ 0101 & \leftarrow \text{1's complement of}(1010)_2 \\ \cline{2-2} \text{carry} \leftarrow & \underline{1}\ 0100 & \\ \text{Add carry} \rightarrow & 1 & \\ \cline{2-2} & 0101 & \end{array}$$

   (ii) 2's complement subtraction:

   The 2's complement of a binary number can be obtained by adding 1 to it's 1's complement.

Example. Subtract $(10\,1101)_2$ from $(11\,0110)_2$

$$\begin{array}{rrl} & 11\ 0110 & \\ & +\ 01\ 0011 & \leftarrow \text{2's complement of}(10\,1101)_2 \\ \hline \text{carry} \leftarrow & \underline{1}00\ 1001 & \end{array}$$

The carry is discarded. Thus answer is $(1001)_2$

Note: The 2's complement method is also for subtractions of a large number from a smaller one.

Chapter 2

Graph Theory

2.1 Graph and Multigraph

A graph G consists of two things:

(i) A set $V = V(G)$ whose elements are called vertices, point or nodes of G.

(ii) A set $E = E(G)$ of unordered pairs of distinct vertices called edges of G.

We denote such a graph by $G(V, E)$.

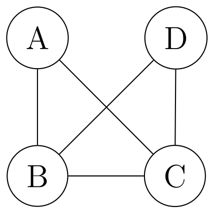


Figure 2.1: G_1

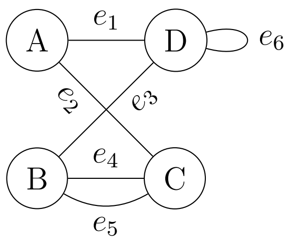


Figure 2.2: G_2

2.2 Multigraphs

Look at the graph 2.2, the edges e_4 and e_5 are called multiple edges since they connect the same endpoints and the edge e_6 is called a loop since its endpoints are the same vertex. Such a diagram is called multigraph.

2.3 Degree

The degree of a vertex v in a graph G written $deg(v)$ is equal to the number of edges in G which contains v, that is which are incident on v. A vertex with degree zero is called isolated. In graph G_2, $deg(D) = 4$, $deg(C) = 3$.

A multigraph is said to be finite if it has a finite number of vertices and a finite number of edges. The finite graph with one vertex and no edges is called the trivial graph.

2.4 Subgraph

Consider a graph $G = G(V, E)$. A graph $H = H(V', E')$ is called a subgraph of G if the vertices and edges of H are contained in the vertices and edges of G.

2.5 Isomorphic Graphs and Homeomorphic Graph

Graphs $G(V, E)$ and $G^*(V^*, E^*)$ are said to be homeomorphic if the can be obtained from the same graph or isomorphic graphs by dividing an edge with additional vertices.

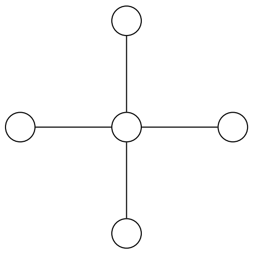


Figure 2.3: Graph A

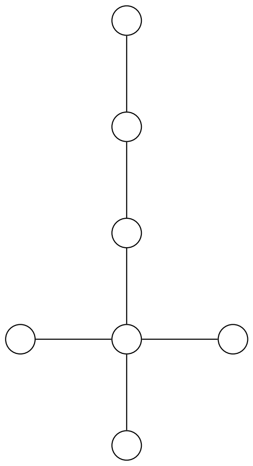


Figure 2.4: Graph B

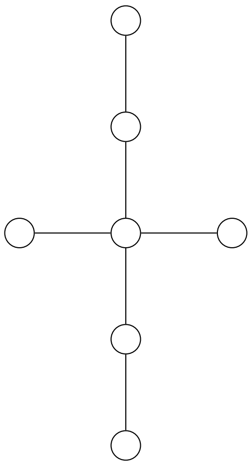


Figure 2.5: Graph C

Graphs 2.4 and 2.5 are homeomorphic since they can be obtained from 2.3 by adding appropriate vertices.

Graphs G and G^* are said to be isomorphic if there exists a one-to-one correspondence $f : V \to V^*$ such that $\{u, v\}$ is an edge of G if and only if $\{f(u), f(v)\}$ is an edge of G^*.

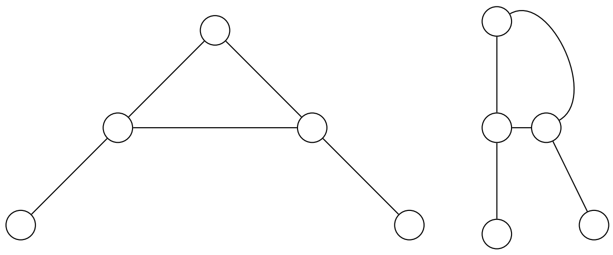


Figure 2.6: Isomorphic Graphs

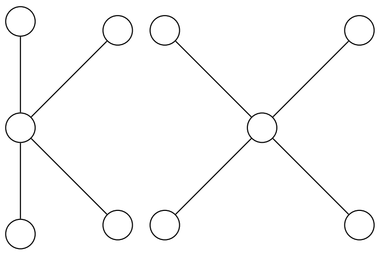


Figure 2.7: Isomorphic Graphs

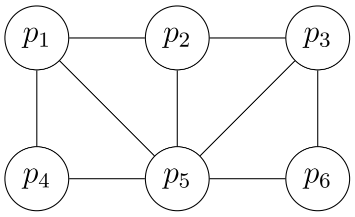
2.6 Paths, Connectivity

A path in a multigraph G consists of vertices and edges of the form,

$v_0,\ e_1,\ v_1,\ e_1,\ v_2,\ e_2, \ldots,\ e_{n-1},\ v_{n-1},\ e_n,\ v_n.$

Where each edge e_i contains the vertices v_{i-1} and v_i. The number n of edges is called the length of the path.

- The path is closed if $v_0 = v_n$.
- A simple path is a path in which all vertices are different.
- A path in which all edges are different will be called a trail.
- A cycle is a closed path in which all vertices are distinct except $v_0 = v_n$.



$\alpha = \{p_4, p_1, p_2, p_5, p_1, p_2, p_3, p_6\}$ is not a trail
$\beta = \{p_4, p_1, p_5, p_2, p_6\}$ is not a path
$\gamma = \{p_4, p_1, p_5, p_2, p_3, p_5, p_6\}$ is a trail but not simple
$\delta = \{p_4, p_1, p_5, p_3, p_6\}$ is simple but not shortest

- A graph G is connected if there is a path between any two of its vertices.
- The distance between vertices u and v in G written $d(u, v)$ is the length of the shortest path between u and v and the diameter of G, written $diam(g)$ is the maximum distance between any two points in G.

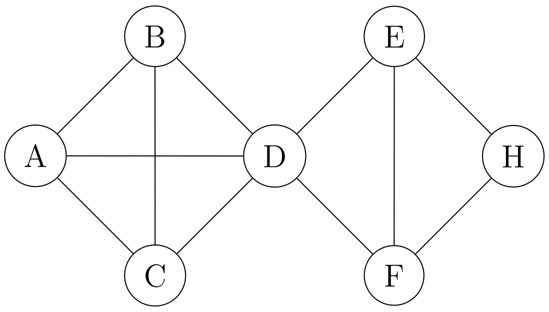


Figure 2.8: G_1

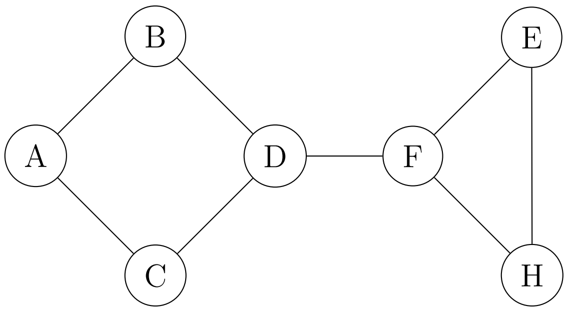
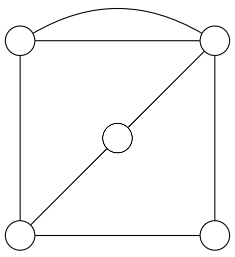
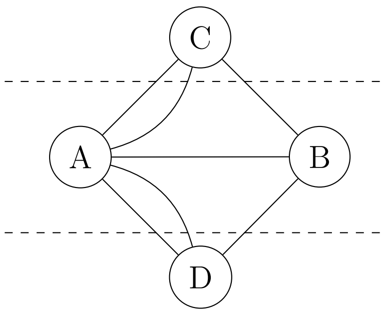


Figure 2.9: G_2

In G_1(figure 2.8), $d(A,\ F) = 2$ and $diam(G_1) = 3$
In G_2(figure 2.9), $d(A,\ F) = 3$ and $diam(G_2) = 4$

- A vertex v in G is called a cutpoint if $G - v$ is disconnected.
- An edge e is called a bridge if $G - e$ is disconnected.
(*D* in graph G_1 is cutpoint and $e = \{D, F\}$ is a bridge in G_2.)
- A multigraph is said to be traversable if it can be drawn without any breaks in the curve and without repeating any edges.





A multigraph with more than two odd vertices cannot be traversable. The famous königsberg bridge problem has four odd vertices.

- A graph G is called an Eulerian graph if there exists a closed traversable trail.

- A finite connected graph is Eulerian if and only if each vertex has even degree.
- A Hamiltonian circuit in a graph G named after the nineteenth-century Irish mathematician William Hamilton, is a closed path that visits every vertex in G exactly once.

  A graph G that admits a Hamiltonian circuit is called a Hamiltonian graph.

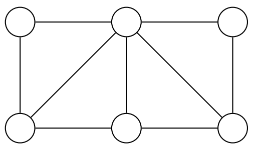


Figure 2.10: Hamiltonian and non-Eulerian

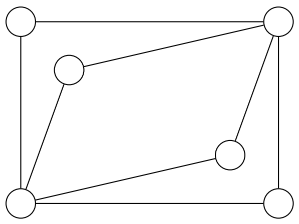


Figure 2.11: Eulerian and non-Hamilton

- A graph G is called a labeled graph if its edges are assigned data of one kind.

  A graph G is called a weighted graph if each e of G is assigned a non-negative number $w(e)$ called the weight or length of v.
- A graph G is said to be complete if every vertex in G is connected to every other vertex in G. The complete graph with n vertices is denoted by k_n.

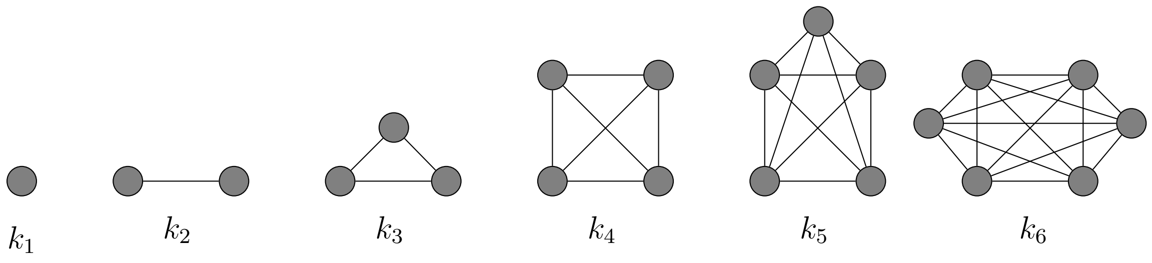


Figure 2.12: Some complete graphs

- A graph G is k-regular if every vertex has degree k.

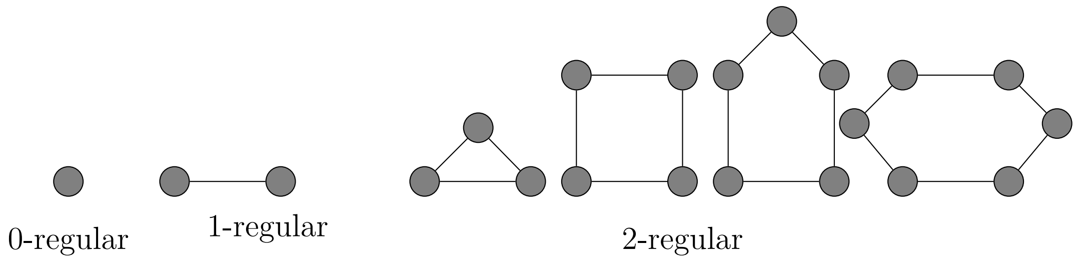
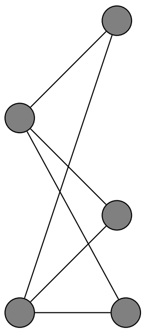


Figure 2.13: Some regular graphs

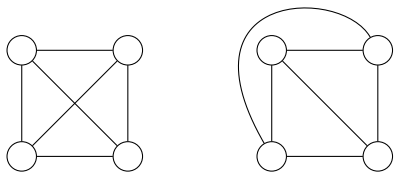
- A graph G is said to be bipartite if its vertices V can be partitioned into two subsets M and N such that each edge of G connects a vertex of M to a vertex of N.

  By $k_{m,n}$ we mean that each vertex of M is connected to each vertex of N, a complete bipartite graph.



$k_{2,3}$

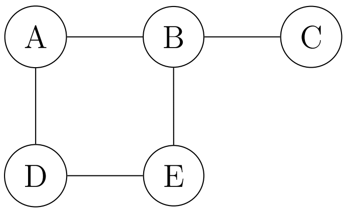
- A graph or multigraph which can be drawn in the plane so that its edges do not cross is said to be planar.



- A particular planar representation of a finite planar multigraph is called a map.
- Let G be a connected planar graph with p vertices and q edges, where $p \geq 3$. Then $q \leq 3p - 6$.
- Suppose G is a graph with m vertices and suppose the vertices have been ordered say, $v_1, v_2, \ldots, v_m$. Then the adjacency matrix $A = [a_{ij}]$ of the graph G is the $m \times m$ matrix defined by

$$a_{ij} = \begin{cases} 1 & \text{if } v_i \text{ is adjacent to } v_j \\ 0 & \text{Otherwise} \end{cases}$$

  Adjacency matrix is symmetric.

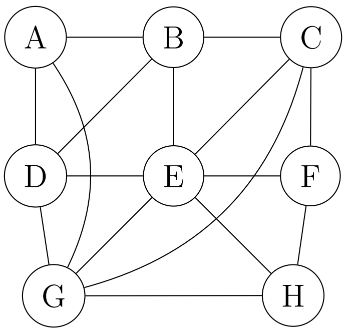


$$\begin{array}{c|ccccc} & A & B & C & D & E \\ \hline A & 0 & 1 & 0 & 1 & 0 \\ B & 1 & 0 & 1 & 0 & 1 \\ C & 0 & 1 & 0 & 0 & 0 \\ D & 1 & 0 & 0 & 0 & 1 \\ E & 0 & 1 & 0 & 1 & 0 \end{array}$$

- A vertex coloring or simply a coloring of G is an assignment of colors to the vertices of G such that adjacent vertices have different colors.

  The minimum number of colors needed to paint G is called the chromatic number of G and is denoted by $\lambda(G)$.

  chi χ



1. Ordering the vertices of G according to decreasing degrees. Here they are E, G, B, C, D, A, F, H
2. Assign first color c_1 to first vertex and assign c_1 to each vertex which is not adjacent to first vertex.
3. Repeat step 2 with second color c_2.
4. Repeat step 3 and 3 until no vertex left.
5. Exit.

First color c_1 to E, A
Second color c_2 to G, B, F
Third color c_3 to C, D, H
Since every vertex is adjacent to every other vertex, k_n requires n colors in any coloring. Thus, $\lambda(k_n) = n$.

2.7 Four Color Theorem

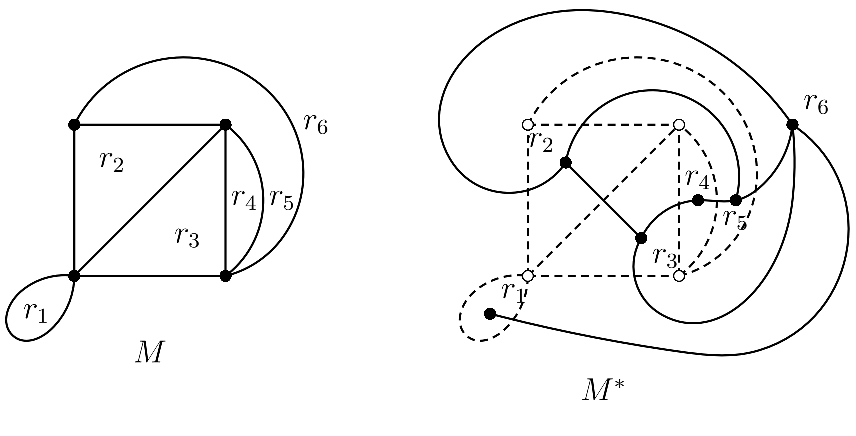
Any planar graph is four colorable.

2.7.1 Dual maps and the four color theorem

If the regions of any map M are colored so that adjacent regions have different colors, then no more than four colors are required.

Consider a map M.

- Two regions of M are said to be adjacent if they have an edge in common.
- By a coloring of M we mean an assignment of a color to each region of M such that adjacent regions have different colors.



- M is 3-colorable. Because r_1 red, r_2 white, r_3 red, r_4 white, r_5 red, r_6 blue.
- In each region of M we choose a point and if two regions have an edge in common then we connect the corresponding points with a curve through the common edge. These curves can be drawn so that they are non-crossing. Thus, we obtain a new map M^*, called the dual of M, such that each vertex of M^* corresponds to exactly one region of M.
- A graph T is called a tree if T is connected and T has no cycles.

2.8 Spanning Tree

A subgraph T of a connected graph G is called a spanning tree of G if T is a tree and T includes all the vertices of G.

Suppose G is a connected weighted graph. Then any spanning tree T of G is assigned a total weight obtained by adding the weights of the edges in T.

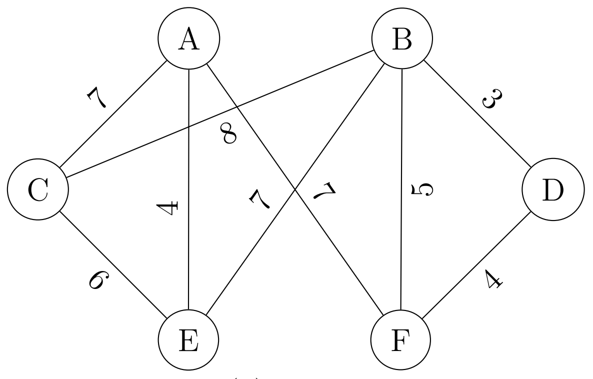
A minimal spanning tree of G is a spanning tree whose total weight is as small as possible.

2.8.1 Kruskal's algorithm for minimal spanning tree

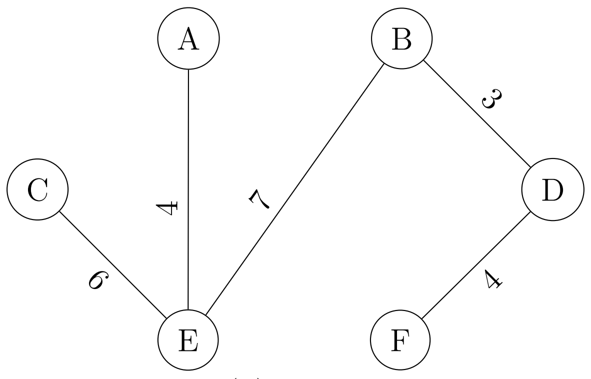
step 1: Arrange the edges of G in order of increasing weights.

step 2: Starting only with the vertices of G and proceeding sequentially, add each edge which does not result in a cycle until $(n-1)$ edges are added.

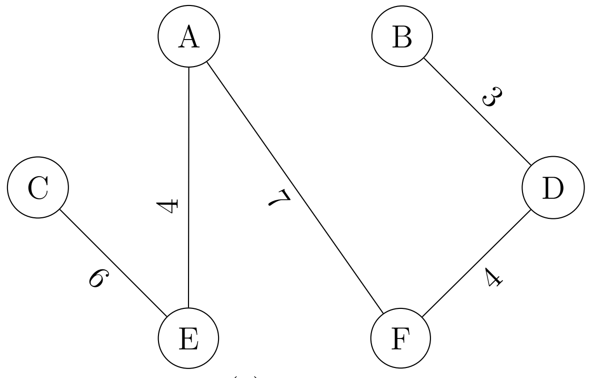
step 3: Exit.



(a)



(b)



(c)

First we order the edges by decreasing weights and then we successively delete edges without disconnecting Q until five edges remain. This yields the following data:

Edges	BC	AF	AC	BE	CE	BF	AE	DF	BD
Weight	8	7	7	7	6	5	4	4	3
Delete	Yes	Yes	Yes	No	No	Yes	No	No	No

Thus the minimal spanning tree of Q which is obtained contains the edges

$$BE,\ CE,\ AE,\ DF,\ BD$$

2.9 Traversing Binary Tree

There are three standard ways of traversing a binary tree T with root R. these three algorithm called pre order, in order and post order.

2.9.1 Pre order

1. Process the root R.
2. Traverse the left subtree of R in pre-order.
3. Traverse the right subtree of R in pre-order.

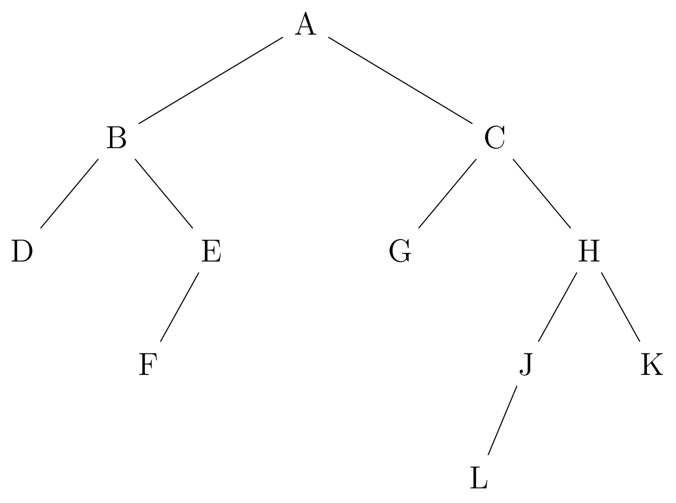
2.9.2 In order

1. Traverse the left subtree of R in in order.
2. Process the root R.
3. Traverse the right subtree of R in in order.

2.9.3 Post order

1. Traverse the left subtree of R in post order.
2. Traverse the right subtree of R in post order.
3. Process the root R.

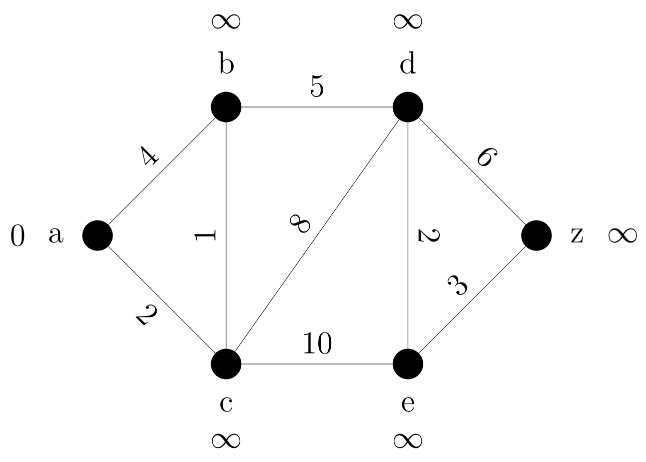
Example. Traverse the tree of the following figure.



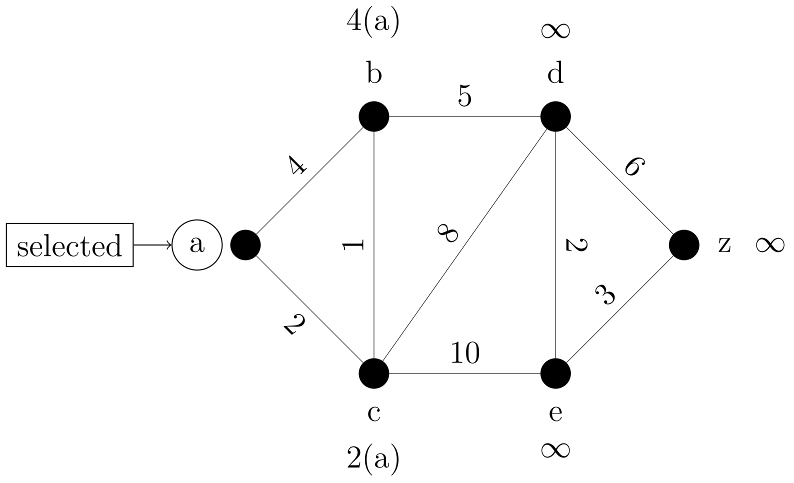
Pre-order traversal = A, B, D, E, F, C, G, H, J, L, K
In-order traversal = D, B, F, E, A, G, C, L, J, H, K
Post-order traversal = D, F, E, B, G, L, J, K, H, C, A

Example. Using Dijkstra's algorithm to find the shortest path from a to z.

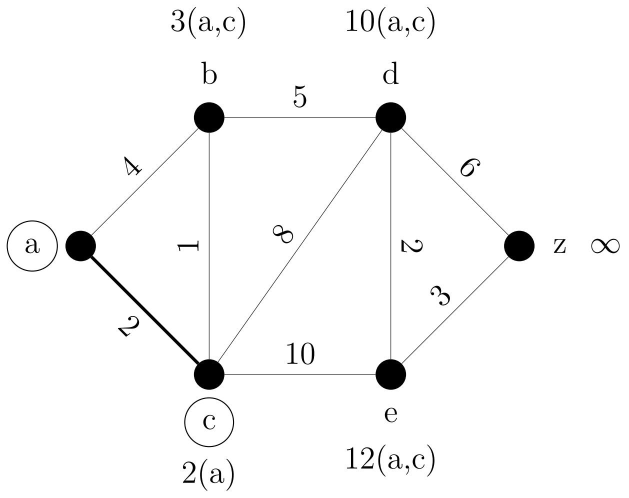
Step 1:



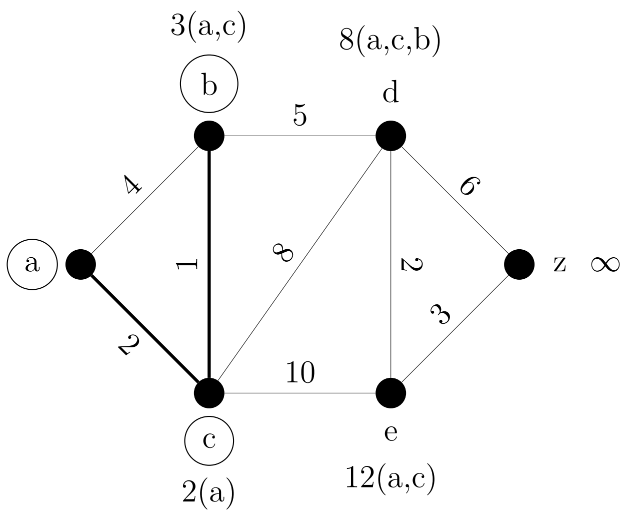
Step 2:



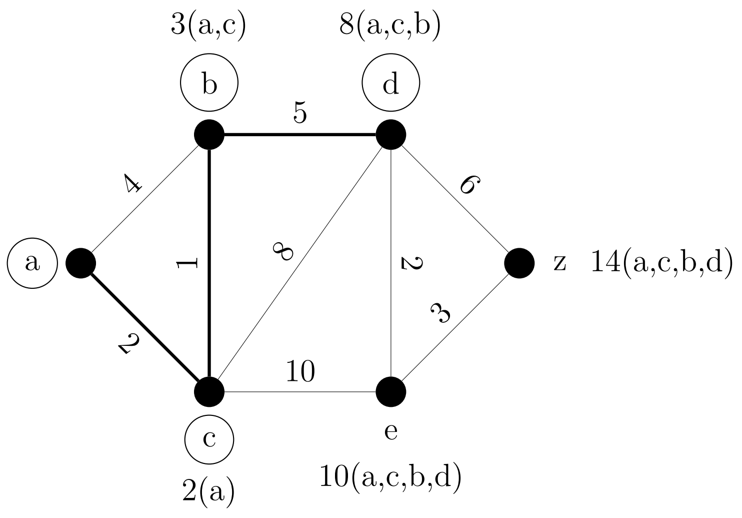
Step 3:



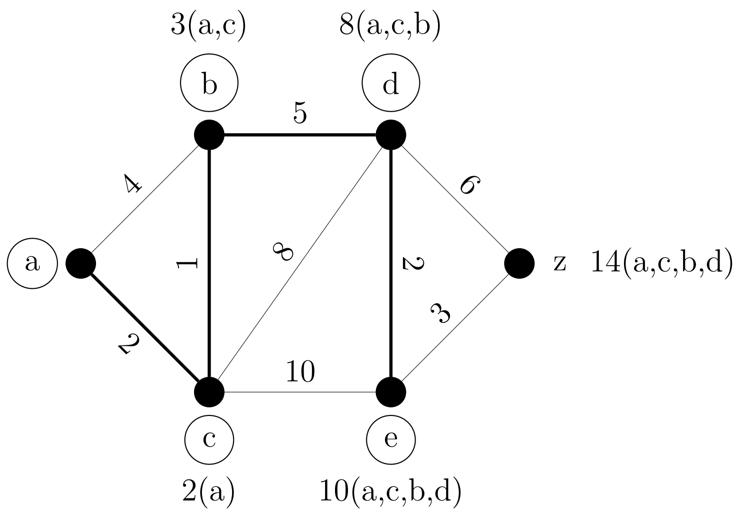
Step 4:



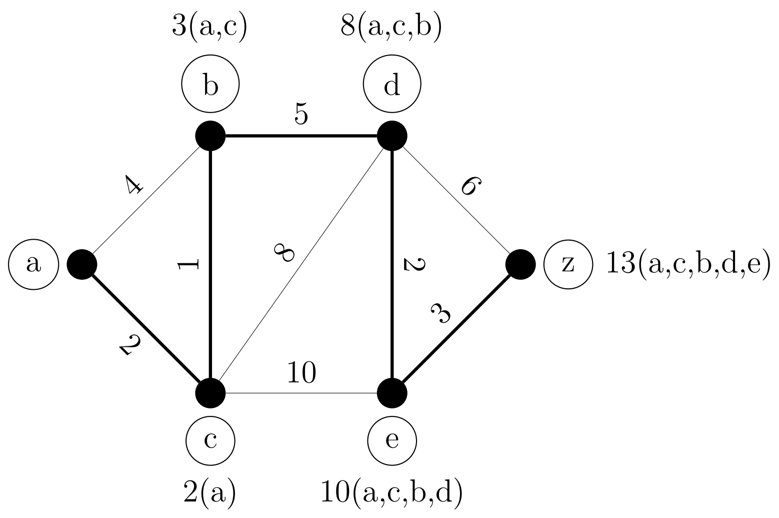
Step 5:



Step 6:



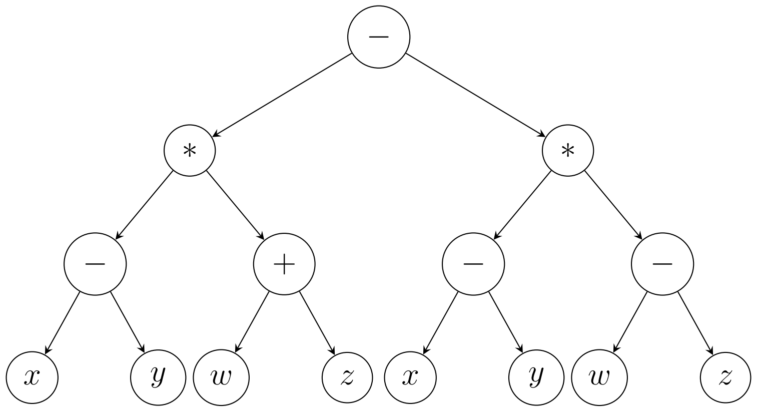
Step 7:



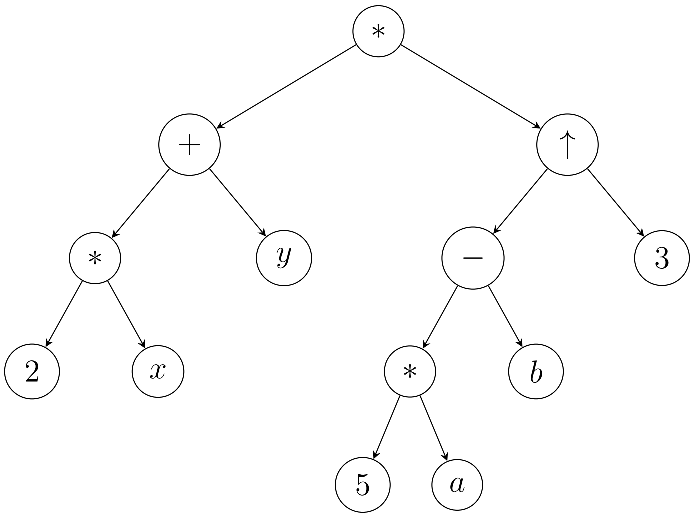
The shortest path: $a \rightarrow c \rightarrow b \rightarrow d \rightarrow e \rightarrow z$

2.10 Graphical Representation of an Expression

In compiler construction an expression such as $'(x-y)*(w+z)*(x-y)*(w-z)'$ can be represented by the directed acyclic graph[1].



Example. Draw the tree which corresponds to the expression $E = (2x+y)(5a-b)^3$



[1]See tree traversal, infix postfix expression, expression tree.

Chapter 3

Boolean Algebra

3.1 Boolean Expressions

Definition 28. Let $x_1, x_2, \ldots, x_n$ be a set of n variables (or letters or symbols). A *Boolean Polynomial (Boolean expression, Boolean form or Boolean formula)* $p(x_1, x_2, \ldots, x_n)$ in the variables $x_1, x_2, \ldots, x_n$ is defined recursively as follows:

1. The symbols 0 to 1 are Boolean polynomials
2. $x_1, x_2, \ldots, x_n$ are all Boolean polynomials
3. if $p(x_1, x_2, \ldots, x_n)$ and $q(x_1, x_2, \ldots, x_n)$ are two Boolean polynomials, then so are
$$p(x_1, x_2, \ldots, x_n) \vee q(x_1, x_2, \ldots, x_n)$$
and
$$p(x_1, x_2, \ldots, x_n) \wedge q(x_1, x_2, \ldots, x_n)$$
4. If $p(x_1, x_2, \ldots, x_n)$ is a Boolean polynomial, then so is $(p(x_1, x_2, \ldots, x_n))'$.
5. There are no Boolean polynomials in the variables $x_1, x_2, \ldots, x_n$ other than those obtained in accordance with rules 1 to 4.

Thus, Boolean expression is an expression built from the variables given using Boolean operations $\vee$, $\wedge$ and $'$.

For example, for variable x, y, z, the expression

$$\begin{aligned} p_1(x, y, z) &= (x \vee y) \wedge z \\ p_2(x, y, z) &= (x \vee y') \wedge (y \wedge 1) \\ p_3(x, y, z) &= (x \vee (y' \wedge z)) \vee (x \wedge (y \wedge 1)) \end{aligned}$$

are Boolean expressions.

Notice that a Boolean expression is n variable may or may not contain all the b variables. Obviously, an infinite number of Boolean expressions may be constructed in n variables.

Definition 29 (Literal)**.** A *literal* is a variable or complemented variable such as x, x', y, y', and so on.

Definition 30 (Fundamental Product)**.** A *fundamental product* is a literal or a product of two or more literal in which no two literal involve same variable.

For example,

$$x \wedge z',\ x \wedge y' \wedge z,\ x,\ y'\ x' \wedge y \wedge z$$

are fundamental products whereas

$$x \wedge y \wedge x' \wedge z \text{ and } x \wedge y \wedge z \wedge y$$

are not fundamental products.

Remark. Fundamental product is also called a *minterm* or *complete product.* In what follows we shall denote $x \wedge y$ by xy.

Any product of literals can be reduced to either 0 or a fundamental product.

For example, consider $x\ y\ x'\ z$. Since, $x \wedge x' = 0$ by complement law, we have $xyx'z = 0$.

Similarly, if we consider $x\ y\ z\ y$ then since $y \wedge y = y$ (idempotent law), we have $xyzy = xyz$, which is a fundamental product.

Definition 31. A fundamental product P_1 is said to be contained in (or included in) another fundamental Product P_2 if the literals of P_1 are also literals of P_2.
For example, $x'z$ is contained in $x'yz$ but $x'z$ is not contained in $xy'z$ since x' is not a literal of $xy'z$.

Observe that if P_1 is contained in P_2, say $P_2 = P_1 \wedge Q$, then, by the absorption law,

$$P_1 \vee P_2 = P_1 \vee (P_1 \wedge Q) = P_1$$

For example,

$$x'z \vee x'yz = x'z$$

Definition 32. A Boolean expression E is called a sum-of-products expression(disjunctive Normal Form or DNF) if E is a fundamental product or the sum (join) of two or more fundamental products none of which is contained in another.

Definition 33. Two Boolean expression $P(x_1, x_2, \ldots, x_n)$ and $Q(x_1, x_2, \ldots, x_n)$ are called equivalent (or equal) if one can be obtained from the other by a finite number of applications of the identities of a Boolean algebra.

Definition 34. Let E be any Boolean expression. A sum of product form of E is an equivalent Boolean sum of products expression.
Example: Consider the expression

$$E_1(x, y, z) = xz' + y'z + xyz'$$

Although the expression E_1 is a sum of products, it is not a sum-of-products expression because, the product xz' is contained in the product xyz'. But, by absorption law, E_1 can be expressed as

$$E_1(x, y, z) = xz' + y'z + xyz' = xz' + xyz' + y'z = xz' + y'z$$

, which is a sum-of-product form for E_1.

3.2 Algorithm for Finding Sum-of-Products Forms

The input is a Boolean expression E. The output is a sum-of-products expression equivalent to E.

Step 1. Use De Morgan's Law and involution to move the complement operation into any parenthesis until finally the complement operation only applies to variables. Then E will consist only sums and products of literals.

Step 2. Use the distributive operation to next transform E into a sum of products.

Step 3. Use the commutative, idempotent, and complement laws to transform each product in E into 0 or a fundamental product.

Step 4. Use the absorption law and identity law to finally transform E into a sum of products expression.

For example, we apply the above Algorithm to the Boolean expression,

$$E = ((xy)'z)'((x' + z)(y' + z'))'$$

Step 1. Using De Morgan's laws and involution, we obtain

$$\begin{aligned} E &= ((xy)'' + z')((x' + z) + (y' + z')') \\ &= (xy + z')(xz' + yz) \end{aligned}$$

Thus E consists only of sum and products of literals.

Step 2. Using the distributive laws, we obtain

$$\begin{aligned} E &= (xy + z')(xz' + yz) \\ &= xyxz' + xyyz + xz'z' + yzz' \end{aligned}$$

Thus E is now a sum of products.

Step 3. Using commutative idempotent and complement law, we obtain

$$E = xyz' + xyz + xz' + 0$$

Each term in E is a fundamental product or 0.

Step 4. The product xz' is contained in xyz'; hence, by the absorption law,

$$xz' + (xz'y) = xz'$$

Thus we may delete xyz' form the sum. Also, by the identity law for 0, we may delete 0 from the sum. Accordingly,

$$E = xyz + xz'$$

E is now represented by a sum-of-products expression.

3.3 Consensus of Fundamental Products

Let P_1 and P_2 be fundamental products such that exactly one variable say x_k appears uncomplemented in one of P_1 and P_2 and complemented in the other. Then the *consensus of* P_1 *and* P_2 is the product (without repetitions) of the literals of P_1 and P_2 after x_k and x_k' are deleted. (we do not define the consensus of $P_1 = x$ and $P_2 = x'$)

Lemma 1. Suppose Q is the consensus of P_1 and P_2. Then $P_1 + P_2 + Q = P_1 + P_2$.

Proof. Since the literals commute, we can assume without loss of generality that

$$P_1 = a_1a_2\ldots a_rt \qquad P_2 = b_1b_2\ldots b_st' \qquad Q = a_1a_2\ldots a_rb_1b_2\ldots b_s$$

Now, $Q = Q(t + t') = Qt + Qt'$. Because Qt contains P_1, $P_1 + Qt = P_1$; and because Qt' contains P_2, $P_2 + Qt' = P_2$. Hence,

$$\begin{aligned} P_1 + P_2 + Q &= P_1 + P_2 + Qt + Qt' \\ &= (P_1 + Qt) + (P_2 + Qt') \\ &= P_1 + P_2. \end{aligned}$$

□

Problem 3.3.1. Find the consensus Q of P_1 and P_2, where

(i) $P_1 = xyz's$, $P_2 = xy't$

(ii) $P_1 = xy'$, $P_2 = y$

(iii) $P_1 = x'yz$, $P_2 = x'yt$

(iv) $P_1 = x'yz$, $P_2 = xyz'$.

Solution.

(i) $P_1 = xyz's$, $P_2 = xy't$
Delete y and y' and then multiply the literals of P_1 and P_2 (without repetition) to obtain $Q = xz'st$

(ii) $P_1 = xy'$, $P_2 = y$ Delete y and y' and then multiply the literals of P_1 and P_2 (without repetition) to obtain $Q = x$

(iii) $P_1 = x'yz$, $P_2 = x'yt$ In this case, no variable appears uncomplemented in one of the products and complemented in the other. Hence, P_1 and P_2 have no consensus.

(iv) $P_1 = x'yz$, $P_2 = xyz'$ Each of x and z appear complemented in one of the products and uncomplemented in the other. Hence, P_1 and P_2 have no consensus.

3.4 Consensus Method For Finding Prime Implicants

The following algorithm, known as *consensus Method* is used to find the prime implicants of a Boolean expression.

3.4.1 Algorithm (Consensus Method)

The input is a Boolean expression $E = P_1 + P_2 + \cdots + P_m$, where P_m are fundamental products. The output expresses E as a sum of its prime implicants.

Step 1. Delete any fundamental product P_i which includes any other fundamental product P_j (this is permissible by the absorption law).

Step 2. Add the consensus of any P_i and P_j providing Q does not include any of the P_i (this is permissible by the lemma $P_1 + P_2 + \cdots + P_n + Q = P_1 + \cdots + P_n$).

Step 3. Repeat step 1/or step 2 until neither can be applied.

Example. Let $E(x, y, z) = xyz + x'z' + xyz' + x'y'z + x'yz'$.
Then

$$\begin{aligned}
E &= xyz + x'z' + xyz' + x'y'z && (x'yz' \text{ includes } x'z')\\
&= xyz + x'z' + xyz' + x'y'z + xy && (\text{ consensus of } xyz, \text{ and } xyz' \text{ added})\\
&= x'z' + x'y'z + xy && (xyz \text{ and } xyz' \text{ include } xy)\\
&= x'z' + x'y'z + xy + x'y' && (\text{ consensus } x'y' \text{ of } x'z' \text{ and } x'y'z \text{ added})\\
&= x'z' + xy + x'y' && (x'y'z \text{ includes } x'y')\\
&= x'z' + xy + x'y' + yz' && (\text{ consensus of } x'z' \text{ and } xy, \text{ which is } yz', \text{ added})
\end{aligned}$$

After this none of the step in the consensus method will change E. Thus, E is the sum of its prime implicants $x'z'$, xy, $x'y'$, and yz'.

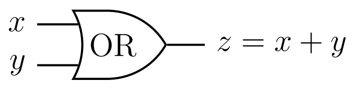
3.5 Logic Gates and Circuits

Definition 35 (Logic Circuit). *Logic circuits* (also called *logic networks*) are structures which are built up from certain elementary circuits called logic gates.

3.5.1 Logic Gates

There are three basic logic gates. The lines (wires) entering the gate symbol from the left are input lines and the single line on the right is the output line.

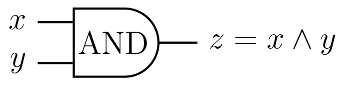
1. *OR Gate*: An OR gate has input x and y and output $z = x \vee y$ or $z = x + y$ where addition (or Join) is defined by the truth table. In this case the output $z = 0$ only when inputs $x = 0$ and $y = 0$.
   The symbol and the truth table for OR gate are shown in the diagram below:



x	y	$x + y$
1	1	1
1	0	1
0	1	1
0	0	0

Table 3.1: Truth table for OR gate

2. *AND gate*: In this gate the inputs are x and y and output is $x \wedge y$ or $x \cdot y$ or xy, where multiplication is defined by the truth table.

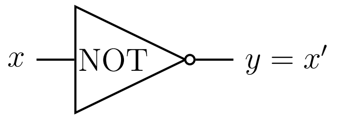


x	y	$x \wedge y$
1	1	1
1	0	0
0	1	0
0	0	0

Table 3.2: Truth table for AND gate

   Thus output is 1 only when $x = 1$, $y = 1$, otherwise it is zero. The AND gate may have more than two inputs. The output in such a case will be 1 if all inputs are 1.

3. *NOT Gate (inverter)*: The diagram below shows NOT gate with input x and output $y = x'$, where inversion, denoted by the prime, is defined by the truth table:



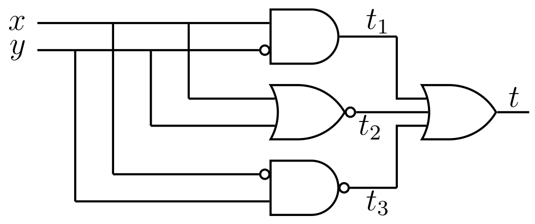
x	$y = x'$
1	0
0	1

Table 3.3: Truth table for NOT gate

   For example, if $x = 10101$, then output x' in NOT gate shall be $x' = 01010$.

Theorem 3.5.1. Logic circuits form a Boolean Algebra.

Problem 3.5.1. Express the output of the logic circuit below as a Boolean expression. (Here small circle represents complement(NOT))



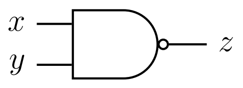
Solution. We note that

$$\begin{aligned} t_1 &= xy' \\ t_2 &= (x+y)' \\ t_3 &= (x'y)' \end{aligned}$$

and so we have

$$\begin{aligned} t &= t_1 + t_2 + t_3 \\ &= xy' + (x+y)' + (x'y)' \end{aligned}$$

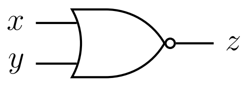
NAND and NOR gates are frequently used in computers.
NAND gate: It is equivalent to AND gate followed by a NOT gate. The symbol and the truth table for NAND gate are shown in the diagram below:



x	y	xy	$z = (xy)'$
1	1	1	0
1	0	0	1
0	1	0	1
0	0	0	1

Table 3.4: Truth table for NAND gate

Thus, the output of a NAND gate is 0 if and only if all the inputs are 1. *NOR gate*: This gate is equivalent to OR gate followed by a NOT gate. The symbol and the truth table for NAND gate are shown in the diagram below:



x	y	$x+y$	$(x+y)'$
1	1	1	0
1	0	1	0
0	1	1	0
0	0	0	1

Table 3.5: Truth table for NOR gate

Thus, the output of a NOR gate is 1 if and only if all the inputs are 0.

3.6 Boolean Function

We know that ordinary polynomials could produce functions by substitution. For example, the polynomial $xy + yz^3$ produces a function $f : \mathbb{R}^3 \to \mathbb{R}$ by letting $f(x, y, z) = xy + yz^3$. Thus, $f(3, 4, 2) = 3 \cdot 4 + 4 \cdot 2^3 = 44$. In a similar way, Boolean polynomials involving n variables produce functions from B_n to B.

Definition 36 (Boolean Function)**.** Let $(B, \cdot, +, ', 0, 1)$ be a Boolean algebra. A function $f : B_n \to B$ which is associated with a Boolean expression (polynomial) in n variables is called a Boolean function.

Thus, a Boolean function is completely determined by the Boolean expression $\alpha(x_1, x_2, \ldots, x_n)$ because it is nothing but the evaluation function of the expression. It may be mentioned here that every function $g : B_n \to B$ needs not be a Boolean function.

If we assume that the Boolean algebra B is of order 2^m for $m \geq 1$, then the number of function from B_n to B is greater than 2^{2n} showing that there are functions from B_n to B which are not Boolean functions. On the other hand, for $m = 1$, that is, for a two element Boolean algebra, the number of function from B_n to B is B^{2n} which is same as the number of distinct Boolean expression in n variable. Hence, every function from B_n to B in this case is a Boolean function.

Problem 3.6.1. Show that the following Boolean expression are equivalent to one-another. Obtain their sum-of-product canonical form.

(a) $(x + y)(x' + z)(y + z)$

(b) $(xz) + (x'y) + (yz)$

(c) $(x + y)(x' + z)$

(d) $xz + x'y$

Solution. The binary valuation of the expression are

x	y	z	$x+y$	$x'+z$	$y+z$	(a)	(c)	xz	$x'y$	yz	(b)	(d)
0	0	0	0	1	0	0	0	0	0	0	0	0
0	0	1	0	1	1	0	0	0	0	0	0	0
0	1	0	1	1	1	1	1	0	1	0	1	1
0	1	1	1	1	1	1	1	0	1	1	1	1
1	0	0	1	0	0	0	0	0	0	0	0	0
1	0	1	1	1	1	1	1	1	0	0	1	1
1	1	0	1	0	1	0	0	0	0	0	0	0
1	1	1	1	1	1	1	1	1	0	1	1	1

Since the values of the given Boolean expression are equal over every triple of the two element Boolean algebra, they are equal.

Two find the sum-of-product canonical (complete) form, we note that (d) is in sum-of-product form. Therefore, to find complete sum-of-product form, we have

$$\begin{aligned}
\text{(d)} &= (xz) + (x'z) \\
&= xz(y + y') + x'z(z + z') \\
&= xzy + xzy' + x'yz + x'yz'
\end{aligned}$$

3.7 Method to Find Truth Table of A Boolean Function

Consider a logic circuit consisting of 3 input devices x, y, z. Each assignment of a set of three bits to the input x, y, z yield an output bit for z. There are $2n = 2^3 = 8$ possible ways to assign bits to the input as follows:

$$000,\ 001,\ 010,\ 011,\ 100,\ 101,\ 110,\ 111.$$

The assumption is that the sequence of first bits is assigned to x, the sequence of second bits to y, and the sequence of third bits to z. Thus, the above set of inputs may be rewritten in the form

$$x = 00001111,\ y = 00110011,\ z = 01010101$$

These three sequences (of 8 bits) contain the eight possible combination of the input bits.
The truth table $T = T(L)$ of the circuit L consists of the output t that corresponds to the input sequences x, y, z.
The truth table is same as we generally have written in vertical columns. The difference is that here we write x, y, z and t horizontally.

Consider a logic circuit L with n input devices. There are many ways to form n input sequences $x_1, x_2, \ldots, x_n$ so that they contain 2^n different possible combinations of the input bits (Each sequence must contain 2^n bits).

The assignment scheme is:

x_1: Assign 2^{n-1} bits which are 0 followed by 2^{n-1} bits which are 1.

x_2: Assign 2^{n-2} bits which are 0 followed by 2^{n-2} bits which are 1.

x_3: Assign 2^{n-3} bits which are 0 followed by 2^{n-3} bits which are 1.

and so on.

The sequence obtained in this way is called "Special Sequence". Replacing 0 by 1 and 1 by 0 in the special sequences yields the complements of the special sequence.

Example. Suppose a logic circuit L has $n = 4$ input devices x, y, z, t. Then $2^n = 2^4 = 16$ bit special sequences for x, y, z, t are
$x = 0000000011111111$ ($2^3 = 8$ zeros followed by 8 ones)
$y = 0000111100001111$ ($2^{n-2} = 2^{4-2} = 4$ zeros followed by 4 ones)
$z = 0011001100110011$ ($2^{n-3} = 2^{4-3} = 2$ zeros followed by 2 ones)
$z = 0101010101010101$ ($2^{n-3} = 2^{4-4} = 2^0 = 1$ zeros followed by 1 ones)

3.7.1 Algorithm For Finding Truth Table For A Logic Circuit L Where Output T is Given by a Boolean Sum-Of-Product Expression in the Inputs

The input is a Boolean sum-of-products expression $t(x_1, x_2, \ldots)$.

Step 1: Write down the special sequences for the inputs $x_1, x_2, \ldots$ and their complements

Step 2: Find each product appearing in $t(x_1, x_2, \ldots)$ keeping in mind that $x_1, x_2, \cdots = 1$ is a position if and only if all $x_1, x_2, \ldots$ have 1 in the position.

Step 3: Find the sum t of the products keeping in mind that $x_1 + x_2 + \cdots = 0$ in a position if and only if all $x_1, x_2, \ldots$ have 0 in the position.

3.8 Representation of Boolean Functions using Karnaugh Map

Karnaugh Map is a graphical procedure to represent Boolean function as an "or" combination of minterms where minterms are represented by squares. This procedure is easy to use with functions $f : B_n \to B$, if n is not greater than 6. We shall discuss this procedure for $n = 2, 3$, and 4.

A Karnaugh map structure is an area which is subdivided into 2^n cells, one for each possible input combination for a Boolean function of n variables. Half of the cells are associated with an input value of 1 for one of the variables and the other half are associated with an input value of 0 for the same variable. This association of cell is done for each variable, with the splitting of the 2^n cells yielding a different pair of halves for each distinct variable.

Case of 1 variables: In this case, the Karnaugh map consists of $2^1 = 2$ squares.

$$\begin{array}{|c|c|}\hline 0 & 1 \\ \hline \multicolumn{1}{c}{x'} & \multicolumn{1}{c}{x}\end{array}$$

The variable is represented by the right square and its complement x' by the left square.

Case of 2 Variables: For $n = 2$, the Boolean function is of two variable, say x and y. We have $2^2 = 4$ squares, that is, a 2×2 matrix of squares. Each squares contain one possible input from B_2.

The variable x appears in the first row of the matrix as x' whereas x appears in the second row as x. Similarly, y appears in the first column as y' and as y in the second column.

$$\begin{array}{c|c|c|}\multicolumn{1}{c}{} & \multicolumn{1}{c}{0} & \multicolumn{1}{c}{1} \\ \cline{2-3} 0 & 00 & 01 \\ \cline{2-3} 1 & 10 & 11 \\ \cline{2-3}\end{array} \qquad \begin{array}{c|c|c|c}\multicolumn{1}{c}{} & \multicolumn{1}{c}{y'} & \multicolumn{1}{c}{y} & \\ \cline{2-3} x' & x'y' & x'y & \\ \cline{2-3} & xy' & xy & x \\ \cline{2-3}\end{array}$$

Figure 3.1: 2 variable Karnaugh Map

In this case, x is represented by the points in lower half of the map and y is represented by the points in the right half of the map.

Definition 37. Two fundamental products are said to be *adjacent* if they have the same variables and if they differ in exactly one literal. Thus, there must be an uncomplemented variable in one product which is complemented in the other.

For example, if $P_1 = xyz'$ and $P_2 = xy'z'$, then they are adjacent.
The sum of two such adjacent products will be a fundamental product with one less literal.
For example, in the case of above-mentioned adjacent products,

$$P_1 + P_2 = xyz' + xy'z' = xz'(y + y') = xz'(1) = xz'$$

We note that two squares in Karnaugh map above are adjacent if and only if squares are geometrically adjacent, that is, have a side in common.

We know that a complete sum-of-products Boolean expression $E(x, y)$ is a sum of minterms and hence can be represented in the Karnaugh map by placing checks in the appropriate square. A prime implicant of $E(x, y)$ will be either a pair of adjacent squares in E or an isolated square (a square which is not adjacent to other square of $E(x, y)$). A minimal sum of products for $E(x, y)$ will consists of a minimum number of a prime implicants which cover all the square of $E(x, y)$.

Problem 3.8.1. Find the prime implicants and a minimal sum-of-products form from each of the following complete sum-of-products Boolean expression:

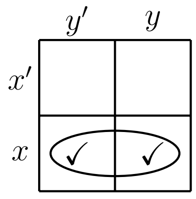
(a) $E_1 = xy + xy'$

(b) $E_2 = xy + x'y + x'y'$

(c) $E_3 = xy + x'y'$

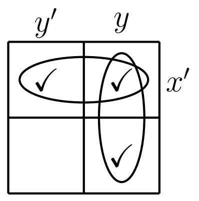
Solution.

(a) The Karnaugh map for E_1 is



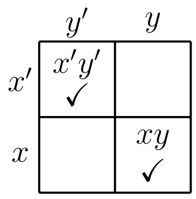
Check the squares corresponding to xy and xy'. We note that E_1 consists of one prime implicant, the two adjacent square designated by the loop. The pair of adjacent square represents the variable x. So x is the only prime implicant of E_1. Consequently, $E_1 = x$ is its minimal sum.

(b) The Karnaugh map for E_2 is



Check the squares corresponding to xy, $x'y$, $x'y'$. The expression E_2 contains two pairs of adjacent squares (designated by two loops) which include all the squares of E_2. The vertical pair represents y and the horizontal pair x'. Hence, y and x are the prime implicants of E_2. Thus, $E_2(x, y) = x' + y$ is minimal sum.

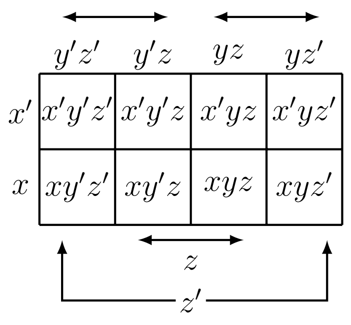
(c) The Karnaugh map for E_3 is



Check (tick) the squares corresponding to xy and $x'y'$. The expression E_3 consists of two isolated squares which represent xy and $x'y'$. Hence xy and $x'y'$ are the prime implicants of E_3 and so $E_3 = xy + x'y'$ is its minimal sum.

Case of 3 variables: We now turn to the case of a function $f : B_3 \to B$ which is function of x, y and z. The Karnaugh map corresponding to Boolean expression $E(x, y, z)$ is shown in the diagram below:

	00	01	11	10
0	000	001	011	010
1	100	101	111	110



Here x, y, z are respectively represented by lower half, right half and middle two quarters of the map.
Similarly, x', y',z' are respectively represented by upper half, left half and left and right quarter of the map.

Definition 38. By a Basic Rectangle in the Karnaugh map with three variables, we mean a square, two adjacent squares or four squares which form a one-by four, or a two by-two rectangle. These basic rectangles correspond to fundamental products of three, two and one literal respectively.

Further, the fundamental product represented by a basic rectangle is the product of just those literals that appear in every square of the rectangle.

Let a complete sum of products Boolean expression $E(x, y, z)$ is represented in the Karnaugh map by placing checks in the appropriate squares. A prime implicant of E will be a maximal basic rectangle of E, i.e., a basic rectangle contained in E which is not contained in any larger basic rectangle in E.

A minimal sum-of-products form for E will consist of a minimal cover of E, i.e., a minimal number of maximal basic rectangles of E which together include all the square of E.

Problem 3.8.2. Find the prime implicants and a minimal sum-of-products form for each of the following complete sum-of-products Boolean expressions:

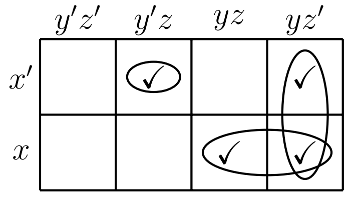
(a) $E_1 = xyz + xyz' + x'yz' + x'y'z.$

(b) $E_2 = xyz + xyz' + xy'z + x'yz + x'y'z.$

(c) $E_3 = xyz + xyz' + x'yz + x'y'z.$

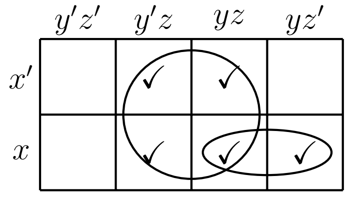
Solution.

(a) The Karnaugh map for E_1 is



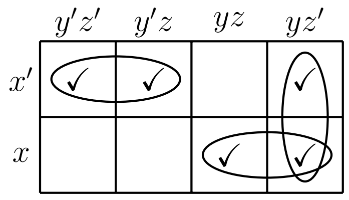
We check the four squares corresponding to four summands in E_1. Here E_1 has three prime implicants (maximal basic rectangles) which are encircled. These are xy, yz' and $x'y'z$. All three are needed to cover E_1. Hence, minimal sum for E_1 is $E_1 = xy + yz' + x'y'z$.

(b) The Karnaugh map for E_2 is



Check the squares corresponding to the five summands. E_2 has two prime implicants which are circled. One is the two adjacent squares which represent xy, and the other is the two-by-two square which represents z. Both are needed to cover E_2 so the minimal sum for E_2 is $E_2 = xy + z$.

(c) The Karnaugh map for E_3 is



Check the squares corresponding to the five summands. Here E_3 has three prime implicants xy, yz', $x'y'$. All these are needed in a minimal cover of E_3. Hence, E_3 has minimal sum as $E_3 = xy + yz' + x'y'$.

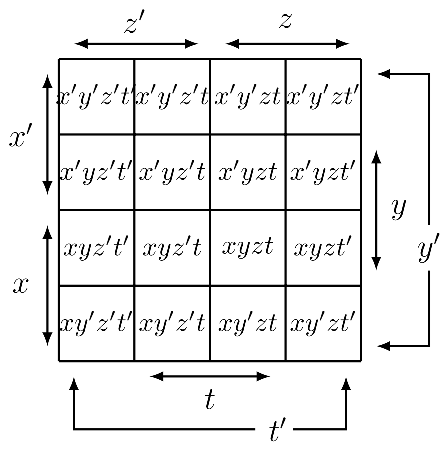
Remark. To find the fundamental product represented by a basic rectangle, find literals which appear in all the squares of the rectangle.

Case of 4 variables: We consider a Boolean function $f : B_4 \to B$, considered as a function of x, y, z and t. Each of the 16 squares (2^4) corresponds to one of the minterms with four variables.

$$xyzt,\ xyzt', ;x'yz't$$

We consider first and last columns to be adjacent, and first and last rows to be adjacent, both by Wrap around, and we look for rectangles with sides of length some power of 2, so the length is 1, 2 or 4. The expression for such rectangles is given by intersecting the large labelled rectangles.

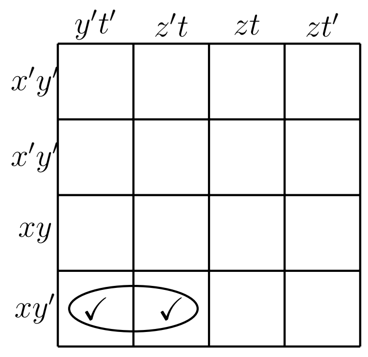
	00	01	11	10
00	0000	0001	0011	0001
01	0100	0101	0111	0110
11	1100	1101	1111	1101
10	1000	1001	1011	1010



	z'	z'	z	z
x'	$x'y'z't'$	$x'y'z't$	$x'y'zt$	$x'y'zt'$
x'	$x'yz't'$	$x'yz't$	$x'yzt$	$x'yzt'$
x	$xyz't'$	$xyz't$	$xyzt$	$xyzt'$
x	$xy'z't'$	$xy'z't$	$xy'zt$	$xy'zt'$

A basic rectangle in a four variable Karnaugh map is a square, two adjacent squares, four squares which form one-by-four or two by two rectangle or eight square squares which form a two by four rectangles. These rectangles correspond to fundamental product with four, three, two and one literal respectively. Maximal basic rectangles are prime implicants.

Problem 3.8.3. Find the fundamental product P represented by the basic rectangle in the Karnaugh map given below:



Solution. We find the literals which appear in all the squares of the basic rectangle. Then P will be the product of such literals.
Here x, y', z appear in both squares. Hence,

$$P = xy'z'$$

is the fundamental product represented by the basic rectangle in question.

Chapter 4

Finite State Machines

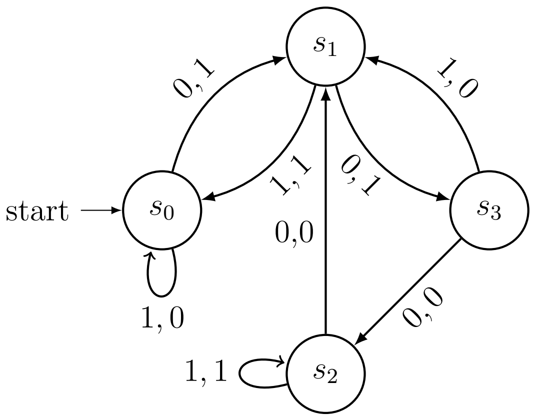
4.1 Finite state machines

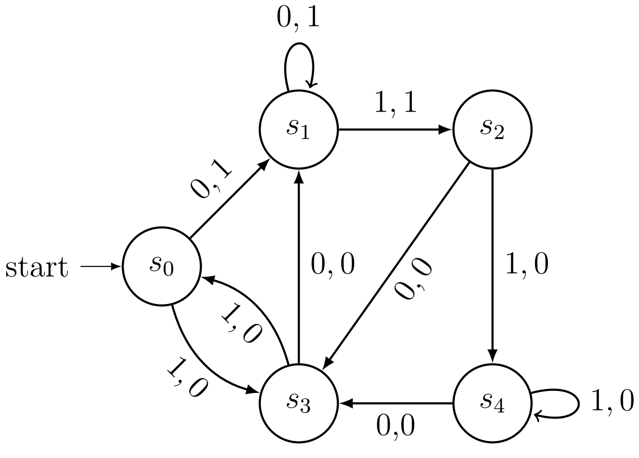
A finite state machine $M = (S, I, O, f, g, s_0)$ consists of a finite set S of states, a finite input alphabet I, a finite output alphabet O, a transition function f that assigns to each state and input pair a new state, an output function g that assigns to each state and input pair an output and an initial state s_0.

- We can use a state table to represent the values of the transition function f and the output function g for all pairs of states and input.

- A state diagram is a directed graph with labeled edges that represents a finite state machine.

	f		g	
	Input		Input	
State	0	1	0	1
s_0	s_1	s_0	1	0
s_1	s_3	s_0	1	1
s_2	s_1	s_2	0	1
s_3	s_2	s_1	0	0





	f		g	
	Input		Input	
State	0	1	0	1
s_0	s_1	s_3	1	0
s_1	s_1	s_2	1	1
s_2	s_3	s_4	0	0
s_3	s_1	s_0	0	0
s_4	s_3	s_4	0	0

- If the input string is 101011, the output string is 001000.

- Let $M = (S, I, O, f, g, s_0)$ be a finite state machine and $L \subseteq I^*$ (the set of all input string over I). We say that M recognizes (or accepts) L if an input string x belongs to L if and only if the last output bit produced by M when given x as input as a 1.

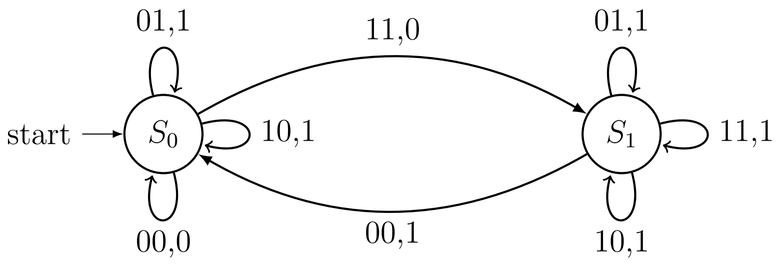
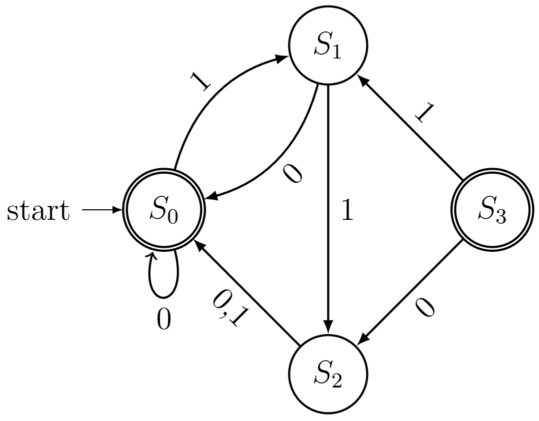


Figure 4.1: A finite state machine for binary addition

4.2 Finite state machines with no output

A finite state automaton $M = (S, I, f, s_0, F)$ consists of a finite set S of states, a finite input alphabet I, a transition function f that assigns a next state to every pair of state and input (so that $f : S \times I \to S$), an initial or state s_0 and a subset F if S consisting of final (or accepting states).

	f	
	Input	
State	0	1
s_0	s_0	s_1
s_1	s_0	s_2
s_2	s_0	s_0
s_3	s_2	s_1



$M = (S, I, f, s_0, F)$, $S = \{s_0, s_1, s_2, s_3\}$, $I = \{0, 1\}$, $F = \{s_0, s_3\}$ and f is given in the table.

- A string x is said to be recognized or accepted by the machine $M = (S, I, f, s_0, F)$ if it takes the initial state s_0 to a final state, that is, $f(s_0, x)$ is a state in F.

- The language recognized or accepted by the machine M, denoted by $L(M)$, is the set of all strings that are recognized by M.

- Two finite state automata are called equivalent if they recognize the same language.

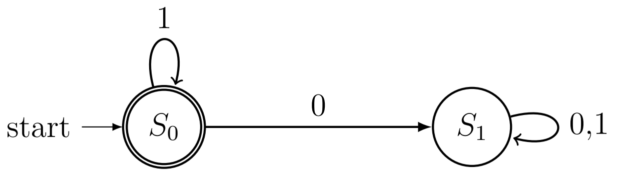


Figure 4.2: M_1

$$L(M_1) = \{1^n \mid n = 0, 1, 2, \dots\}$$

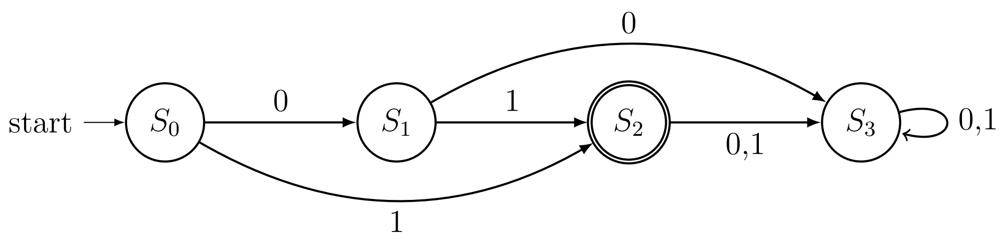


Figure 4.3: M_2

$$L(M_2) = \{1, 01\}$$

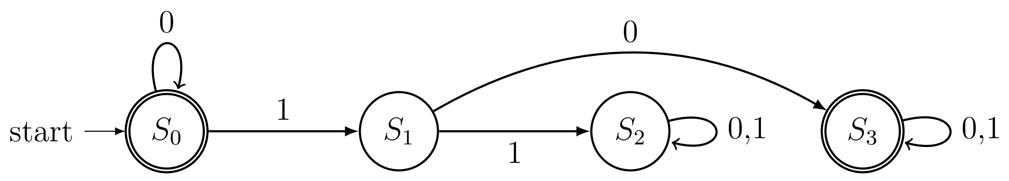


Figure 4.4: M_3

$$L(M_3) = \{0^n,\ 0^n10x \mid n = 0, 1, 2, \ldots \text{ and } x \text{ is any string}\}$$

Problem 4.2.1. Construct deterministic finite set automata that recognize each of these languages:

(a) The set of bit strings that begin with two 0's.

(b) The set of bit strings that contains two consecutive 0's.

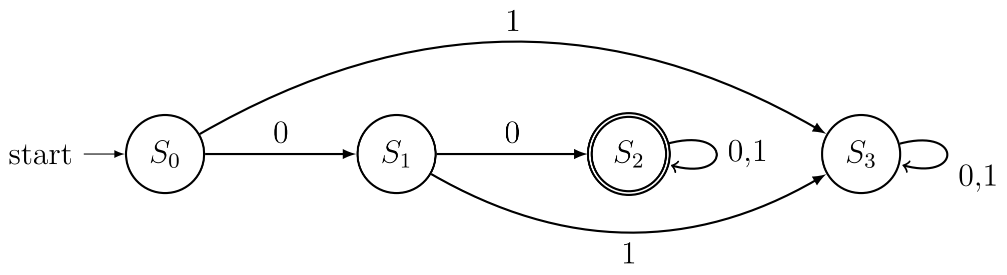
(c) The set of bit strings that do not contain two consecutive 0's.

(d) The set of bit strings that end with two consecutive 0's.

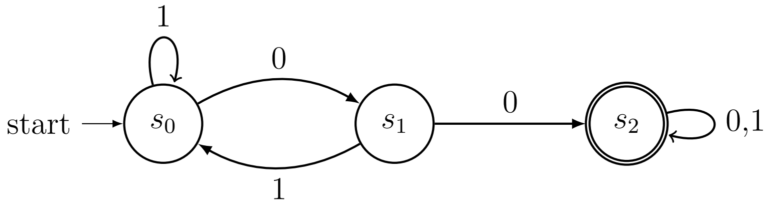
(e) The set of bit strings that contains at least two 0's.

Solution.

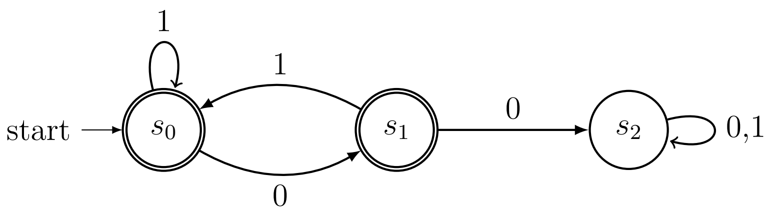
(a)



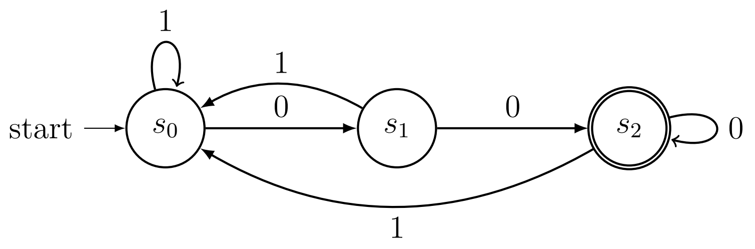
(b)



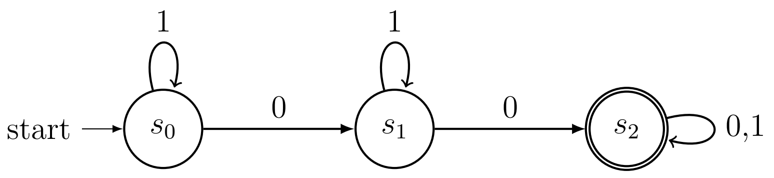
(c)



(d)



(e)



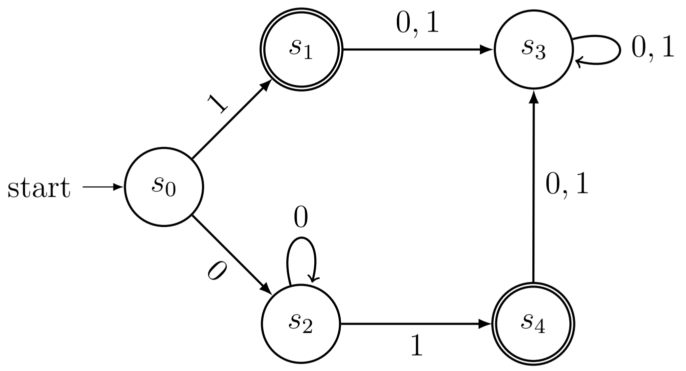


Figure 4.5: M_0

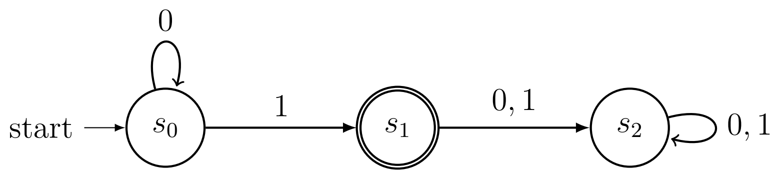


Figure 4.6: M_1

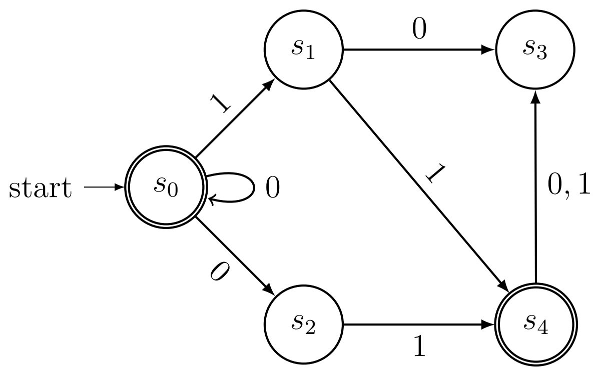
Problem 4.2.2. Show that M_0 and M_1 are equivalent.

Solution. For a string x to be recognized by M_0, x must take us from s_0 to the final state s_1 or the final state s_4.

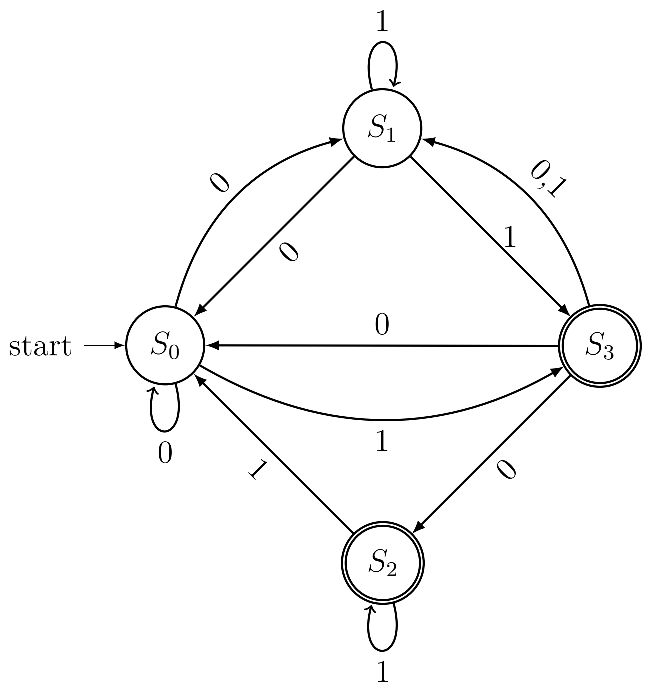
The only string that takes us from s_0 to s_1 is the string 1.

The strings that take us from s_0 to s_4 are those strings that begin with a 0, which takes us from s_0 to s_2, followed by zero or more additional zero which keep the machine in state s_2 followed by a 1, which takes us from s_2 to the final state s_4.

H.W. Nondeterministic finite state automata



	f	
	Input	
State	0	1
s_0	s_0, s_2	s_1
s_1	s_3	s_4
s_2		s_4
s_3	s_3	
s_4	s_3	s_3



	f	
	Input	
State	0	1
s_0	$s_0,\ s_1$	s_3
s_1	s_0	$s_1,\ s_3$
s_2		$s_0,\ s_2$
s_3	$s_0,\ s_1,\ s_2$	s_1