B.E. (TU, PU, PoU, KU), B.Sc. CSIT

Insights on DISCRETE STRUCTURE

Er. Shree Krishna Sulu

CONTENTS

Chapter – 1

LOGIC, INDUCTION AND REASONING

1.1 Discrete Mathematics 1
1.2 Logic, Induction and Reasoning 3
1.3 Proposition and Truth Function 6
1.4 Propositional Logic 7
1.5 Expressing Statements in Propositional Logic 16
1.6 The Predicate Logic 17
1.7 Arguments, Argument Forms and Validity 24
1.8 Rules of Inference 25
1.9 Formal and Informal Proof 28
1.10 Informal Deduction in Predicate Logic 33
1.11 Informal Proofs and Formal Proofs 36
1.12 Elementary Induction and Complete Induction 47
1.13 Consistency and Completeness of the System 55
1.14 Method of Tableaux 55
SOLVED NUMERICALS 63

Chapter – 2

FINITE STATE AUTOMATA

2.1 Sequential Circuits and Finite state Machine 73
2.2 Finite state Automata 76
2.3 Language and Grammar 80
2.4 Language and Automata 86
2.5 Non-deterministic Finite State Automata 89
2.6 Regular Expression and its Characteristics 93
SOLVED NUMERICALS 101

Chapter – 3

RECURRENCE RELATION

3.1 Recursive Definition of Sequences 110
3.2 Solution of Linear Recurrence Relations 112
3.3 Solution of Nonlinear Recurrence Relations 121
3.4 Application to Algorithm Analysis 127
SOLVED NUMERICALS 130

Chapter – 4

GRAPH THEORY

4.1 Graph 136
4.2 Undirected and Directed Graphs 138
4.3 Graphical Terms and Terminologies 139
4.4 Bipartite Graphs, Planar Graphs, Regular Graphs 153
4.5 Walk Paths, Circuits, Components 160
4.6 Connectedness Algorithm 162
4.7 Cut Sets and Cut Vertices 163
4.8 Eulerian Graph 165
4.9 Hamiltonian Graph 167
4.10 A shortest Path Algorithm 168
4.11 Tree as a Directed Graph 175
4.12 Binary Search Tree 181
4.13 Prefix Codes 183
4.14 Spanning Tree 186
4.15 Graph Coloring 198
4.16 Max Flow and Min Cut 200
SOLVED NUMERICALS 204

Logic, Induction, and Reasoning

1.1 Discrete Mathematics

Discrete mathematics is one of the branches of mathematics that deals with discrete objects and structures. *Discrete objects* are distinct objects (separated or not connected or opposite of continuous objects) with separate value such as tree, people, house, etc. *Discrete structure* is the abstract mathematical structure used to represent discrete objects and relationships between the objects e.g., set, relations, graphs, trees, and finite state machines. Discrete structures are structures that are used in describing discrete mathematics.

Discrete mathematics is finite mathematics contrast to continuous mathematics. In finite mathematics the values are separate (integer) whereas in continuous mathematics values are in range (real number). The objects are continuous and vary smoothly in continuous mathematics, e.g., in calculus.

Discrete mathematics is the mathematical language of computer science as information is stored and manipulated in a discrete form by computing machines. It is mathematics of integers and collection of objects that triggers the operation of digital computer. Discrete mathematics is widely used in all fields of computer science for programming and reasoning about data structures, algorithms and complexity. It enables to formulate problems, precisely, solve the problems, apply formal proof techniques, and explain their reasoning clearly. Discrete mathematics deals with the questions like:

- In how many ways the valid password can be chosen?
- What is the probability of winning a lottery?
- How can a list of integers be sorted in increasing order?
- How the shortest path between two cities can be determined?

Uses of Discrete Math in Computer Science

Discrete mathematics is widely used in various field of computer science like in Networking, Database, Image Processing, Programming Languages, Compilers & Interpreters, Software Engineering, Artificial Intelligence, Computer Architecture, Operating Systems, Security & Cryptography, Advanced Algorithms & Data Structures, and Graphics & Animation.

Reason Behind Studying Discrete Mathematics:

There are two reasons behind studying discrete mathematics:

i. To develop mathematical maturity

ii. To enter to more advanced mathematical sciences

i. To develop mathematical maturity:

The study of discrete mathematics is necessary to upgrade the ability to understand and create mathematical arguments. This skill is important to get very far in the studies in mathematical sciences.

ii. To enter to more advanced mathematical sciences

Discrete mathematics is the gateway to more advanced courses in all parts of mathematical sciences. Discrete mathematics provides mathematical foundations for many computer science courses including data structure algorithms, database theory, automata theory, formal language, compiler theory, computer security, operating systems, math courses based on the material studied in discrete mathematics including logic, set theory, number theory, liner algebra, abstract algebra, combinatorics, graph theory, probability theory. Discrete math contains necessary mathematical background for solving problems in operations research, chemistry, engineering, biology. Discrete mathematics develops mathematical reasoning and problem solving.

Aspects of Discrete Mathematics:

1. Logical thinking (propositional, predicate logic, proof method, prolog)

2. Relational thinking (graph theory, sets, functions, database relations)
3. Recursive thinking (recurrence relations, recursive data structures (binary search trees))
4. Quantitative thinking (counting, combinations, permutations, pigeonhole, big-Oh notation)
5. Analytical thinking (apply previous tools, algorithm complexity, analysis of algorithms)
6. Applied thinking (problem solving skills, modeling)

Discrete Mathematics is essential to Computer Science. The various aspects of discrete mathematics form the foundation for modeling computing structures, designing programs and algorithms, reasoning about programs and algorithms, solving real-world problems using the computer in particular, solving problems in information technology i.e., problems involving computers, communication, information.

1.2 Logic, Induction and Reasoning

1.2.1 Logic

Logic is a particular way of thinking, especially one that is reasonable and based on good judgment. More precisely, logic is the study of reasoning or method of reasoning. Logic is a language for reasoning. It is a collection of rules that is used in logical reasoning. Logic distinguishes valid and invalid arguments. Logic is

- Crucial for mathematical reasoning
- Important for program design
- Used for designing electronic circuitry

1.2.1.1 Types of logic

1. Propositional logic (logic of sentences)

Propositional logic is the study of propositions, where a proposition is a statement that is either true or false. Propositional logic may be used to encode simple arguments that are expressed in natural language, and to determine their

validity. The validity of an argument may be determined from truth tables, or using inference rules to establish the conclusion via deductive steps.

2. **Predicate logic (logic of objects)**

Predicate logic can represent quantity that allows complex facts about the world to be represented, and new facts may be determined via deductive reasoning. Predicate calculus includes predicates, variables and quantifiers, and a *predicate* is a characteristic or property that the subject of a statement can have. The universal quantifier is used to express a statement such as that all members of the domain of discourse have property P, and the existential quantifier states that there is at least one value of x has property P.

1.2.2 Induction

Deduction is to find the value from known to known. E.g, Man is mortal. Ram is a man. So, Ram is mortal. It is top to bottom, general to specific approach. It is guaranteed.

Induction is to find the value from known to unknown. It is bottom to top, specific to general approach. E.g., "The coin I pulled from the bag is a penny. That coin is a penny. A third coin from the bag is a penny. Therefore, all the coins in the bag are pennies."

Induction is the act of bringing forward something (such as facts or particulars). More Precisely induction is the act of causing or bringing on or about.

1.2.3 Reasoning

Reason is the action of thinking about something in a logical, sensible way. Logical reasoning is the term for different types of reasoning like deductive reasoning, inductive reasoning and abductive reasoning. They are based on deduction, induction and abduction respectively.

Deductive reasoning

Deductive reasoning is a basic form of valid reasoning. Deductive reasoning, or deduction, starts out with a general statement, or hypothesis, and examines the possibilities to reach a

specific, logical conclusion. Deductive reasoning usually follows steps. First, there is a premise, then a second premise, and finally an inference. For example, the premise "Every A is B" could be followed by another premise, "This C is A." Those statements would lead to the conclusion "This C is B." Syllogisms are considered a good way to test deductive reasoning to make sure the argument is valid.

For example, "All men are mortal. Hari is a man. Therefore, Hari is mortal." For deductive reasoning to be sound, the hypothesis must be correct. It is assumed that the premises, "All men are mortal" and "Hari is a man" are true. Therefore, the conclusion is logical and true. In deductive reasoning, if something is true of a class of things in general, it is also true for all members of that class.

Inductive reasoning

Inductive reasoning is the opposite of deductive reasoning. Inductive reasoning makes broad generalizations from specific observations. Basically, there is data, then conclusions are drawn from the data. In inductive inference, we go from the specific to the general. We make many observations, discern a pattern, make a generalization, and infer an explanation or a theory. An example of inductive logic is, "The coin I pulled from the bag is a penny. That coin is a penny. A third coin from the bag is a penny. Therefore, all the coins in the bag are pennies."

Inductive reasoning has its place in the scientific method. Scientists use it to form hypotheses and theories. Deductive reasoning allows them to apply the theories to specific situations.

Abductive reasoning

Another form of scientific reasoning that doesn't fit in with inductive or deductive reasoning is abductive. Abductive reasoning usually starts with an incomplete set of observations and proceeds to the likeliest possible explanation for the group of observations. It is based on making and testing hypotheses using the best information available. It often entails making an educated guess after observing a phenomenon for which there is no clear explanation.

For example, a person walks into their living room and finds torn up papers all over the floor. The person's dog has been alone in the room all day. The person concludes that the dog tore up the papers because it is the most likely scenario. Now, the person's sister may have brought by his niece and she may have torn up the papers, or it may have been done by the landlord, but the dog theory is the more likely conclusion. Abductive reasoning is useful for forming hypotheses to be tested. Abductive reasoning is often used by doctors who make a diagnosis based on test results and by jurors who make decisions based on the evidence presented to them.

1.3 Proposition and Truth Function

1.3.1 Proposition

Proposition is a declarative sentence that is either true or false but not both. True value is denoted by T and false value is denoted by F. Sometimes it is denoted by 1 and 0 in bit operation. Exclamations and commands are not propositions.

Propositions are denoted by a lower case letter, like, p, q, r, s, or t. The variables p, q, r, s, or t are Boolean variable (or variables).

The following sentences are propositions

1. Today is Monday.
2. Sun rises from the east.
3. Sun rises from the west.
4. Pokhara is the capital of Nepal.
5. $1 + 2 = 3$
6. $1 + 2 > 5$
7. If x is an integer, then x^2 is a positive integer

The following sentences are not propositions

1. $x + 3 = 9$ (x is unknown)
2. Let him go! (Exclamation)
3. Do your work. (Imperative sentence)
4. Obama was a great president of USA. (Opinion)

5. Close the window! (Command)
6. What is your mobile number? (Interrogation)

Basicaly, proposition is denoted by variable

p : Today is Sunday

q : $1 + 1 = 3$

Atomic Proposition

A proposition consisting of only single proposition variable or a single propositional constant is called an atomic or primary or primitive or simply proposition, that can't be further subdivided.

Compound propositions:

A proposition obtained from the combination of two or more proposition by means of logical operators or connectives of two or more proposition or by negating the single proposition is referred to molecular or complete or compound proposition.

1.3.2 Truth Value, Truth Table, and Truth Function

Truth Value:

The truthness or falsehood of a proposition is called its *truth value*. We denote T for True and F for False.

Truth Table:

Table of all possible combination of variables and their result that shows the truth values of a proposition is called *truth table*.

Truth Function

A *truth function* is a function that accepts truth values as input and produces a truth value as output. Propositional logic is an example of truth function, wherein a compound statement is constructed by one or two statements connected by a logical connective. The compound statement is called a truth function, and the logical connective is said to be truth functional.

1.4 Propositional Logic

Propositional logic is a system based on propositions. Propositional logic, also known as sentential logic and statement logic, is the branch of logic that studies ways of joining and/or

modifying entire propositions, statements or sentences to form more complicated propositions, statements or sentences, as well as the logical relationships and properties that are derived from these methods of combining or altering statements.

1.4.1 Connectives

Connectives are those notations which are used to get new propositions from given propositions. The common used connectives are as follows:

1. Conjunction (AND) denoted by $\wedge$
2. Disjunction (OR) denoted by $\vee$
3. Negation (NOT) denoted by $\neg$ or $\sim$
4. Implication (if ...then...) denoted by $\rightarrow$
5. Double implication (if and only if) denoted by $\leftrightarrow$
6. Exclusive denoted by $\oplus$
7. NOR (not OR) denoted by $\downarrow$
8. NAND (not AND) denoted by $\uparrow$

1. Conjunction ($\wedge$)

Let p and q be two propositions. The proposition 'p and q' is denoted by $p \wedge q$ and is true only when p and q are true.

Truth table of $p \wedge q$

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

Let p is proposition 'I am in college'. q is proposition 'It is cold'.

$p \wedge q$ is I am in college and it is cold.

Sometime **but** is used instead of **and**.

E.g., The sun is shining but it is raining.

2. Disjunction ($\vee$)

Let p and q be two propositions. The proposition 'p or q' is denoted by $p \vee q$ and it is false when both p and q are false otherwise it is true.

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

p : Today is Wednesday

q : It is raining.

3. Negation ($\neg$)

Let p be a proposition. The statement 'It is not the case p' is the negation of p. It is denoted by $\neg p$.

P	$\neg p$
T	F
F	T

p : Today is Monday.

$\neg p$: Today is not Monday.

$\neg p$: It is not the case that today is Monday.

4. Implication ($\rightarrow$) (conditional implication)

Let p and q be two propositions. The conditional statement $p \rightarrow q$ is the proposition 'if p then q'.

The proposition $p \rightarrow q$ is false only when p is true and q is false otherwise true.

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

If I become HOD, I will give laptop to every student.

p: I become HOD.

q: I will give laptop to every student.

In this example, the first part, p:If I become HOD, is called hypothesis or antecedent or premise or sufficient condition.

The second part, q: I will give laptop to every student, is called conclusion or consequence or necessary condition.

The following statements are the ways to express the implication or conditional statement.

1. If p then q
2. If p, q
3. p is sufficient for q
4. q if p
5. q when p
6. p implies q
7. p only if q
8. a necessary condition for p is q
9. a sufficient condition for q is p
10. q whenever p
11. q is necessary for p.
12. q follows from p.
13. q unless ¬p.

Converse

The converse is computed by interchanging the hypothesis and conclusion. If we have conditional statement $p \rightarrow q$ then the converse is $q \rightarrow p$

Inverse

The inverse of conditional statement is the negation of both the hypothesis and the conclusion. If we have conditional statement $p \rightarrow q$ then the inverse is $\neg p \rightarrow \neg q$

Contrapositive

The contrapositive is computed by interchanging the hypothesis and conclusion of inverse. If we have conditional statement $p \rightarrow q$ then the inverse is $\neg q \rightarrow \neg p$

Among these three conditional statement formed from $p \rightarrow q$, only contra positive is logically equivalent to $p \rightarrow q$. (i.e. $\neg q \rightarrow \neg p$ has the same truth value as $p \rightarrow q$)

Q ***Show that implication and its contra positive are logically equivalent.***

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

5. **Bi-conditional or bi-implication (↔)**

Let p and q be two propositions, the bi-conditional statement $p \leftrightarrow q$ is the proposition 'p if and only if q'. Thebi-conditional statement $p \leftrightarrow q$ is true when p and q have same truth values otherwise false.

$p \leftrightarrow q$ is expressed as

- p if and only if q
- p iff q
- p is necessary and sufficient condition for q
- if p then q and conversely

$(p \leftrightarrow q) \equiv (p \rightarrow q) \wedge (q \rightarrow p)$

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

Example, you can take the flight if and only if you buy a ticket.

6. **Exclusive OR (⊕)**

Let p and q be propositions, $p \oplus q$ is true when p is true and q is false or q is true and p is false. It is false when both p and q are false or both are true.

Example, Student who have taken calculus or computer science but not both can enroll in this class.

P	q	p⊕q
T	T	F
T	F	T
F	T	T
F	F	F

7. **NOR (not OR) (↓)**

NOR is negation of OR. For two propositions p and q,

$p \downarrow q = \neg (p \vee q)$

8. **NAND (not AND)(↑)**

NAND is negation of AND. For two propositions p and q,

$p \uparrow q = \neg(p \wedge q)$

Precedence of operator

1. ¬
2. ∧
3. ∨
4. →
5. ↔

$\neg p \wedge q$ means $(\neg p) \wedge q$

$p \wedge q \rightarrow r$ means $(p \wedge q) \rightarrow r$

Q) Construct the truth table of compound proposition $(\neg p \wedge q) \rightarrow (p \wedge q)$

P	q	¬p	¬p∧q	(p∧q)	(¬p∧q) → (p∧q)
T	T	F	F	T	T
T	F	F	F	F	T
F	T	T	T	F	F
F	F	T	F	F	T

Logic and bit operation

X	y	x∧y	x∨y	x → y	x↔y
0	0	0	0	1	1
0	1	0	1	1	0
1	0	0	1	0	0
1	1	1	1	1	1

Q. Find bitwise OR, bitwise XOR of bit strings 0110110110 and 1100011101. ***[Do it yourself]***

1.4.2 Tautology, Contradiction and Contingency

Tautology

A compound proposition that is always true no matter what the truth values of the proposition occur in it, is called tautology.

e.g., $p \vee \neg p$, $\neg p \wedge (p \vee q) \rightarrow q$, $[(p \rightarrow q) \wedge p] \rightarrow q$

Contradiction

A compound proposition that is always false no matter what the truth values of the proposition occur in it, is called contradiction.

e.g., $p \wedge \neg p$, $(p \vee q) \wedge [(\neg p) \wedge (\neg q)]$

Contingency

A compound proposition that is neither tautology nor contradiction is called contingency.

e. g., $(p \wedge \neg q) \rightarrow (p \wedge q)$, $(p \vee q) \wedge (\neg p)$

Logical equivalence (≡)(⇔)

Two statements x and y are logically equivalent if any of the following two condition holds

i. The truth table of each statement have same values

ii. The biconditional statement x↔y is tautology

The compound propositions that have same truth values in all possible cases are called logically equivalent.

Let p and q be propositions. They are logically equivalent if $p \leftrightarrow q$ is tautology.

Q. ***Show following proposition are logically equivalent***

1. $\neg(p \vee q)$ and $\neg p \wedge \neg q$
2. **$p \rightarrow q$ and $\neg p \vee q$**
3. $p \vee (q \wedge r)$ and $(p \vee q) \wedge (p \vee r)$
4. $\neg(p \vee (\neg p \wedge q))$ and $\neg p \wedge \neg q$

$(p \rightarrow q) \Leftrightarrow (\neg p \vee q)$

p	Q	$p \rightarrow q$	$\neg p$	$\neg p \vee q$
T	T	T	F	T
T	F	F	F	F
F	T	T	T	T
F	F	T	T	T

$p \leftrightarrow q \equiv (p \rightarrow q) \wedge (q \rightarrow p)$

p	q	$p \leftrightarrow q$	$p \rightarrow q$	$q \rightarrow p$	$(p \rightarrow q) \wedge (q \rightarrow p)$
T	T	T	T	T	T
T	F	F	F	T	F
F	T	F	T	F	F
F	F	T	T	T	T

Logical Equivalences

S.N.	Equivalence	Name
1.	$p \wedge T \Leftrightarrow p$ $p \vee F \Leftrightarrow p$	Identity law
2.	$p \vee T \Leftrightarrow T$ $p \wedge F \Leftrightarrow F$	Domination law
3.	$p \vee p \Leftrightarrow p$ $p \wedge p \Leftrightarrow p$	Idempotent law
4.	$\neg(\neg p) \Leftrightarrow p$	Double negation law
5.	$p \vee q \Leftrightarrow q \vee p$ $p \wedge q \Leftrightarrow q \wedge p$	Commutative law
6.	$(p \vee q) \vee r \Leftrightarrow p \vee (q \vee r)$ $(p \wedge q) \wedge r \Leftrightarrow p \wedge (q \wedge r)$	Associative law
7.	$p \vee (q \wedge r) \Leftrightarrow (p \vee q) \wedge (p \vee r)$ $p \wedge (q \vee r) \Leftrightarrow (p \wedge q) \vee (q \wedge r)$	Distributive law
8.	$\neg(p \wedge q) \Leftrightarrow (\neg p \vee \neg q)$ $\neg(p \vee q) \Leftrightarrow \neg p \wedge \neg q$	De-morgan's law
9.	$p \vee (p \wedge q) \Leftrightarrow p$ $p \wedge (p \vee q) \Leftrightarrow p$	Absorption law
10.	$p \vee \neg p \Leftrightarrow T$ $p \wedge \neg p \Leftrightarrow F$	Negation law

Useful logical equivalence

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

Q) ***Show that***

1. ***$\neg(p \vee (\neg p \wedge q)) \Leftrightarrow (\neg p \wedge \neg q)$ (to prove is logically equivalent)***
2. ***$p \wedge q) \rightarrow (p \vee q)$ is tautology***
3. ***$(p \rightarrow q) \wedge (p \rightarrow r) \equiv p \rightarrow (q \wedge r)$***

Solutions:

1. $\neg(p \vee (\neg p \wedge q))$

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

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

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

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

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

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

2. $(p \wedge q) \rightarrow (p \vee q)$

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

$= \neg p \vee \neg q \vee p \vee q$

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

$= T \vee T$

$= T$

1.5 Expressing Statements in Propositional Logic

Translate the sentence in propositional logic

Q. ***It snows*** ***whenever*** ***the wind blows from the northeast.***

Conclusion ↓ Hypothesis ↓

If the wind blows from the north-east then it snows

p : The wind blows from the northeast

q : It snows

$p \rightarrow q$

Q. ***To get tenure as a professor, it is sufficient to be world famous.***

If you are world famous then you can get tenure as a professor.

p: You are world famous

q : You can get tenure as a professor

$p \rightarrow q$

Q. ***The automated reply cannot be sent when the file system is full. [conclusion is statement before when]***

p : The automated reply can be sent.

q : The file system is full.

$q \rightarrow \neg p$

Q. ***Ram is either smart or honest but Ram is not honest if he is smart.***

P: Ram is smart

q: Ram is honest

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

Q. ***Ram is smart is necessary and sufficient for Ram to be honest.***

P: Ram is smart

q: Ram is honest

$p \leftrightarrow q$

1.6 The Predicate Logic

Predicates and Quantifiers

A *predicate* is a declarative sentence whose truthiness or falsehood depends on one or more variables. Predicate is propositional function

"x is greater than 4"

Subject: x

Predicate: "is greater than 4" (property of statement)

P(x) :"x greater than 4"

where x is the variable and p is the predicate "is greater than 4"

Assign a value to x and P(x) becomes a proposition.

P(x) is also called the value of propositional function at x.

P(x) denotes x>3, what are truth value of p(x) and p(1)

P(4), setting x = 4, 4>3 is True.

P(1), setting x = 1, 1>3 is False.

- Propositional logic is not sufficient to express the meaning of statements in mathematics.
- Predicate logic is more powerful type of logic.
- Predicates are also used in verification that computer programs always produces the desired output when given valid input. The statements that describe valid inputs are preconditions and the conditions that output should satisfy when the program has run are called post condition.

temp = x

x = y

y = temp

For precondition, we need to express that x and y have particular values before we run the program. For precondition we use the predicate P(x, y) is statement where x = a and y =b. As we want to verify that program swaps the values of x and y, for post

condition we use Q(x, y) where Q(x, y) is statement x = b and y = a. To verify, suppose that the precondition we use Q(x, y) statement x = a and y = b is true. In 1st step, temp = x, after this step we know x = a, temp = a and y =b. after 2nd step, x = y we know x = b, temp = a, y = b. Finally after 3rd step, we know that x = b, temp = a and y = a. After this program is run the post condition Q(x, y) holds, that is, the statement "x = b and y = a" is true.

Quantifiers

Quantifier defines the quantity or quantification. *Quantification* is a way to create a proposition from a propositional function. It expresses the extent to which a predicate is true over a range of elements.

Two types of quantification

1. Universal quantification
2. Existential qualification

Universal quantification

Universal quantification tells us that a predicate is true for every element under consideration. The universal quantification of P(x) for a particular domain is the proposition that asserts that P(x) is true for all values of x in this domain. Domain specifies the possible values of the variable x.

The universal quantification of P(x) for all values of x in the domain is denoted by $\forall x\, P(x)$.

$\forall x P(x)$ is true when P(x) is true for every x.

$\forall x P(x)$ is false when there is a x for P(x) is false. An element for which P(x) is false is called a counter example of $\forall x\, P(x)$. If the domain is empty $\forall x P(x)$ is true for any propositional function P(x), since there are no counter example in the domain.

When all elements in domain can be listed as $x_1, x_2, x_3, \ldots, x_n$, the universal quantification $\forall x P(x)$ is same the conjunction.

If the domain is finite $\{x_1, x_2, x_3 \ldots\ldots, x_n\}$

$\forall x P(x)$ is the same as $p(x_1) \wedge p(x_2) \ldots\ldots \wedge p(x_n)$.

Examples

Q. ***Let p(x) be "$x^2>10$", what is the truth value of $\forall x\, P(x)$***

i. ***The set of real number*** (Ans. F)

ii. ***The set of positive number not exceeding {1, 2, 3, 4}*** (Ans. F)

$= p(1) \wedge p(2) \wedge p(3) \wedge p(4)$

$= F \wedge F \wedge F \wedge F \wedge T$

$= F$

iii. ***Let set of real number in the interval [10,50]*** (Ans. T)

Q. ***Let Q(x) be the statement 'x<2', what is the truth value of the quantification $\forall x\, Q(x)$ where domain consists of all real numbers.***

Q(x) is not true for every real number x. Q(3) is false, that is Q(3) is a counter example of statement $\forall x\, Q(x)$. Thus $\forall x\, Q(x)$ is false.

Existential Qualification

Existential quantification tells us that there is one or more element under consideration for which predicate is true.

The existential quantification of P(x) is the statement "There is an element x in the domain such that P(x) is true." It is denoted by $\exists x P(x)$.

$\exists x P(x)$ is true when P(x) is true for one or more x in the domain.

$\exists x P(x)$ is false when P(x) is false for all values of x in the domain.

The element for which P(x) is true is called a witness of $\exists x P(x)$. If the domain is empty $\exists x P(x)$ is false for any propositional function P(x) since there is not witness in the domain.

$\exists$ is called existential quantifier

- Words that express existential quantification are "there exists", "for at least one", "there is".

- If the domain is finite {x1, x2, …. , xn} or listed as x1.x2….. xn, the existential quantification. P(x1)∨p(x2)∨……p(xn).

Examples

Q. ***Let P(x) be "x^2>10". what is the truth value of ∃xP(x).***

1. ***The set of real number T***
2. ***The set of positive number not exceeding {1, 2, 3, 4}***

   = P(1) ∨ P(2) ∨ P(3) ∨ P(4)

   = F ∨ F ∨ F ∨ T

   = T

3. ***The set of real number in the interval [0,$\sqrt{9.8}$]***

   False (($\sqrt{9.8}$)^2>10)

   9.8>10 False

Q. ***P(x) denotes statement "x>3", what is truth value of quantification ∃xP(x)? where domain consists of all real number?***

Because "x>3" is sometime true, for instant when x = 4, the existential quantification of P(x), that is ∃xP(x) is true.

Q. ***Q(x) denote statement "x = x + 1". Domain consists of real number.***

Because Q(x) is false for every real number x, the existential quantification of Q(x) which is ∃xQ(x) is false.

Binding variables and scope

When a quantifier is used on the variable x, we say that this occurrence of x is said to be bound. An occurrence of a variable that is not bound by an quantifier or set equal to a particular value is said to be free.

For example,

∃x(x + y = 1), variable x is bound by existential quantification.

Show that ∀x(P(x) ∧ Q(x)) and ∀xP(x) ∧∀x Q(x) are logically equivalent (where the same domain is used throughout)

Let ∀x(P(x) ∧∀Q(x)) is true. This means that if a is in the domain, P(a) ∧Q(a) is true. Hence P(a) is true and Q(a) is true. Because P(a) is true and Q(a) is true for every element in the domain, we can conclude that ∀xP(x) and ∀xQ(x) are both true. This concludes ∀xp(x) ∧∀xQ(x) is true.

Let ∀x p(x) ∧∀x Q(x) is true. ∀xP(x) is true and ∀xQ(x) is true. Hence, if a is in domain, then P(a) is true and Q(a) is true. It follows that for all a, p(a) ∧Q(a) is true.

∀x(p(x) ∧ Q(x)) ≡ ∀xP(x) ∧ ∀xQ(x)

Negating Quantified Expression

1. ***Every student in this class has taken a course in calculus***

   ∀xP(x), let P(x) be the statement "x has taken a course in calculus". And domain consists of the students in this class.

   Negation

   The negation of this statement is "It is not the case that every student in this class has taken a course in calculus". This is equivalent to "There is a student in this class who has not taken a course in calculus". And it is the existential quantification of the negation of the negation of original propositional function. i. e.

   ¬∀xP(x)≡∃x¬P(x)

   ¬∀xP(x) is true, if ∀xP(x) is false.

   ∀xP(x) is false, if there is an element x in the domain for which P(x) is false. ¬P(x) is true.

   Finally there is an element x in the domain for which ¬P(x) is true if and only if ∃x¬P(x) is true.

   ∀xP(x) is true if and only if ∃x¬P(x) is true.

2. ***¬∃xP(x) ⇔ ∀x¬P(x)***

   "There is a student in this class who has taken a course in calculus."

   ∃xP(x).

Negation

"It is not the case that there exist a student in this class who has taken a course in calculus."

$\neg\exists x P(x)$

"Eevery student in this class has not taken a course in calculus."

$\neg\forall x P(x)$

Find the negations

1. $\forall x(x^2 > x)$
2. $\exists x(x^2 = 2)$

Negation of $\forall x(x^2 > x)$

$= \neg\forall x(x^2 > x)$

$= \exists x \neg (x^2 > x)$

$= \exists x(x^2 <= x)$

Negation of $\exists x(x^2 = 2)$

$= \neg\exists x(x^2 = 2)$

$= \exists x \neg(x^2 = 2)$

$= \forall x(x^2 \neq 2)$

Restricting the domain of a quantifier

1. $\forall x>0\ (x^2>2) \Leftrightarrow \forall x\ (x>0 \rightarrow x^2>0)$
2. $\forall y\neq 0\ (y^3 = 0)$ is same as $\forall y\ (y = 0 \rightarrow y^3\neq 0)$
3. $\exists z>0\ (z^2 = 2)$ is same as $\exists x(z = 0 \wedge z^2 = 2)$

Translate into logical equivalence

1. **Every student in this class has studied calculus.**

   Here,

   Domain = all student.

   For every student in this class, that student has studied calculus.

   Let x be the variable, so that for every student x in this class, x has studied calculus.

   Let, p(x) be 'x has studied calculus.'

   If we change the domain to consist of all people.

   For every person, if person x is a student in this class, then x has studied calculus.

   If Q(x) represent statement the person s is student.

   $\forall x Q(x) \rightarrow P(x)$

2. **Some student in this class has visited pokhara**

   There is a student that hass visited pokhara

   Q(x) : student x has visited pokhara.

   S(x) : x is a student in this class.

   There is a student x, x is student in the class and x has visited pokhara.

3. **Every student in this class has visited either Canada or mexico**

   $\forall x(P(x) \rightarrow Q(x) \vee R(x))$

4. **Every mail message larger than one megabyte will be compressed.**

   Let x be mail message.

   Let y be megabyte.

   Mail message x is larger than one megabyte. i. e.

   S(x, y)

   Message will be compressed. i. e. C(x)

   $\forall x\ (S(x, y) \rightarrow C(x))$

5. Consider the statement

   "all lions are fierce."

   "Some lions do not drink coffee."

   P(x) : x is lion.

   Q(x) : x is fierce

   $\forall x\ (P(x) \rightarrow Q(x))$

   $\exists x(P(x) \wedge \neg R(x))$

   $\exists x(Q(x) \wedge \neg R(x))$

Nested quantifiers

1) $\forall x \forall y\ (x + y = y + x)$

For every x and for every y, $x + y = y + x$

2) $\forall x \exists y (x + y = 0)$

For all real no. x, there is a real no. y such that $x + y = 0$

3) Let Q(x, y) denote $(x + y = 0)$. What is truth values of quantification$\exists y \forall x Q(x, y)$ and $\forall x \exists y Q(x, y)$. where domain consist of all real number.

$\exists y \forall x\ Q(x, y)$ is False

$\forall x \exists y Q(x, y)$ is True

why?

$\exists y \forall x Q(x, y)$ means that "There is a number y such that for every real number x, $Q(x + y = 0)$." which is false.

$\forall x \exists y Q(x, y)$ means that "For all real no. x, there is a real no. y, such that Q(x, y) that is $x + y = 0$." which is true.

Statement	When True?	When False?
$\forall x \forall y P(x, y)$ $\forall y \forall x P(x, y)$	For every pair (x, y), P(x, y) is true.	There exists a pair x, y such that p(x, y) is false.
$\forall x \exists y P(x, y)$	For every values of x, there is a y such that P(x, y) is true.	There exists a x suh that for for every y, P(x, y) is false
$\exists x \forall y P(x, y)$	There exists a x such that P(x, y) for every y,	For every x there exists a y such that P(x, y) is false.
$\exists x \exists y P(x, y)$ $\exists y \exists x P(x, y)$	For some pair (x, y), P(x, y) is true.	For all pairs (x, y), P(x, y) is false.

1.7 Arguments, Argument Forms and Validity

Definition:

An *argument* in propositional logic is sequence of propositions. All but not the final proposition 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.

An argument form in propositional logic is a sequence of compound propositions involving propositional variables. An argument form is valid if no matter which propositions are substituted for the propositional variables in its premises, if the premises are all true, then the conclusion is true.

In other words, an argument form with premises $p_1, p_2, \ldots, p_n$ and conclusion q is valid if and only if $(p_1 \wedge p_2 \wedge \cdots \wedge p_n) \rightarrow q$ is a tautology.

Validity is the quality of being logically or factually sound; soundness or cogency. It is the state of being acceptable.

1.8 Rules of Inference

Rules of Inference

An argument is a sequence of statement that ends with a conclusion. The argument is valid if the conclusion (final statement) follows from the truth of the preceding statement (premises). Rules of inferences are templates for building valid arguments. These will be the main ingredient needed in formal proofs.

Example:

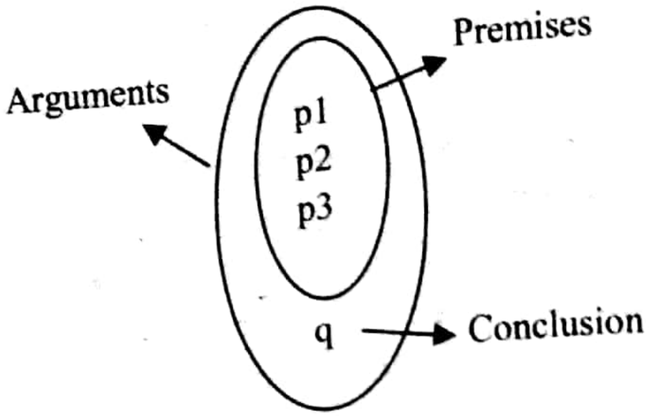
'If you have a current password, then you can log onto the network.'

Let p represents "you have a current password."

q represents "you can log onto the network."

$$\begin{array}{c} p \rightarrow q \\ p \\ \hline q \end{array}$$

$((p \rightarrow q) \wedge p) \rightarrow q$ is tautology.



When both $p \rightarrow q$ and p are true, q must be true. We say this form of argument is valid because whenever all premises are true, the conclusion must also be true.

An argument form in prepositional logic is a sequence of compound prepositions involving preposition variables. An argument formed with premises p1, p2,…., pn and conclusion q is valid iff $(p1 \wedge p2 \wedge \ldots pn) \rightarrow q$is tautologies.

P	Q	$p \rightarrow q$	$(p \rightarrow q) \wedge p$	$((p \rightarrow q) \wedge p) \rightarrow q$
T	T	T	T	T
T	F	F	F	T
F	T	T	F	T
F	F	T	F	T

Hence the argument is valid.

S.N.	Rules of inference	Tautology	Name
1.	P $p \rightarrow q$ q	$(p \wedge (p \rightarrow q)) \rightarrow q$	Modus ponens(mode that affirms)
2.	$\neg q$ $p \rightarrow q$ $\neg p$	$(\neg q \wedge (p \rightarrow q)) \rightarrow \neg p$	Modus tollens mode that denies)
3.	$p \rightarrow q$ $q \rightarrow r$ $p \rightarrow r$	$[(p \rightarrow q) \wedge (q \rightarrow r)] \rightarrow (p \rightarrow r)$	Hypothetical syllogism

S.N.	Rules of inference	Tautology	Name
4.	$p \vee q$ $\neg p$ q	$[(p \vee q) \wedge \neg p] \rightarrow q$	Disjunctive syllogism
5.	P $p \vee q$	$p \rightarrow (p \vee q)$	Addition
6.	$p \wedge q$ p	$(p \wedge q) \rightarrow p$	Simplification
7.	P q $p \wedge q$	$[(p) \wedge (q)] \rightarrow (p \wedge q)$	Conjunction
8.	$p \vee q$ $\neg p \vee r$ $q \vee r$	$[(p \vee q) \wedge (\neg p \vee r)] \rightarrow (q \vee r)$	Resolution

Which rule of inference is used?

Q. ***It is below freezing and raining now. Therefore it is either below freezing or raining now.***

Let p be proposition "It is below freezing now" and q be proposition "it is raining now."

p

pvq

This is an argument that uses additive rule.

Q. ***If it rains today, then we will not have barbecue today. If we do not have barbecue today, then we will have a barbecue tomorrow. Therefore, if it rains today, then we will have a barbecue tomorrow.***

Let p be proposition "it rains today."

q be proposition "we will not have barbecue today."

r be proposition "we will have barbecue tomorrow."

$p \rightarrow q$

$q \rightarrow r$

$p \rightarrow r$

This argument is hypothetical syllogism.

1.9 Formal and Informal Proof

1.9.1 Formal Proof

A *formal proof* of a conclusion q with given hypotheses p_1, p_2, p_3 . . . , p_n is a sequence of steps, each of which applies some inference rule to hypotheses or previously proven statements (antecedents) to yield a new true statement (the consequent).

A formal proof demonstrates that if the premises are true, then the conclusion is true. A formal proof is based simply on symbol manipulation (no need of thinking, just apply rules). A formal proof is rigorous but so can be a proof that does not rely on symbols! Formal proof example (rules of inference to build arguments)

1.9.2 Informal Proof

The steps of the informal iproofs are not expressed in any formal language as e.g. propositional logic. Steps are argued less formally using English, mathematical formulas and so on. One must always watch the consistency of the argument made; logic and its rules can often help us to decide the soundness of the argument if it is in question. We use informal proofs to illustrate different methods of proving theorems.

Q. ***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 canoe trip, then we will be home by sunset." Lead to the conclusion "we will be home by sunset."***

Let p be the proposition "It is sunny this afternoon."

q denotes "It is colder than yesterday."

r denotes "we will go swimming."

s denotes "we will take canoe trip."

t denotes "we will be home by sunset."

Then, hypothesis

i. $\neg p \wedge q$

ii. $r \rightarrow p$

iii. $\neg r \rightarrow s$

iv. $s \rightarrow t$

Conclusion

S. N.	Steps	Reasons
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 ponens using (4) and (5)
7.	$s \rightarrow t$	Hypothesis
8.	T	Modus ponens using (6) and (7)

Q. ***"If you send me an email message then I will finish writing the program", "if you do not send me an email message then I will go to sleep early" and "If I go to sleep early then I will wake up feeling refreshed" lead to the conclusion " If I do not finish writing the program then I will wake up feeling refreshed."***

Propositions :

p : you send me an email message.

q : I will finish writing the program.

r : I will go to sleep early.

s : I will wake up feeling refreshed.

Hypothesis :

$p \rightarrow q$, $\neg p \rightarrow r$, $r \rightarrow s$

Conclusion :

$\neg q \rightarrow s$

S. N.	Step	Reason
1.	$p \rightarrow q$	Hypothesis
2.	$\neg q \rightarrow \neg p$	Contrapositive of (1)
3.	$\neg p \rightarrow r$	Hypothesis

S. N.	Step	Reason
4.	$\neg q \rightarrow r$	Using hypothetical syllogism of (2) and (3)
5.	$r \rightarrow s$	Hypothesis
6.	$\neg q \rightarrow s$	Hypothetical syllogism

1.9.3 Resolution Principle

- Another way of proving that an argument is correct, is resolution principle.
- The variable or negation of a variable is called literal.
- Clause is a disjunction of literals.

For any two clauses C1 and C2, if there is a literal L1 in C1 that is complementary to a literal L2 in C1, then delete L1 and L2 from C1 and C2 respectively and construct the disjunction of the remaining clauses suppose.

C1 = PVQVR

C2 = PV¬SVT

Then P and ¬P is complementary to each other, the resolvent of C1 and C2 respectively, and finding the disjunction of the remaining literals. In this case it is QVRV¬SVT`

E.g., Consider modus ponens

P

$\underline{P \rightarrow Q}$

Q

Now, from resolution principle,

C1: P

C2: $\neg P \vee Q$

C3: $\neg Q$

From C1 and C2 by resolution you can get Q and from C3 by resolution ϒ is arrived. So, it is valid.

It can be written as follows:

C1: P (hypothesis)

C2: $\neg P \vee Q$ (hypothesis)

C3: $\neg Q$ (negation of conclusion)

C4: Q (C1 and C2 resolution)

C5: ϒ

Steps for resolution

1. Change implies ($\rightarrow$) to disjunction ($\vee$)
2. Negate the conclusion
3. Use rules
4. If empty clause ϒ comes, it is proved.

A deduction of ϒ (empty clause) is called refutation of proof of a set of clause.

Example

Is argument correct? Prove by resolution principle.

Q. ***If today is Tuesday, I have a test in mathematics or economics. If my economics professor is sick, I will not have a test in economic. Today is Tuesday and my economics professor is sick. Therefore, I have a test in mathematics.***

T denotes "Today is Tuesday."

M denotes "I have a test in mathematics."

E denotes "I have a test in economics."

S denotes "My economic professor is sick."

Premises are

$T \rightarrow (M \vee E)$

$S \rightarrow \neg E$

$T \wedge S$

conclusion`

M

Solution putting in clause form

Clause	Reason
C1 : $\neg T \vee (M \vee E)$	Hypothesis
C2 : $\neg S \vee \neg E$	Hypothesis
C3 : T	Using Simplification in $T \wedge S$

C4 : S	Using Simplification in $T \wedge S$
C5 : $\neg M$	Negation of Conclusion
C6 : $\neg T \vee M \vee \neg S$	From C1 and C2
C7 : $M \vee \neg S$	From C3 and C6
C8 : M	From C4 and C7
C9 : Υ	From C5 and C8

Hence argument is correct.

1.9.4 Fallacy

Fallacy is misconception resulting from incorrect argument.

Fallacy of affirming the conclusion based on

$((p \rightarrow q) \wedge q) \rightarrow p$ which is NOT a tautology

Is the following argument valid?

Q. ***If you do every problem in this book then you will learn discrete mathematics. You learned discrete mathematics. Therefore, you did every problem in this book.***

$((p \rightarrow q) \wedge q) \rightarrow p$ is not a tautology because it is false when p is false and q is true.

Premises $(p \rightarrow q)$ and q

Conclusion q.

This type of incorrect reasoning called fallacy of affirming the conclusion.

Fallacy of denying the hypothesis

Based on

$((p \rightarrow q) \wedge \neg p) \rightarrow \neg q$ which is not tautology.

If you do every problem in this book, then you will learn D.S. you did not do every problem in this book. Therefore, you didn't learn D.S.

1.9.5 Translation of sentences in predicate logic

"Every student in this class has studied D.S."

For every student x in this class, x has studied D.S.

D(x) : "x has studied D. S."

$\forall x D(x)$

Different connectives are used in universal quantifier and existential quantifier.

i. Use implies ($\rightarrow$) for universal quantifier

ii Use conjunction ($\wedge$) for existential quantifier

(Q) ***For every person x, if a person x is student on this class then x has studied Discrete Structure.***

S(x): "x is a student in this class."

D(x): "x has studied discrete structure."

$\forall x(S(x) \rightarrow D(x))$

(Q) ***Some student in this class has visited Mexico.***

S(x): "x is a student in this class."

M(x): "x has visited Mexico."

$\exists x(S(x) \wedge M(x))$

1.10 Informal Deduction in Predicate Logic

The informal deduction of a square implies that it has four congruent sides and four congruent angles and with the properties of a square such as opposite sides parallel and the diagonals perpendicular bisectors. The following rules of inference are informal deduction in predicate logic.

Rules of inference for Quantified statement

S. N.	Rules of inference	Name
1.	$\frac{\forall x p(x)}{p(c)}$	Universal instantiation
2.	$\frac{p(c)}{\forall x\, p(x)}$	Universal generalization
3.	$\frac{\exists x p(x)}{p(c)}$	Existential instantiation
4.	$\frac{p(c)}{\exists x p(x)}$	Existential generalization

Show that the premises

- A student in section A has taken the course has not read the book.
- Everyone in section A has taken the course passed the first exam.

Imply the conclusion

Someone who passed the first exam has not read the book.

A(x) :"x is in section who has taken the course"

B(x) : "x read the book"

P(x) : "x passed the first exam."

Hypothesis

$\exists x(A(x) \wedge \neg B(x)), \forall x(A(x) \rightarrow P(x))$

Conclusion:

$\exists x\ (P(x) \wedge \neg B(x))$

S. N.	Step	Reason
1.	$\exists x(A(x) \wedge \neg B(x))$	Hypothesis
2.	$A(a) \wedge \neg B(a)$	Existential instantiation from (1)
3.	$A(a)$	Simplification from (2)
4.	$\forall x(A(x) \rightarrow P(x))$	Hypothesis
5.	$A(a) \rightarrow P(a)$	Universal instantiation from (4)
6.	$P(a)$	Modus ponens from (3) and (5)
7.	$\neg B(a)$	Simplification from (2)
8.	$P(a) \wedge B(a)$	Conjuction from (6) and (7)
9.	$\exists x(Px)) \wedge \neg B(x))$	Existential generalization from (8)

Q. ***Show that premises "everyone in his discrete mathematics class has taken a course in computer science." And "Marla is a student in this class" imply the conclusion "Marla has taken a course in computer science."***

Let D(x) denote "x is the discrete mathematics class, and let C(x) denote "x has taken a course in computer science." Then the premises are $\forall x(D(x) \rightarrow C(x))$ and D(Marla). The conclusion is C(Marla).

S.N.	Step	Reason
1.	$\forall x(P(x) \rightarrow C(x))$	Premise
2.	$P(marla) \rightarrow C(marla)$	Universal instantiation
3.	D(marla)	Premise
4.	C(marla)	Modus ponens from (2) and (3)

Q. ***"A student in this class has not read the book" and "Everyone in this class passed the first exam" implies that conclusion "someone who passed the first exam has not read the book."***

Propositions :

C(x) :xis student in this lass.

R(x) : x has read the book

P(x) : x has passed the exam.

Hypothesis :

$\exists x(C(x) \wedge \neg R(x)), \forall x(C(x) \rightarrow P(x)$

Conclusion :

$\exists x(P(x) \wedge \neg R(x))$

S. N.	Step	Reason
1.	$\exists x(C(x) \wedge \neg R(x))$	Hypothesis
2.	$(C(a) \wedge \neg R(a))$	Using existential instantiation in (1)
3.	$\forall x(C(x) \rightarrow P(x))$	Hypothesis
4.	$C(a) \rightarrow P(a)$	Using universal instantiation on (3)
5.	$C(a)$	Using simplification on (2)
6.	$P(a)$	Using modus ponens on (4) and (5)
7.	$\neg R(a)$	Using simplification on (2)
8.	$P(a) \wedge \neg R(a)$	Using conjunction on (6) and (7)
9.	$\exists x(P(x) \wedge \neg R(x)$	Using existential generation on (8)

Q. ***Lions are dangerous animals. There are lions. Therefore there are dangerous animals.***

L(x) : x is lion.

D(x) : x is dangerous animals.

Hypothesis :

$\forall x\,(L(x) \rightarrow D(x)), \exists x L(x)$

Conclusion:

$\exists x D(x)$

S. N.	Steps	Reasons
1.	$\forall x(L(x) \rightarrow D(x))$	Hypothesis
2.	$L(a) \rightarrow D(a)$	Universal instantiation on (1)
3.	$\exists x(L(x))$	Hypothesis
4.	$L(a)$	Existential instantiation on (3)
5.	$D(a)$	Modus ponens on (3) and (4)
6.	$\exists x D(x)$	Existential generalization on (5)

1.11 Informal Proofs and Formal Proofs

The steps of the *informal proofs* are not expressed in any formal language as e.g. propositional logic. Steps are argued less formally using English, mathematical formulas and so on. One must always watch the consistency of the argument made; logic and its rules can often help us to decide the soundness of the argument if it is in question. We use informal proofs to illustrate different methods of proving theorems.

A *formal proof* of a conclusion q with given hypotheses p_1, p_2, p_3 . . . , p_n is a sequence of steps, each of which applies some inference rule to hypotheses or previously proven statements (antecedents) to yield a new true statement (the consequent).

A formal proof demonstrates that if the premises are true, then the conclusion is true. A formal proof is based simply on symbol manipulation (no need of thinking, just apply rules). A formal proof is rigorous but so can be a proof that does not rely on symbols! Formal proof example (rules of inference to build arguments)

1.11.1 Proof Methods

A proof is a valid argument that establishes the truth of a mathematical statement, using the hypotheses of the theorem, if any, axioms assumed to be true, and previously proven theorems.

Using these ingredients and rules of inference, the proof establishes the truth of the statement being proved.

We move from formal proofs, as seen in the previous section, to informal proofs, where more than one inference rule may be used at each step, where steps may be skipped, and where axioms and rules of inference used are not explicitly stated.

Some Terminology

- **Definition :** A precise and unambiguous description of the meaning of mathematical term.
- **Theorem:** A statement that can be shown to be true using vigorous (strong) mathematical reasoning.

  A statement that can be shown to be true (sometimes referred to as facts or results).

  Less important theorems are often called propositions.
- **Lemma :** A lemma is a less important theorem, used as an auxiliary result to prove a more important theorem.

  It is used as an auxiliary result to prove a theorem. It is stepping stone to prove theorem.
- **Corollary:** It is a theorem proven as an easy consequence of a theorem.
- **Conjecture :** A conjecture is a statement that has been proposed a true statement. If proved later then it can be used as a theorem but it may be false

- **Axiom** (or postulates) are statements that we assume to be true v A proof is a valid argument that establishes the truth of a theorem.
- **Rules of inference, Axiom (or postulates):** Axiom or postulates are statements that we assume to be true (algebraic axioms specify rules for arithmetic like commutative laws) without proof. These are building blocks to prove theorem.
- **Proof :** A proof is a valid argument that establishes the truth of a theorem. The statements used in a proof include axioms, hypotheses (or premises), previously proven theorems and definitions. Rules of inferences, together with definition of terms, are used to draw conclusions from other assertions, tying together the steps of a proof.

Three methods of showing statements of the type $p \rightarrow q$ are true:

1. **Direct proofs:** Assume p is true; the last step establishes q is true.
2. **Proof by Contraposition:** Uses a direct proof of the contrapositive of $p \rightarrow q$, which is $\neg q \rightarrow \neg p$. That is, assume $\neg q$ is true; the last step established $\neg p$ is true.
3. **Proof by Contradiction:** To prove that P is true, we assume $\neg P$ is true and reach a contradiction, that is $(r \wedge \neg r)$ is true for some proposition r. In particular, to prove $(p \rightarrow q)$, we assume $(p \rightarrow q)$ is false, and get as a consequence a contradiction. Assuming that $(p \rightarrow q)$ is false $= (\neg p \vee q)$ is false $= (p \wedge \neg q)$ is true.
4. **Formal and Informal Proof**

   A formal direct proof of a conditional statement $p \rightarrow q$ works as follows:

   Assume p is true, build steps using inference rules, with the final step showing that q is true.

   In informal direct proof, we assume that p is true, and use axioms, definitions and previous theorems, together with rules of inference to show that q must be true.

1.11.2 Direct proof

Assume p is true and the last step establishes q is true.

Before we give example of direct proof, we need a definition.

Definition

The integer n is even if there exits an integer k such that $n = 2k$ and n is odd if there exits an integer to such that $n = 2k + 1$.

Theorem (use direct proof)

1. **If n is an odd integer, then n^2 is odd.**

   Let p(n) : 'n is an odd integer ' is true

   then,

   $n = 2k + 1$ (from definition)

   then,

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

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

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

   here, $(2k^2 + 2k)$ is an integer, so,

   $2(2k^2 + 2k)$ is even

   n^2 is odd.

   Definition

   An integer is a perfect square if the is an integer b such that $a = b^2$

Theorem

2. **If m and n are both perfect square the mn is also a perfect square.**

   Let m and n be perfect square, then there exist integers s and t such that,

   $m = s^2$ and

   $n = t^2$

   $mn = s^2t^2 = (st)^2$

   Hence, R.H.S, is

mn is a perfect square.

1.11.3 Indirect proof

Direct proofs lead from the hypothesis of a theorem to the conclusion. They begin with the premises, continue with a sequence of deductions, and end with the conclusion. We need other methods of proving theorems of the form $\forall x(P(x) \rightarrow Q(x))$

Proofs of theorem of this type that are not direct proofs, i.e., that do not start with the hypothesis and end with the conclusion, are called indirect proofs.

Example of indirect proofs are proof by contraposition, proof by contraposition.

1.11.2.1 Proof by contraposition

1. **If n is an integer and 3n + 2 is odd, then n is odd.**

$p \rightarrow q$ (if 3n + 2) is odd then n is odd)

$\neg q \rightarrow \neg p$ (if n is not odd then (3n + 2) is even)

let n is even,

$n = 2k$

now, $3n + 2 = 3*2k + 2$

$= 6k + 2$

$= 2(3k + 1)$

Thus (3n + 2) is even

The negation of conclusion of conditional statement implies that the hypothesis is false, so, the original conditional statement is true.

2. **Prove that if n = ab where a and b are positive integers then $a \leq \sqrt{n}$ or $b \leq \sqrt{n}$**

Here,

If (n = ab) then ($a \leq \sqrt{n}$ or $b \leq \sqrt{n}$)

The contrapositive of $p \rightarrow q$ is $\neg q \rightarrow \neg p$

So,

$\neg q$ means ($a > \sqrt{n}$ V $b > \sqrt{n}$)

Now, $ab > \sqrt{n}\sqrt{n}$

$ab > n$

$ab \neq n$, which contradicts the original conditional statement

$n = ab$

Proof strategy

When we want to prove a statement of the form $\forall x\ (p(x) \rightarrow Q(x))$, first we try to prove by direct proof. If a direct proof does not see to go anywhere, try the same thing with a proof by contraposition.

Definition

A real number is rational if there exist integers p and q with $q \neq 0$ such that $r = p/s$. A real number that is not rational is called irrational.

1. **Prove that the sum of two rational numbers rational.**

If r and s are rational numbers then to prove is r + s is also rational

Let $r = p/q$ and $s = t/u$ where p, q, t, u are integers such that $q \neq 0$ and $u \neq 0$

$r + s = p/q + t/u = (pu + qt)/qu$

$qu \neq 0$

Hence, r + s is rational (proof succeeded by direct proof)

2. **Prove that if n is an integer and n^2 is odd,**

Then n is odd.

Here, we first attempt by a direct proof. Suppose n is an integer and n^2 is odd.

$n^2 = 2k + 1$

$n = \pm\sqrt{2k + 1}$

The above equation is not useful to show n is odd. Because this attempt to use a direct proof did not bear fruit, we next have a proof by contraposition.

$\neg q \rightarrow \neg p$

if n is not odd then n^2 is not odd

if n is even then n^2 is even

$n = 2k$

$n^2 = (2k)^2 = 2k(2k)^2$ even

1.11.2.2 Proof by contradiction

Suppose we want to prove that a statement p is true. Suppose that we can find a contradiction q such that $\neg p \rightarrow q$ is true, only when p is false that is p is true.

For $(p \rightarrow q)$ we use $(p \wedge \neg q) \rightarrow F$

1. **$\sqrt{2}$ is irrational (prove by proof by contradiction)**

Let $\sqrt{2}$ is rational.

Then 2 can be represented as a ratio of integer a and b such that $b \neq 0$. Where a and b have no common factor.

$\sqrt{2} = a/b$

$\sqrt{2}\ b = a$

Squaring both side,

$2b^2 = a^2$

a^2 is even number, so a is also even

$a = 2c$

Again, $2b^2 = 4c^2$

$b^2 = 2c^2$

b^2 is even, b is even

Both a an b are even, 2 divides both a and b. $\sqrt{2} = a/b$. when a and b have no common factor it means 2 does not divid both a and b. Because $\neg p$ leads to contradiction that 2 divide both a and b. so, $\neg p$ is false. 2 is irrational.

2. **If 3n + 2 is odd, then n is odd.**

Let us assume that n is even and 3n + 2 is odd.

Then, $n = 2k$

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

$= 6k + 2$

$= 2(3k + 1)$

Its even

Hence, our assumption came to wrong.

So, if 3n + 2 is odd then n is odd.

Q. ***Prove if a^2 is even then a is even (by contrapositive)***

For $p \rightarrow q$ the contrapositive is $\neg q \rightarrow \neg p$

So, the contrapositive g the given question is

If a is odd then a^2 is odd

$a = 2k + 1$

$a^2 = 4k^2 + 4k + 1 = 2(2k^2 + 2k) + 1$ which is odd`(see the difference in proof by contrapositive are proof contradiction)

Hence proved.

If a^2 is even then a is even.

$a = 2c$ where c is any integer

$2b^2 = (2c)^2$

$b^2 = 2c^2$

1.11.3 Vacuous proof

$p \rightarrow q$ is true when p is false. If we can show that p is false then we have a proof , called vacuous proof.

We can quickly prove that a conditional statement $p \rightarrow q$ is true when we know that p is false, because $p \rightarrow q$ must be true when p is false. Consequently, if we can show that p is false, then we have a proof, called a vacuous proof, of the conditional statement $p \rightarrow q$.

Q. ***Show that the proposition P(0) is true where the domain consists of the integer numbers and P(n) is "If $n \geq 1$ then $n^2 > n$." Note: vacuous proof: when p is false $p \rightarrow q$ is true, regardless of the value of q.***

Solution:

Note that P(0) is "If 0 > 1, then 02 > 0." We can show P(0) using a vacuous proof, because the hypothesis 0 > 1 is false. Hence P(0) is automatically true.

1.11.4 Trivial proof

A proof of $p \rightarrow q$ that uses the fact that q is true is called trivial proof.

By showing that q is true, it quickly follows that $p \rightarrow q$ must also be true. A proof of $p \rightarrow q$ that uses the fact that q is true is called a trivial proof.

Q. ***Let P(n) is 'If a and b are positive integers with $a \geq b$ then $a^n \geq b^n$ where domain consists of all integers. Show that p(0) is true.***

The proposition P(0) is "If $a \geq b$, then $a^0 \geq b^0$" Because $a^0 = b^0 = 1$, the conclusion of the given conditional statement is true.

Thus p(0) is true.

1.11.5 Proof by equivalence

To prove $p \rightarrow q$ show that $p \rightarrow q$ is true and $q \rightarrow p$ is true. The validity of this approach is based on the tautology

$$(p \leftrightarrow q) \leftrightarrow [(p \rightarrow q) \wedge (q \rightarrow p)]$$

Example: Prove that if n is a positive integer, then n is odd if and only if n2 is odd.

Sometimes, a theorem states that several propositions are equivalent. Such a theorem states that propositions p1, p2, p3.....pn are equivalent. This can be written as $p1 \leftrightarrow p2 \leftrightarrow \ldots\ldots \leftrightarrow pn$

which states that all propositions have the same truth values and consequently for all i and j with $1 \leq i \leq n$ and $1 \leq j \leq n$, pi and pj are equivalent.

This can be proved as following $v(p1 \leftrightarrow p2 \leftrightarrow \ldots\ldots \leftrightarrow pn) \leftrightarrow (p1 \rightarrow p2) \wedge (p2 \rightarrow p3) \wedge \ldots\ldots\ldots\ldots \wedge (pn \rightarrow p1)$

Q. ***Prove that "If n is an integer, then n is odd iff n^2 is odd".***

p: n is odd

q: n^2 is odd

$p \rightarrow q$ [T]

$n = 2k + 1$

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

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

$q \rightarrow p$

$n = 2k$

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

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

if n is even, then n^2 is even

(proof by contraposition)

Q. ***Show that these statement about integer n are equivalent***

p1: n is even

p2: n-1 is odd

p3: n^2 is even

$p1 \rightarrow p2$, $p2 \rightarrow p3$ and $p3 \rightarrow p1$

For $p1 \rightarrow p2$

Let n= 2k

then n-1= 2k-1=2k-1+1-1=2(k-1)+1=even+1=odd

For $p2 \rightarrow p3$

Let n-1=2k + 1

Then n= 2k+2

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

$p3 \rightarrow p1$ i.e. if n^2 is even n is even is proved earlier.

Hence these propositions are equivalent

$p1 \leftrightarrow p2 \leftrightarrow p3$ ------ $\leftrightarrow pn) \leftrightarrow (p1 \rightarrow p2) \wedge (p2 \rightarrow p3) \wedge \ldots \wedge (pn \rightarrow p1)$

1.11.6 Proof by cases

Sometimes it is difficult to use a single argument that holds for all cases.

Proof by cases uses the following equivalence:

$[(p1 \vee p2 \vee \cdots \vee pn) \rightarrow q] \equiv [(p1 \rightarrow q) \wedge (p2 \rightarrow q) \wedge \cdots \wedge (pn \rightarrow q)]$

Example: Prove that if n is integer then $n^2 \geq n$.

We split the proof into three cases.

Case (i) $n = 0$.

In this case, $n^2 = 0^2 = 0 = n$.

Case (ii) $n \geq 1$

In this case, when we multiply both sides of $n \geq 1$ by n we obtain $n \cdot n \geq n \cdot 1$. This simplies $n^2 \geq n$.

Case (iii) $n \leq -1$.

In this case, $n \leq -1$, but $n^2 \geq 0$. Therefore, $n^2 \geq 0 \geq -1 \geq n$, and so $n^2 \geq n$.

$[(p1 \vee p2 \vee \ldots\ldots\ldots \vee pn) \rightarrow q] \equiv [(p1 \rightarrow q) \wedge (p2 \rightarrow q) \ldots\ldots \wedge (pn \rightarrow q)]$

This shows that the original conditional statement with a hypothesis made up of a disjunction of the propositions p1, p2,....., pn can be proved by proving each of the n conditional statement $pi \rightarrow q$, $i = 1,2,\ldots.n$ individually. Such an argument is called a proof by cases.

1.11.7 Exhaustive proof

This is a special form of a proof by cases, when there is a finite and small number of examples for which we need to prove a fact.

Some theorems can be proved by examining a relatively small number of examples. Such proofs are called exhaustive proof because these proofs proceed by exhausting all possibilities. It is special type of proof by cases.

Q1. Prove that $(n + 1)^2 \geq 3^n$ if n is a positive integer with $n \leq 2$.

We use a proof by exhaustion, by examining the cases n = 1, 2.

For $n = 1, (n + 1)^2 = 2^2 = 4 \geq 3^n = 3^1 = 3$.

For $n = 2, (n + 1)^2 = 3^2 = 9 \geq 3^2 = 3^n$.

Q2. Prove that $(n + 1)^2 \geq 3^n$ if n is positive integer with $n \leq 4$.

We only need verify the inequality $(n + 1)^2 \geq 3^n$

When n = 1,2,3 and 4

For $n = 1$, $(n + 1)^2 = 2^2 = 4$ and $3^n = 3^1 = 3$

For $n = 2$, $(n + 1)^2 = 3^2 = 9$ and $3^n = 3^2 = 9$

For $n = 3$, $(n + 1)^2 = 4^3 = 64$ and $3^n = 3^3 = 27$

For $n = 4$, $(n + 1)^2 = 5^3 = 125$ and $3^n = 3^4 = 81$

In each cases, we see that $(n + 1)^2 \geq 3^n$

Mistakes in Proofs

What is the problem with the following proof that 1 = 2?

Use the following steps, where a and b are two equal positive integers.

1. $a = b$ Given
2. $a^2 = ab$ Multiply both sides of (1) by a
3. $a^2 - b^2 = ab - b^2$ Subtract b^2 from both sides of (2)
4. $(a - b)(a + b) = b(a - b)$ Factoring both sides of (3)
5. $a + b = b$ Divide both sides of (4) by a−b
6. $2b = b$ Replace a by b in (5) , since a = b
7. $2 = 1$ Divide both sides of (6) by b Therefore 2 = 1.

Solution:

Step 5 where we divided both sides by a-b is invalid. The error is that a - b equals zero; (Divide by Zero)

1.12 Elementary Induction and Complete Induction

In weak induction, we test the base case putting n=0 or 1 and assume that particular statement holds at k^{th} step, while in strong induction, we assume that the particular statment holds at all the steps from the base case to k^{th} step. The difference between weak (elementary) induction and strong (complete) induction only appears in induction hypothesis. In weak induction, we only assume that particular statement holds at k^{th} step, while in strong induction, we assume that the particular statment holds at all the steps from the base case to k^{th} step.

Example of strong induction:

Proposition: Every integer greater than 1 can be written as the product of prime numbers.

Let P(n) be the statement that n can be written as the product of prime numbers.

Then the proposition says: P(n) is true for each integer greater or equal to 2.

Proof (by strong induction):

1 **Base case**: 2 is a prime, so it is the product of a single prime.

2. **Strong inductive step**: Suppose for some $k \geq 2$ that each integer n with $2 \leq n \leq k$ may be written as a product of primes. We need to prove that k + 1 is a product of primes.

Case (a): Suppose k + 1 is a prime. Then we are done.

Case (b): Suppose k + 1 is a not prime. Then by the fact stated above, there exist integers a and b with $2 \leq a, b \leq k$ such that

$k + 1 = a \cdot b$.

By the strong inductive hypothesis, since $2 \leq a, b \leq k$, both a and b are the product of primes. Thus $k + 1 = a \cdot b$ is the product of primes.

Mathematical Induction

Mathematical induction is a means of proving a theorem by showing that if it is true of any particular case it is true of the next case in a series, and then showing that it is indeed true in one particular case

Suppose we have some statement P(n) and we want to demonstrate that P(n) is true for all n.

To prove that p(n) is true for all positive integer n where p(n) is a propositional function then we do two steps

Basic step: we verify that p(0) or p(1) is true

Inductive hypothesis: we assume that p(k) is true

Inductive step: we prove p(k + 1) is true

We show $p(k) \rightarrow p(k + 1)$ is true for all positive integer k assuming p(k) is true (Inductive hypothesis)

As rule of inference, this can be stated as,

$[p(1) \wedge \forall k(p(k) \rightarrow p(k + 1)] \rightarrow \forall n p(n)$

1. ***Show that if n is a positive integer then***

$1 + 2 + \ldots + n = \frac{n(n+1)}{2}$

Let p(n) be the proposition, that sum of the 1st n positive integers is n(n + 1)/2

Basic step: p(1) is true because

$1 = 1(1 + 1)/2 = 1$

Inductive step :

Assume p(k) is true for an arbitrary positive integer k, i.e.

$1 + 2 + 3 + \ldots + k = k(k + 1)/2$

Now, we have to show p(k + 1) is true, i.e.

$1 + 2 + 3 + \ldots k + (k + 1) = (k + 1)(k + 2)/2$

L.H.S. $= 1 + 2 + 3 + \ldots k + (k + 1)$

$= k(k + 1)/2 + (k + 1)$ (inductive hypothesis)

$= k(k + 1) + 2k + 2/2$

$= k(k + 1) + 2(k + 1)/2$

$= (k + 1)(k + 2)/2$

From the principle of mathematical induction, p(n) is true for all positive integers n.

2. ***Conjecture a formula for the sum of the first n positive odd integers. Then prove your conjecture using mathematical induction.***

The sum of the first positive odd integers for n = 1, 2, 3, 4, 5 are

$1 = 1$

$1 + 3 = 4$

$1 + 3 + 5 = 9$

$1 + 3 + 5 + 7 = 16$

$1 + 3 + 5 + 7 + 9 = 25$

Hence the sum of the first n positive odd integers is n^2 i.e.

$1 + 3 + 5 + \ldots (2n-1) = n^2$

Let p(n) denote the proposition that the sum of the first n positive odd integers is n^2

Basis step : p(1) is true a

$1 = 1^2$ (is true)

Inductive step : let p(k) is true, i.e. $1 + 3 + 5 + \ldots + (2k-1) = k^2$ is true.

Now, p(k + 1) is the statement

$1 + 3 + 5 + \ldots\ldots + (2k - 1) + (2(k + 1)-1) = (k + 1)^2$

$1 + 3 + 5 + \ldots\ldots (2k - 1) + (2k + 1) = (k + 1)^2$

Here,

$1 + 3 + 5 + \ldots\ldots + (2k-1) + (2k + 1) = k^2 + (2k + 1)$

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

k^2(inductive hypothesis)

$P(k) \rightarrow p(k + 1)$ is true, p(n) is true for all positive integers n

3. ***Show that***

$1 + 2 + 2^2 + \ldots\ldots + 2^n = 2^{n+1}-1$ for all non-negative integers.

Let p(n) be the proposition

$1 + 2 + 2^2 + \ldots\ldots + 2^n = 2^{n+1}-1$

Basis step : p(0) is true as

L.H.S.= $2^0 = 1$and R.H.S. = $2^{0+1}-1 = 2-1 = 1$

L.H.S.= R.H.S.

Inductive step :

Let $1 + 2 + 2^2 + \ldots\ldots + 2^k = 2^{k+1}-1$ holds for k

Then, p(k + 1) is

$1 + 2 + 2^2 + \ldots\ldots + 2^k + 2^{k+1} = 2^{(k+1)+1}-1 = 2^{k+2}-1$

Now,

$1 + 2 + 2^2 + \ldots + 2^k + 2^{k+1} = (2^{k+1}-1) + 2^{k+1}$ (using inductive hypothesis)

$= 2.2^{k+1}-1 = 2^{k+2}-1$

$P(k) \rightarrow p(k + 1)$ is true

P(n) holds for all n(all non-negative integer)

4. $\sum_{1=0}^{n} ar^j = ar^n = a + ar + ar^2 + \ldots + ar^n = \frac{ar^{n+1}-a}{r-1}a$ ***where $r \neq 1$***

Basis step : p(0) is true as

$\frac{ar^{0+1}-a}{r-1} = \frac{ar-a}{r-1} = \frac{a(r-1)}{r-1} = a$

Inductive step :

let p(k) holds for k, where k is non negative integer. i.e.

$a + ar + ar^2 + \ldots + ar^k = \frac{ar^{k+1}-a}{r-1}$ is true

Now p(k + 1) is the statement

$a + ar + ar^2 + \ldots + ar^k + ar^{k+1} = \frac{ar^{(k+1)+1}-a}{r-1} = \frac{ar^{(k+2)}-a}{r-1}$

L.H.S. = $a + ar + ar^2 + \ldots + ar^k + ar^{k+1}$

$= \frac{ar^{(k+1)}-a}{r-1} + ar^{k+1}$

$= \frac{ar^{(k+1)}-a+ar^{(k+2)}-ar^{(k+1)}}{r-1}$

$= \frac{ar^{(k+2)}-a}{r-1}$

$p(k) \rightarrow p(k + 1)$ is true, p(n) is true for all non-negative integers.

5. ***Prove $n<2^n$ for all positive integers n.***

Let p(n) be the proposition that $n<2^n$

Basis step : p(1) is true as $1<2^1 = 2$ is true

Inductive step : assume for positive integer k, p(k) is true, i.e. $k<2^k$

p(k + 1) is the statement $(k + 1) <2^k + 1$

we have, $k<2^k$

$k + 1 < 2^k + 1$ (adding both side 1)

note that $1<2^k$

$k + 1 < 2^k + 1$

$<2^k + 2^k$

$<2.2^k$

$<2^{k+1}$

$k + 1 < 2^{k+1}$

Since $p(k) \rightarrow p(k + 1)$ is true, $p(n)$ is true for all positive integer n.

6. ***Prove that $2^n < n!$ for every positive integer n with n>4***

Let p(n) be the proposition $2^n < n!$

Basis step : p(4) is true as $2^4 = 16 < 4! = 24$

Inductive step : Let p(k) be true for positive integer k with $k \geq 4$ i.e. $2^k < k!$

p(k + 1) is the statement $2^{k+1} < (k + 1)!$

we have, $2^{k+1} = 2^k 2 = 2.2^k < 2.k!$ (inductive hypothesis)

$< (k + 1)k!$ $\quad (2 < (k + 1))$

$< (k + 1)!$ Factorial definition.

Hence, $2^{k+1} < (k + 1)!$

$P(k) \rightarrow p(k + 1)$ is true, p(n) holds for all $n \geq 4$

7. ***An inequality for harmonic numbers. The harmonic numbers H_j, j = 1, 2, 3,..... are defined by $H_j = 1 + 1/2 + 1/3 + \ldots\ldots + 1/j$***

Show that $H_2^n \geq 1 + n/2$ where n is a non-negative integer.

Solution:

Let p(n) be the proposition that $H_2^0 \geq 1 + 0/2$

Basis step,

$H_1 \geq 1$

$1 \geq 1$

Inductive step : assume p(k) is true $H_2^k \geq 1 + k/2$ for a non-negative integer k.

Then p(k + 1) is the statement $H_2^{k+1} \geq 1 + (k + 1)/2$

Now, from definition of harmonic number

$H_2^{k+1} = 1 + 1/2 + 1/3 + + (1/2^k) + (1/2^k + 1) + (1/2^k + 2) + \ldots + 1/2^{k+1}$

$= H_2^k + (1/2^k + 1) + (1/2^k + 2) + \ldots + 1/2^{k+1}$

$\geq (1 + k/2) + (1/2^k + 1) + (1/2^k + 2) + \ldots + 1/2^{k+1}$

$\geq (1 + k/2) + 2^k * (1/2^{k+1})$

$\geq (1 + k/2) + 1/2$

$\geq (1 + (k+1)/2)$

8. ***Prove $n^3 - n$ is divisible by 3 wherever n is a positive integer.***

Let p(n) denotes $n^3 - n$ is divisible by 3

Basis step: p(1) is true as $1^3 - 1 = 0$ is divisible by 3.

Inductive step : let p(k) is true, i.e. $k^3 - k$ is divisible by 3 for a positive integer k.

Then, p(k + 1) is the statement $(k + 1)^3 - (k + 1)$ is divisible by 3.

$(k + 1)^3 - (k + 1) = (k^3 + 3k^2 + 3k + 1) - k - 1$

$= (k^3 - k) + 3(k^2 + k)$

$(k^3 - k)$ is divisible by 3 (from hypothesis) and $3(k^3 + k)$ is also divisible by 3 and so their sum is also divisible by 3.

$P(k) \rightarrow p(k + 1)$ is true p(n) is true for a positive integer n.

9. ***If S is a finite set with n element where n is a non-negative integer, then S has 2^n subsets.***

Let p(n) be the proposition that a set with n elements has 2^n subsets.

Basis steps : p(0) is true, because $2^0 = 1$ subsets which agrees with the fact that a set with zero element i.e. empty set has only one subset namely itself.

Inductive step : we assume for non-negative integer k that every set with k element has 2^k subsets. p(k + 1) is the statement that every set with k + 1 elements has 2^{k+1} subsets.

Let T be a set with k + 1 elements, then we can write $T = S \cup \{a\}$ where a is one of the element of T and $S = T - \{a\}$

(/S/ = k) subsets of T can be obtained in the following way

For each subsets of there are exactly two subsets of T, namely, X and $X \cup \{a\}$, these constitute all the subsets of T and are all distinct.

But there are 2^k subsets of S, so there are $2.2^k = 2^{k+1}$ subsets of T.

$S = \{a, b\}$ $T = \{a, b, c\}$

$X = [\{\}, \{a\}, \{b\}, \{a, b\}]$

$T = \{a, b, c\}$

Subset of T

$= [\{\},\{a\},\{b\},\{a, b\},\{\}\cup\{a\},\{a\}\cup\{c\},\{b\}\cup\{c\},\{a, b\}\cup\{c\}]$

Summarization of proof techniques

S. N.	Proof Technique	Approach to prove $p \rightarrow q$	Remarks
1.	Exhaustive proof	Demonstrate $p \rightarrow q$ for all possibilities	May only be used for finite number of cases
2.	Direct proof	Assume p, deduce q	The standard approach usually the thing to try.
3.	Proof by contraposition	Assume $\neg q$, derive $\neg p$	Use this q if as a hypothesis seems to give more ammunition then q would.
4.	Proof by contradiction	Assume $p \wedge \neg q$, deduce a contradiction	Use this when q says something is not true
5.	Vacuous proof	Show p is false	Give an example to make p is false, in $p \rightarrow q$.
6.	Trivial proof	Show q is true	Give an example to make q is true.
7.	Proof by cases	Demonstrate $p \rightarrow q$ for all cases	May only be used for known cases.
8.	Proof by equivalence	To prove $p \leftrightarrow q$ show $(p \rightarrow q) \wedge (q \rightarrow p)$	$(p_1 \leftrightarrow p_2 \leftrightarrow p_3 \ldots\ldots \leftrightarrow p_n) \leftrightarrow (p_1 \rightarrow p_2) \wedge (p_2 \rightarrow p_3) \ldots\ldots \wedge (p_n \rightarrow p_1)$

1.13 Consistency and Completeness of the System

Valid Formula

Valid formulae are those formulae which are valid under all valuations ($p \vee \neg p$) is valid formula. Valid propositional formulae are called tautologies.

Satisfiable Formula

These can be made true under at least one valuation, though not necessarily all valuations. An example of satisfible formula is $p \vee q$ which is true at p = T and q = F. Every valid formula is satisfiable.

Unsatisfiable Formula

The unsatisfiable formulas come out false under all valuations.

Soundless and Completeness of the system

The axioms are all valid and each inference rule has the following properties: If both premises are valid then the conclusion is also valid. It follows this, that, only valid formulae can be proved in a sound logical system.

If ϕ can be proved in a sound logical system then ϕ is valid. If there is a proof of ϕ then ϕ is called a theorem.

The soundless of a logical system is expressed as

If ϕ is a theorem (if ϕ can be proved), then ϕ is valid.

The completeness of the system is expressed as

If ϕ is valid then ϕ is a theorem (ϕ can be proved)

1.14 Method of Tableaux

The main problem with the axiomatic system we have described is that although it is relatively easy to check that a proof of a formula is correct, it is much harder to construct a proof. We need intuition to know which axioms to pick and when to apply modus ponens. It is even worst, if you do not know whether a formula can be proved or not.

It is the method to prove the satisfiabilty of a set of formula.

Reasons for using tableau

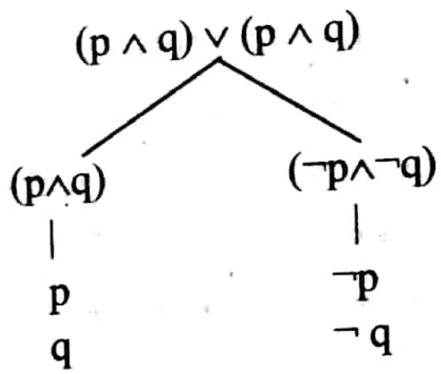
- It is quite easy to use (though some may argue, that natural deduction is easier) and
- It is very suitable for implementation on a computer.
- It breaks each formula into component upto simplest one, where contradiction is easy to spot.
- It creates a tree structure called tableau.

Parsing

When we parse a formula, we look at the string of symbols and work out how it was built up from the propositions. In other words, we recover the structure of the formula. The term comes from linguistics where the act of parsing is to determine the grammatical structure of a sentence of natural language.

Example

Let ϕ be the formula $(p \wedge q) \vee (\neg p \wedge \neg q)$ then we can parse as



Tableaux method

Note:

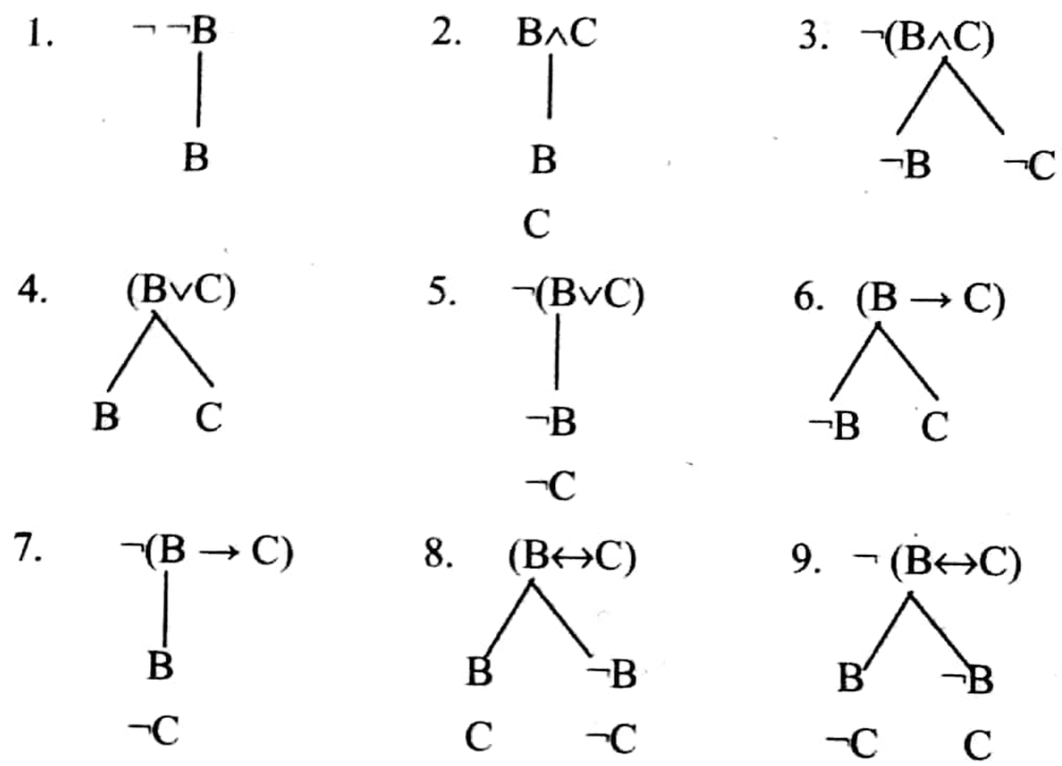
1. If only set of formula is given (i.e. conclusion is not given), if any branch is open, then the set is satisfiable sometimes called soundness.
2. If the argument (i.e. premises and conclusion) are given, we negate the conclusion. If the all branches of tree (made up of premises and negation of conclusion) then it is valid called completeness.

Tableaux construction rules

If a formula ϕ is not a literal, it has one of the following forms.

1. $\phi = \neg\neg B$
2. $\phi = B \wedge C$
3. $\phi = \neg(B \wedge C)$
4. $\phi = (B \vee C)$
5. $\phi = \neg(B \vee C)$
6. $\phi = (B \rightarrow C)$
7. $\phi = \neg(B \rightarrow C)$
8. $\phi = (B \leftrightarrow C)$
9. $\phi = \neg(B \leftrightarrow C)$

Derivation Rules in Tableaux Form



Some derivations:

$$
\begin{aligned}
(B \leftrightarrow C) &\equiv (B \rightarrow C) \wedge (C \rightarrow B) \\
&\equiv (\neg B \vee C) \wedge (\neg C \vee B) \\
&\equiv (\neg B \wedge \neg C) \vee (\neg B \wedge B) \vee (C \wedge \neg C) \vee (C \wedge B) \\
&\equiv (\neg B \wedge \neg C) \vee (C \wedge B) \\
&\equiv (B \wedge C) \vee (\neg B \wedge \neg C) \\
\neg(B \leftrightarrow C) &\equiv \neg [(B \wedge C) \vee (\neg B \wedge \neg C)] \\
&\equiv \neg (B \wedge C) \wedge \neg (\neg B \wedge \neg C) \\
&\equiv (\neg B \vee \neg C) \wedge (B \vee C)
\end{aligned}
$$

$\equiv (\neg B \wedge B) \vee (\neg B \wedge C) \vee (\neg C \wedge B) \vee (\neg C \wedge C)$

$\equiv (B \wedge \neg C) \vee (\neg B \wedge C)$

Steps in Tableaux method

Given a finite set of formulas $\phi_1, \phi_2, \ldots\ldots \phi_m$, a tableau for $\phi_1, \phi_2, \ldots\ldots \phi_m$ is a tree of formulas constructed as follows:

1. First of all, list the formulas

   ϕ_1

   ϕ_2

   ϕ_3

   .

   .

   ϕ_m

2. Starting from any formula oi, apply the derivation rule, say

   $\phi_i = (B \rightarrow C)$

   $\phi\ 1$

   ϕ_i

   .

   .

   .

   ϕ_j

   .

   ϕ_m

   B $\quad$ $\neg B$

   C $\quad$ $\neg C$

3. Now, along each branch of the tree constructed hitherto, repeat the previous steps, say $\phi_j = (\neg C \wedge D)$

   ϕ_1

   ϕ_i

   .

   ϕ_j

   .

   .

   ϕ_m

   B $\quad$ $\neg B$

   C $\quad$ $\neg C$

   $\neg C$ $\quad$ $\neg C$

   D $\quad$ D

4. After each step, we check for satisfiability along branches. Thus, the left- most branch contains a formula and its negation, namely, C and $\neg C$.

   There are, the left-most branch is not satisfiable. So, we can close-off this branch and we will not analyze this further.(indicated by bar)

   Note that we will not apply further derivations rules to it.

   We apply the previous steps along each branch that is not yet closed off until either

   i. Every branch is closed or

   ii. All non-literals along each unclosed branch is checked.

   Then in case (i), the initial set $\phi_1, \phi_2, \ldots\ldots \phi_m$ is not satisfiable. On the other hand, if (i) fails that is some branch is not closed off but (ii) holds, then the set of formulas $\phi\ 1, \phi\ 2, \ldots\ldots \phi\ m$ is satisfiable.

5. From open branch we can check for validity by valuation. For all literals p we define valuation V as

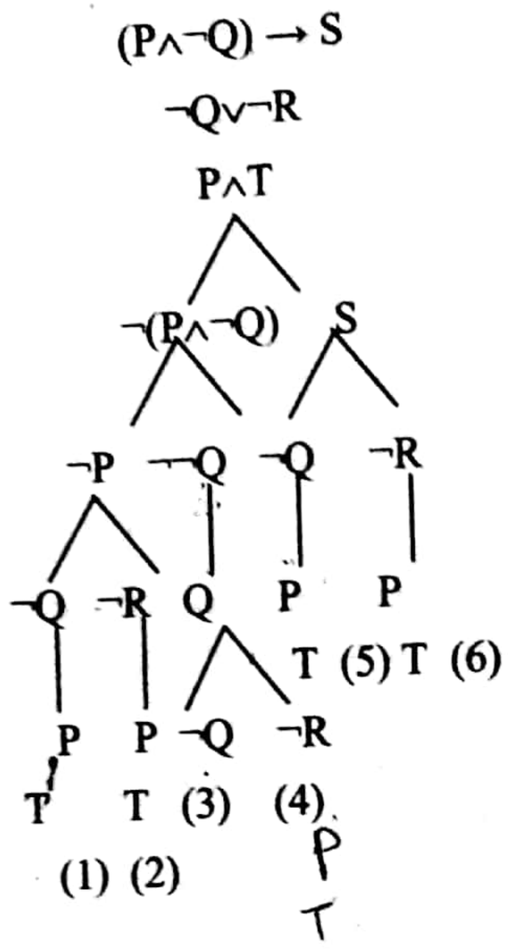
   V(p) = T if p appears along the branch

   = F if p appears along the branch

   = T or F if both p and $\neg p$ not appears in branch

Q. ***Consider the set $\phi = \{(P \wedge \neg Q) \rightarrow S, \neg Q \vee \neg R, P \wedge T\}$***

A tableaux for ϕ



For open branch (4)

V(T) = T

V(P) = T

V(R) = F

V(S) = T or F

V(Q) = T

Then

V((P ∧ ¬Q) → S) = T

V(¬Q ∨ ¬R) = T

V(P ^ T) = T

Soundness of propositional tableaur

When we say that the tableau method is sound we mean that if you can construct a closed tableau for ¬ ϕ then ϕ must certainly valid formula. This means that anything we prove by the tableau method is certainly valid. Soundness is usually the easy part to check for most inference systems.

Completeness

When we say that the tableau method is complete we mean that if ϕ is any valid formula then you can construct a closed tableau for ¬ ϕ. This means that any valid formula can be proved using the tableau method showing completeness is usually the hard part to check for most inference systems.

Corollary:

If a completed tableau for a formula ϕ is open, then ϕ is satisfiable. If a completed tableau for ϕ is closed, then ϕ is unsatisfiable.

By following corollary and the two theorems we know that the tableau method is both sound and complete.

Definition:

An argument

A1

A2

.

.

An

B

is said to be valid if the conclusion B is a tautological consequence of the premises A1, A2,.....An

Now, {A1, A2,..... , An}=B

Q. ***P → (Q∧R)***

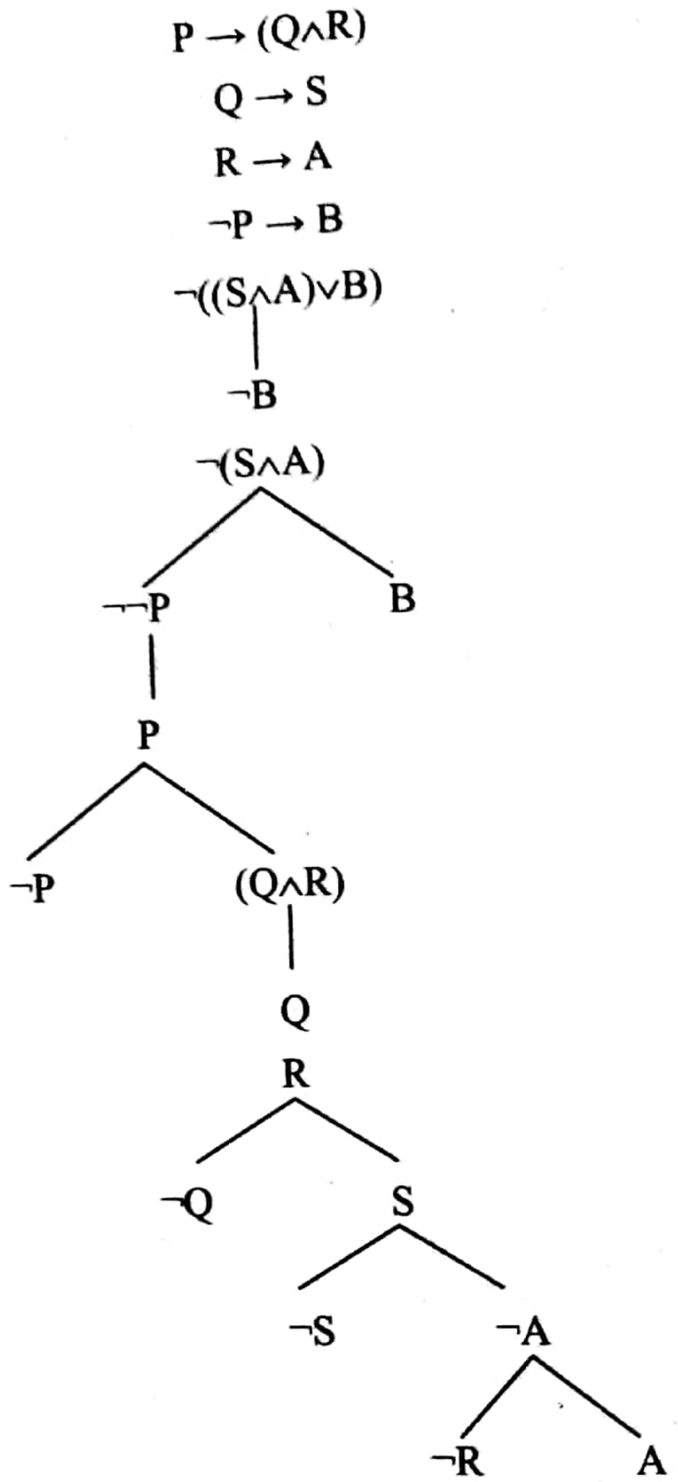
Q → S

R → A

¬P → B

(S∧A)∨B

This argument is valid iff for any valuation 'V' satisfying the hypothesis, V also satisfied the conclusion. Or in other words if the hypothesis plus the negation of the conclusion is not satisfiable. So, we check for satisfiability.



Every branch closes hence the argument is valid.

SOLVED NUMERICALS

1. ***"A student in this class has not read the book ", and "Everyone in this class passed the first exam" implies that conclusion "someone who passed the first exam has not read the book."*** ***[2076 Baisakh, Back]***

Solution:

Propositions:

C(x): x is student in this class

R(x): x has read the book

P(x): x has passed the first exam

Hypothesis:

$\exists x(C(x) \wedge \neg R(x))$

$\forall x(C(x) \rightarrow P(x))$

Conclusion:

$\exists x(P(x) \wedge \neg R(x))$

S.N.	Statement	Reason
1.	$\exists x(C(x) \wedge \neg R(x))$	Hypothesis
2.	$C(a) \wedge \neg R(a)$	Using Existential Instantiation in 1.
3.	$\forall x(C(x) \rightarrow P(x))$	Hypothesis
4.	$C(a) \rightarrow P(a)$	Using Universal Instantiation in 3.
5.	$C(a)$	Using Simplification in 2.
6.	$P(a)$	Using Modus Ponens in 4 and 5.
7.	$\neg R(a)$	Using Simplification in 2.
8.	$P(a) \wedge \neg R(a)$	Using Conjunction in 6 and 7.
9.	$\exists x(P(x) \wedge \neg R(x))$	Using Existential generalization in 8.

2. ***Using rules of inference, prove that the hypotheses "If I get my Christmas bonus and my friend are free, I will take a road trip with my friends.", "If my friends don't find a job after Christmas, then they will be free." and "I got my***

Christmas bonus and my friend did not find a job after Christmas " Lead to the conclusion "I will take a road trip with my friends." ***[073 Bhadra]***

Solution:

Propositions:

p: I get my Christmas bonus

q: My friend are free

r: I will take a road trip with my

s: My friend find a job after Christmas

Hypothesis

$(p \wedge q) \rightarrow r$, $\neg s \rightarrow q$, $p \wedge \neg s$

Conclusion

r

S. N.	Statement	Reason
1.	$\neg s \rightarrow q$	Hypothesis
2.	$p \wedge \neg s$	Hypothesis
3.	$\neg s$	From 2 Simplification
4.	Q	From 1 and 2 Modus Ponens
5.	P	From 2 Simplification
6.	$p \wedge q \rightarrow r$	Hypothesis
7.	$p \wedge q$	From 4 and 5 Conjunctions
8.	R	From 6 and 7 Modus Ponens

3. ***Using resolution principle, prove that the hypotheses "If today is Tuesday then I will have a test in Discrete Structure or Microprocessor". "If my Microprocessor teacher is sick then I will not have a test in Microprocessor." and "Today is Tuesday and my Microprocessor teacher is sick." lead to the conclusion that "I will have a test in Discrete Structure".*** ***[072 Ashwin]***

Solution:

Propositions:

p: Today is Tuesday

q: I will have a test in Discrete Structure

r: I will have a test in Microprocessor

s: My Microprocessor teacher is sick

Premises:

$p \rightarrow (q \vee r)$

$s \rightarrow \neg r$

$p \wedge s$

Conclusion:

q

Puttinng in clause form:

c1 : $\neg p \vee (q \vee r)$	Disjunction form of premises $p \rightarrow (q \vee r)$
c2: $\neg s \vee \neg r$	Disjunction form of Premises $s \rightarrow \neg r$
c3: p	From premises $p \wedge s$
c4: s	From premises $p \wedge s$
c5: $\neg q$	Neagating Conclusion
c6: : $\neg p \vee q \vee \neg s$	From c1 and c2
c7: $q \vee \neg s$	From c3 and c6
c8: q	From c4 and c6
c9: Υ	From c7 and c8

4. ***Use resolution to show the hypothesis "It is not raining or Sita has her umbrella," "Sita does not have her umbrella or she does not get wet," and "It is raining or Sita does not get wet" imply that "Sita does not wet."*** ***[071 Bhadra]***

Solution :

r: It is raining

u: Sita has her umbrella

w: Sita gets wet

hypothesis:

$\neg r \vee u$, $\neg u \vee \neg w$, $r \vee \neg w$

Conclusion:

$\neg w$

Putting in clause form

c1: $\neg r \vee u$	hypothesis
c2: $\neg u \vee \neg w$	hypothesis
c3: $r \vee \neg w$	hypothesis
c4: w	negating conclusion
c5: $\neg r \vee \neg w$	from c1 and c2
c6: $\neg w$	from c3 and c6
c7: Υ	from c4 and c6

hence, the argument is true.

5. ***Construct an argument using rules of inference to show that the hypothesis "all movies produced a movie are wonderful.. John Sayles produced a movie about coal miners" imply the conclusion, "There is a wonderful movie about coal miners." You are required to show each step and give reasons for those steps before you come to the desired conclusion from the hypothesis.*** ***[071 Magh]***

Solution:

J(x): x is movies produced by John Sayles

C(x): x is movie about coal miners.

W(x): x is wonderful movie.

Hypothesis:

$\forall x(J(x) \rightarrow W(x)),\ \exists x(J(x) \wedge C(x))$

Conclusion:

$\exists x(W(x) \wedge C(x))$

S.N.	Statement	Reason
1.	$\forall x(J(x) \rightarrow W(x))$	Hypothesis
2.	$\exists x(J(x) \wedge C(x))$	Hypothesis
3.	$J(a) \wedge C(a)$	Using existential instantiation on 2.
4.	$J(a) \rightarrow W(a)$	Using universal instantiation on 1.
5.	$J(a)$	Using simplification on 3.
6.	$W(a)$	Using modus ponens on 4 and 5.
7.	$C(a)$	Using simplification on 3.
8.	$W(a) \wedge C(a)$	Using conjunction on 6 and 7.
9.	$\exists x\ (W(x) \wedge C(x))$	Using existential generalization on 8.

6. ***Let Q(x, y) denote (x + y = 0). What is truth values of quantification $\exists y \forall x Q(x, y)$ and $\forall x \exists y Q(x, y)$.*** ***[2076 Baisakh, Back]***

Solution:

Here,

Q(x, y) denotes x + y = 0

$\exists y \forall x Q(x, y)$ means that " There is a number y such that for every real number x, for Q(x, y) i.e. x + y = 0.

It is false when for all value of y, there exists a x such that Q(x, y) is false.

It is true when there exists a y such that Q(x, y) is true for every x.

$\forall x \exists y Q(x, y)$ means that "For all real number x, there is a real number y, such that Q(x, y) i.e. x + y = 0.

It is true when for every value of x there exists a y such that Q(x, y) is true.

It is false when there exists a x such that for every y Q(x, y) is false.

7. ***Using mathematical induction technique, prove that the following statement is true: $3 + 3*5 + 3*5^2 + 3*5^n = 3(5^{n+1}-1)/4$ whenever n is nonnegative integer.*** ***[072 Ashwin]***

Hints:

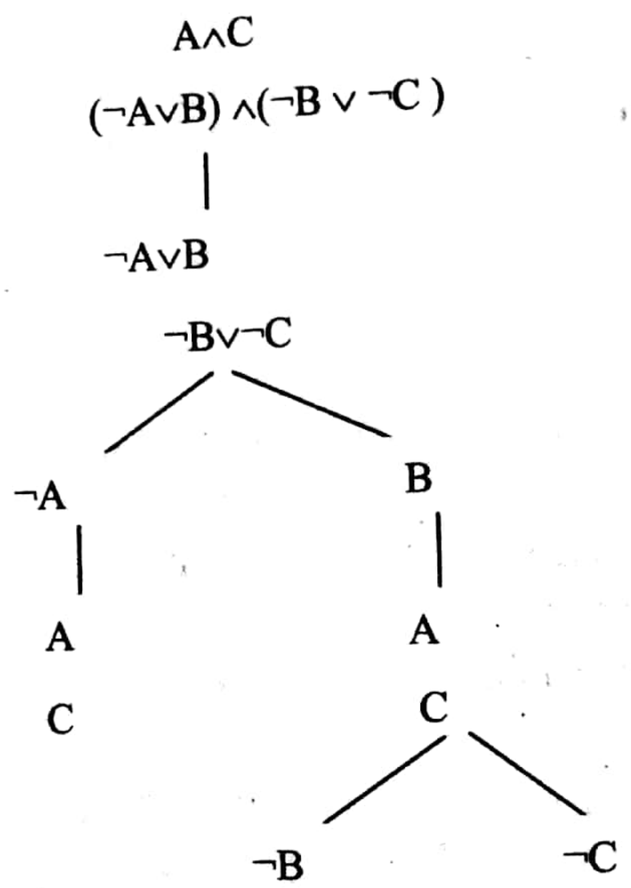
$$p(k+1) = 3*(5^{k+1}-1)/4 + 3*5^{k+1}$$
$$= (3*(5^{k+1}-1) + 12*5^{k+1})/4$$
$$= (15*(5^{k+1})-3)/4$$

$= (3*5*(5^{k+1})-3)/4$

$= 3*(5^{k+2}-1)/4$

8. ***Show that {A ∧ C, (¬A ∨ B) ∧(¬B ∨ ¬C)} unsatisfiability of the given set using method of tableaus. Prove that " If n is an integer and 3n + 2 is odd, then n is odd. [2076 Baisakh, Back]***

Solutions:



In all branches there is proposition and its negation exists, so it is unsatisfiable.

Here,

p → q(if 3n + 2 is odd then n is odd)

¬q → ¬ p(if n is not then 3n + 2 is even)

Let n is even,

N = 2k

Now, 3n + 2 = 3*2k + 2

= 2(3k + 1)

Thus 3n + 2 is even

The negation of conclusion of conditional statement implies that the hypothesis is false, so the original conditional statement is true.

9. ***Use mathematical induction to prove that $7^{n+2} + 8^{2n+1}$ is divisible by 57 for every non negative integer n. [2076 Baisakh, Back]***

Solution:

Let P(n) denotes $7^{n+2} + 8^{2n+1}$ is divisible by 57

Basis step:

P(1) is true as $7^{0+2} + 8^{2*0+1} = 49 + 8 = 57$

Inductive step: Let P(k) is true

i.e. $7^{k+2} + 8^{2k+1}$ is divisible by 57 for every non-negative integer k.

Then P(k + 1) is the statement $7^{(k+1)+2} + 8^{2(k+1)+1}$ is divisible by 57

$7^{(k+1)+2} + 8^{2(k+1)+1}$

$= 7^{k+2}\,7^1 + 8^{2k+1}8^2$

$= 7^1*7^{k+2} + 8^{2k+1}*64$

$= 7*7^{k+2} + 7*8^{2k+1} + 57*8^{2k+1}$

$= 7*(7^{k+2} + 8^{2k+1}) + 57*8^{2k+1}$

It is divisible by 57 because in first part $(7^{k+2} + 8^{2k+1})$ is divisible by 57 as from inductive step and in second part is multiple of 57, so the second part is also divisible by 57. So, $7^{n+2} + 8^{2n+1}$ is divisible by 57 for every non negative integer n.

10. ***Use mathematical induction to show that 6 divides n^3-n whenever n is a non negative integer.[071 magh]***

Solution:

Let p(n) denotes "n^3-n is divisible by 6"

Basis step:

P(1) is true as 1_3-1 = 0 is divisible by 6.

Inductive step:

Let p(k) is true, i.e. k^3-k is divisible by 6 for a non negative integer k.

Then, p(k + 1) is the statement (k + 1)3-(k + 1) is divisible by 6.

$(k + 1)^3 - (k + 1) = (k^3 + 3k^2 + 3k + 1) - k - 1$

$= (k^3 - k) + 3(k^2 + k)$

$= (k^3 - k) + 3k(k + 1)$

Here, (k^3-k) is divisible by 6 (from hypothesis)and $3k(k^3 + 1)$ is also divisible by 6 because k(k + 1) is always even number as k is non negative integer. If k is odd then (KH) becomes even and if k is even then obviously k(k + 1) becomes even.

Since k(k + 1) is always even number, so, we can write it as 2^n where n is some integer.

Then,

$p(k + 1) = 6M + 3.2N$

$= 6(M + n)$

$= 6A.$

$p(k) \rightarrow p(k + 1)$ is true, p(n) is true for a non negative integer n.

11. ***Use mathematical Induction to show that***

$1^3 + 2^3 + \ldots\ldots + n^3 = [n(n + 1)/2$ ***[071 Bhadra]***

Solution:

Let p(n) denotes $1^3 + 2^3 + \ldots\ldots + n^3 = [n(n + 1)/2]^2$

Basis step

p(1) is true as $1^3=(1(1 + 1)/2)^2$

1 = 1

Inductive step:

Let p(k) is true, i.e.

$1^3 + 2^3 + \ldots + k^3 = [k(k + 1)/2]^2$

Then p(k + 1) is

$1^3 + 2^3 + 3^3 + \ldots + k^3 + (k + 1)^3 = [(k + 1)(k + 2)/2]^2$

L.H.S

$1^3 + 2^3 + 3^3 + \ldots\ldots + k^3 + (k + 1)^3$

$= (k(k + 1)/2)^2 + (k + 1)^3$

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

$= (k + 1)^{2(}k^2 + 4k + 4)/4$

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

$= ((k + 1)(k + 2)/2)^2$

= R.H.S

$P(k) \rightarrow P(k + 1)$ is true, p(n) is true.

12. ***If P = F, Q = T, S = T, R = F, then find truth value of***

a. $(S \rightarrow (P \wedge \neg R)) \wedge ((P \rightarrow (R \vee Q)) \wedge S)$

b. $((P \wedge \neg Q) \leftrightarrow s(Q \wedge R)) \rightarrow (S \vee \neg Q)$ ***[071 Magh]***

Solution:

a. $S \rightarrow (P \wedge \neg R)) \wedge ((P \rightarrow (R \vee Q)) \wedge S)$

$= (T \rightarrow (F \wedge \neg F)) \wedge ((F \rightarrow (r \vee T)) \wedge T)$

$= (T \rightarrow (F \wedge T)) \wedge ((F \rightarrow T) \wedge T)$

$= (T \rightarrow F) \wedge (T \wedge T)$

$= F \wedge T$

$= F$

b. $((P \wedge \neg Q) \leftrightarrow (Q \wedge R)) \rightarrow (S \vee \neg Q)$

$= ((F \wedge \neg T) \leftrightarrow (T \wedge F)) \rightarrow (T \vee \neg T)$

$= ((F \wedge F) \leftrightarrow (F)) \rightarrow (T \vee F)$

$= (F \leftrightarrow F) \rightarrow T$

$= T \rightarrow T$

$= T$

13. ***State the converse, contrapositive and inverse for the conditional statement, "I go to the beach whenever it is a sunny summer day." [071 Bhadra]***

Solution:

P: I go to the beach

Q: it is a sunny summer day

Converse: q-p

Whenever I go to the beach, it is a sunny day.

Contrapositive: q-p

Whenever I do not go to beach, it is not a sunny summer day.

Or

If I do not go to the beach whenever it is not a sunny summer day.

14. ***Why is a tableau method important in proposition logic? Draw the tableau for the formula***

$\phi = (p \wedge \neg q) \rightarrow s$

Where $\neg$ denotes the nagation of a variable, $\wedge$ denotes the conjuction of variables and $\rightarrow$ denotes the implication. [071 Bhadra]

Solution:

Tableau method is important in propositional logic because it is quite easy to use and suitable for implementation on a computer.

Tableau method breaks each to use formula into components upto simplest one, where contradiction is easy to spot. This method create a tree structure called tableau, which make easy and unambigious solution of the problem.

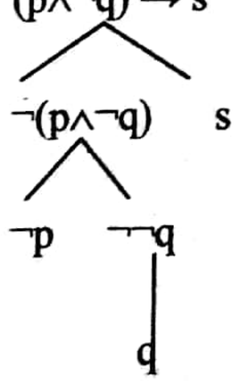
When we parse a formula, we look at the string of systems and work out how it was built up from the propositions. in other words, we recover the structure of the formula.

The main problem with the axiomatic system is that although it is relatively easy to check that a proof of a formula is correct, it is much harder to construct a proof. We need intuition to know which axioms to pick and when to apply modus ponens. It is ever worse if we do not know Whether a formula can be proved or not.

Now,

$\phi = (p \wedge q) \rightarrow s$

$(p \wedge \neg q) \rightarrow s$ since $(p \wedge \neg q) \rightarrow s \equiv \neg(p \wedge \neg q) \vee s$



Finite State Automata

2.1 Sequential Circuits and Finite State Machine

2.1.1 Combinational circuit and Sequential circuit

Combinational circuits have no memory or interm states, their output depend only on the current values of their inputs.

Sequential circuits contain memory element or internal states. The output depends not only on its current inputs but also on the past history of those inputs.

Finite state machines have internal states, so their output may depend not only on its current inputs but also on the past state or past history of those inputs.

2.1.2 Finite State Machine

A finite state machine is a computation model with a primitive internal memory that can be implemented with hardware or software and can be used to simulate sequential logic and some computer programs. Finite state automata generate regular languages.

Definition

A finite state machine M consists of

a. A finite set I of input symbols.
b. A finite set O of output symbols.
c. A finite set S of states.
d. A next state function f from SXI into S.
e. An output function g from SXI into O.
f. An initial state

We write $M = (I, O, S, f, g, \sigma)$

Example

Let $I = \{a, b\}$, $O = \{0,1\}$ and $S = \{\sigma_0. \sigma_1\}$

Define the pair of functions

f: SXI → S and g: SXI → O by the rules given below.

	f		g	
S/I	a	b	a	b
σ_0	σ_0	σ_1	0	1
σ_1	σ_1	σ_1	1	0

Then M = (I, O, S, f, g, σ_0) is a FSM

$f(\sigma_0, a) = \sigma_0$ $\quad g(\sigma_0, a) = 0$

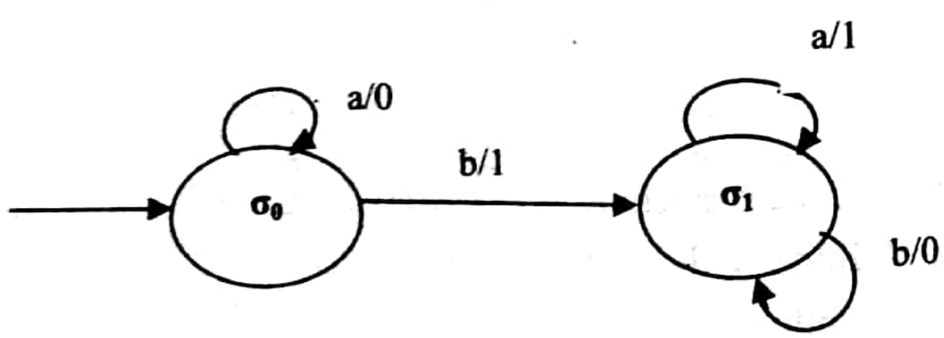
$f(\sigma_0, a) = \sigma 1$ $\quad g(\sigma_0, b) = 1$

$f(\sigma 1, a) = \sigma 1$ $\quad g(\sigma 1, a) = 1$

$f(\sigma 1, a) = \sigma 1$ $\quad g(\sigma 1, b) = 0$

Transition diagram

The transition diagram is a digraph. The vertices are the states. The initial state is indicated by an arrow. If we are in state σ and inputting I causes output O and moves us to state σ', we draw a directed edge from vertex σ to vertex σ' and label it i/o.



Let M = {I, O, S, f, g, σ } be a FSM. The transition diagram of M is a diagraph G whose vertices are the members S. an arrow designates the initial state. A directed edge (σ1, σ2) exist in G. there exists an input I with f(σ1, i)= σ2. In this case, if g(σ1, i) = The edge (σ1, σ2) is labeled i/o.

Definition

Let M = (I, O, S, f, g, σ) be a FSM. An I/P string for M is string over I.

The string

$y_1 \ldots\ldots y_n$

is the o/p string for M corresponding to the i/p string

$x_1 \ldots\ldots x_n$

If there exists states $\sigma_0, \ldots\ldots \sigma_n \in s$

With $\sigma_0 = \sigma$

$\sigma_i = f(\sigma_{i-1}, x_i)$ for i = 1,....., n

$y_i = g(\sigma_{i-1}, x_i)$ for i = 1,....., n

Example

1. ***Design a FSM that performs serial addition serial adder accepts pair of bits The i/p set will be {00,01,10,11}***

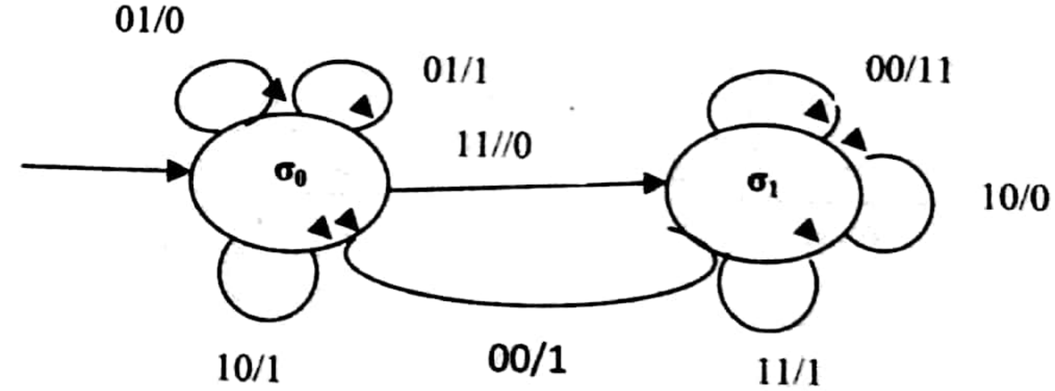
The o/p set is {0,1}

Given an i/p x, y

i. either add x and y or ***ii. add x, y and 1***

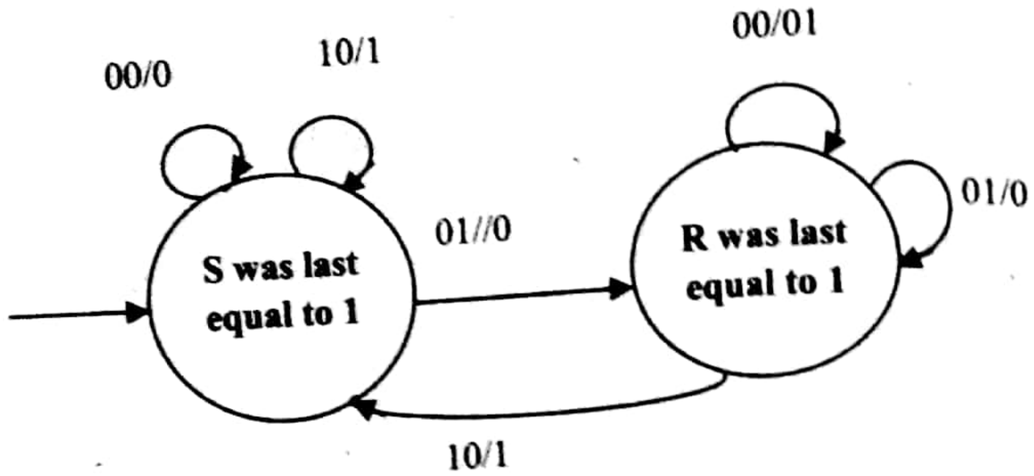
Depending on whether the carry bit was 0 or 1. So, we have two states c (carry) and NC (no carry). The initial state is NC.

Transition diagram:



The SR flip flop

S	R	Q
1	1	Not allowed
1	0	1
0	1	0
0	0	1 if s was last equal to 1 0 if R was last equal to 1



Set means output =1

Reset means output = 0

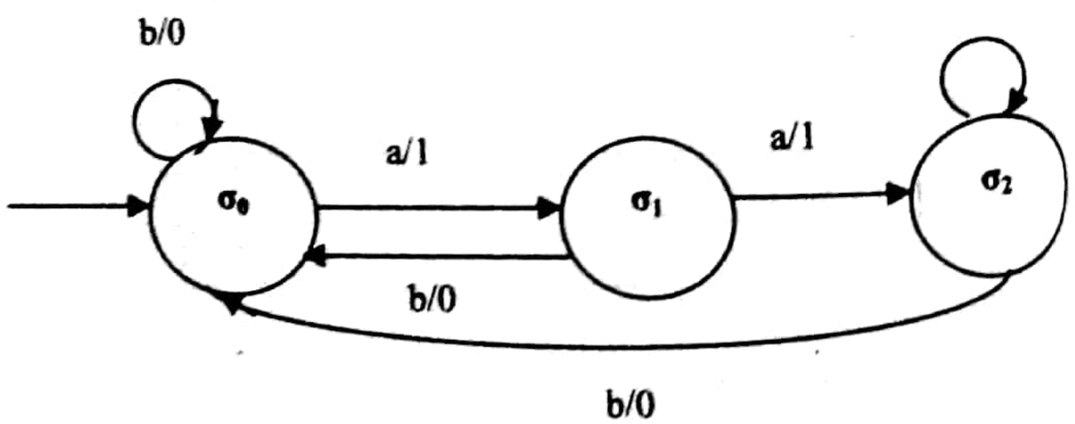
2.2 Finite State Automata

Finite state automaton is a special kind of FSM. It has no output and no output function but has accepting or final state.

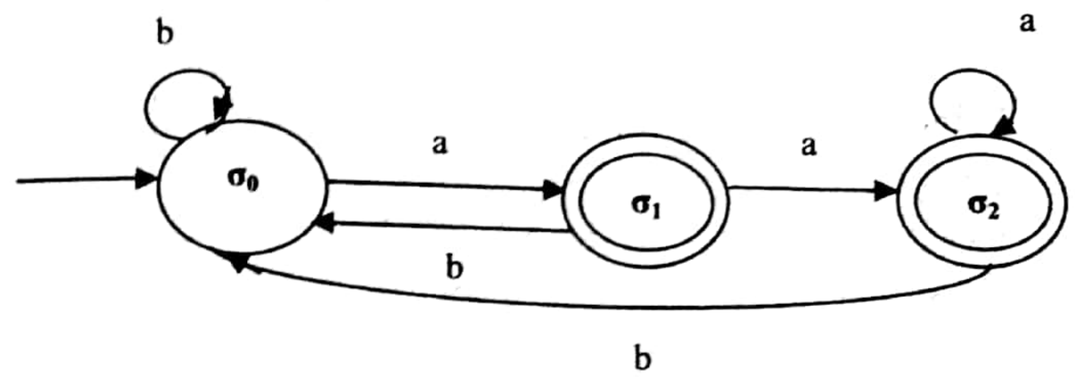
A FSA A = {I, O, s, f, g, σ} is a FSM in which the set of o/p symbols is {0,1} and where the current state determines the last output. Those states for which the last output was 1 are called accepting states.

E.g. The initial state is σ_0

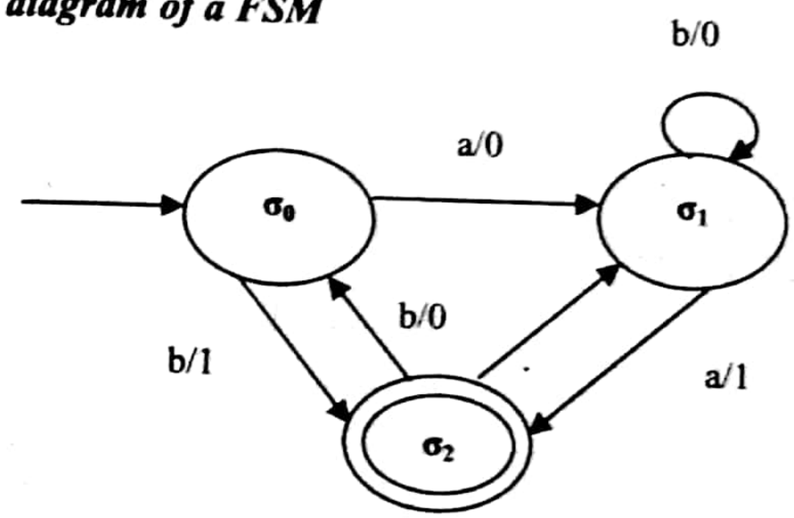
	f		g	
s/I	a	b	a	b
σ_0	σ_1	σ_0	1	0
σ_1	σ_2	σ_0	1	0
σ_2	σ_2	σ_0	1	0

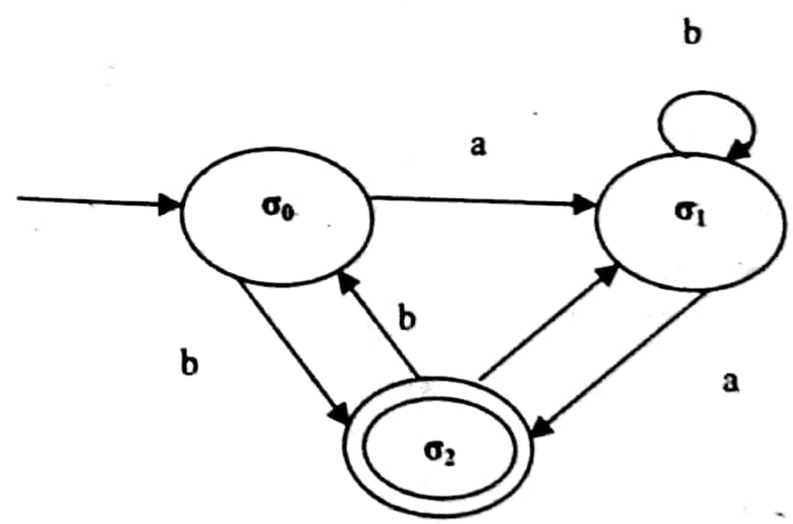


If we are in state σ_0, the last output was 0. If we are in either state σ_1 or σ_2, the last o/p was 1. Thus A is FSA. The accepting states are σ_1 and σ_2.



Q. Draw the transition diagram of the FSA as a transition diagram of a FSM





σ_2 is an accepting states

Its incoming edges with a and b.

The other states σ_0 and σ_1 are not accepting state so that they have incoming along with only a.

Definition

A finite state automaton is similar to finite state machine but with no output, and with a set of states called accepting or final states. FSA can be regarded as.

1. A finite set of input symbols (I)
2. A finite set of states symbols (S)
3. A next state function f from SXI into S.
4. A subset A of S of accepting states
5. An initial state $\sigma \in S$

   $A = (I, S, f, A, \sigma)$

Example

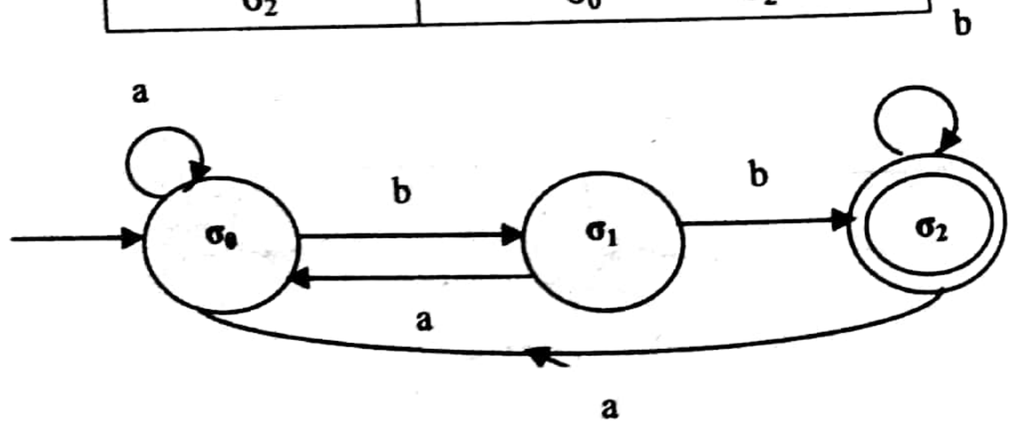
Transition diagram of FSA

$A = (I, S, f, A, \sigma)$ where

$I = \{a, b\}$, $S = \{\sigma_0, \sigma_1, \sigma_3\}$, $A = \{\sigma_2\}$

$\sigma = \sigma_0$ and f is given by table below

	F	
s/I	a	b
σ_0	σ_0	σ_1
σ_1	σ_0	σ_2
σ_2	σ_0	σ_2



Incoming is i/p to a FSA, we will end at either an accepting or a non accepting state. The state of this final state determines whether the string is accepted by FSA.

Let A(I, S, f , A, σ) be a FSA. Let $\alpha = x_1, \ldots\ldots x_n$ be a string over I. If there exists states $\sigma_0 \ldots\ldots \sigma_n$ satisfying.

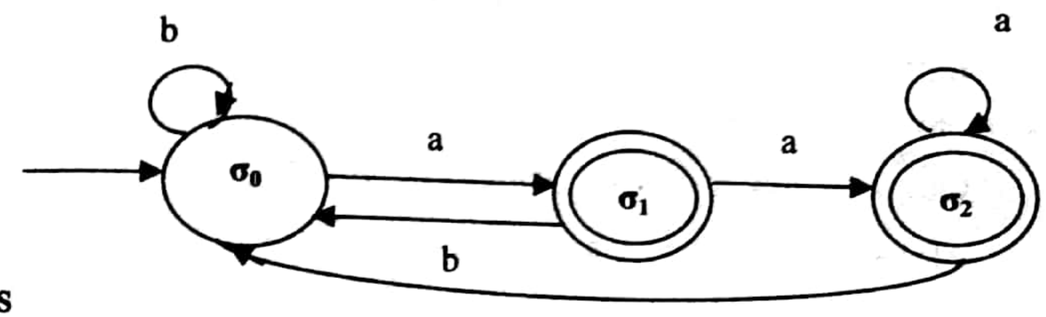
a. $\sigma_0 = \sigma$

b. $f(\sigma_{i-1}, x_i) = \sigma_i$ for $i = 1, \ldots\ldots, n$

c. $\sigma_n \in A$

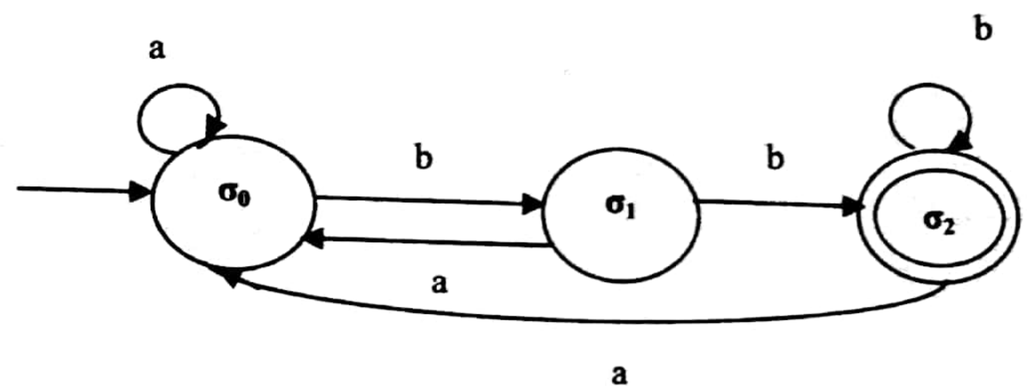
We say that α is accepted by A. The null string is accepted iff $\sigma \in A$.

Is the string abaa accepted by FSA



Yes

Is the string abbabba accepted by FSA



No

Q. ***Design FSA that accepts precisely those strings over {a , b} that contain an odd number of a's.***

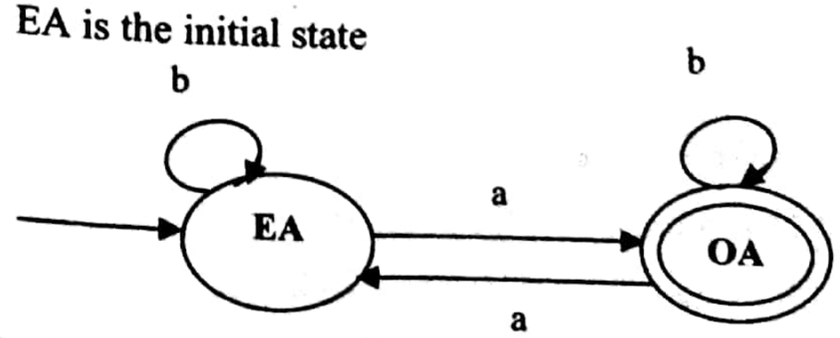
Two states

Odd A = (OA)

Even A = (EA)

OA is accepted state

EA is the initial state



If two FSA accept precisely the same things, we say that the automata are equivalent. FSA A and B are equivalent if Ac (A) = Ac (B)

Algorithm

This algorithm determines whether a string over {a, b} is accepted by FSA.

Input: n, the length of the string (n = 0 designation the null string); s_1 s_2 ..... s_n, the string.

Output : 'Accept' if the string is accepted

'Reject' if the string is not accepted procedure fsa (s, n)

State : 'E'

For i = 1 to n do

begin

If state = 'E' and S_i = 'a' then

State = '0'

If state ='0' and S_i = 'a' then

state ='E'

End

If state = '0' then

Return ("Accept")

Else

Return ("Reject")

End fsa

2.3 Language and Grammar

Alphabet or vocabulary

Alphabet or vocabulary is a finite non empty set of systems.

It is denoted by $\sum$ or A or v

V = {0,1} is a binary alphabet

V = {a, b,....z} is a set of lower case letter.

String(w)

A finite sequence of symbols taken from some alphabet is called string.

Word or sentence is string of finite length of element of symbols.

0100 is string from alphabet v = {0,1}

Concatenation (vw): vw is string v, then string w.

W^n: string w repeated n times

V^+: all possible string using v.

Empty string (λ or ∈ or e)

Zero occurance of symbols is null or empty string.

Length of string

The number of symbols present in the string.

W = 0100

|w| = 4

|∈| = 0

2.3.1 Language

Natural Language is a method of combining words used to understand by a considerable community. Formal languages are used to model natural languages and to communicate with computers.

The rules of a natural language are very complex and difficult to characterize completely.

It is possible to specify completely the rules by which certain formal languages are constructed.

Definition

The set of all words or string including zero or v is denoted by V*

Language is set of strings all of which are taken from V*

A formal language over v is subset of V^*

$L \subseteq V^*$

$V = \{0, 1\}$

$V^0 = \{\ \}$

$V^1 = \{0,1\}$

$V^2 = VUV$

$= \{0,1\} U \{0,1\}$

$= \{00,01,10,11\}$

$V^* = U_{i=0}^{\infty} V^i$

$V^+ = U_{i=1}^{\infty} V^i$

V^+ consist of all words formed by concatenating a finite number but not null.

2.3.2 Grammar

A way to determine the structure of a language is with a grammar. In order to define a grammar, we need two kinds of symbols: non terminal, used to represent given subsets of the language, and terminal, the final systems that occur in the strings of the language.

A grammar defining formal language l is a quadtruple

$G = (N, T, R, S)$

N = set of non terminal

T = set of terminals

R = set of production

S = start symbol

T is alphabet

A grammar is a way to specify the set of all legal sentences of a language.

We write $G = (N, T, P, \sigma)$. A production $(A, B \in P)$ is usually written $A \rightarrow B$

A must include at least one non terminal symbol whereas

B can consists of any combination of non terminal and terminal symbols.

Let,

$N = \{\sigma, S)$

$T = \{a, b\}$

$P = \{\sigma \rightarrow b\sigma, \sigma \rightarrow as, s \rightarrow bs, s \rightarrow b\}$

Then $G = (N, T, P, \sigma)$ is a grammar.

The language generated by G, written L(G) consists of all strings over T derivable from σ

For above grammar

$\sigma = b\sigma = bb\sigma = bbas = bbab$

The only derivations from σ are

$\sigma = b\sigma ... = b^n\sigma = b^n as (n \geq 0)$

$= b^n ab^m \quad (n \geq 0, m \geq 1)$

Thus, L(G) consists of the strings over {a, b} containing precisely one a that end with b.

Types of grammar (Chomsky hierarchy of grammar)

1. **Unrestricted grammar (Type 0)**

   If there is no restriction in the every production of the grammar , it is called unrestricted grammar.

   Production rule

   $\alpha \rightarrow \beta$

   Where,

   $\alpha \in (N \cup T)^*$

   $\beta \in (N \cup T)^*$

   It is recognized by Turing machine

2. **Context Sensitive Grammar (Type 1)**

   If every production is of the form

   $\alpha A \beta \rightarrow \alpha\gamma\beta$

(we may replace A with γ in the context of α and β)

$\alpha, \beta \in (N \cup T)^*$

$A \in N$

The language generated by this grammar is recognized by Linearly Bounded Automata.

3. **Context Free Grammar (Type 2)**

If the production is of the form

$A \rightarrow \gamma$

Where $A \in N$

$\gamma \in (N \cup T)^*$

Left side = single non terminal

Right side = a word in one or more symbols.

$N = \{S, A, B\}$

$T = \{a, b\}$

$P = \{S \rightarrow aA,\ A \rightarrow aAB, B \rightarrow b, A \rightarrow a\}$

Start symbol S.

The language generated by this grammar is recognized by Push down Automata.

4. **Regular Grammar (Type 3)**

If every production is of the form

$A \rightarrow a$ or $A \rightarrow aB$ or $A \rightarrow \gamma$

Where, $A, B \in N$

$a \in T$

Left side = single non terminal

Right side = either single terminal or terminal followed by a non terminal.

Recognized by Finite State Automata.

$N = \{S, A\}$

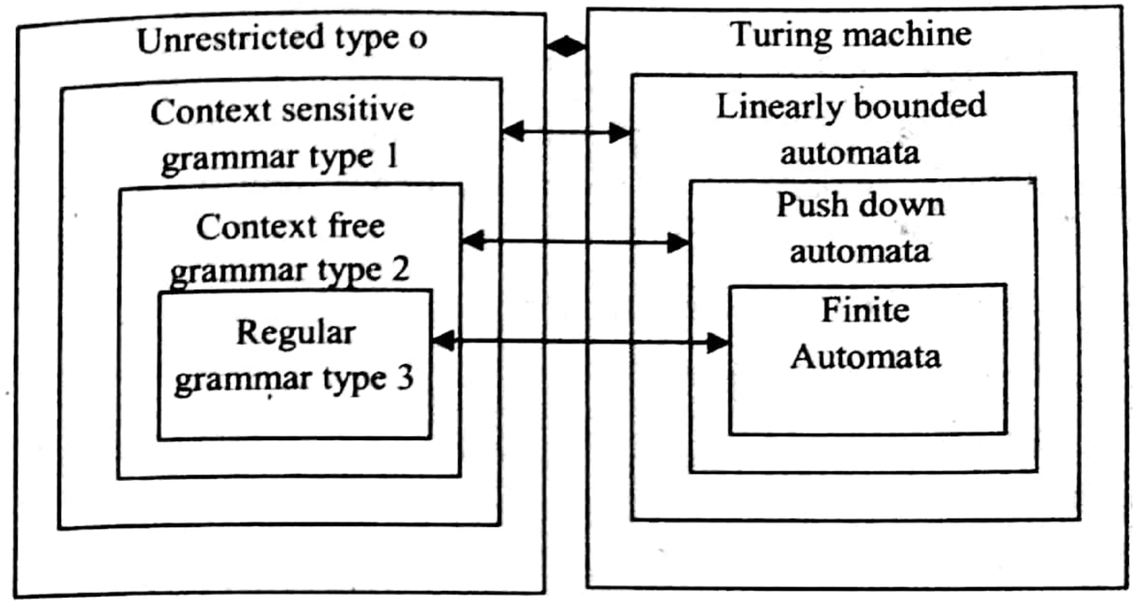
$T = \{a, b\}$

$P = \{S \rightarrow bS, S \rightarrow aA, A \rightarrow aS, A \rightarrow bA, A \rightarrow a, S \rightarrow b\}$

Starting symbol S

Grammar

Mathematical model



BNF

An alternative way to state or represent the productions context free grammar is using

1. Backus naur form and
2. Derivation tree

In BNF the non terminal symbols typically being with "<" and end with ">" the production $S \rightarrow T$ is written S::=T

Production of the from $S::=T_1$, $S::=T_2$..... $S::= T_n$ may be combined as $S::=T_1|T_2|$ ..... T_n

Grammar for integers

<digit>:: = 0|1|2|3|4|5|6|7|8|9

<integer>::=<signed integer>|< unsigned integer>

<signed interger>::=+<digit>|-<unsigned integer>

<unsigned integer>::=<digit>|<digit><unsigned integer>

Derivation of -108

<integer>::=<signed integer>::=<unsigned integer>

::= -<digit><unsigned integer>::=-<digit><digit><unsigned integer>

::= -<digit><digit><digit

::= -1<digit><digit>

::= -10<digit>

::= -108

2.4 Language and Automata

A regular grammar is a context free grammar and that a context-free grammar with no production of the form $A \rightarrow \lambda$ is a context sensitive grammar.

If every production is of the form

$A \rightarrow \alpha$ or $A \rightarrow \alpha B$ or $A \rightarrow \lambda$

Where $A, B \in N$, $\alpha \in T^* - \{\lambda\}$ then G is also a regular grammar.

Q. ***The grammar G defined by T = {a, b.c},***
N = {σ, A, B, C, D, E}, with productions
$\sigma \rightarrow aAB$, $\sigma \rightarrow aB$, $A \rightarrow aAC$, $A \rightarrow ac$, $B \rightarrow DC$, $D \rightarrow b$, $CD \rightarrow CE$, $CE \rightarrow DE$, $DE \rightarrow DC$, $Cc \rightarrow Dcc$ and starting symbol σ is context sensitive

CD = CE = DE = DC (derivation of dC from CD)

The string $a^3b^3c^3$ is in L(G)

$aAB \rightarrow aaACB = aaaCCDc = aaaCDCc$

$= aaaDCCc = aaaDCDcc = aaaDDCccc$

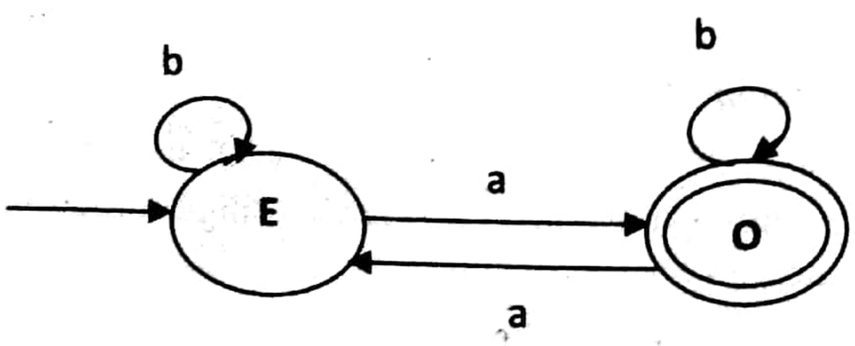
$= aaaDDDccc = aaabbbccc$

The string $a^3b^3c^3$ is in L(G)

It can be show that

$L(G) = \{a^nb^nc^n \mid n = 1,2,.....\}$

Q. ***Write the regular grammar given by FSA given below***



The terminal symbols are the input symbols {a, b}. The states E and O become the non terminal symbols. The initial state E becomes the starting symbol. The production corresponds to the directed edges.

If there is an edge labeled x from S to S',

We write

$S \rightarrow xS'$

In addition, if S is an accepting state we includes the production

$S \rightarrow \lambda$

In case

$E \rightarrow aO$, $E \rightarrow bE$, $O \rightarrow aE$, $O \rightarrow bO$

since O is accepting state we also include $O \rightarrow \lambda$

Then the grammar G = (N, T, P, E) with N = {O, E}, T = {a, b} and P consisting of above productions generate the language L(G) which is the same as the set of string accepted by above FSA.

Theorem

Let A be a FSA given as a transition diagram. Let σ be the initial state. Let T be the set of input symbols and N be the set of states.

Let P be the set of productions.

$S \rightarrow xS'$

If there is an edge labeled x from S to S' and

$S \rightarrow S'$

If S is an accepting state let G be the regular grammar.

$G = (N, T, P, \sigma)$

Then set of strings accepted by A is equal to L(G).

Reverse case

Given a regular grammar G, we want to construct a FSA to that L(G) is precisely the set of string accepted by A.

E.g.

Consider the regular grammar defined by

$T = \{a, b\}$

$N = \{\sigma, c\}$

With productions,

$\sigma \rightarrow b\sigma$, $\sigma \rightarrow aC$, $C \rightarrow bC$, $C \rightarrow b$ and starting symbol σ

The non terminal become state with σ as the initial state. For each production of the form

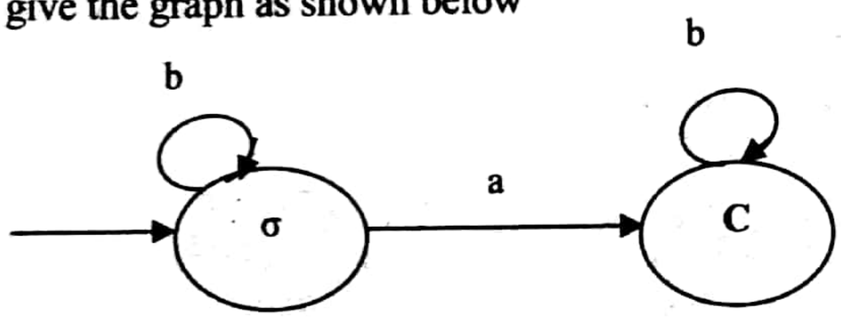
$S \rightarrow xS'$

we draw an edge from state s to S' and label it x.

The productions:

$\sigma \rightarrow b\sigma$, $\sigma \rightarrow aC$, $C \rightarrow bC$

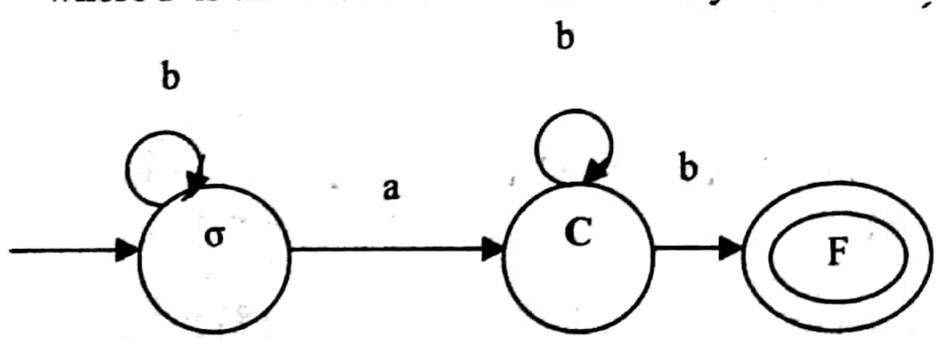
give the graph as shown below



The production $C \rightarrow b$ is equivalent to two productions.

$C \rightarrow bF$ and $F \rightarrow \lambda$

where F is an additional non terminal symbol.



The production $F \rightarrow \lambda$ tells us that F should be an accepting state.

This is not FSA

i. vertex C has no outgoing edge label a.

ii. vertex F has no outgoing at all

iii. vertex C has two outgoing edges labeled b

This is NDFA (Non Deterministic Finite Automata)

2.5 Non-deterministic Finite State Automata

Definition:

NDFSA 'A' consists of

a. A finite set I of input symbols.

b. A finite set S of states

c. A next-state function f from SxI into P(S) (subset state S)

d. A subset A of S accepting states

e. An initial state $\sigma \in S$

We write $A = (I, S, f, A, \sigma)$

In DFSA, the next-state function takes us to a uniquely defined state, whereas in NDFSA, the next-state function takes us to a set of states.

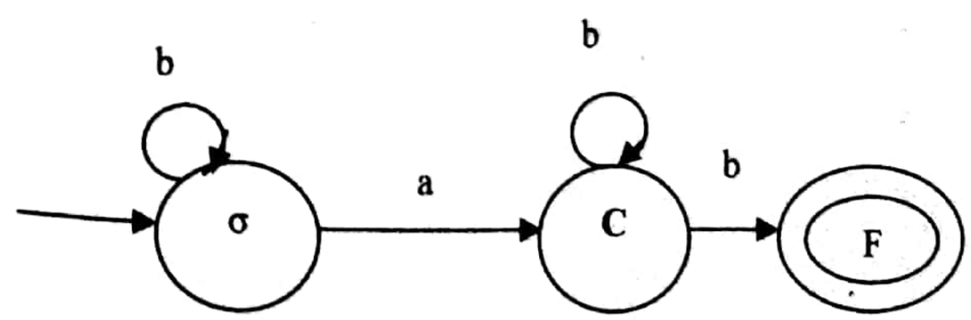
Example

$I = \{a, b\}$

$S = \{\sigma, C, F\}$

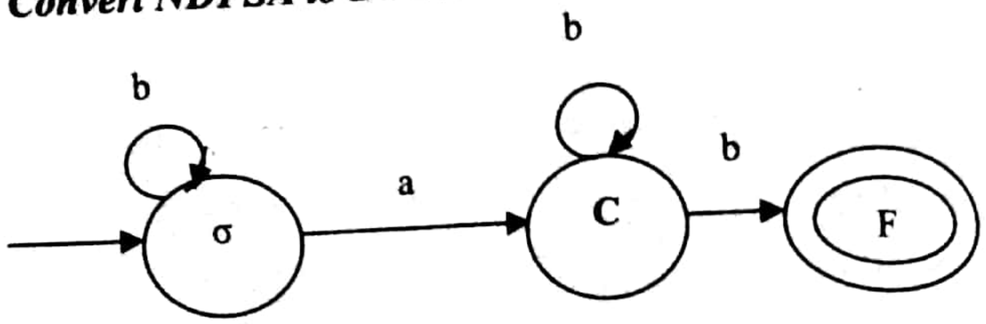
$A = \{F\}$

	f	
S \ I	a	b
Σ	{c}	{σ}
C	ϕ	{c, f}
F	ϕ	ϕ

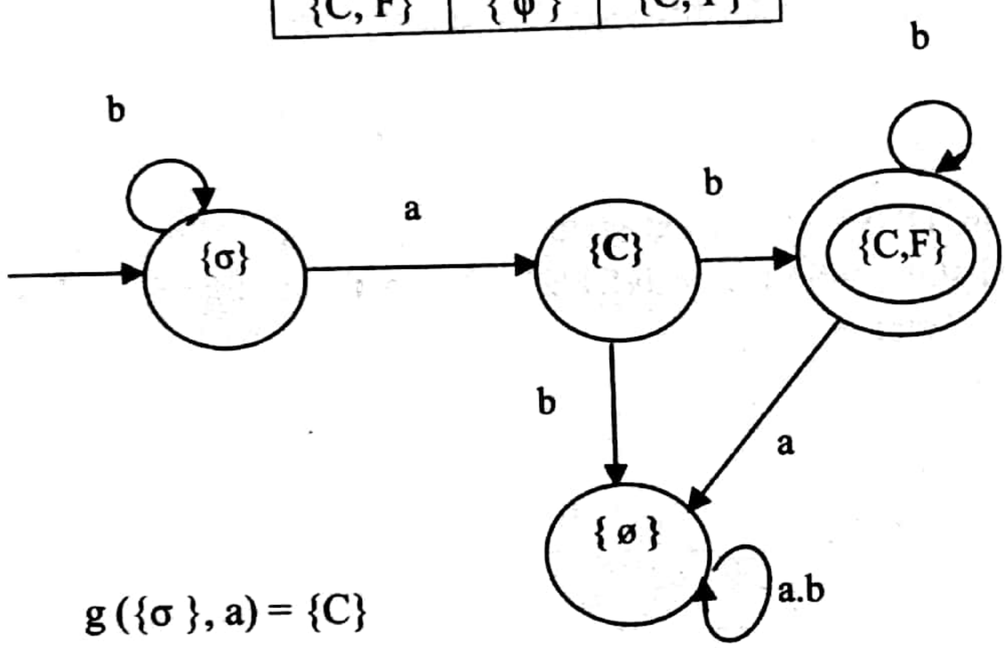


The string α = bbabb is accepted because there is at least one path representing α that ends at an accepting state.

Q. *Convert NDFSA to DFSA*



S \ I	A	B
{σ}	{c}	{σ}
{C}	{ϕ}	{C, F}
{C, F}	{ϕ}	{C, F}



g ({σ }, a) = {C}

g ({σ }, b) = { σ }

g ({C}, a) = ϕ

g ({C}, b) = E(C)UE(F) = {C, F}

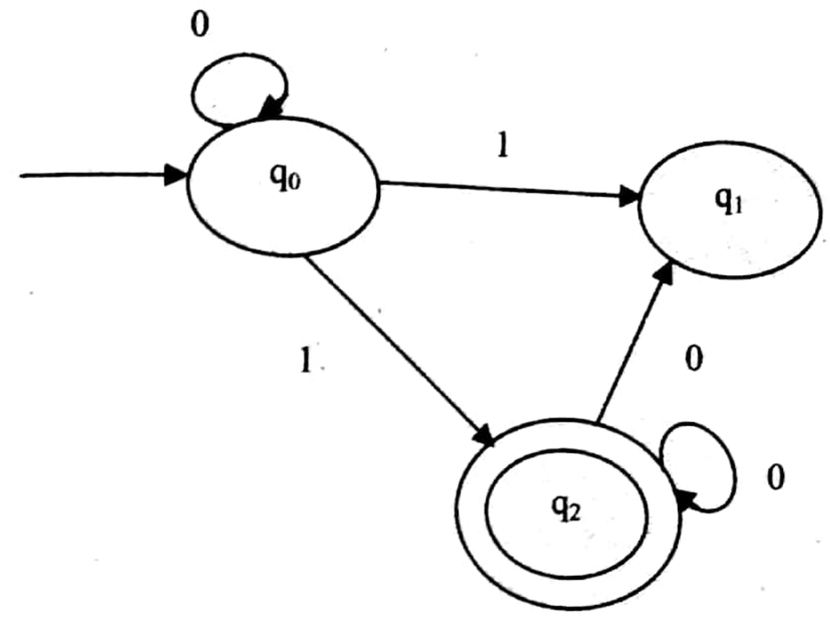
g({C, F}, a) = ϕ

g({C, F}, b) = E(C)UE(F) = {C, F}

g({F}, a) = ϕ

g({F}, b) = ϕ

Q. *Change NDFA to DFA*



Q	0	1
→ q_0	$\{q_0\}$	$\{q_1, q_2\}$
q_1	ϕ	ϕ
q_2	$\{q_1, q_2\}$	ϕ

$\delta'(q_0,0) = \delta(q_0, 0) = \{q_0\}$

$\delta'(q_0,1) = \delta(q_0,1) = \{q_0, q_1\}$ new state

$\delta'\{(q_1, q_2),0\} = \delta(q_1, 0) \cup \delta(q_2, 0)$

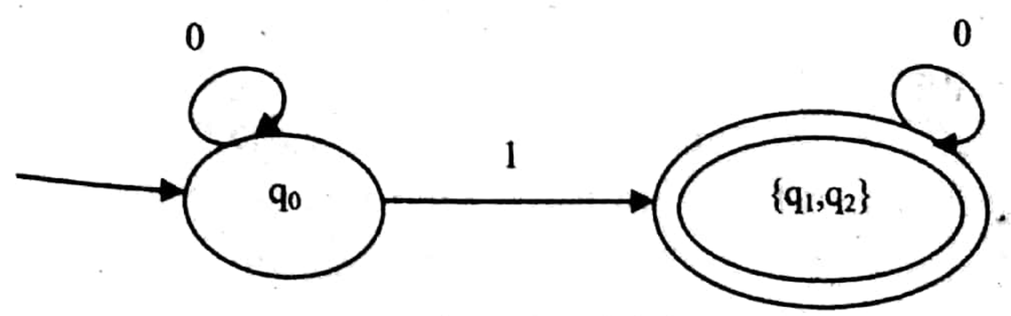
$= \phi \cup \{q_1, q_2\}$

$= \{q_1, q_2\}$ old state

$\delta'(\{q_1, q_2\}, 1) = \delta(q_1,1) \cup \delta(q_2, 1)$

$= \phi \cup \phi$

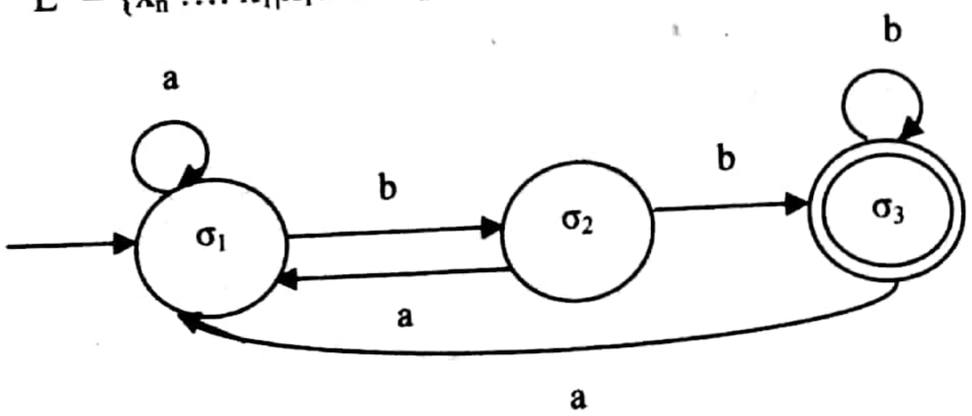
$= \phi$



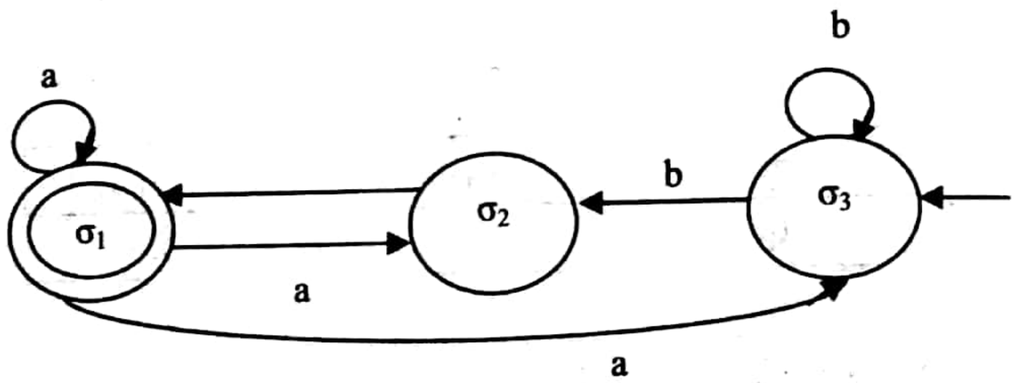
Equivalent DFA

Q. ***Let L be the set of strings accepted by the FSA shown below construct a FSA that accepts the strings***

$L^R = \{x_n \ldots x_1 | x_1 \ldots x_n \in L\}$



Reserve all arrows in above FSA and make σ_3 the starting state and σ_1 the accepting state. The result is a NDFSA that accepts L^R



Change into DFSA

$E(\sigma_3) = \{\sigma_3\}$, $E(\sigma_2) = \{\sigma_2\}$, $E(\sigma_1) = \{\sigma_1\}$

Transitions

(σ_3, b, σ_3)	(σ_2, b, σ_1)	(σ_1, a, σ_1)
(σ_3, b, σ_2)		(σ_1, a, σ_2)
		(σ_1, b, σ_3)

$g(\{(\sigma_3\}, a) = \phi$

$g(\{(\sigma_3\}, b) = E(\sigma_3) U E(\sigma_2) = \{\sigma_3, \sigma_2\}$

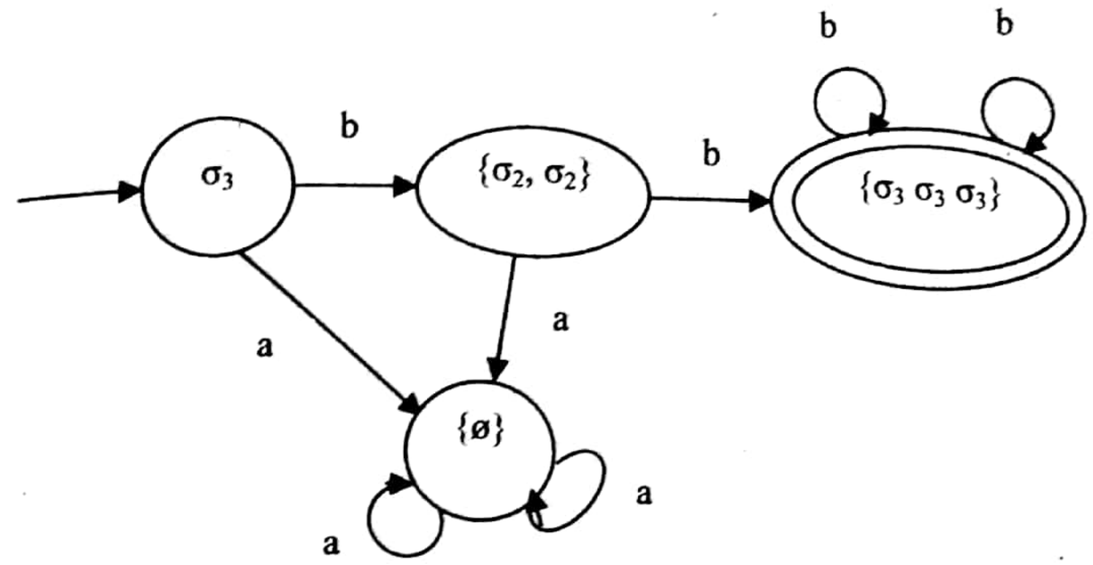
$g(\{(\sigma_3, \sigma_2\}, a) = \phi$

$g(\{(\sigma_3, \sigma_2\}, b) = E(\sigma_3) U E(\sigma_2) U E((\sigma_1) = \{\sigma_3, \sigma_2, \sigma_1\}$

$g(\{(\sigma_3, \sigma_2, \sigma_1\}, a) = E(\sigma_1) U E(\sigma_2) U E((\sigma_3) = \{\sigma_3, \sigma_2, \sigma_1\}$

$g(\{(\sigma_3, \sigma_2, \sigma_1\}, b) = E(\sigma_1) U E(\sigma_2) U E((\sigma_3) = \{\sigma_3, \sigma_2, \sigma_1\}$

$g(\{\phi\}, a) = \{\phi\}$

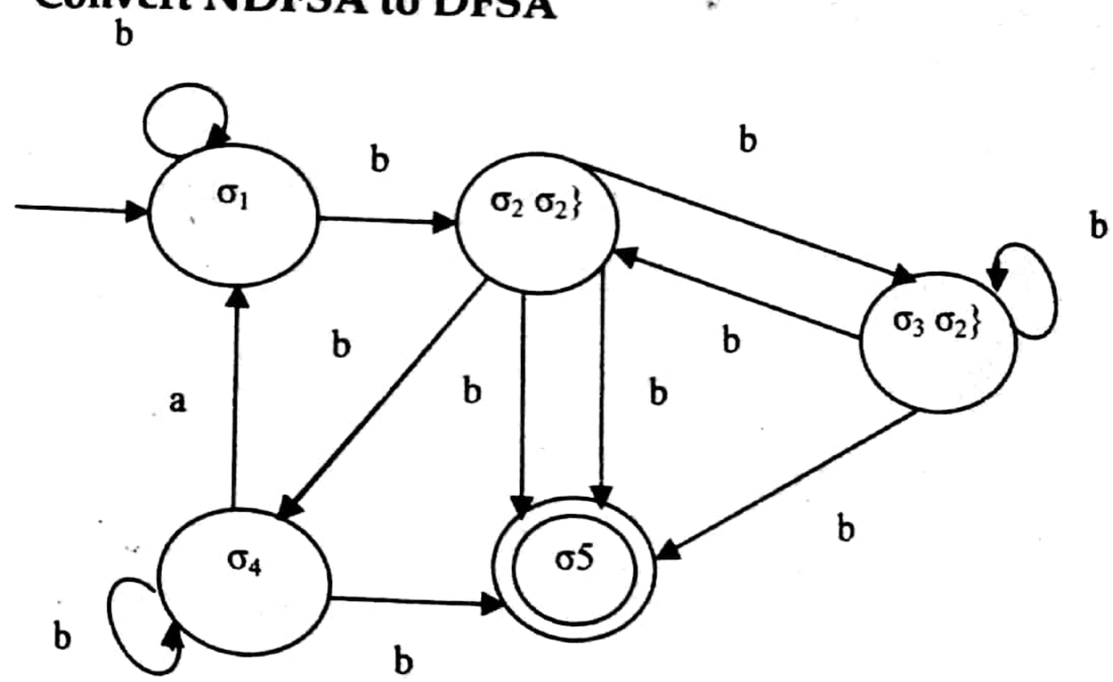


$g(\{\phi\}, b) = \{\phi\}$

Non Deterministic Finite State Automata

A non-deterministic finite automaton is a generalization of a finite state automaton so that at each state there might be several possible choices for the "next state' instead of just one.

Convert NDFSA to DFSA



2.6 Regular Expression and its Characteristics

Regular expressions are useful for representing certain set of strings in an algebraic manner. Actually this describes the language accepted by finite state automata.

The regular expressions over an alphabet $\sum$ are the strings over the alphabet $\sum$ U{), c, , U, }such that the following hold ($\sum$ consists of alphabet $\sum = \{a, b, \ldots\}$ I/p symbols)

1. The empty set ϕ and each member of $\sum$ is a regular expression.
2. If α and β are regular expressions then their concatenation $\alpha\beta$ is also a regular expression.
3. If α and β are regular expressions then their union (α U β) is also a regular expressions.
4. If α is a regular expressions then so is the closure (iteration) α^* . If α is R.E (α) is also RE.
5. Nothing is a regular expression unless it follows from (1) through (4).

Regular languages can be characterized as languages defined by regular expressions.

The language represented by regular expression is given a function L, such that if α is any regular expressions then L(α) is the language represented by α.

1. $L(\phi) = \phi$, $L(a) = \{a\}$, for each $a \in \sum$
2. If α and β are regular expressions then $L(\alpha\beta) = L(\alpha).\ L(\beta)$
3. If α and β are regular expressions

   Then

   $L(\alpha U\beta) = L(\alpha)UL(\beta)$
4. If α is a regular expression

   Then

   $L(\alpha^*) = L(\alpha)^*$

What are the strings specified by the regular expression?

Regular expression is **language generator** it generates the set of all strings for finite automata.

(FSA, pumping lemma are **language recognizer**.)

1. 10^* = A '1' followed by any no. of 0's including non zero.

2. $(10)^*$ = Any number of copies of 10 (including the null string)
3. 0U01 = The string 0 or the string 01.
4. $0(0U1)^*$ = Any string not beginning with 0.
5. $(0^*1)^*$ = Any string not ending with 0`
6. $\sum^* = \{0.1\}^* = (0 + 1)^* = \{\in, 0, 1, 00, 01, 10, 11..\}$
7. $0^* + 1^* = \{\in, 0, 00, 000, \ldots\}$

   $= \{\in, 1, 11, 111, \ldots\}$ (can it give combination only either 0 or 1.)
8. $0^*1^* = \{\in, 0, 1, 01, 001, 0011, .\}$(zero never come after 1)
9. $(01)^* = \{\in, 01, 0101, 010101, \ldots\}$
10. $01^* + 10^*$ =either single 0 followed by any number of 1 or vice versa.
11. $(0 + 1)^*.00$

A language is regular iff it is accepted by FSA.

There are three operations on language that the operators of regular expression represent.

These operations are;

1. **Union:**

   The union of two languages L and M denoted by L + M is the set of strings that are in either L or M or both

   E.g. $L = \{001, 10, 111\}$ and $M = \{\in, 001\}$

   Then,

   $L + M = \{\in, 10, 001, 111\}$
2. **Concatenation:**

   The concatenation of language L and M is the set of strings that can be formed by taking any string in L and concatenation it with any string in M and is denoted by L.M

   E.g. $L = \{001, 10, 111\}$, $M = \{\in, 001\}$

   Then,

   $L.M = \{001, 10, 111, 001001, 10001, 111001\}$

3. **The closure (or star or kleene star) of language**

L is denoted by L and represents the set of those strings that can be formed by taking any number of strings from L, possibly with repetations (i.e. the same string may be selected more than once) and concatenating all of them.

E.g. L = {0, 1} then L* is the set of all the strings of 0's and is with ∈ or λ

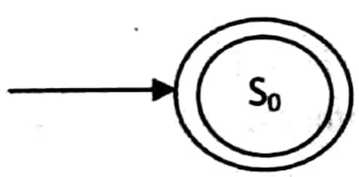
i.e. L = {0, 1}

L* = {0, 1}*= {λ, 0, 1, 01, 10, 11, 111, ...] .

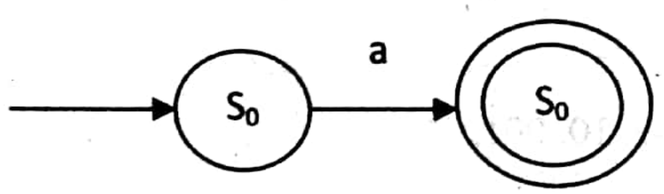
The class of languages accepted by finite state automata (i.e. R.L.) is closed under.

1. Union
2. Concatenation
3. Kleene star
4. Complementation
5. Intersection

An automata with no final sets accepts or recognizes ϕ



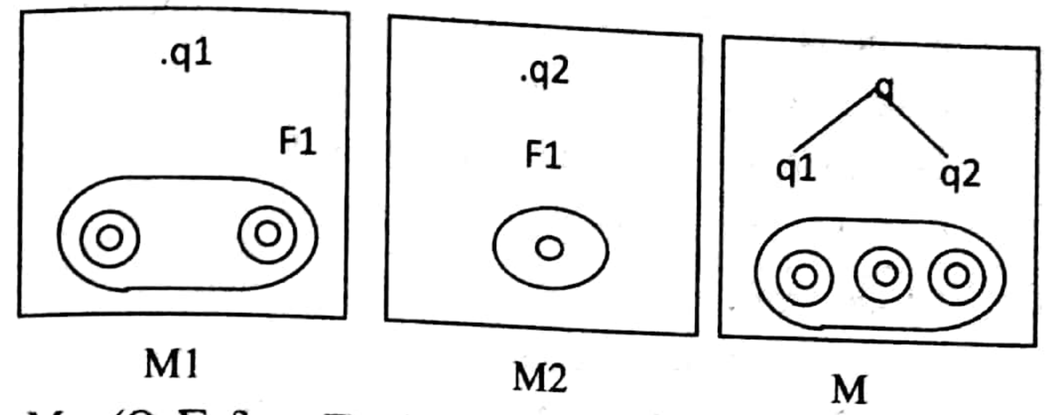
Empty string



Accepts the alphabet a

1. **Union :**

Let L1 and L2 be languages accepted by non-deterministic automata M1 and M2 respectively. Let M_1= (Q_1, Σ, δ_1, q_1, F_1) and M_2= (Q_2, Σ, δ_2, q_2, F_2). We assume that Q_1 and Q_2 are disjoint sets. We construct NDFSA M that accepts $L(M_1)UL(M_2)$ as follows



M = (Q, Σ, δ , q, F) where q is new state not in q_1 or Q_2

Q = $Q_1UQ_2U\{q\}$ [Set of states of M]

F = F_1UF_2 [Final states of M]

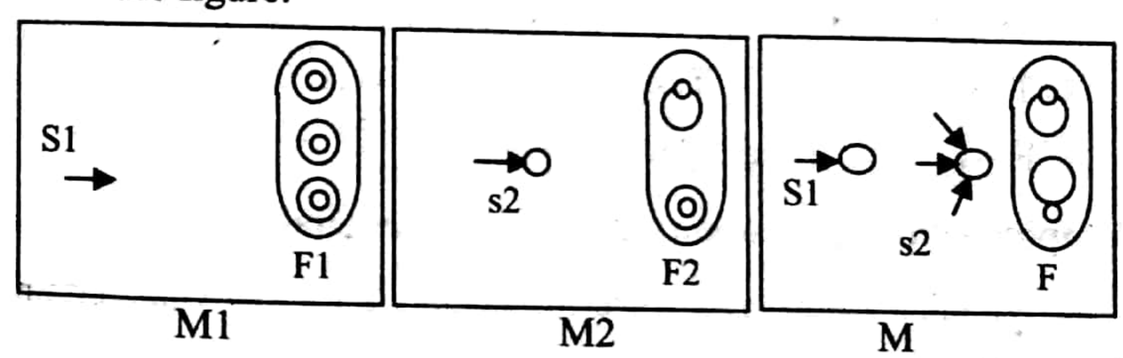
Transition function of M = $\delta_1U\delta_2U\{(q, \in) \rightarrow ((q, \in) \rightarrow q_2)\}$

Or,

$F_1UF_2U\{s, e, s_1),(S, e, s_2)\}$

2. **Concatenation :**

Let M1 and M2 be NDFA. We construct a NDFA M such that L(M)= L(M1) → L(M2). The construction is shown in the figure.



M operates by simultaning M_1 for a while and then jumping non-deterministically from a finite state of M_1 to the initial state of M_2. Therefore M imitates M_2.

Formally,

Let,

M1 = (S'_1, I, f_1, s_1, F_1) or Q_1, , , q_1, F_1) or (Q1, Σ, δ1, q1, F1)

M2=(S'_2, I, f_2, s_2, F_2) s_1 and s_2 are disjoint

L(M) = L(M1).L(M2)

M = (S, I, f, σ , F)

Set of states of M = S = $S'_1 U S'_2$=union of M_1 and M_2

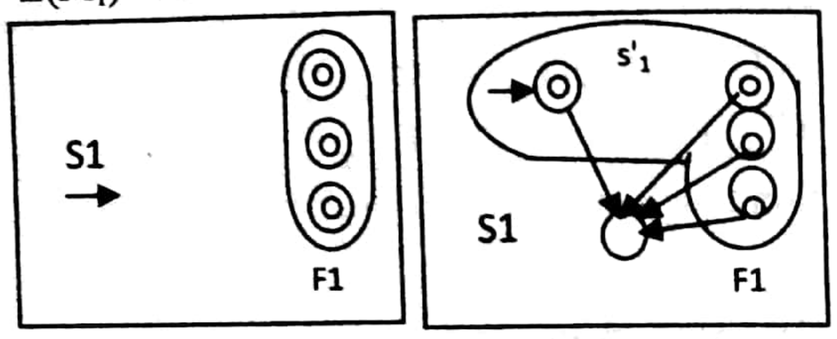
Final states (F)= final states of M_2 (F_2)

Transition S = Union of M1 and M2 plus ($F_1 X\{e\} X\{s_2\}$)

$F = F_1 U F_2 U\{(F_1, c) \rightarrow S_2\}$

3. **Kleene star :**

Let M_1 be a NDFA we construct a NDFA M such that L(M) = $L(M_1)^*$. The construction is show below.



M_1 M

L(M) = $L(M_1)$

States of M = states of M_1 plus s'_1

Final states of M = Final states of M_1 plus s'_1

Transition of M = Transition of M_1 plus ($FX\{e\}\}X\{s_1\}$)

$F = F_1 U F_2 U\{(F, e, S_1)\}$

Pumping lemma (Theorem) for Regular Language

Pumping lemma is a powerful technique for showing certain language to be non-regular.

Statement

Let L be a regular language. Then there exit a constant n (which depend on L) such that for every string w in L, such that $|w| \geq n$.

We can break w into three sub strings

1. $y \neq \in$
2. $|xy| \leq n$
3. For all $i \geq 0$, string xy^iz is also in L

Technique to identify the class of language for finite automata is pumping lemma.

Show that the language

$L = \{a^n b^n \ n > 0\}$ is not regular

$w \in L$

$w = a^p b^p$

According to the pumping lemma

$x = a^q, \ y = a^r, \ z = a^{p-(q+r)} b^p$

Since,

$xy \in L$

Now,

xy^2z

$= a^q (a^r)^2 (a^{p-(q+r)} b^p)$

$= a^q a^{2r} a^{p-(q+r)} b^p$

$= a^{p+r} b^p$ L

Since xy^2z is not the from $a^p b^p$ therefore $xy^2z \in L$. It is contradiction. Hence L is not regular.

Closure properties of Regular Language

If L and M be regular language. Then following languages are all regular.

1. Union : L U M, L + M
2. Intersection : $L \cap M$
3. Complement : $\bar{N}$, $\bar{L}$ and $\bar{M}$, $i = \Sigma^* \backslash L$
4. Difference : $L \backslash M = L \cap \bar{M}$
5. Reversal $L^R = W^R : w \in L$
6. Closure: L^*
7. Concatenation : L.M
8. Homomorphism

   $h(L) = \{h(w) \mid w \in L, h \text{ is homomorphism}\}$

9. Inverse homomorphism

$h^{-1}(L) = \{w \in \mid h(w)L, h:\Sigma \rightarrow \Delta$ is a homomorphism)

homomorphism

$h: \Sigma^* \rightarrow \theta * ($ alphabet Σ , θ)

$w: a_1 a_2 \ldots\ldots\ldots\ldots a_n \in \Sigma^*$

$h(w) = h(a_1)\, h(a_2) \ldots\ldots\ldots h(a_n)$

$h(L) = \{h(w) | w \in L\}$

Example

let $h: \{0,1\}^* \rightarrow \{a, b\}^*$ be defined by

$h(0) = a\,b$ and $h(1) = \epsilon$

now,

$h(0011) = abab$

$h(L(10^*1)) = L((ab^*))$

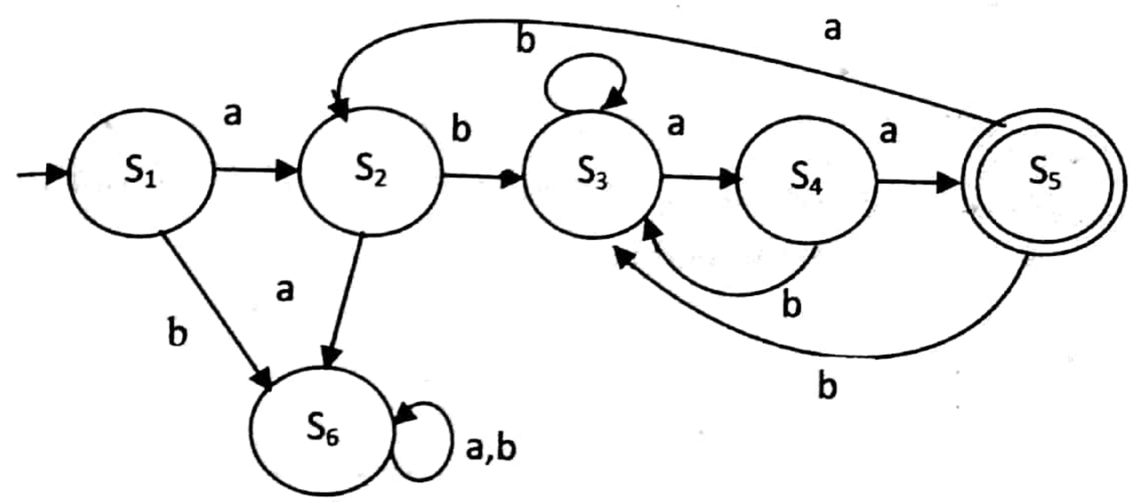
SOLVED NUMERICALS

1. ***Design FSA transition diagram that accepts the given set of string over {a, b}.*** ***[2076 Baisakh, Back]***

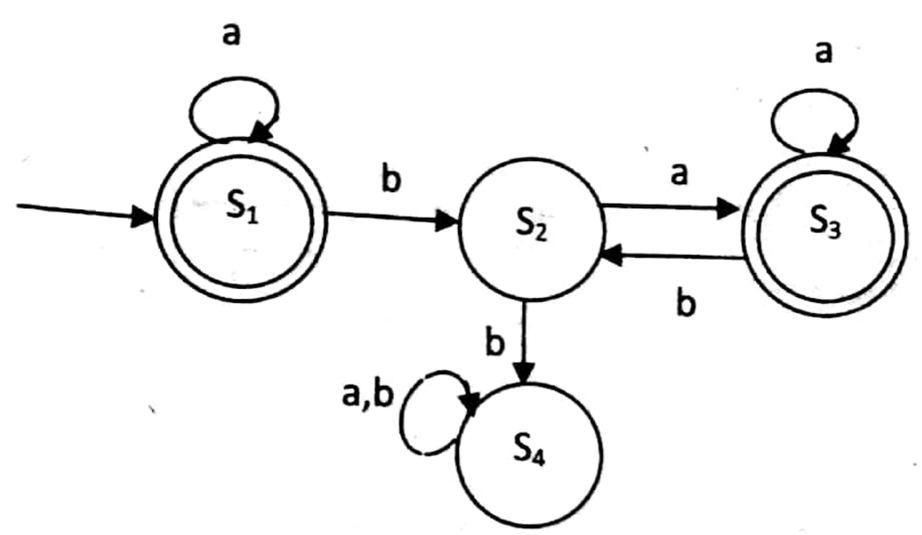
i. ***Which starts with ab and ends with baa***

Solution:

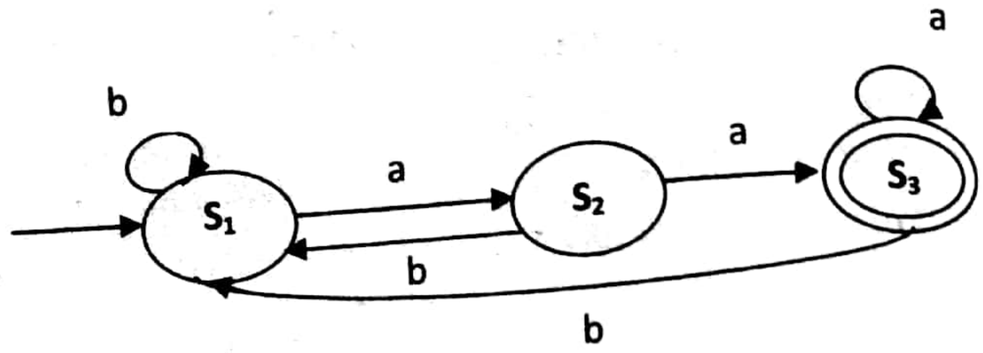
FSA which starts with ab and ends with ba



ii. ***Where every b is followed by a***



2. ***How do you define a Finite State Automation (FSA)? Design a finite state automata that accepts precisely those strings over {a, b} that end with substring aa. Your design should include the proper definition of the finite state automation, transition table and the transition diagram.*** ***[2073 Bhadra]***



3. ***Write a grammar that generates the string having the given property.***

   a. String over {a, b} ending with ba

   b. String over {a, b} starting with a ***[2073 Bhadra]***

a. String over {a, b} ending with ba

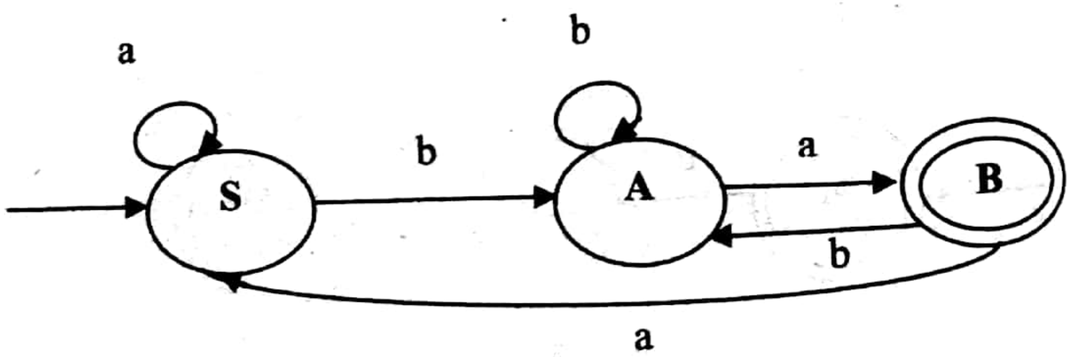
Let the grammar G be G = (N, T, R, S)

N ={S, A, B}

T = {a, b}

R={S → aS|bA, A → aB|bA, B → aS|bA}

Starting symbol is S



b. String over {a, b} starting with a

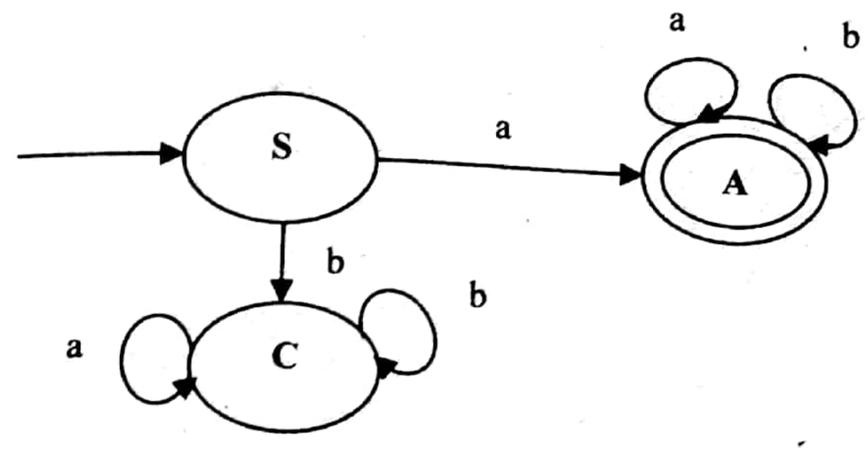
Let the grammar G be G = (N, T, R, S)

N={S, A, B}

T = {a, b}

R={S → aA|bB, A → aA|bA, B → aB|bB}

Starting symbol is S



4. ***Design a Grammar to generate Palindrome for Binary Number. Consider the right linear Grammar G = {N, T, P, S}, where N = set of non-terminal = {A, B, S}, T = Set of Terminal = {a, b, c}, and P consists of the following rules: {S → abA/ bbB/ a, A → aA/ bB/b, B → BaB/aaaA}. Construct the NDFA equivalent to the given grammar.***

[2076 Baisakh, Back]

Solution:

Grammar to generate palindrome for binary number

G = (N, T, R, S)

N = Set of non – terminal = {S}

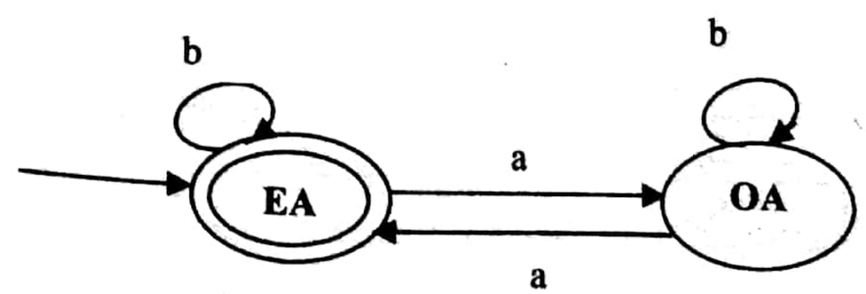
T = Set of terminal {0, 1}

R = Set of Production rule = {S → 0S0| 1S1 |ϵ | 1 | 0}

S = Start Symbol = {S}

5. ***Differentiate between a Finite State Machine and a Finite State Automation. Design a Finite State Automata that accepts precisely those string over {a, b} that contains an even no. of a's. Your design should include the proper definition of the Finite State Automata, transition table and the transition diagram.*** ***[072 Ashwin]***

Hints:



6. ***Define Non-Deterministic Finite State Automata. Design a Finite State Automata that accepts precisely those strings over {a, b} that do not contain two consecutive a's.***

[071 Magh]

Solution:

- Non deterministic gives a machine multiple options for its moves. In a non deterministic finite state automata, for each state there can be zero, one, two or more transitions corresponding to a particular symbol.

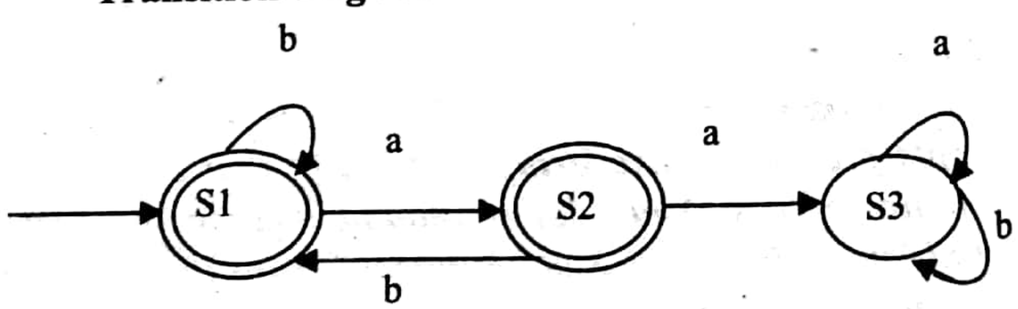
  i.e.in put-state pair.

- In DFSA, the next-state function takes us to a uniquely defined state, whereas in NDFSA, the next state function takes us to a set of states.

  A. non-deterministic finite state automata is a 5 tuple (I, S, f, σ, A) where

  - I is a finite set of input symbols
  - S is a finite set of states
  - F is a next state function or transition function from SXI to 2^S
  - $\sigma \in s$ is an initial state
  - $A \subseteq S$ is accepting states.

Transition diagram:



Transition table:

S/I	a	b
$\rightarrow S_1$	S_2	S_1
S_2	S_3	S_1
S_3	S_3	S_3

Table:1

Fig(1) and table 1 are transition diagram and transition table respectively for a finite state automata that accepts precisely those strings over {a, b} that do not contain two consecutive a's.

7. ***Construct the regular grammar to generate integers. Your construction should include the proper definitions of the grammar, which includes properly defined non-terminal symbols, terminal symbols, production rules and starting symbol.***

[071 Magh]

Solution:

- Grammar is a way to determine or specify the structure or set of sentences of a language.
- A grammar defining formal language L is a quadtuple

  G = (N, T, P, S)

  N is set of non terminal symbols, used to represent given subsets of the language.
- Non terminals can be placed.
- Non terminals (or syntactic symbols) are replaced by groups of terminal symbols according to the production rules.

T is set of terminal symbols

- Terminal symbols are literal symbols which may appear in the outputs of production rule or may occur in the outputs of production rule or may occur in the strings of the language.

  The lexical elements terminals and non-terminals of a particular grammar are two disjoint sets.

P is set of production rule

- A grammar is defined by production rule that specify which symbols may replace with other symbol to generate string.
- A symbol substitution can be recursively performed to generate a symbol sequence.
- Each production rule has a head on left hand side which consists of the string that may be replaced, and a body, or

right-hand side, which consists of a s∈. The type of grammar is distinguished by the properties of production rule.

In regular grammar, every production is of the form.

$A \rightarrow a$ or $A \rightarrow aB$ or $A \rightarrow \lambda$

Where A, B∈N

A∈T

i.e. **In left side** there is single terminal

In right side there may be either single terminal or terminal followed by a non-terminal or just null string.

Now, to construct the regular grammar to generate integers,

N = {S}

T = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}

P = {s → 0 |1| 2| 3| 4| 5| 6| 7| 8| 9| 0s|1s| 2s| 3s| 4s| 5s| 6s| 7s| 8s| 9s}

Starting symbol is S.

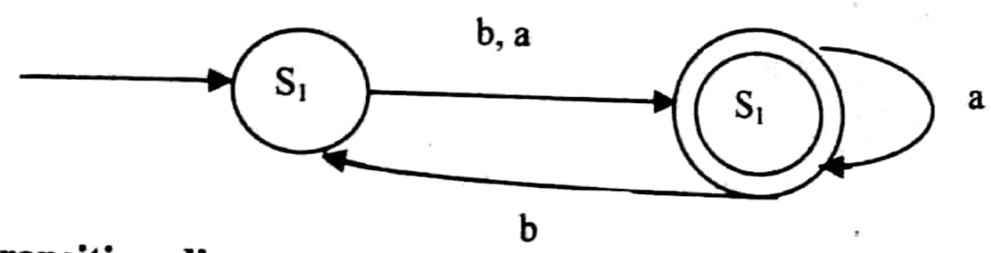
As mentioned earlier, N is set of terminal symbol, T is set of non terminal symbols, P is production rule and s is start symbol. The grammar G = (N, T, P, S) with above values is the regular grammar that generate integers.

8. ***Differentiate between finite state machines and finite state automata. Design a finite state automata that accepts precisely those strings over {a, b} that accepts precisely those strings over {a, b} that contain an odd number of b's. Your design should include the proper definition of the finite-state automation, transition table and the transition diagram.*** *[071 Bhadra]*

Solution:

S. N.	Finite State Machine	Finite State Automata
1.	A finite state machine is an abstract model of a machine with a primitive internal memory.	A finite state automata is similar to finite state machine but with no output and output function but it has accepting or final states.

S. N.	Finite State Machine	Finite State Automata
2.	A finite state machine M consists of a. Finite set of input symbols (I) b Finite set of output symbols (O) c. Finite set of states (s) d. Next state function (f) e. Output function (g) f. An initial state σ∈ s M = (I, O, S, f, g, σ)	A finite state automata A consist of a. Finite set of input symbols (I) b. finite set of states (s) c. next state function (f) d. accepting state A is subset of S. e. an initial state σ∈s A = (I, S, F, A, σ)
3.	Example: FSM that performs serial adder that takes pairs of bits as input.	Example: FSA that accepts precisely those strings over {a, b} that contain an odd number of a's.



Transition diagram

The transition diagram is a digraph. The vertices are the states. The initial state is indicated by arrow.

Here, the initial state is s1. If we are in state s1 and inputinga or b we move to s2 states. The double rounded state is final or accepting state. S2 is accepting state.

Transition table:

s/I	a	b
→ S_1	S_2	S_2
S_2	S_2	S_1

9. ***Consider the regular grammar G = (N, T, P, σ) where N = set of non-terminals = {σ, A , B}, T = set of terminals = {a, b} with productions σ → aA, σ → bB, A → a, B → a and starting symbol σ .***

Construct a non-deterministic finite state automata equivalent to the above given regular grammar and convert this into equivalent deterministic finite state automata. [07 Bhadra]

Solution:

Here,

G = (N, T, P, σ)

N = set of non-terminals = { σ, A, B}

T = set of terminals {a, b}

P = production rules

={ σ → aA, σ → bB, A → a, B → b}

σ = starting symbol

The non -terminals become starts with σ as the initial state

For each productions of the form

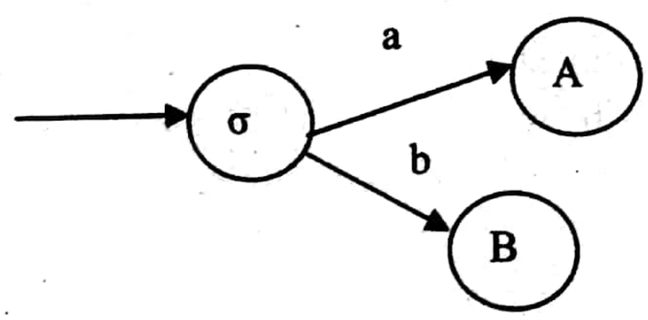
s → xs'

we draw an edge from state s to s' and label it x.

here, the productions are

σ → aA, σ → bB

Give the graph as shown below



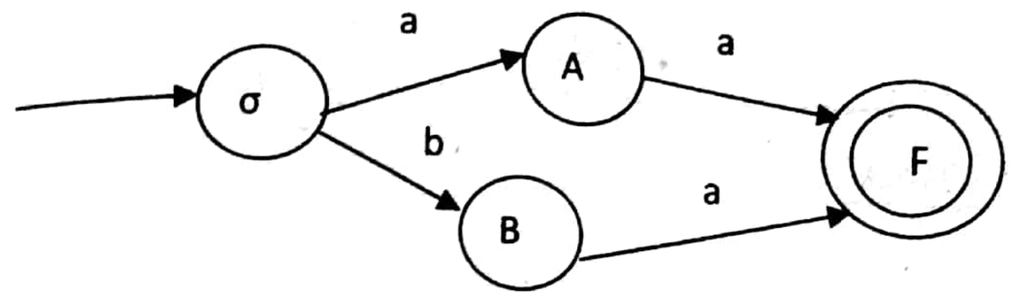
The production A-a is equivalent to two productions

A → aF and F → λ

Similarly production B-a is equivalent to two productions

B → aF and F → λ

Where p is an additional non terminal symbol.



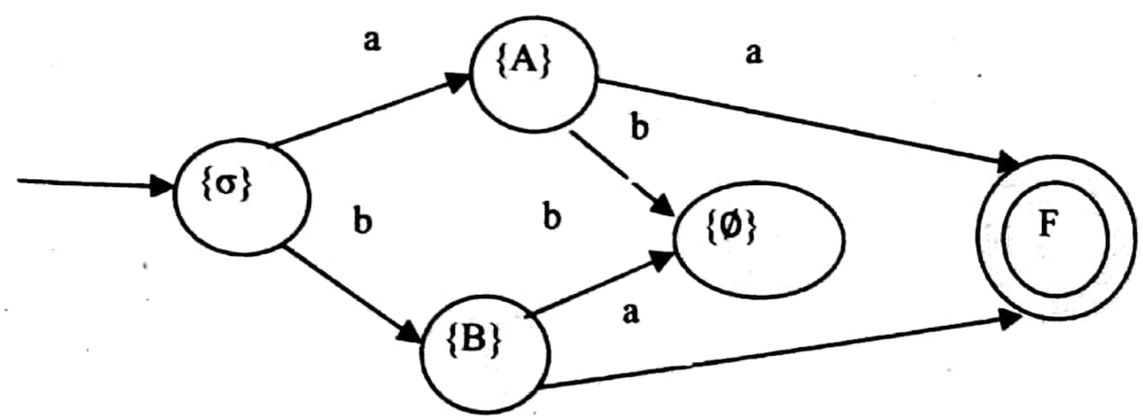
The production F → λ tell us that I should be an accepting state.

Transition table

s/I	a	b
σ	{A}	{B}
A	{F}	{Ø}
B	{F}	{Ø}

Now, convert NDFSA to DFSA

s/I	a	b
{σ}	{A}	{B}
{A}	{F}	{Ø}
{B}	{F}	{Ø}
{F}	{Ø}	{Ø}



Recurrence Relation

3.1 Recursive Definition of Sequences

A sequence can be defined by giving a general formula for its nth term by writing a few of its terms. An alternative approach is to represent the sequence by finding a relationship among its terms. Such relations are referred as recurrences.

Recursively defined sequences are often referred to as recurrence relations.

A recurrence relation for the sequence $\{a_n\}$ is an equation that expresses a_n in terms of one or more of the previous terms of the sequence namely $a_0, a_1, a_2, \ldots\ldots a_{n-1}$ for all integers n with $n \geq n_0$ where n_0 is a non-negative integer.

A sequence is called a solution of a recurrence relation if their terms satisfy the recurrence relation.

Example

Consider the following instructions for generating a sequence

1. ***Start with 3***
2. ***Given any term, add 3 to get the next term.***

The term of sequence obtained is

3 ,6, 9, 12, 15, 18,.....

If we denote the sequence as $a_1, a_2, \ldots$

$a_1 = 3$ ..... (1)

Instruction (2) can be rephrased as

$a_n = a_{n-1} + 3 \quad n \geq 2$ ..... (A)

By using a_1 and a_n, we can compute any term in the sequence.

Equation (A) is an example of recurrence relation (1)

Example

2. ***Let $\{a_n\}$ be a sequence that satisfies the recurrence relation. $a_n = a_{n-1} - a_{n-2}$ for $n = 2,3,4$ and suppose that $a_0 = 3$ and $a_1 = 5$ what are a_2 and a_3?***

From the recurrence relation.

$a_n = a_{n-1} - a_{n-2}$

$a_2 = a_1 - a_0 = 5-3 = 2$

$a_3 = a_2 - a_1 = 2-5 = -3$

Similarly a_4, a_5 and also each successive terms. The sequence forms in such a way is 3,5,2,-3,....

3. ***Determine whether the sequence $\{a_n\}$ is a solution of the recurrence relation $a_n = 2a_{n-1} - a_{n-2}$ for $n = 2,3,4,\ldots$ where $a_n = 3n$ for every non negative integer n. Answer the same question where $a_n = 2^n$ and where $a_n = 5$.***

Suppose that $a_n = 3n$ for every n. therefore $n \geq 2$, $2a_{n-1} - a_{n-2}$

$= 2(3(n-1) - 3(n-2))$

$= 6n - 6 - 3n + 6$

$= 3n$

$= a_n$

Since $\{a_n\}$, where $a_n = 3n$, is a solution of the recurrence relation.

Suppose $a_n = 2^n$ for every n. note $a_0 = 2^0 - 1$

And $a_1 = 2^1 = 2$ and $a_2 = 2^2 = 4$

$2a_{n-1} - a_{n-2}, \quad (n = 2)$

$= 2a_1 - a_0$

$= 2.2 - 1 = 3 \neq a_2$, we see that $\{a_n\}$,

where $a_n = 2^n$ is not a solution of recurrence relation. $a_n = 5$

4. ***A customer deposits 10,000 in a bank and he gets 11% interest per annual and the interest amount is accumulated every year. Find the recurrence relation to compute total amount customer receive after 10 year. And also calculate amount that customer receive after 30 years.***

$p_0 = 10{,}000$

Interest = 11%

$p_1 = p_0 + 0.11 * p_0$

$= 1.11p_0$

$= 11100$

$p_2 = p_1 * 0.11 + p_1$

$= 1.11p_0 * 0.11 + p_0 * 1.11$

$= 1.11p0(0.11 + 1)$

$= (1.11)p_0 * 1.11$

$= (1.11)^2 p0$

$p_3 = 1.11p_2 = (1{,}11)^3 p_0$

$p_n = (1{,}11)p_{n-1} = (1.11)^n p_0$

$p_n = (1.11)^n\ 10{,}000.$

3.2 Solution of Linear Recurrence Relations

A linear homogeneous recurrence relation of degree k with constant coefficient is a recurrence relation of the form

$a_n = c_1 a_{n-1} + c_2 a_{n-2} + \ldots\ldots\ldots\ldots\ldots + c_k a_{n-k}$

Where $c_1, c_2, \ldots. C_k$ are real numbers and $c_k \neq 0$.

The recurrence relation in the definition is linear since R.H.S is a sum of multiples of the previous terms of the sequence.

The recurrence relation is homogeneous since no term occur that are not multiple of the a_js.

The degree is k because a_n is expressed in terms of the previous k term of the sequence.

If $f(n) \neq 0$ for the recurrence relation

$a_n = c_1 a_{n-1} + c_2 a_{n-2} + \ldots. + c_k a_{n-k} + f(n)$

Then it is called non-homogeneous recurrence relation.

If $f(n) = 0$ then it is called linear homogeneous recurrence relation.

Solving linear homogeneous recurrence relation

Method of characteristic equation

We know that the linear homogeneous recurrence relation of degree k with constant coefficients is of the form.

$a_n = c_1 a_{n-1} + c_2 a_{n-2} + \ldots\ldots. + c_k a_{n-k}$

Let $a_n = r^n$ is the solution of this equation then

$r^n = c_1 r^{n-1} + c_2 r^{n-2} + \ldots\ldots + c_k a_{n-k}$

Dividing both by r^{n-k}

$r^k = c_1 r^{k-1} + c_2 r^{k-2} + \ldots.. + c_k$

$\mathbf{r^k - c_1 r^{k-1} - c_2 r^{k-2} - \ldots.. - c_k = 0}$

This is the characteristic equation of degree k of the recurrence relation.

Let us consider the liner homogeneous recurrence relations of degree 2.

i.e.

$a_n = c_1 a_{n-1} + c_2 a_{n-2}$

Its characteristic equation is of the form

$r^2 - c_1 r - c_2 = 0$

The roots of this equation can be

1. **Distinct roots, say r_1 and r_2 (i.e. $r_1 \neq r_2$)**

   If roots are distinct then $\{a_n\}$ is the solution of the given linear homogeneous recurrence relation if and only if a_n is of the form

   $a_n = \alpha_1 r_1^n + \alpha_2 r_2^n$ where α_1 and α_2 are constant

2. **Repeated roots, say r_0 (i.e. $r_1 = r_2 = r_0$)**

   If roots are repeated root, say r_0 then $\{a_n\}$ is the solution of the given linear homogeneous recurrence relation if and only if a_n is of the form

   $a_n = \alpha_1 r_0^n + \alpha_2 n r_0^n$

Theorem:

Let c_1 and c_2 be real numbers suppose that $r^2-c_1r-c_2=0$ has two distinct roots r_1 and r_2, then the sequence $\{a_n\}$ is a solution of the recurrence relation $a_n=c_1a_{n-1}+c_2a_{n-2}$ iff $a_n=\alpha_1r_1^n+\alpha_2r_2^n$ for n = 0, 1, 2, ... where α_1 and α_2 are constant. To find α_1 and α_2 initial condition is used.

1. ***What is the solution of recurrence relation***

$a_n=a_{n-1}+2a_{n-2}$ with $a_0=2$ and $a_1=7$

Let $a_n=r^n$

Substituting in the equation $a_n=a_{n-1}+2a_{n-2}$

we get,

$r^n=r^{n-1}+2r^{n-2}$.....(i)

Dividing equation (i) by r^{n-2}, we get

$r^2-r-2=0$

$r^2-(2-1)r-2=0$

$r^2-2r+r-2=0$

$r(r-2)+1(r-2)=0$

$(r-2)(r+1)=0$

Its roots are r = 2 and r = -1. Hence the sequence $\{a_n\}$ is a solution to the recurrence relation iff

$a_n=\alpha_1r_1^n+\alpha_2r_2^n$

$a_n=\alpha_12^n+\alpha_2(-1)^n$

For some constants α_1 and α_2. From initial conditions it follows that,

$a_0=\alpha_1+\alpha_2=2$

$a_1=\alpha_12+\alpha_2(-1)$

$7=\alpha_12+\alpha_2(-1)$

i.e. $\alpha_1+\alpha_2=2$ ..... (ii)

$2\alpha_1-\alpha_2=7$ .....(iii)

Solving these two equation

$\alpha_1=3$ and $\alpha_2=-1$

Hence the solution to the recurrence relation and initial conditions is the sequence $\{a_n\}$ with $a_n=3.2^n-(-1)^n$

2. ***Find the explicit formula for the fibonacci numbers.***

OR

Find the solution of recurrence relation of fibonacci series.

$a_0=0$

$a_1=1$

$a_n=a_{n-1}+a_{n-2}$

The characteristic equation of $a_n=a_{n-1}+a_{n-2}$ is

$r^n=r^{n-1}+r^{n-2}$

(since assume $a_n=r^n$ then $r_n=r^{n-1}+r^{n-2}$)

$r^n/r^{n-2}=r^{n-1}/r^{n-2}+r^{n-2}/r^{n-2}$

$r^2-r-1=0$

$$r=\frac{+1\pm\sqrt{(-1)^2-4(1)(-1)}}{2*1}$$

$$r=\frac{1\pm\sqrt{5}}{2}$$

$$r_1=\frac{1+\sqrt{5}}{2}$$

$$r_2=\frac{1-\sqrt{5}}{2}$$

Then fibonacci numbers are given by

$a_n=\alpha_1r_1^n+\alpha_2r_2^n$

$=\alpha_1((1+\sqrt{5})/2)^n+\alpha_2((1+\sqrt{5})/2)^n$

For some constants α_1 and α_2, using $a_0=0$ and $a_1=1$

When n = 0, $a_0=0$

$a_0=\alpha_1+\alpha_2$

$\alpha_1=-\alpha_2$

When n = 1, $a_1=1$

$a_1=\alpha_1((1+\sqrt{5})/2)^1+\alpha_2((1+\sqrt{5})/2)^1$

$1=-\alpha_2((1+\sqrt{5})/2)+\alpha_2((1+\sqrt{5})/2)$

$=\alpha_2(-1/2-\sqrt{5}/2+1/2-\sqrt{5}/2)$

$=\alpha_2(-2\sqrt{5}/2)$

$\alpha_2 = -1/\sqrt{5}$

$\alpha_1 = 1/\sqrt{5}$

Hence, $a_n = 1/\sqrt{5}\ ((1+\sqrt{5})/2)^n + (-1/\sqrt{5}\ )((1-\sqrt{5})/2)^n$

3. ***What is the solution of recurrence relation***

$a_n = 6a_{n-1} + 9a_{n-2}$ with $a_0 = 3$ and $a_1 = 7$

Solution,

$a_n = 6a_{n-1} + 9a_{n-2}$

Let $a_n = r^n$

We get,

$r^n = 6r^{n-1} + 9r^{n-2}$ ..... (1)

Dividing equation (1) by r^{n-2}, we get

$r_2 - 6r - 9 = 0$

Solving,

We get....

....

Theorem:

Let c_1 and c_2 be real number with $c_2 \neq 0$. Suppose that $r^2 - c_1 r - c_2 = 0$ has only one root r_0. A sequence $\{a_n\}$ is a solution of recurrence relation $a_n = c_1 a_{n-1} + c_2 a_{n-2}$ iff $a_n = \alpha_1 r_0^n + \alpha_2 n r_0^n$, for n = 0,1,2,.... Where α_1 and α_2 are constant.

Theorem:

Let c_1, c_2, ... c_k be real numbers. Suppose that the characteristic equation.

$r^k - c_1 r^{k-1} \ldots\ldots - c_k = 0$

has k distinct roots $r_1, r_2, \ldots\ldots r_k$ then sequence $\{a_n\}$ is a solution of recurrence relation.

$a_n = c_1(a_{n-1}) + c_2 a_{n-2} + \ldots\ldots + c_k a_{n-k}$ iff

$a_n = \alpha_1 r_1^n + \alpha_2 r_2^n + \ldots\ldots + \alpha_k r_k^n$ for n = 0,1,2.... where $\alpha_1, \alpha_2, \alpha_3, \ldots\ldots \alpha_k$ are constants.

Theorem :

Let c_1, $c_2, \ldots\ldots c_k$ be real numbers, suppose that the characteristics equation

$r^k - c_1 r^{k-1} \ldots\ldots - c_k = 0$

has t distinct roots r_1, r_2 ..... r_k with multiplication $m_1, m_2, m_3, \ldots\ldots m_t$ respectively so that $m_i \geq 1$ for 1,2,....t and $m_1 + m_2 + \ldots + m_k = k$. Then a sequence $\{a_n\}$ is a solution of the recurrence relation

$a_n = c_1 a_{n-1} + c_2 a_{n-2} + \ldots\ldots + c_k a_{n-k}$ iff

$a_n = (\alpha_{1,0} + \alpha_{1,1} n + \ldots\ldots\ldots + \alpha_{1,m1-1} n^{m1-1}) r_1^n +$

$(\alpha_{2,0} + \alpha_{2,1} n + \ldots\ldots\ldots + \alpha_{2,m2-1} n^{m2-1}) r_2^n + \ldots$

$\ldots\ldots + (\alpha_{t,0} + \alpha_{t,1} n + \ldots\ldots\ldots + \alpha_{t,mt-1} n^{mt-1}) r_t^n$

For n = 0, 1, 2,.... Where $\alpha_{i,j}$ are constants for $1 \leq i \leq t$ and $0 \leq j \leq m_{i-1}$

Suppose that roots of the characteristics equation of a linear homogeneous recurrence relation are 2, 2, 2, 5, 5 and 9 (that is there are three roots, the root 2 with multiplicity two three, the root 5 with multiplicity two and the root 9 with multiplicity one.

What is the form of general solution?

$(\alpha_{1,0} + \alpha_{1,1} n + \alpha_{1,2} n^2) 2^n + (\alpha_{2,0} + \alpha_{2,1} n) 5^n + \alpha_{3,0} 9^n$

1. ***$a_n = 6a_{n-1} - 9a_{n-2}$ with $a_0 = 1$, $a_1 = 6$***

Find the solution of given recurrence relation.

Solution:

$a_n = 6a_{n-1} - 9a_{n-2}$

Let $a_n = r^n$, we get

$r^n = 6r^{n-1} - 9r^{n-2}$, we get,

$r^2 = 6r - 9$

$r^2 - 6r + 9 = 0$ .....(1)

This is a characteristic equation,

Solving the equation (1)

r = 3, 3

Hence,

$a_n = \alpha_1 3^n + \alpha_2 n3^n$ for some constants, applying initial condition.

Here, $a_0 = 1$

$a_0 = \alpha_1$

$\alpha_1 = 1$

Here $a_1 = 6$

$a_1 = \alpha_1 3 + \alpha_2 3$

$6 = 3 + \alpha_2 3$

$\alpha_2 = 1$

Hence the solution is $a_n = 3^n + n3^n$

2. ***Find the solution to the recurrence relation $a_n = 6a_{n-1} - 11a_{n-2} + 6a_{n-3}$***

With the initial conditions $a_0 = 2$, $a_1 = 5$ and $a_2 = 15$.

Solution:

The characteristic equation is given by

$r^3 - 6r^2 + 11r - 6 = 0$

or, $r^3 - r^2 - 5r^2 + 5r + 6r - 6 = 0$

or, $r^2(r - 1) - 5r(r - 1) + 6(r - 1) = 0$

or, $(r - 1)(r^2 - 5r + 6) = 0$

or, $(r - 1)(r^2 - 3r - 2r + 6) = 0$

or, $(r - 1)[r(r - 3) - 2(r - 3)] = 0$

or, $(r - 1)(r - 2)(r - 3) = 0$

$r_1 = 1, r_2 = 2, r_3 = 3.$

The solution is then given by

$a_n = \alpha_1.1^n + \alpha_2.2^n + \alpha_3.3^n$

For some constants α_1, α_2, and α_3

Finding α_1, α_2 and α_3 using initial conditions

$a_0 = 2 = \alpha_1 + \alpha_2 + \alpha_3$.....(1)

$a_1 = 5 = \alpha_1 + \alpha_2.2 + \alpha_3.3$.....(2)

$a_2 = 15 = \alpha_1 + \alpha_2.4 + \alpha_3.9$.....(3)

Solving equation (1),(2), and (3), we get

$\alpha_1 = 1$, $\alpha_2 = -1$, $\alpha_3 = 2$

Hence the solution is given by

$a_n = 1 - 2^n + 2.3^n$

3. ***Find the solution to the recurrence relation $a_n = -3a_{n-1} - 3a_{n-2} - a_{n-3}$ with initial conditions $a_0 = 1$, $a_1 = -2$ and $a_2 = -1$***

The characteristic of this recurrence relation is

$r^3 + 3r^2 + 3r + 1 = 0$

$(r + 1)^3 = 0$

$r_1 = r_2 = r_3 = -1$ i.e. there is a single root $r = -1$ of multiplicity three of characteristic equation.

The solution is given by,

$a_n = (\alpha_{1,0} + \alpha_{1,1}n + \alpha_{1,2}n^2)(-1)^n$

$= (\alpha_{1,0}(-1)^n + \alpha_{1,1}n(-1)^n + \alpha_{1,2}n^2(-1)^n$

To find the constants $\alpha_{1,0}$, $\alpha_{1,1}$ and $\alpha_{1,2}$ use the initial condition. This gives

$a_0 = 1 = \alpha_{1,0}$.....(1)

$a_1 = -2 = \alpha_{1,0}(-1) + \alpha_{1,1}(-1) + \alpha_{1,2}(-1)$

$a_1 = -2 = -\alpha_{1,0} - \alpha_{1,1} - \alpha_{1,2}$.....(2)

$a_2 = -1 = (\alpha_{1,0}(-1)^2 + \alpha_{1,1}2(-1)^2 + \alpha_{1,2}2^2(-1)^2$

$a_2 = -1 = \alpha_{1,0} + 2\alpha_{1,1} + 4\alpha_{1,2}$.....(3)

Solving (1),(2) and (3), we get,

$\alpha_{1,0} = 1$; $\alpha_{1,1} = 3$, $\alpha_{1,2} = -2$

Hence, the solution is given by,

$a_n = 1(-1)^n + 3n(-1)^n + (-2)n^2(-1)^n$

$a_n = (1 + 3n - 2n^2)(-1)^n$

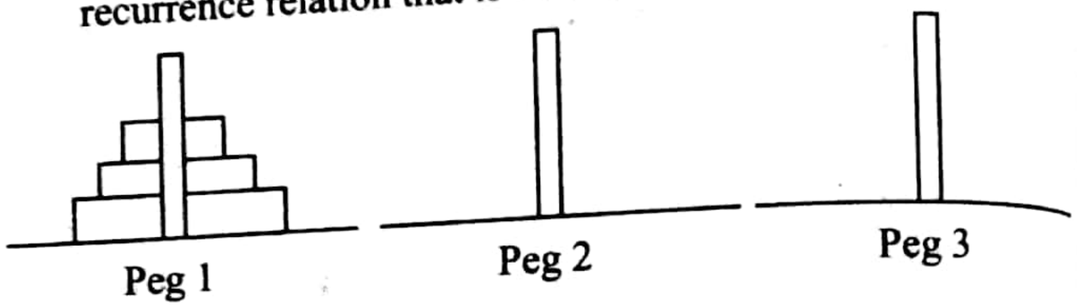
Tower of Hanoi Puzzle

There are 3 pegs and n circular disks of increasing on one peg, with the largest on the bottom. These disks are to be transferred one at a time onto the another peg, with the provision that larger disk is not allowed to place on a smaller disk. The

problem is to determine the number of moves necessary for the transfer.

Solution:

Let h_n be the number of moves required to transfer n disk. We verify that $h_0 = 0$, $h_1 = 1$ and $h_2 = 3$. We have to find a recurrence relation that is satisfied by h_n.



To transfer n disks to another peg, we first transfer the top n-1 disks to a peg, transfer the largest disk to the vacant peg and then transfer all n-1 disks to the peg, which now contain the largest disk, this h_n satisfies $h_n = 2h_{n-1} + 1 (n \geq 1)$ where $h_0 = 0$

This is a recurrence relation of order 1 with constant coefficients, but it is not homogeneous, because of the presence of the quantity.

1. **To find h_n, we iterate the above equation.**

$$h_n = 2h_{n-1} + 1$$
$$= 2(2h_{n-2} + 1) + 1$$
$$= 2^2h_{n-2} + 1$$
$$= 2^2(2h_{n-3} + 1) + 2 + 1$$
$$= 2^{n-1}(2h_0 + 1) + 2^{n-2} + \ldots + 2^2 + 2 + 1$$
$$= 2^{n-1} + 2^{n-2} + \ldots + 2^2 + 2 + 1$$
$$= \frac{1(2^n-1)}{2-1} \quad (\text{since } s_n = \frac{a(r^n-1)}{r-1})$$
$$= 2^n - 1$$

Therefore, the no. h_n are the partial sums of the geometric sequence $1, 2, 2^2, \ldots 2^n$ and hence satisfy.

$$h_n = \frac{2^n-1}{2-1} = 2^n - 1 = 2_{n-1}$$

We have to verify this formula for h_n and it can be easily verified by mathematical induction.

Since $h_0 = 0$, it holds for $n = 0$

Assume that it holds for n. we then show it holds with n replaced by n + 1

i.e. $h_{n+1} = 2h_n + 1$

$$= 2(2^n - 1) + 1$$
$$= 2^{n+1} - 1$$

Since it also holds for n + 1, hence the formula is verified.

3.3 Solution of Nonlinear Recurrence Relations

To solve the non-homogeneous recurrence relation with constant coefficient, we need to solve its associated liner homogeneous equation and to find a particular solution for the given non-homogeneous equation.

A recurrence relation of the form

$$a_n = c_1a_{n-1} + c_2a_{n-2} + \ldots + c_ka_{n-k} + F(n)$$

Where $c_1, c_2, \ldots c_k$ are real numbers and F(n) is a function not identically zero depending only on n is linear non-homogeneous recurrence relation. The recurrence relation

$$a_n = c_1a_{n-1} + c_2a_{n-2} + \ldots c_ka_{n-k}$$

is called the associated homogeneous recurrence relation.

For example,

1. $a_n = a_{n-1} + 2^n$
2. $a_n = a_{n-1} + a_{n-2} + n^2 + n + 1$
3. $a_n = 3a_{n-1} + n3^n$
4. $a_n = a_{n-1} + a_{n-2} + a_{n-3} + n$

These are examples of linear non-homogeneous recurrence relation with constant coefficients corresponding associated linear homogeneous recurrence relation are

1. $a_n = a_{n-1}$
2. $a_n = a_{n-1} + a_{n-2}$

3. $a_n = 3a_{n-1}$
4. $a_n = a_{n-1}+a_{n-2}+a_{n-3}$

Theorem :

If $\{a_n^{(p)}\}$ is a particular solution of the non-homogeneous liner recurrence relation with constant coefficients.

$a_n = c_1a_{n-1}+c_2a_{n-2}+.....+c_ka_{n-k} + f(n)$

Then every solution is of the form $\{a_n^{(p)} + a_n^{(h)}\}$ where $a_n^{(h)}$ is a solution of the associated homogeneous recurrence relation.

$a_n = c_1a_{n-1}+c_2a_{n-2}+.....+c_ka_{n-k}$

Theorem:

Suppose $\{a_n\}$ satisfies the linear recurrence relation.

$a_n = c_1a_{n-1}+c_2a_{n-2}+....+c_ka_{n-k} + f(n)$

Where $c_1, c_2,.....c_k$ are real numbers and

$f(n) = (b_tn^t + b_{t-1}n^{t-1}+.....+ b_1n + b_0)s^n$

Where $b_0, b_1.....bt$ and s are real numbers. When s is not a root of the characteristics equation of the associated linear homogeneous recurrence relation there is a particular solution of the form.

$(p_tn^t + p_{t-1}n^{t-1}+.....+p_1n + p_0)s^n$

When s is a root of this characteristic equation and its multiplicity is m, there is a particular solution is the form

$n^m(p_tn^t + p_{t-1}n^{t-1} + ... + p_1n + p_0)s^n$

1. ***What form does a particular solution of the linear non-homogeneous recurrence relation $a_n = 6a_{n-1}-9a_{n-2}+f(n)$.***

have when $f(n) = 3^n$

$f(n) = n3^n$

$f(n) = n^22^n$

$f(n) = (n^2 + 1)3^n$

Solution

The associated linear homogeneous recurrence relation is $= 6a_{n-1} - 9a_{n-2}$. Its characteristics equation, $r^2 - 6r + 9 = (r-3)^2 = 0$ has a single root 3, of multiplicity two.

For $f(n) = 3^n$

Because s = 3 is a root with multiplicity 2, particular solution has the form $p_0n^23^n$ if $f(n) = 3^n$

For $f(n) = n3^n$

The form is $n^2(p_1n + p_0)3^n$ if $f(n) = n3^n$

For $f(n) = n^22^n$

The form is $(p_2n^2 + p_1n + p_0)2^n$ if $f(n) = n^22^n$

For $f(n) = (n^2 + 1)3^n$

The form is $n^2(p_2n^2+p_1n + p_0)3^n$ if $f(n) = (n^2 + 1)3^n$

2. ***Find all solutions of the recurrence relation $a_n = 3a_{n-1} + 2n$. what is the solution with $a_1 = 3$?***

Solution,

The associated linear homogeneous equation is $a_n = 3a_{n-1}$

Its solution are $a_n^{(h)} = \alpha 3^n$

Where α is constant

Now, finding particular solution

f(n) = 2n is a polynomial of degree one

Comparing with $f(n)= p(n)\, s^n$

we get, s = 1

The root of the characteristic equation is 3

Here, s is not the root of characteristic equation of the associated linear homogeneous recurrence relation, there is a particular solution of the form

$(p_0 + p_1n +.....+ p_{t-1}n^{t-1} + p_tn^t)s^n$

$= (p_0 + p_1n)s^n$

$= (p_0 + p_1n)$

New, equation $a_n = 3a_{n-1} + 2n$ becomes

$(p_0 + p_1n) = 3(p_0 + p_1(n - 1)) + 2n$

or, $(p_0 + p_1n) = 3p_0 + 3p_1n - 3p_1 + 2n$

or, $n(p_1 - 3p_1 - 2) = 2p_0 - 3p_1$

or, $n(-2p_1 - 2) = 2p_0 - 3p_1$

or, $n(2p_1 + 2) + 2p_0 - 3p_1 = 0$

It follows that cn + d is a solution if and only if $2 + 2p_1 = 0$ and $2p_0 - 3p_1 = 0$

$\therefore p_1 = -1, p_0 = -3/2$

$a_n^{(p)} = -n - 3/2$

$a_n = a_n^{(p)} + a_n^{(h)}$

$= -n - 3/2 + \alpha.3^n$

To find solution with $a_1 = 3$

$n = 1, a_1 = 3$

$3 = -1 - 3/2 + 3\alpha$

$6 = -2 - 3 + 6\alpha$

$\alpha = 11/6$

$a_n = -n - 3/2 + (11/6)3^n$

3. ***Find the solution of recurrence relation $a_n = 5a_{n-1} - 6a_{n-2} + 7^n$***

Solution,

The solution of its associated homogeneous recurrence relation

$a_n = 5a_{n-1} - 6a_{n-2}$

Here,

$f(n) = 7^n$

Comparing with $f(n) = p(n)\, s^n$

$s = 7$

Here s is not the root of characteristic equation, there is particular solution in the form

$p_0 = 7^n$

Now, equation $a_n = 5a_{n-1} - 6a_{n-2} + 7^n$ becomes

$p_0 7^n = 5.p_0 7^{n-1} - 6.p_0 7^{n-2} + 7^n$

$p_0 = 49/20$

$a_n^{(p)} = (49/20) * 7^n$ is a particular solution

$a_n = a_n^{(p)} + a_n^{(h)}$

$= (49/20) * 7^n + \alpha_1 3^n + \alpha_2 2^n$

4. ***Let a_n be the sum of the first n positive integer so that $a_n = \sum_{k=1}^{n} k$***

a_n satisfies the linear non-homogeneous recurrence relation

$a_n = a_{n-1} + n$ ..... (1)

The associated linear homogeneous recurrence relation are given by

$a_n^{(h)} = c(11)^n = c,$

Where c is constant

Here,

We have,

$f(n) = n$

Comparing with $f(n) = p(n)s^n$

$s = 1$

Here, s is the root of characteristic equation. The particular solution is in the form

$n(p_1 n + p_0)$

$= p_1 n^2 + p_0 n$ (here m = 1)

Now,

Equation (1) becomes

$p_1 n^2 + p_0 n = p_1(n-1)^2 + p_0(n-1) + n$

Simplifying we see,

$n(2p_1 - 1) + (p_0 - p_1) = 0$ which means

$2p_1 - 1 = 0$ and $p_0 - p_1 = 0$

$p_1 = p_0 = 1/2$

Hence,

$a_n^{(p)} = n^2/2 + n/2 = n(n+1)/2$ is a particular solution

$a_n = a_n^{(p)} + a_n^{(h)}$

$a_n = n(n+1)/2 + c$

We have, $a_1 = 1$, $1 = a_1 = c + (1/2)*2 = c + 1$

$c = 0$

$a_n = n(n+1)/2$

Solving recurrence relation using substitution

Solve recurrence relation

$a_n^2 - 2a^2_{n-1} = 1$ for $n \geq 1$ where $a_0 = 1$

This non-linear equation can be converted into linear equation by using substitution $b_n = an^2$

Now equation become $b_n - 2b_{n-1} = 1$ with $b_0 = a_0^2 = 1^2 = 1$

Solving $b_n - 2b_{n-1} = 1$ with $b_0 = 1$, the homogeneous solution is $\alpha.2^n$ and particular solution is c.

So, $b_n = a2^n + c$ where a and c are constants.

Using fact $b_0 = 1$ and $b_1 = 3$ we get $a + c = 1$ and $2a + c = 3$

$\therefore a = 2$ and $b = -1$.

Hence the solution is $b_n = 2.2^n - 1$

$= 2.2^n - 1 = 2^{n+1} - 1$

Substituting back $an^2 = 2^{n+1} - 1$

$a_n = \sqrt{2^{n+1} - 1}$

Divide and conquer recurrence relations

Divide and conquer relation is of the form $a_n = c^{an}/d + f(n)$. Hence this equation can be looked at as $b_k = cb_{k-1} + f(d^k)$ using $n = d^k$ and solved. Initial condition have to be changed appropriately. For substituting back, we use $k = \log_d n$.

Q.. Solve the divide and conquer relation $a_n = 3a_n/2+n$

Where, $n = 2^k$ for $k \geq 1$ and $a_1 = 1$

Using change of variables the equation becomes

$b_k = 3b_{k-1} + 2^k$

The homogeneous solution $b_k^{(h)} = A3^k$ and the particular solution is $b_k^{(p)} = B2^k$

We get,

$b_k = A3^k + B2^k$

Since,

$a_1 = 1, b_0 = 1$. So $A + B = 1$

$a_2 = b_1 = 5$,

We get,

$A = 3$ and $B = -2$

So, solution is,

$b_k = 3*3^k + 2*2^k$

Substituting back,

$a_n = 3.3^{\log_2 n} - 2.2^{\log_2 n}$

$= 3.3^{\log 2n} - 2n$

But,

$3^{\log 2n} = n^{\log_2 3}$

So,

$a_n = 3n^{\log_2 3} - 2n$

3.4 Application to Algorithm Analysis

Binary search:

Binary search looks for a value in a sorted sequence and returns the index of the value if it is found or 0 if it is not found.

Input: A sequence $s_i, s_{i+1}, \ldots, sj$, $i \geq 1$ sorted in increasing order, a value key, i and j.

Output: The output is an index k for which s_k = key or if key is not in the sequence the output is the value 0.

1. Procedure binary search (s, i, j, key)
2. If i>j then return(0) //not found.
3. K = (i + j)/2
4. Key = s_k then return (k) //found
5. If key <s_k then j = k-1 //search left half
6. Else i = k + 1 //search right half
7. (repeat the process) return (binary_search (s. i. j. key))
8. End binary search

This algorithm uses the divide and conquer approach. The sequence is divided into two nearly equal parts. (line 3)

- If the item is found at the dividing point (line 4), the algorithm terminates.

- If the item is not found, because the sequence is sorted, an additional comparison will located the half of the sequence in which the item appears if it is present.
- We then recursively invoke binary search (line 7) to continue the search.

Example

Input

$s_1 = 2, s_2 = 4, s_3 = 5, s_4 = 6$ and key 6

- $i = 1, j = 4, k =$ index
- Since $(i > j)$ $(1 > 4)$ is false
- $k = 2$
- Since key(6) is not equal to $s_4(4)$
- Key $(6)<s_k$ $(6<4)$ is false
- We set, $i = k + 1 = 3$

$i = 3, j = 4$

$s_3 = 5, s_4 = 6$

$i > j$ $(3 > 4)$ is false

$k = (3 + 4)/2 = 3$

- Since key (6) is not equal to $s_3(5)$
- Key $(6)<s_3$ $(6<5)$ is false

We set

$i = k + 1 = 3 + 1 = 4$

$i = 4, j = 4, s_4 = 6$

- $i>j$ $(4>4)$ is false
- Set $k = (4 + 4)/2 = 4$

Key (6) is equal to s4 (6). So we return 4, the index of key in the sequence s.

Thus, binary search reduces the search of an element in a search sequence of size n to the binary search sequence of the element in a search sequence of size n/2 when n is even. Hence the problem of size n has reduce to one problem of size n has reduced to one problem of size n/2. Two comparison are needed to

implement this reduction (one to determine which half of the list to use and other to determine whether any term of the list remain). Hence if f(n) is number of comparison required to search for an element in a search sequence of size n, then

$$f(n) = f(n/2) + 2$$

Selection sort

This algorithm sorts the sequence

$s_1, s_2, \ldots, s_n$

in increasing order by first selecting the largest item and placing it last and then recursively sorting the remaining elements.

Input : $s_1, s_2, \ldots\ldots, s_n$ and length n of the sequence

Output ; $s_1, s_2, \ldots\ldots, s_n$ arranged in increasing order

1. Procedure selection sort (s, n)
2. Base case, if n = 1 then return
3. // find largest value

   max_index = 1 // assime
4. For i = 2 to n do

   If $s_i > s_{max\text{-}index}$ then // found larger so up date max_index = i
5. // more largest to end

   Swap (s_n, s_{max_index})
6. Call selection_sort (s, n-1)
7. End selection_sort

$b_n = n-1+b_{n-1}$

SOLVED NUMERICALS

1. ***Find all the solutions of recurrence relation: $a_n = 3a_{n-1} + 4a_{n-2} + 3^n$ with initial conditions $a_0 = 1$ and $a_1 = 2$. [2076 Baisakh, Back]***

Solution:

The all solutions of the recurrence relation is $a_n = a_n^{(h)} + a_n^{(p)}$

The associated linear homogeneous equation is $a_n = 3a_{n-1} + 4a_{n-2}$

Its solution is $a_n^{(h)}$

$r^2 = 3r + 4$

$r^2 - 3r - 4 = 0$

$r^2 - 4r + r - 4 = 0$

$r(r - 4) + 1(r - 4) = 0$

$r1 = 4$

$r2 = -1$

$a_n^{(h)} = \alpha_1 4^n + \alpha_2 (-1)^n$

Now, Finding particular solution, $a_n^{(p)}$

$f(n) = 3^n$

Comparing with characteristics equation is 4 and 1, which is not equal to s.

Form of particular solution is

$P_0 s^n = P_0 3^n$

Now $a_n = 3a_{n-1} + 4a_{n-2} + 3^n$

$P_{03}^n = P_{03}^{n-1} + 4P_0 3^{n-2} + 3^n$

$P_0 = P_0/3 + 4P_0/9 + 1$

$P_0 = -1/2$

Now $a_n = a_n^{(h)} + a_n^{(p)}$

$= \alpha_1 4^n + \alpha_2 (-1)^n + (-1/2)3^n$

Putting the initial conditions, $a_0 = 1$ and $a_1 = 2$.

$a_n = \alpha_1 4^n + \alpha_2 (-1)^n + (-1/2)3^n$

$1 = \alpha_1 4^0 + \alpha_2 (-1)^0 + (-1/2)3^0$

$1 = \alpha_1 + \alpha_2 + (-1/2)$.....(i)

$2 = \alpha_1 4^1 + \alpha_2 (-1)^1 + (-1/2)3^1$

$2 = 4\alpha_1 - \alpha_2 + (-1/2)3$.....(ii)

From equation (i) and (ii)

$\alpha_1 = 1$

$\alpha_2 = 1/2$

$a_n = 4^n + (1/2)(-1)^n + (-1/2)3^n$

2. ***Find all the solutions of recurrence relation: [073 Bhadra]***

$a_n = 7a_{n-1} - 12a_{n-2} + 3^n$ with initial conditions $a_0 = 1$ and $a_1 = 4$

$r^n = 7r^{n-1} - 12r^{n-2}$

$r^2 - 7r + 12 = 0$

$r^2 - 4r - 3r + 12 = 0$

$r(r - 4) - 3(r - 4) = 0$

$(r - 4)(r - 3) = 0$

$r_1 = 3$

$r_2 = 4$

$a_n^{(h)} = \alpha_1 r_1^n + \alpha_2 r_2^n$

$= \alpha_1 3^n + \alpha_2 4^n$

for particular solution

$a_n^{(p)} = np_0 3^n$

$np_0 3^n = 7(n - 1)p_0 3^{n-1} - 12(n - 2)p_0 3^{n-2} + 3^n$

$9np_0 = 21np_0 - 21np_0 - 12np_0 + 24p_0 + 9$

$3p_0 = -9$

$p_0 = -3$

when $n = 0$, $a_0 = 1$

$1 = \alpha_1 + \alpha_2$

when $n = 1$

$4 = \alpha_1 3 + \alpha_2 4 - 9$

$3\alpha_1 + 4\alpha_2 = 13$

solving, we get,

$\alpha_1 = -9$

$\alpha_2 = 10$

$a_n = (-9)\,3^n + (10)4^n - n3^{n+1}$

3. ***Find all the solutions of the recurrence relation:***

$a_n = 5a_{n-1} - 6a_{n-2} + 2^n$ with initial conditions $a_0 = 1$ and $a_1 = 4$ ***[072 Ashwin]***

Hints:

homogeneous part

$a_n = 5a_{n-1} - 6a_{n-2}$

put $a_n = r^n$

$r = 2, 3$

$an^{(p)} = \alpha_1 2^n + \alpha_2 3^n$

comparing with $f(n)=p(n)s^n$

$n^m(p_0)s^n$

$n(p_0)2^n$

$n(p_0)2^n = 5(n-1)(p_0)2^{n-1} - 6(n-2)(p_0)2^{n-2} + 2^n$

$p_0 = 2/(11-6n)$

$a_n = a_n^{(p)} + a_n^{(h)}$

$= \alpha_1 2^n + \alpha_2 3^n + n*2/(11-6n)*2^n$

put $n = 0$, $a_0 = 1$

and

put $n = 1$, $a_1 = 4$

then get,

$\alpha_1 = -1/5$

$\alpha_2 = 6/5$

$a_n = (-1/5)2^n + (6/5)3^n + n*2/(11-6n)*2^n$

4. ***Find general solution of $\sqrt{a_n} = \sqrt{a_{n-1}} + 2\sqrt{a_{n-2}}$ with initial condition $a_0 = a_1 = 1$*** ***[071 Magh]***

Solution:

Here, $\sqrt{a_n} = \sqrt{a_{n-1}} + 2\sqrt{a_{n-2}}$ with $a_0 = a_1 = 1$

This non-linear equation can be converted into linear equation by using substitution, $b_n = \sqrt{a_n}$

Now, solve $b_n = b_{n-1} + 2b_{n-2}$

Let, $b_n = r^n$ is the solution of the equation

Then $r^n = r^{n-1} + 2r^{n-2}$

Dividing both side by r^{n-2}

$r^n/r^{n-2} = r^{n-1}/r^{n-2} + 2r^{n-2}/r^{n-2}$

Or, $r^2 = r + 2$

Or, $r^2 - r - 2 = 0$

Or, $r^2 - 2r + r - 2 = 0$

Or, $r(r-2) + 1(r-2) = 0$

Or, $(r-2)(r+1) = 0$

Its roots are $r = 2$ and $r = -1$,

Hence the solution is

$b_n = \alpha_1 r_1^n + \alpha_2 r2^n$

$b_n = \alpha_1 2^n + \alpha_2(-1)^n$

To find constant α_1 and α_2, we use the initial condition

$b_0 = \alpha_1 2^0 + \alpha_2(-1)^0$

$1 = \alpha_1 + \alpha_2$ ..... (1)

Again,

$b_1 = \alpha_1 2 + \alpha_2(-1)$

$1 = 2\alpha_1 - \alpha_2$ ..... (2)

Adding equation (1) and (2)

$1 = \alpha_1 + \alpha_2$

$1 = 2\alpha_1 - \alpha_2$

$\alpha_1 = 2/3$

Putting value of

$1 = \alpha_1 + \alpha_2$

$1 = 2/3 + \alpha_2$

$\alpha_1 = 1/3$

Now,

$bn = (2/3)^{2n} + (1/3)^{(-1)n}$

Substituting back $b_n = an$

$a_n = b_n^2$

$=(2^{n+1}/3 + 1/3(-1)^n)^2$

5. ***Find all solution of the recurrence relation $an = 3a_{n-1} + 2^n$ with initial condition $a_0 = 5$*** ***[071 Bhadra]***

Solution:

$a_n = 3a_{n-1} + 2^n$

The associated liner homogeneous equation is

$a_n = 3a_{n-1}$

Let $a_n = r^n$ be solution,

Then, $r^n = 3r^{n-1}$

$r = 3$

The homogeneous solution is $a_n^{(h)} = \alpha 3^n$

Where α is constant,

Now, finding particular solution

$f(n) = 2^n$

Comparing with $f(n) = p(n)s^n$

We get $s = 2$

The root of the characteristic equation is 3. Here, s is not the root of characteristic equation of the associated linear homogeneous recurrence relation. There is particular solution of the form,

$P_0.2^n$

Now, equation

$a_n = 3a_{n-1} + 2^n$ becomes

$p_0.2^n = 3.p_0.2^{n-1} + 2^n$

$p_0.2^n = 3.p_0.2^n/2 + 2^n$

$2p_0 = 3p_0 + 2$

$p_0 = 2$

$a_n^{(p)} = 2.2^n$

$a_n^{(p)} = 2^{(n+1)}$ is the particular solution

$a_n = a_n^{(h)} + a_n^{(p)}$

$= \alpha 3^n + \alpha 2^{n+1}$

To find solution, we use initial condition

$a_0 = 5$

$a_0 = \alpha 3^0 + 2^{0+1}$

$5 = \alpha + 2$

$\alpha = 3$

$a_n = 3.3^n + 2^{n+1}$

$a_n = 3^{n+1} + 2^{n+1}$

Graph Theory

4.1 Graph

Graph is a discrete structure consisting of vertices and edges. Edges connect the vertices.

A graph G = (V, E) consists of two parameter.

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

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

Let us suppose that a publishing company has ten editors in different areas. These ten editors have a standard meeting time during the first Sunday of every month and have divided themselves into seven committees to meet later in the day to discuss specific topics of interest of company.

The 10 editors have devided on seven committees $C_1 = \{1, 2, 3\}$, $C_2=\{1,3,4,5\}$, $C_3=\{2, 5, 6, 7\}$, $C_4=\{4, 7, 8, 9\}$, $C_5=\{2, 6, 7\}$, $C_6=\{8, 9, 10\}$ $C_7=\{1, 3, 9, 10\}$

They have set aside three time periods for the seven committees to meet on those Sundays when ten editors are present. Some committees can not meet during the same period because one or two of the editors are on both committees.

In this figure, there are seven small circles representing the seven committees and a straight line segment is drawn between two circles if the committees they represent have at least one committee member is common. In other words this means two committee with common committee memebe should not be scheduled to meet at a same time.

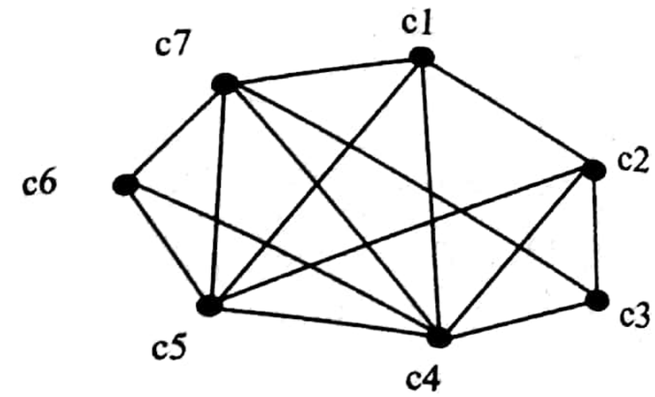


Fig.: Graphcal representation of seven committee of a company

A graph consist of a finite non empty set V of objects called vertices and a set E of element subsets of V called edges. So, graph is a pair of two sets V and E, vertex set and edge set respectively.

Vertices are also called points or nodes. Edges are sometimes called lines or arcs.

The vertex of set G is $V(G) = \{ c_1, c_2, \dots c_7\}$ and the edge set $E(G)= \{ \{ c_1, c_2\}, \{ c_1, c_3\}, \{ c_1, c_5\}, \{ c_1, c_7\}, \{ c_2, c_3\}, \{ c_2, c_4\}, \{ c_2, c_7\}, \{ c_3, c_4\}, \{ c_3, c_5\}, \{ c_4, c_5\}, \{ c_4, c_6\}, \{ c_4, c_7\}, \{ c_6, c_7\}\}$

- The number of vertices in G is often called the order of G while the number of edges is its size. We use n and m for order and size respectively.
- Since the vertex set of every graph is non-empty, the order of every graph is at least 1.
- A graph with exactly one vertex is called a trivial graph.

4.1.1 Types of Graphs

1. Labeled graph
2. Unlabeled graph

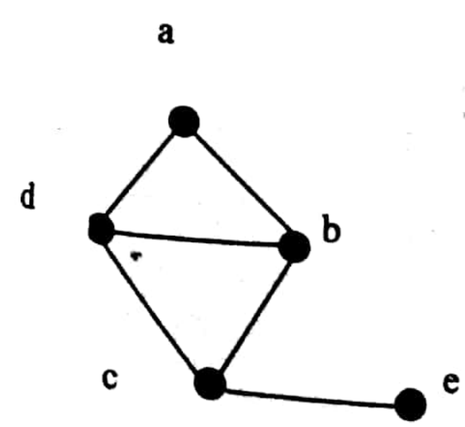


Fig.: Labeled graph

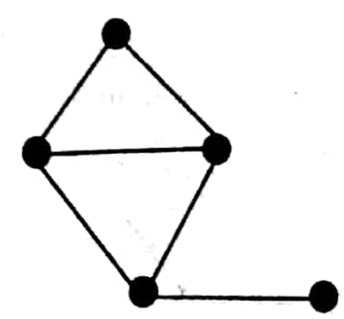


Fig.: Unlabeled graph

4.2 Undirected and Directed Graphs

4.2.1 Simple Graph

Simple graph is 2-tuple consists of non-empty set of vertices V, a set of unordered pairs of distinct elements of vertices called edges. We represent this graph as G = (V, E). This kind of graph has undirected edges, no loops and no multiple edges.

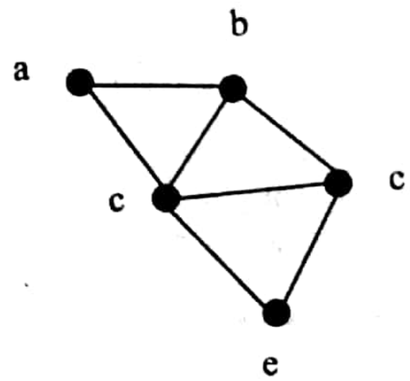


***Fig** : Simple graph*

In above graph G = (V, E) , set of vertices.

V (G) or V = {a, b, c, d, e} and set of edges.

E (G) or E = { {a, b}, {a, c}, {b, c},{b, d},{c, d}, {c, e}, {d, e}}

4.2.2 Multigraph

In a graph, two vertices are either adjacent or they are not, that is two vertices are joined by one edge or no edges. A multigraph M consist of finite non empty set V of vertices and a set of E of edges where every two vertices of M are joined by finite number of edges.

- If two or more edges join the same pair of vertices, then these edges are called parallel edges.
- In multigraph muliple edge occurs.
- In a pseudograph, an edge is also permitted to join a vertex to itself. Such an edge is called a loop.

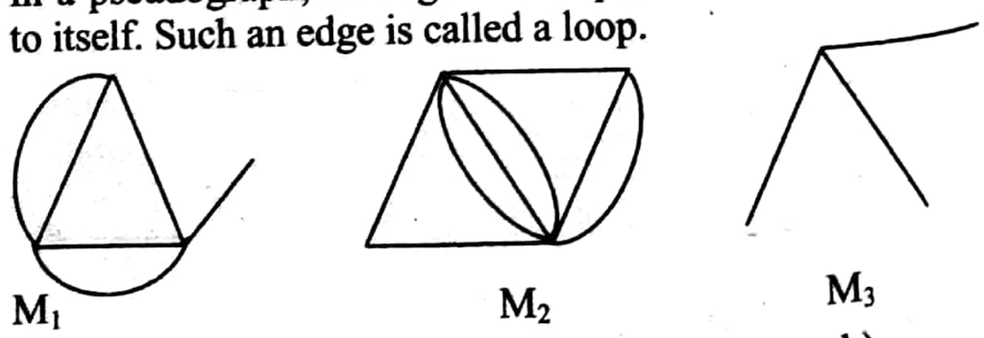
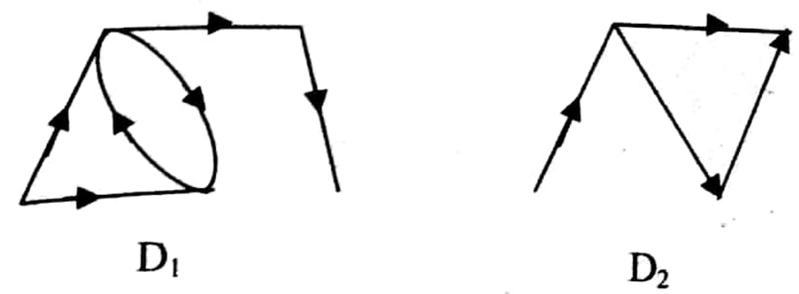


***Fig.:** Multigraph* (Every graph is a multigraph)

4.2.3 Directed Graph (Digraph)

A directed graph (V, E) consists of set of V of vertices, a set E of edges that are ordered pairs of elements of V. In this graph, loop is allowed but no two vertices can have multiple edges in same direction. A directed graph having no symeetric pais of directed edges is called oriented graph. A complete oriented graph is called tournament.



D_2 is oriented graph

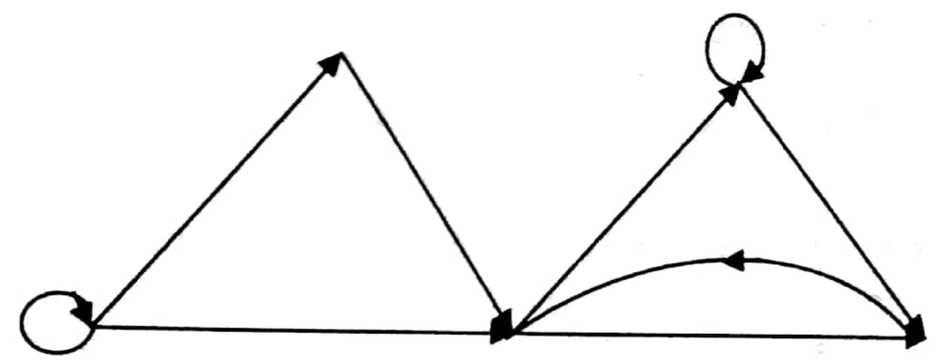


***Fig.:** Directed mutligraph*

4.3 Graphical Terms and Terminologies

4.3.1 Adjacent, Incident, and Degree of Vertex

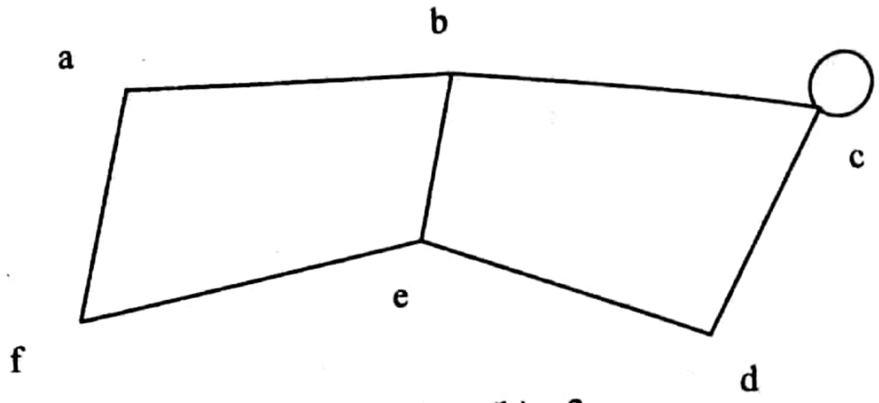
Adjacent: Two vertices u, v in an undirected graph G are called adjacent or neighbor if {u, v} is an edge.

Incident: The edge e is called incident with the vertices u and v if e = {u, v}

Degree of a vertex

The degree of a vertex is undirected graph is a number of edges incident with it, except a loop at a vertex. Degree of a vertex v is denoted by deg (v). A vertex of degree zero is called isolated vertex and are with degree one is called pendant vertex.

Loop in a vertex counts twice to the degree. e.g.: find the degrees.



deg (a) = 2 deg (b)= 3

deg (c) = 4 deg (d) = 2

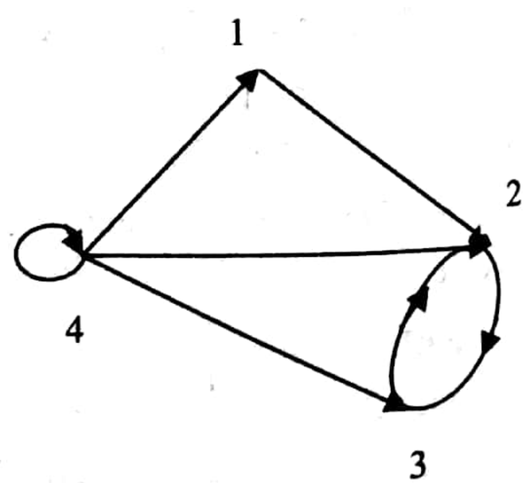
deg (e) = 3 deg (f) = 2

Let (u, v) be an edge representing edge of a directed graph G, u is called adjacent to v and v is called adjacent from u. The vertex v is called initial vertex and the vertex v is called terminal or end vertex. Loop had same initial and terminal vertex.

In directed graph the in-degree of a vertex v, denoted by $\deg^-(v)$, is a number of edges, that have v as their terminal vertex. The out-degree of a vertex v, denoted by $\deg^+(v)$, is the number of edges that have v as their initial vertex.

Loop at a vertex adds up both in-degree and out-degree to one or more than calculated in-degree and out-degree.

Q. Find in-degree and out-degree of each vertex



In-degrees:

$\deg^-(1) = 2$

$\deg^-(2) = 3$

$\deg^-(3) = 2$

$\deg^-(4) = 1$

Out-degrees:

$\deg^+(1) = 1$

$\deg^+(2) = 1$

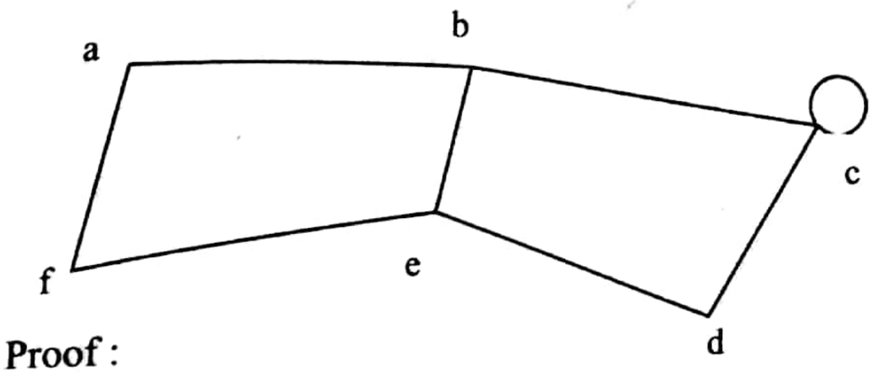
$\deg^+(3) = 1$

$\deg^+(4) = 4$

Theorem 1 :

The Handshaking Theorem

Let G = (V, E) be an undirected graph with a edges Then $2e = \sum_{v \in V} \deg(v)$



Proof :

deg(a) = deg(f) = deg(d) = 2

deg(b) = deg(e) = 3

deg(c) = 4

We have

LHS = 2*e

$= 2 \times 8 = 16$

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

$= 2 + 2 + 2 + 3 + 3 + 4 = 16$

Therefore, LHS = RHS

Theorem 2:

An undirected graph has an even number of vertices of odd degree.

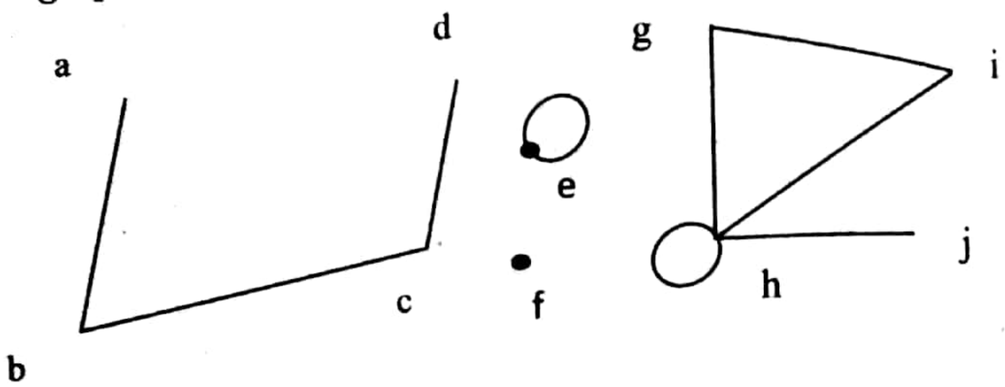
- **Let V_1 and V_2 be set of vertices of even and odd degrees respectively. In an undirected graph G = (V, E)**

We have,

$2e = \sum_{v \in V} \deg(v) = \sum_{v \in V1} \deg(v) + \sum_{v \in V2} \deg(v)$

From equality above, we can say the left part is even i.e. 2e is even, the sum of deg (v) for $v \in V_1$ is even since every vertices has even degree. So for the left hand to be even sum of deg (v) for $v \in V_2$ must be even. Since all vertices in the set V_2 have odd degree the number of such vertices must be even for the sum to be even.

Q. ***Which vertices in following graph are isolated which are pendant and what is maximum degree? What types of graph is it?***



Vertex f is isolated, vertices a, d and j are pendant. Maximum degree is deg (h) = 5. This graph is pseudograph (undirected loops)

Q. ***Determine the number of its edges and sum of the degrees of all its vertices.***

There are 9 edges and sum of degrees is 18.

Theorem 3:

The sum of the in-degrees of all vertices in a digraph = The sum of the out-degrees = The number of edges.

Let G = (V, E) be a graph with directed edges,

Then,

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

4.3.2 Complete graph

- The complete graph of n vertices is the simple graph that contains exactly one edge between each pair of distinct vertices.

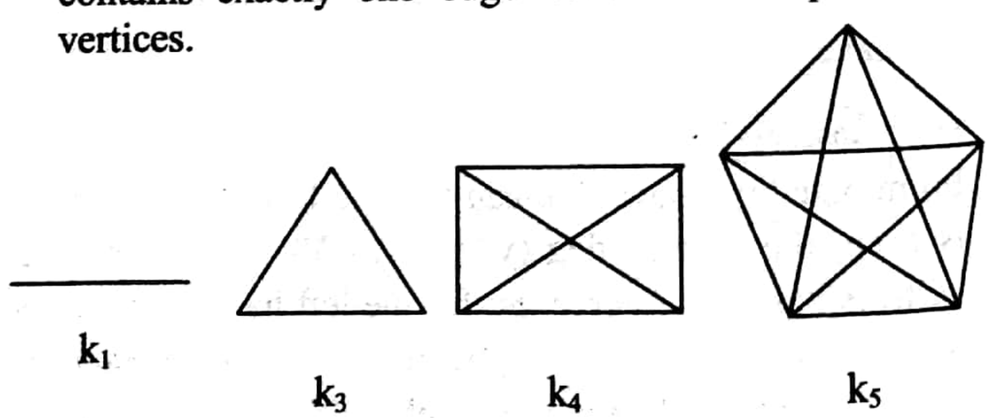


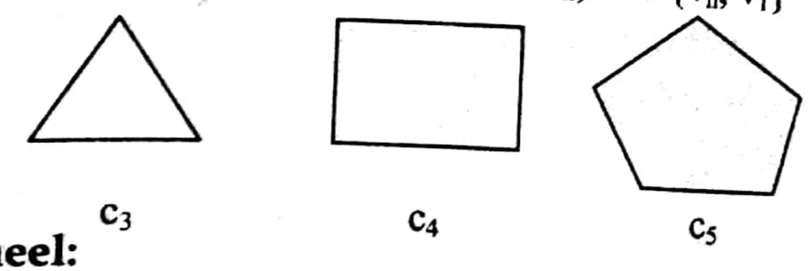
***Fig:** Complete graph*

- A complete graph of order n is denoted by k_n. Since every distinct vertices of k_n are joined by an edge, the number of pairs of vertices in k_n is $\binom{n}{2}$ the size of k_n is

$$\binom{n}{2} = \frac{n(n-1)}{2}$$

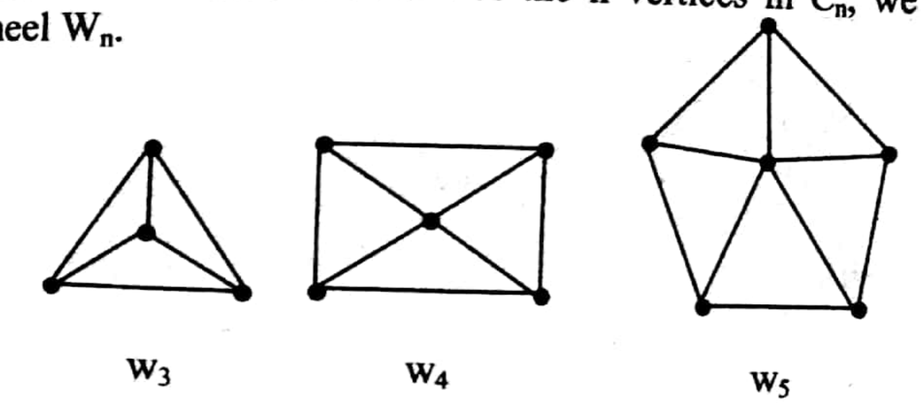
4.3.3 Cycle:

The cycle C_n, $n\geq 3$, consists of n vertices $v_1, v_2, \ldots\ldots v_n$ and edges $\{v_1, v_2\}, \{v_2, v_3\}, \ldots\ldots\{v_{n-1}, v_n\}$ and $\{v_n, v_1\}$



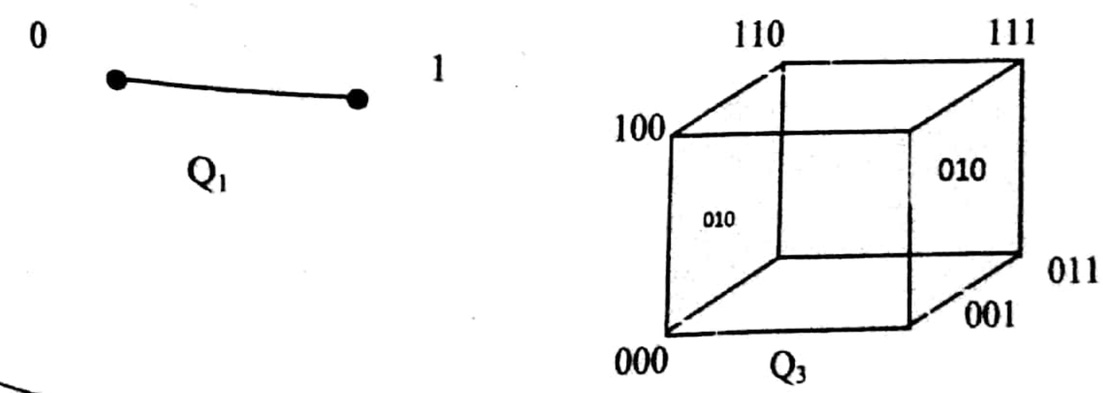
4.3.4 Wheel:

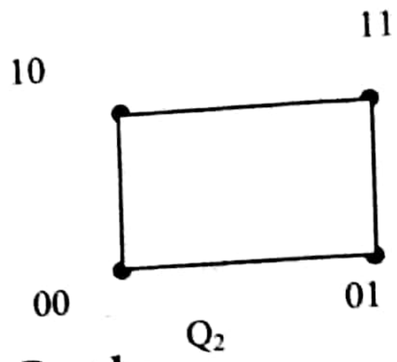
When a new vertex is added to a cycle C_n and this new vertex is connected to each of the n vertices in C_n, we obtain a wheel W_n.



4.3.5 n-cubes:

The n-cube, denoted by Qn, is the graph that has vertices representing the 2^n bit string of length of n. Two vertices are adjacent if and only if the bit strings that they represent differ in exactly one bit position.

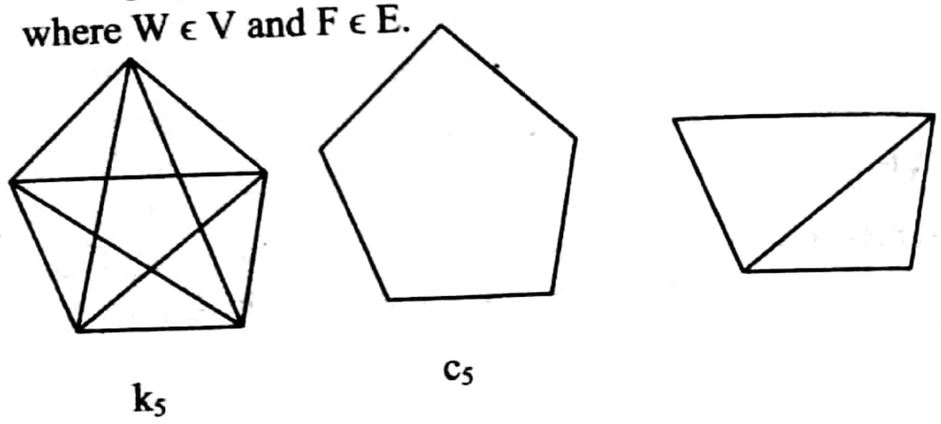




4.3.6 Operations on Graphs

4.3.6.1 Sub graph

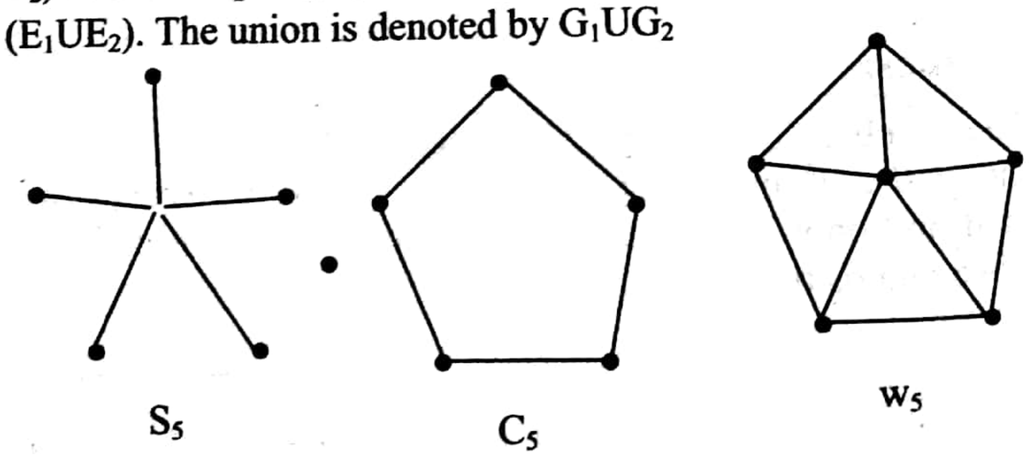
A subgraph of a graph G = (V, E) is a graph H = (W, F), where W ϵ V and F ϵ E.



sub graph of K_5

4.3.6.2 Union

The union of two simple graphs $G_1=(V_1, E_1)$ and $G2 = (V_2, E_2)$ is the simple graph with vertex set $V = (V_1 U V_2)$ and set $E = (E_1 U E_2)$. The union is denoted by $G_1 U G_2$



$S_5 \cup C_5 = W_5$

4.3.7 Representing graphs

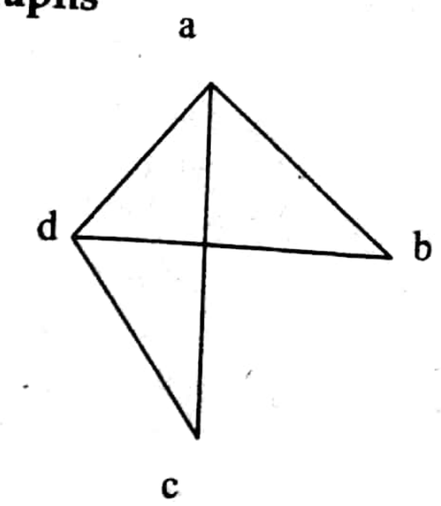


Fig : 1

1. **Edge list for simple graph**

Vertex	Adjacent vertices
a	b, c, d
b	a, d
c	a, d
d	a, b, c

2. **Edge list for Directed graph**

Initial Vertex	End vertex
a	b
b	c
c	b
d	a, b, c, d

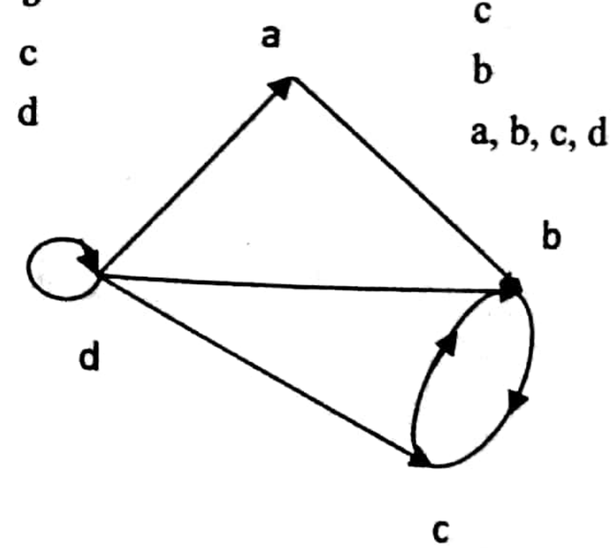


Fig : 2

4.3.8 Adjacency Matrix:

Given a simple graph G = (V, E) with |V|=n of the vertices of the graph are listed in some arbitrary order like $V_1, V_2, \ldots\ldots V_n$, the adjacency matrix A of G, with respect to the order of vertices is n-by-n zero-one matrix ($A = [a_{ij}]$) with the condition

$$a_{ij} = \begin{cases} 1 & if (Vi, Vj) \text{ is an edge of } G \\ 0 & otherwise \end{cases}$$

Adjacency matrix for undirected graph is symmetric, in case of pseudograph or multigraph the representation is similar but the matrix here is not zero-one matrix rather the $(i, j)^{th}$ entry of matrix contains the no. of edges appearing between the pair of vertices.

In case of directed graph, we can extend the same concept as in undirected graph as below

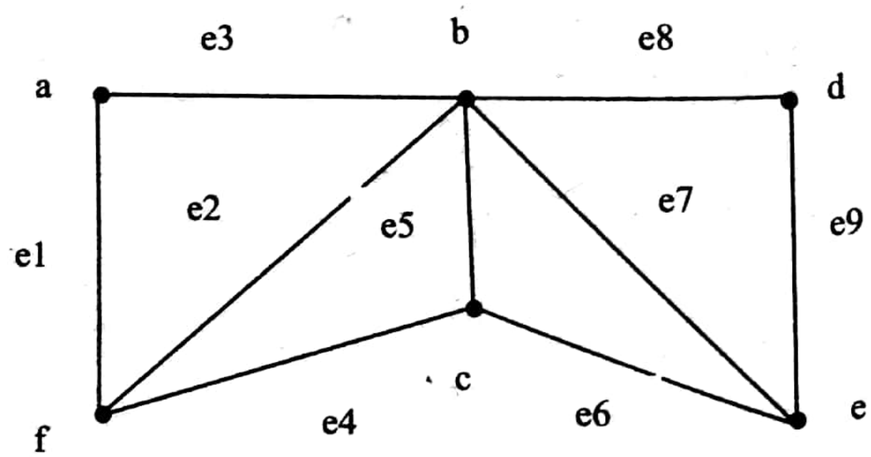
$$a_{ij} = \begin{cases} 1 & if (Vi, Vj) \text{ is an edge of } G \\ 0 & otherwise \end{cases}$$

4.3.9 Incidence Matrix

Given an undirected graph G = (V, E). Assume that vertices of the graph $V_1, V_2 \ldots\ldots\ldots\ldots V_n$ and the edges of graph are $e_1, e_2 \ldots\ldots\ldots\ldots e_m$. The incidence matrix of a graph with respect to the above ordering of V and E is n by m matrix. Where

$$m_{ij} = \begin{cases} 1 & \text{when edge, is incident with V1} \\ 0 & otherwise \end{cases}$$

When the graph is not simple then graph can be represented by using incidence matrix where multiple edges corresponds to two different columns with exactly same entries. Loops are represented with column with only one entry.



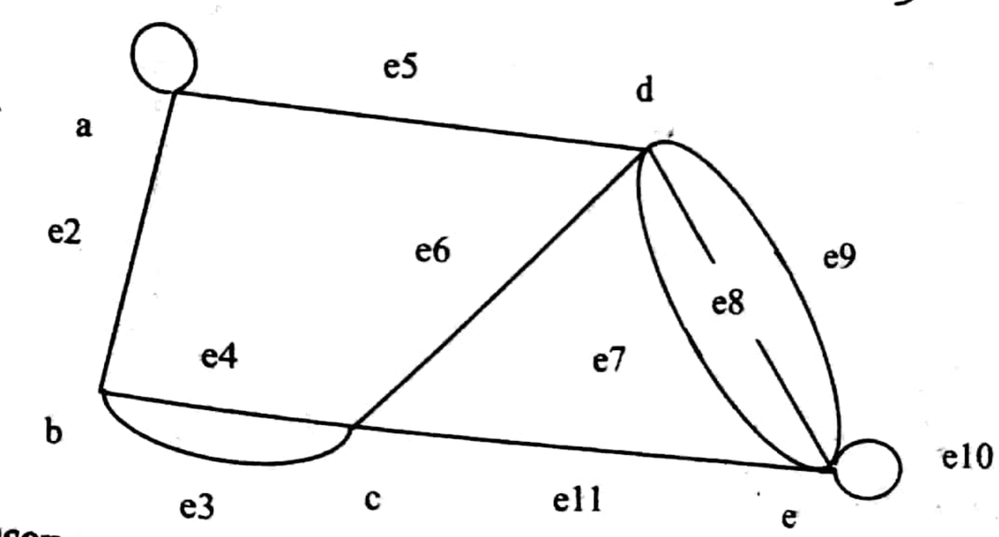
The order of vertices be a, b, c, d, e, f, and edges order be e1, e2, e3, e4, e5, e6, e7, e8, e9

Adjacency Matrix:

$$\begin{array}{c} \\ a \\ b \\ c \\ d \\ e \\ f \end{array} \begin{array}{c} \begin{array}{cccccc} a & b & c & d & e & f \end{array} \\ \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 \end{pmatrix} \end{array}$$

Incidence matrix

$$\begin{array}{c} \\ a \\ b \\ c \\ d \\ e \\ f \end{array} \begin{array}{c} \begin{array}{ccccccccc} e_1 & e_2 & e_3 & e_4 & e_5 & e_6 & e_7 & e_8 & e_9 \end{array} \\ \begin{pmatrix} 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \end{array}$$

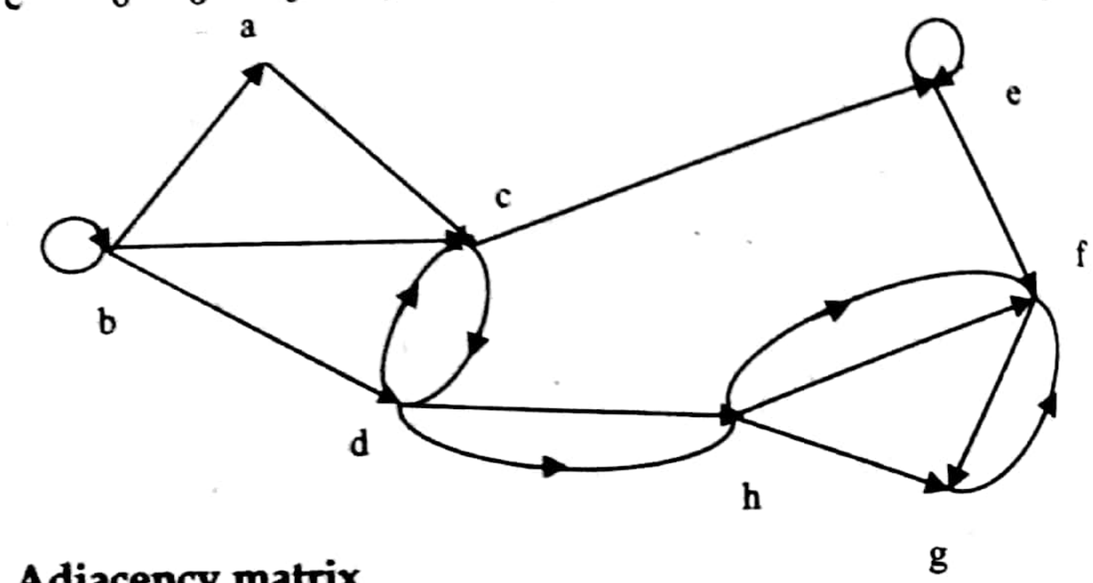


Adjacency matrix

$$\begin{pmatrix} 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 2 & 0 & 0 \\ 0 & 2 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 3 \\ 0 & 0 & 1 & 3 & 1 \end{pmatrix}$$

Incidence matrix

	e_1	e_2	e_3	e_4	e_5	e_6	e_7	e_8	e_9	e_{10}	e_{11}
a	1	1	0	0	1	0	0	0	0	0	0
b	0	1	1	1	0	0	0	0	0	0	0
c	0	0	1	1	0	1	0	0	0	0	1
d	0	0	0	0	1	1	1	1	1	0	0
e	0	0	0	0	0	0	1	1	1	1	1



Adjacency matrix

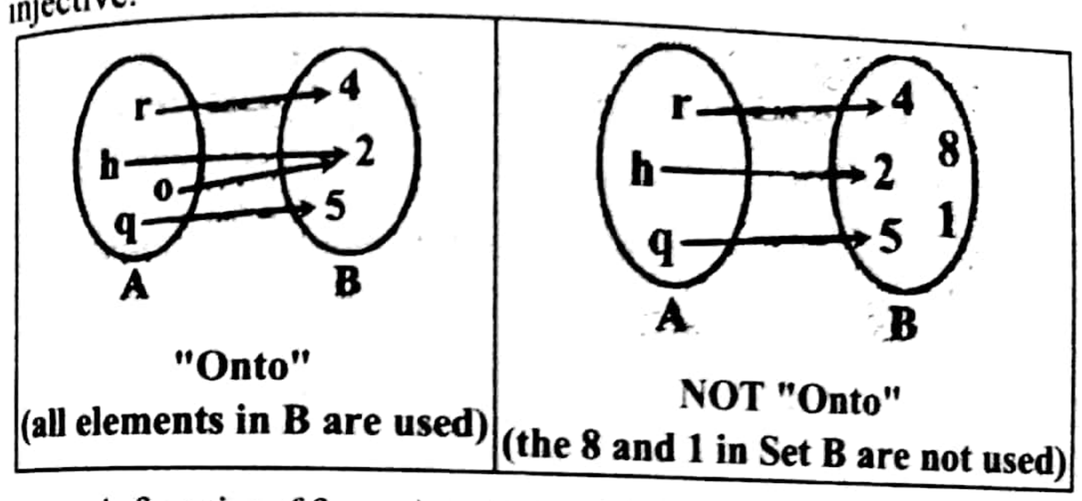
$$\begin{array}{c} \\ a \\ b \\ c \\ d \\ e \\ f \\ g \\ h \end{array} \begin{array}{c} \begin{array}{cccccccc} a & b & c & d & e & f & g & h \end{array} \\ \begin{pmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 2 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 2 & 1 & 0 \end{pmatrix} \end{array}$$

4.3.10 Isomorphism of Graphs

Two simple graphs $G_1=(V_1, E_1)$ and $G_2=(V_2, E_2)$ are isomorphic if there is a **one-to-one and onto function** f from V_1 to V_2 with the property that a and b are adjacent in G_1 if and only if f(a) and f(b) are adjacent in G_2, for all a and b in V_1. Such a function is called isomorphism.

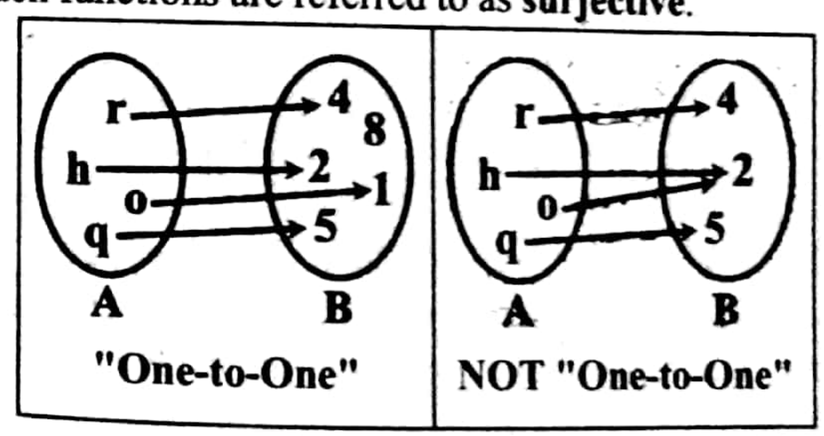
A function f from **A** to **B** is called **one-to-one** (or 1-1) if whenever $f(a) = f(b)$ then $a = b$. No element of **B** is the image of more than one element in **A.**

In a one-to-one function, given any y there is only one x that can be paired with the given y. Such functions are referred to as injective.

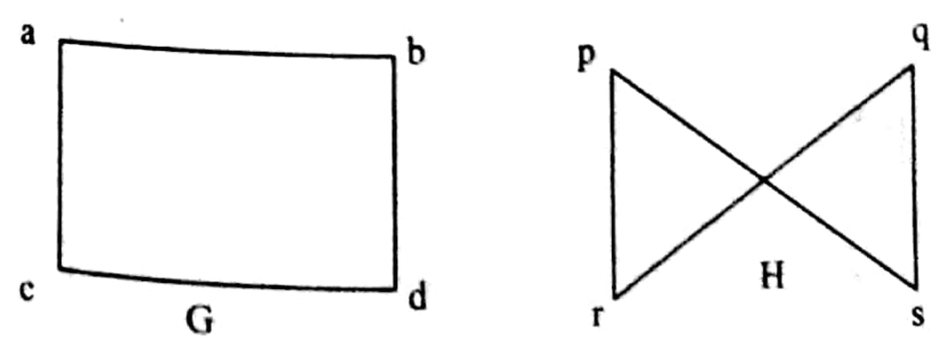


A function f from **A** to **B** is called **onto** if for all b in **B** there is an a in **A** such that $f(a) = b$. All elements in **B** are used.

Such functions are referred to as **surjective.**



E.g. : G = (V, E) and H(W, F)



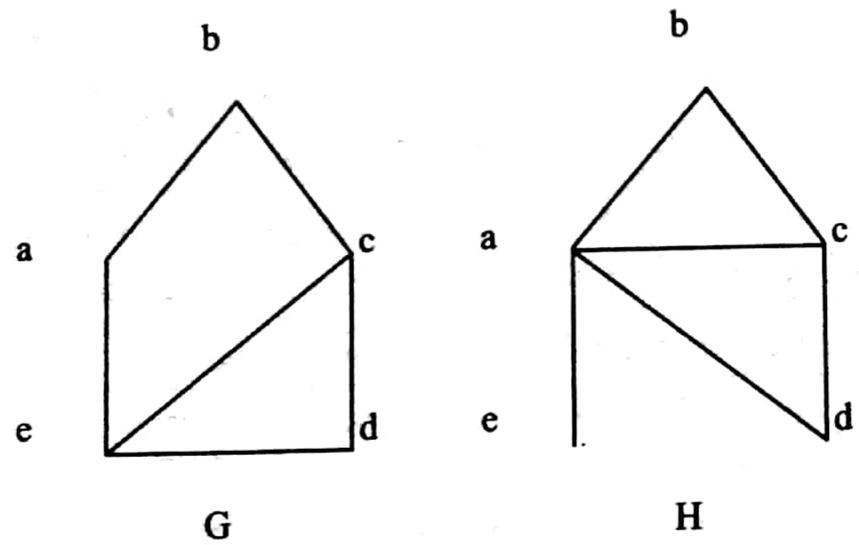
The function f with f(a) = p, f(b) = s, f(c) = r, f(d) =q is a one to one correspondence between V and W.

We can show two simple graphs are not isomorphic by showing they don't share a property is called invariant. These properties are described as follows:

1. Isomorphic simple graphs must have the same number of vertices.
2. Isomorphic simple graphs must have same number of edges.
3. The degree of vertices must be same because the number of edges from the vertex is determined by degree.

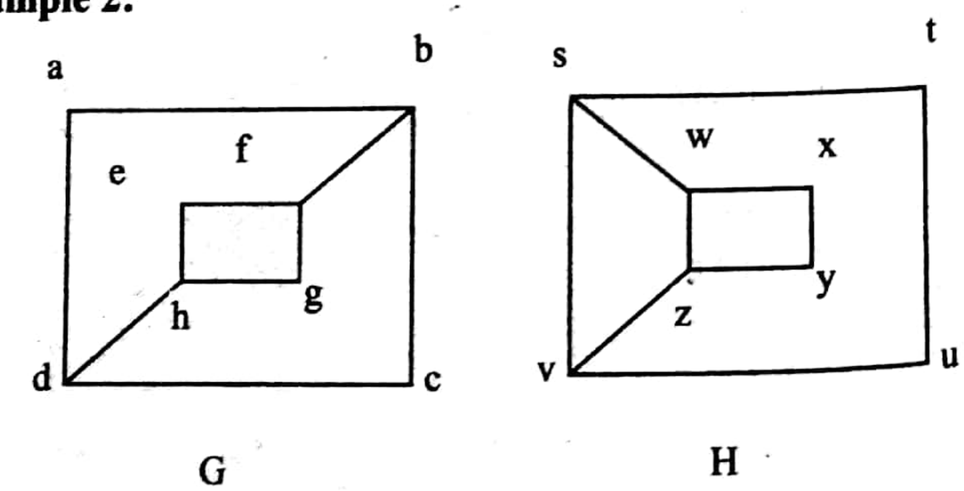
Example 1:

Show that the graphs displayed below are not isomorphic



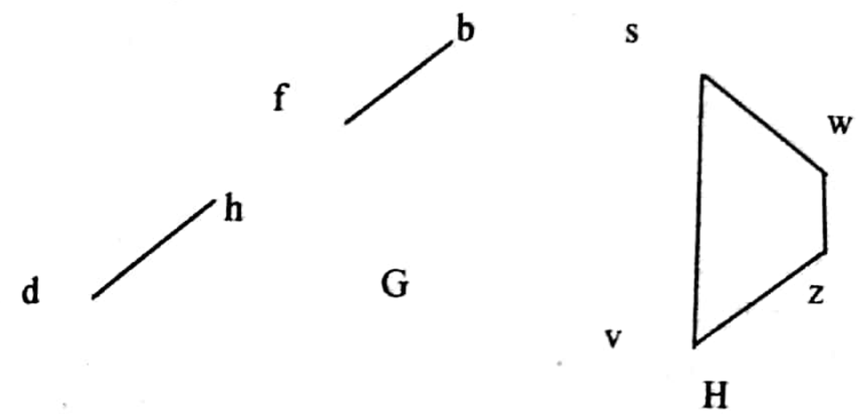
Both graphs have 5 vertices and six edges. But H has a vertex of degree one, namely e, whereas G has no vertices of degree one. Hence G and H are not isomorphic.

Example 2:



Both graphs have eight vertices and ten edges. Also, they both have four vertices of degree two and four vertices of degree three. However, G and H are not isomorphic. Since, deg(a) = 2 in G, a must correspondence to either t, u, x or y in H because these are the vertices of degree two in H. Here, each of these four vertices in H is adjacent to another vertex of degree two in H, which is not true for a in G.

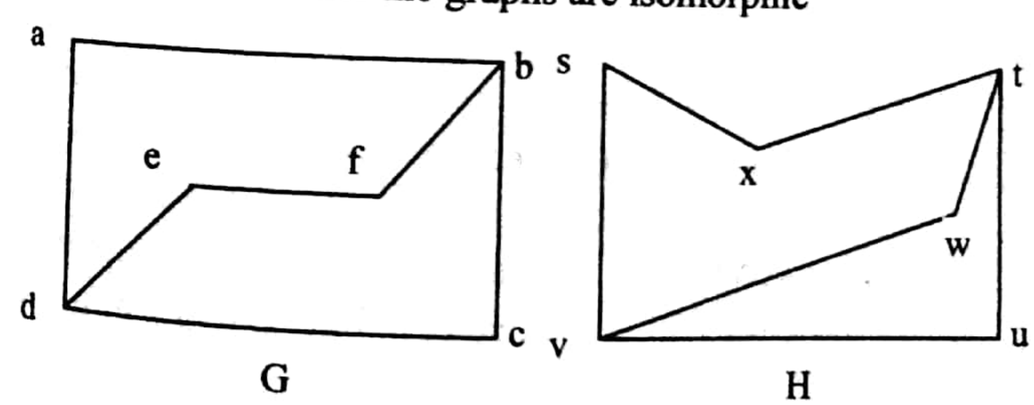
Another way to see that two graphs are not isomorphic is to note that subgraphs formed by connecting the edges from the vertex with same degree in both the graphs are not isomorphic. For e.g.: subgraphs of G and H made up of vertices of degree three and the edges connecting them are not isomorphic.



To show isomorphism of graphs, we can also use adjacency matrix. For this, adjacency matrices of two graphs are same.

Example 3:

Determine whether the graphs are isomorphic

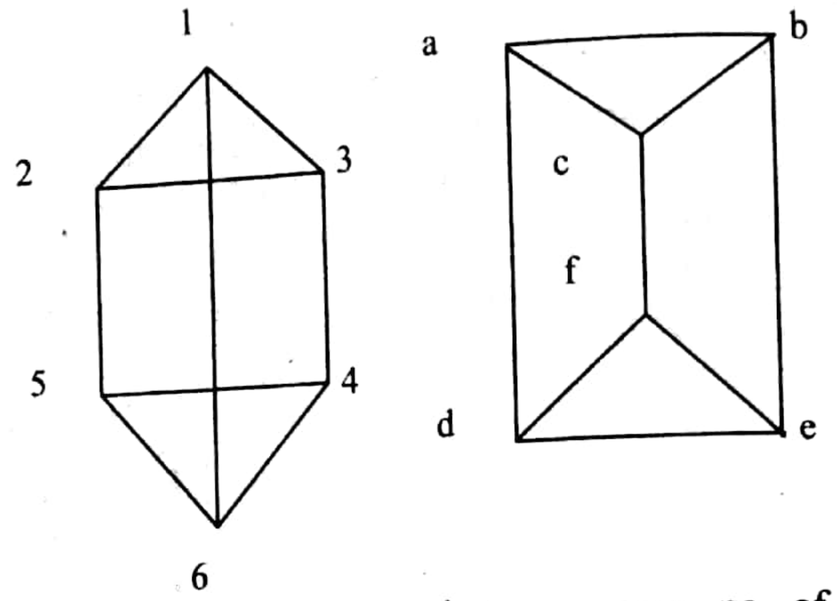


Both graphs have six vertices and seven edges. Both have four vertices of degree two and two vertices of degree three. Also, Subgraphs are isomorphic. Here, we cannot say these graphs are isomorphic or not. For this, we can use adjacency

matrix with the order of vertices a, b, c, d, e, f and w, t, u, v, s, x. we can see similar matrices after drawing adjacency matrices. Hence, these graphs are isomorphic.

Example 4:

Determine whether given two graphs are isomorphic or not?



In above graphs, no.of vertices are same, no. of edges is same and all vetices in both the graphs have degree 3.

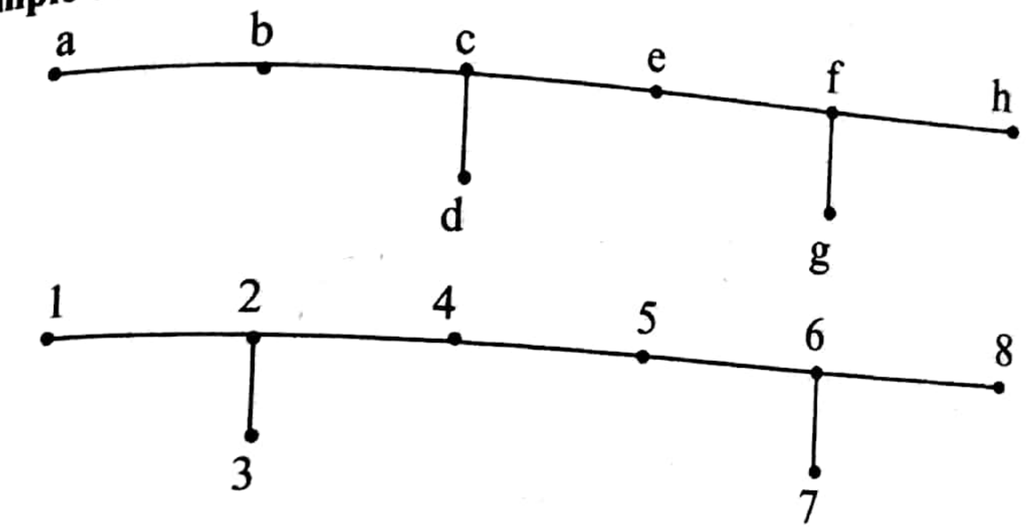
Since the invariants agree on both the graphs, we can try out to find the function that is isomorphism. Take the sequence of vertices from the first graph as 1, 2, 3, 4, 5 and 6. Now, define f(1) = c, f(2) = a here, there is adjacency preservation since we have {1,2}as and edge in the first graph where as {f(1), f(2)} = {c, a} is an edge in second graph. Similarly, we can assign f(3) = b, f(4) = e, f(5) = d, f(6) = f.

Since we found one to one correspondence between verticesof two graphs preserving the adjacency, above graphs are isomorphic. Here, the adjacency matyrices of two graphs in which vertices are ordered in terms of function i.e, 1, 2, 3, 4, 5 and 6 for first graph and c, a, b, e, d and f in the second graph are same.

One to one correspondence between vertices of two graphs preserving the adjacency, above graphs is isomorphic.

Here, the adjacency matyrices of two graphs in which vertices are ordered in terms of function i.e, 1, 2, 3, 4, 5 and 6 for first graph and c, a, b, e, d and f in the second graph are same.

Example 5:



Both graphs have 8 vertices, 7 edges in both graphs two vertices have degree 3, 4 vertices have degree 1 and remaining 2 vertices have degree 2. Since, the invariants agree in both the graphs, we can continue to get the function such that it is isomorphism.

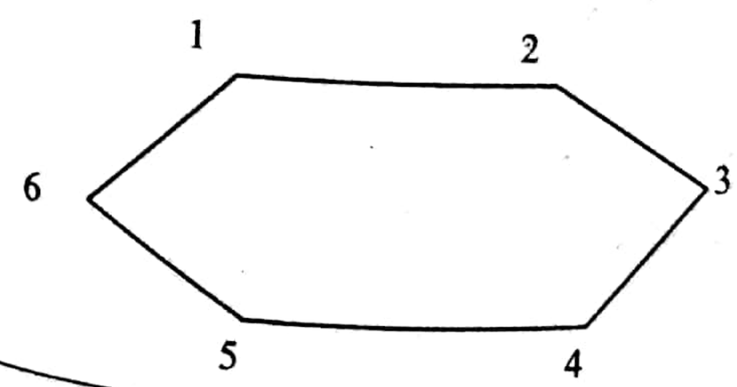
However, in case of first graph, the subgraph containing vertex with degree 3 is not isomorphic with any graph formed by connecting edges with vertex 2 or 6 (both of degree 3). Hence, two graphs are not isomorphic.

4.4 Bipartite Graphs, Planar Graphs, Regular Graphs

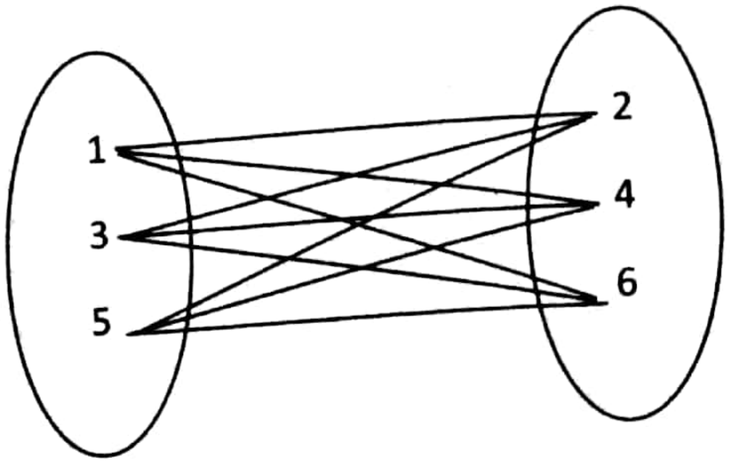
4.4.1 Bipartite Graph

A simple graph G is bipartite if its vertex set can be partitioned into two disjoint subsets V_1 and V_2 such that every edge in the graph connects a vertex from the set V_1 to the vertex of set V_2. No two vertices of the same set are connected by an edge.

Q. ***Is C_6 Bipartite ?***

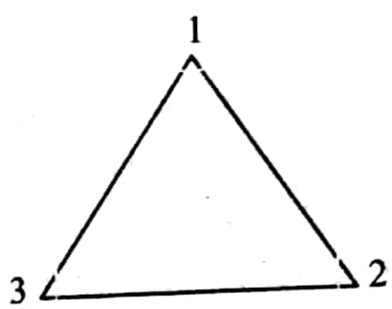


Its vertex set can be partitioned into two sets $V_1 = \{1,3,5\}$ and $V_2 = \{2, 4,6\}$ every edge of C_6 connects a vertex in V_1 with a vertex in V_2.



So, C_6 is a bipartite graph.

Q. ***Is K_3 Bipatite?***

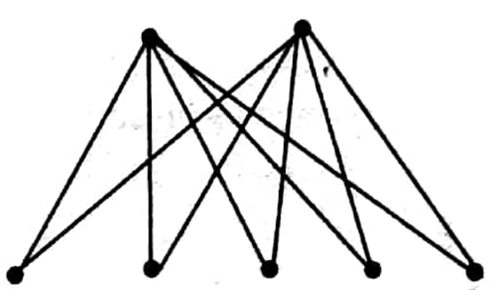


= No

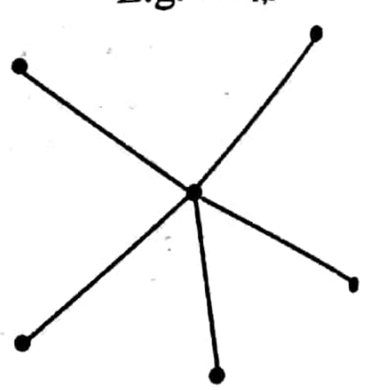
Complete Bipartite Graph

The complete bipartite graph $K_{m,n}$ is the graph where the vertex set is partitioned into two subsets of m and n vertices, respectively. In this graph, there is an edge between two vertices if and only if two vertices are in different subsets of vertices.

E.g. : $K_{2,5}$



E.g. : $K_{1,5}$

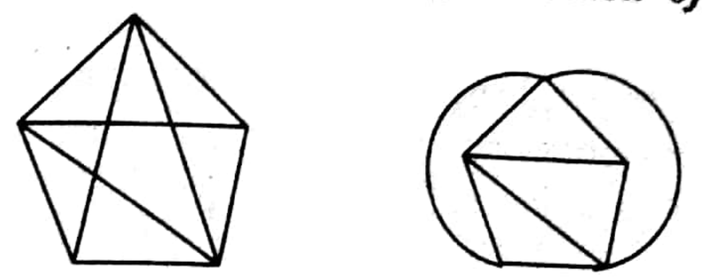


4.4.2 Planar Graph

A graph is called a planar if it can be drawn in the plane without any edges crossing such drawing is called a planar representation of the graph.

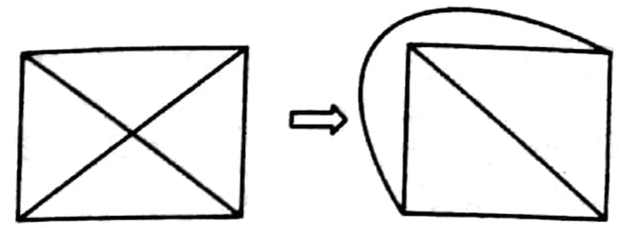
For example:

Draw the graph below as planar representation of the graph.



Solution

2.

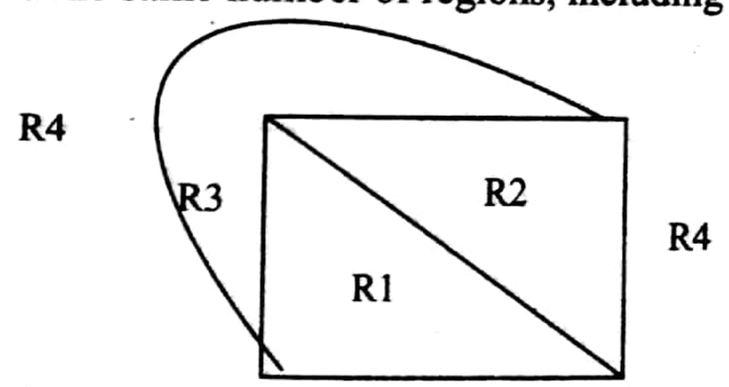


Not all graphs are planar

It may be difficult to show that graph is non-planner. We would have to show that there is no way to draw the graph without any edges crossing.

Regions

Euler showed that all planar representations of a graph split the plane into the same number of regions, including on unbounded region.



Any planar representation of K3,3 V_1 and V_2 connected to V_4 and V_5.

The four edges $\{V_1, V_4\}, \{V_4, V_2\}, \{V_2, V_5\}, \{V_5, V_1\}$ form a closed curve that splits the plane into two regions R_1 and R_2.

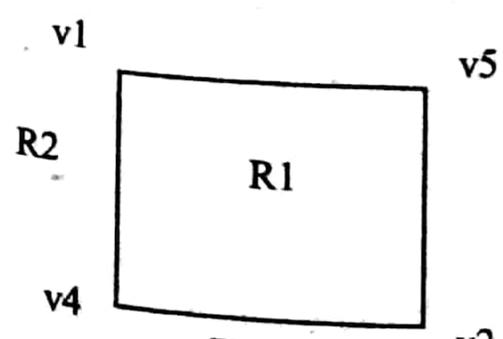


Fig. 2

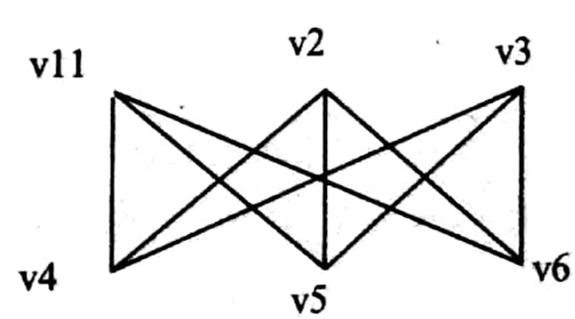
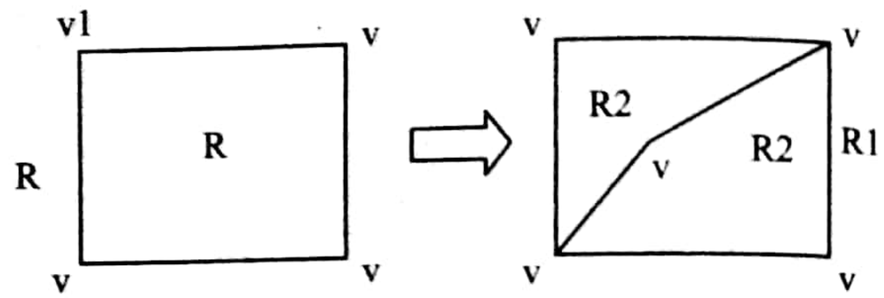


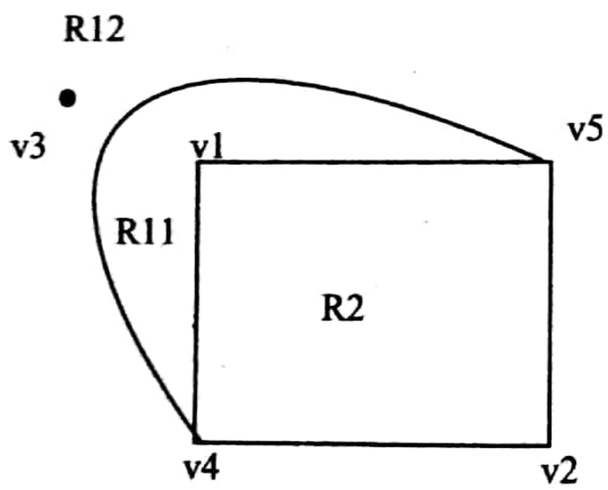
Fig. 1

Next, V_3 must be either in R_1 or R_2. Assume V_3 is in R_2. Then edges (V_3, V_4) and (V_3, V_5) separate R_2 into two sub regions R_{21} and R_{22}.



Now, there is no way to place vertex V_6 without forcing a crossing.

- If V_6 is in R_1, then (V_6, V_3) must cross an edge.
- If V_6 is in R_{21}, then $\{V_6, V_2\}$ must cross an edge.
- If V_6 is in R_{22}, then $\{V_6, V_1\}$ must cross an edge.
- Alternatively assume V_3 in R1. Then the edge (V_3, V_4) and (V_3, V_4) and (V_3, V_5) separate R_1 into two sub-regions R_{11} and R_{12}.



Now there is no way to place vertex V_6 without forcing a crossing.

- If V_6 is in R_2 , then $\{V_6, V_3\}$ must cross an edge.
- If V_6 is in R_{11}, then $\{V_6, V_2\}$ must cross an edge.
- If V_6 is in R_{12}, then $\{V_6, V_1\}$ must cross an edge.

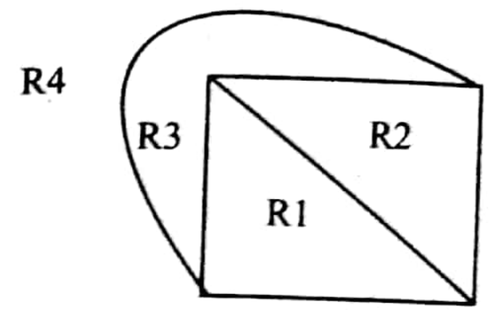
Consequently, the graph $K_{3,3}$ must be non- planar.

Euler devised a formula for expressing the relationship between the number of vertices, edges and regions of a planar graph.

- These may help to determine if a graph can be planar or not.

Euler's Formula:

- Let G be a connected planar simple graph with e edges and v vertices. Let r be the number of regions in a planar representation of G. Then $r = e-v+2$



no. of edge, $e = 6$

no. of vertices, $v = 4$

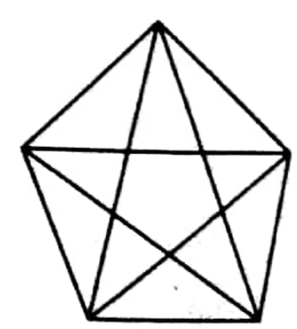
no. of regions, $r = e - v + 2 = 4$

Corollary 1:

If G is a connected planar simple graph with e edges and v vertices where $v \geq 3$, then $e \leq 3v-6$

For example:

Is K_5 planar?



k_5

⇨ K_5 has 5 vertices and 10 edges.

Here , $V \geq 3$

So, if K_5 is planar, it must be true that

$e \leq 3v - 6$

$3v - 6 = 3 \times 5 - 6 = 15 - 6 = 9$

Thus, e must be less than or equal to 9

But e = 10

Therefore K_5 is non-planar.

Corollary 2 :

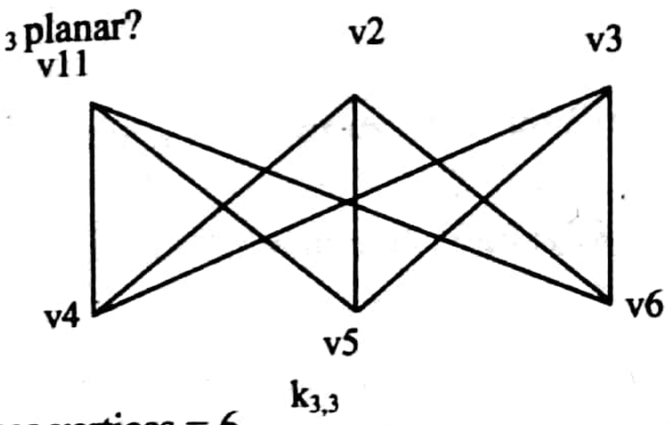
If G is a connected planar simple graph, the G has a vertex of degree not exceeding 5.

Corollary 3:

If a connected planar simple graph has a edges and v vertices with $v \geq 3$ and no circuits of length 3, then $e \leq 2V - 4$

For example:

Is $K_{3,3}$ planar?



$k_{3,3}$

$K_{3,3}$ has vertices = 6

$K_{3,3}$ has edges = 9

Here, $v \geq 3$ and there is no circuits of length 3, If $K_{3,3}$ were planar, then $e \leq 2v - 4$ would be true.

$2V - 4$

$= 2 \times 6 - 4 = 8$

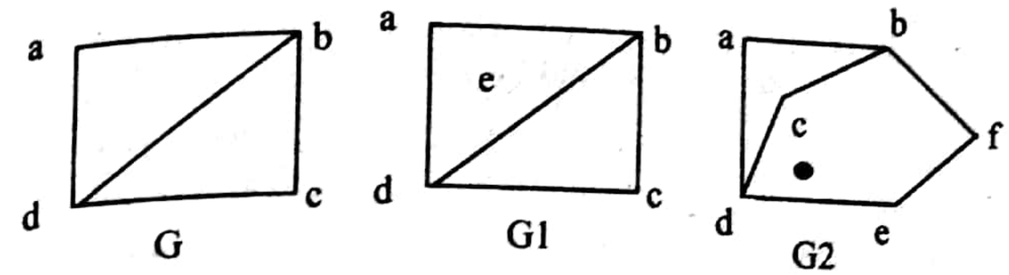
So, e must be ≤ 8

But e = 9

Therefore $K_{3,3}$ is a non-planar.

- If a graph is planar, so will be any graph obtained by removing an edge {u, v} and adding a new vertex w together with edges {u, w} and {w, v}. such an operation is called an elementary sub-division. The graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ are called homeomorphic if they can be obtained from the same graph by a sequence of elementary sub division.

For example :

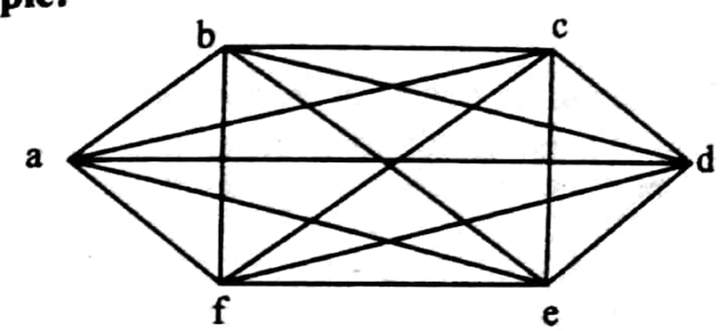


Graphs G_1 and G_2 are homeomorphic to the graph

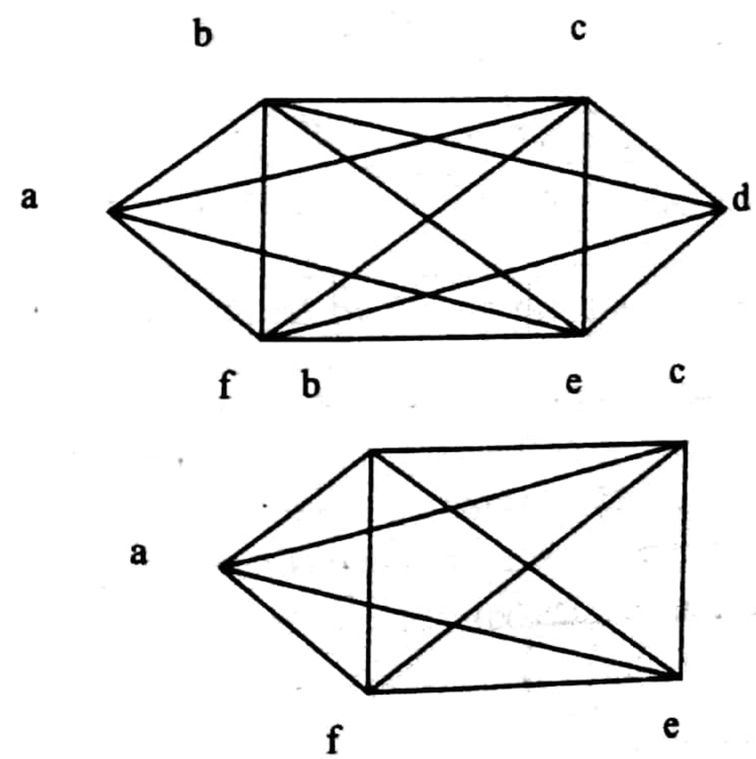
Kuratowski's Theorem (Planarity Testing Algorithm)

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

For example:



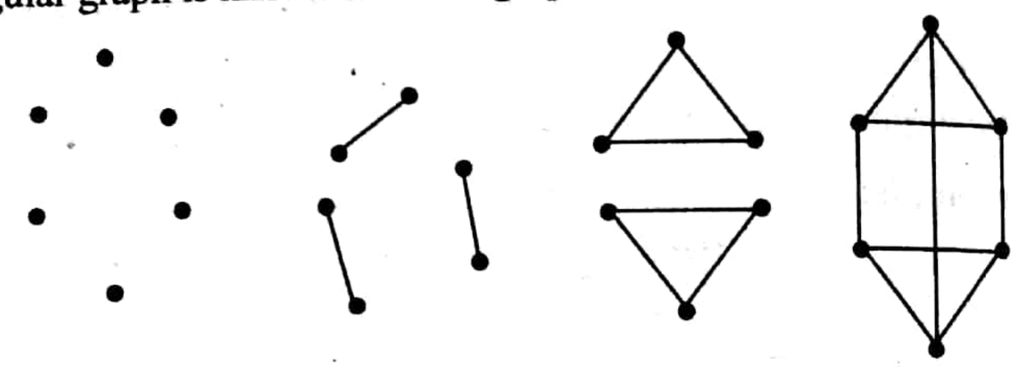
Solution



Above graph is homegrapnic to K_5, the given graph is not planar.

4.4.3 Regular Graph

In graph theory, a regular graph is a graph where each vertex has the same number of neighbors i.e., every vertex has the same degree. A regular graph with vertices of degree k is called a k-regular graph or regular graph of degree k. Regular graph of degree at most 2 are easy to classify. A o-regular graph consists of disconnected vertices a 1-regular graph consists of disconnected edges, and a 2-regular graph consists of disconnected cycles. A 3-regular graph is known as cubic graph.



o-regular graph *1- regular graph* *2-regular graph* *3-regular graph*

4.5 Walk Paths, Circuits, Components

Walk:

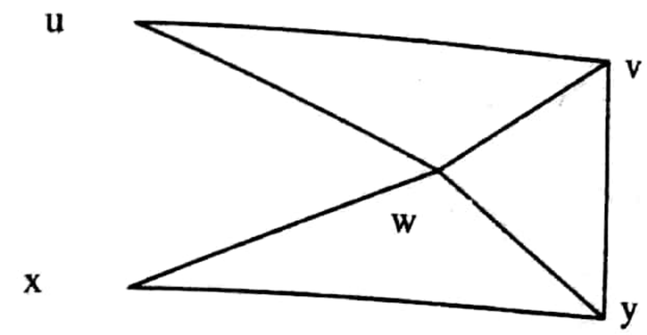
A walk is defined as a finite alternative sequence of vertices and edges of the form.

$v_i\, e_j,\ v_{i+1}\, e_{j+1},\ v_{i+2}\, e_{j+2} \ldots\ldots\ldots\ldots\ldots e_k\, v_m.$

Which begins and ends with vertices, such that

i. Each edge in the sequence is incident on the vertices preceding and following it in the sequence.

A walk that begins and ends at the same vertex is called a closed walk . A walk that is not closed is an open walk. If u be initial vertex, v be terminal vertex. If u = v, then walk w is closed and u≠v it is open . As we move from one vertex of w to next, we are actually encountering or traversing edges of G, Possibly traversing some edges of G more than once. The number of edges encountered in a walk (including multiple occurrence of an edge) is called the length of the walk.



w: x, y, w, y, v, w is therefore a walk. Length of walk is the no. of edges. Here 5 is the length.

A walk of length 0 is a trivial walk.

We define a u-v **trail** in a graph G to be a u-v walk in which **no edge** is traversal more than once.

Since w:x, y, w, y, v, w is not an x-w trail as the edge wy is repeated.

T: u, w, y, x, w, v is a u-v trail.

Trail T repeats the vertex w.

A u-v walk is a graph in which **no vertices** are repeated in u-v **path.**

P : u, w, y, v is a u-v **path.**

If no vertex in a walk is repeated then no edge is repeated either. Hence, every path is a trail.

A circuit in graph G is a closed trail of length 3 or more. Hence, a circuit begins and ends at the same vertex but repeats no edges. In a circuit, vertices can be repeated in addition to the first and last.

For example:

c : y, w, u, v, w, x, y

e: x, y, w, u, v, w, x

c: w, x, y, w, u, v, w

is a circuit.

Cycle

A circuit that repeats no vertex except for the first and last is a cycle. A K-cycle is a cycle of length K. A 3-cycle is also referred to as a triangle. A cycle of odd length is called odd cycle, a cycle of even length is called as even cycle.

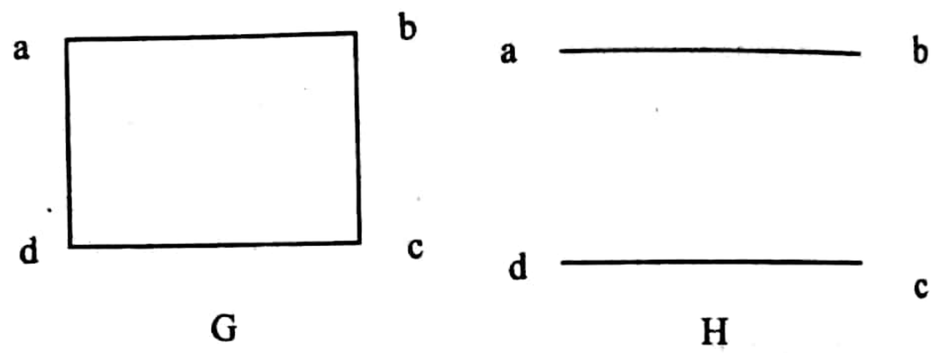
For example:

C' = x, y, v, w, x is a cycle.

4.6 Connectdness Algorithm

Connectedness

- An undirected graph is called connected if there is a path between every pair of distinct vertices of the graph.
- There is a simple path between every pair of distinct vertices of a connected undirected graph.

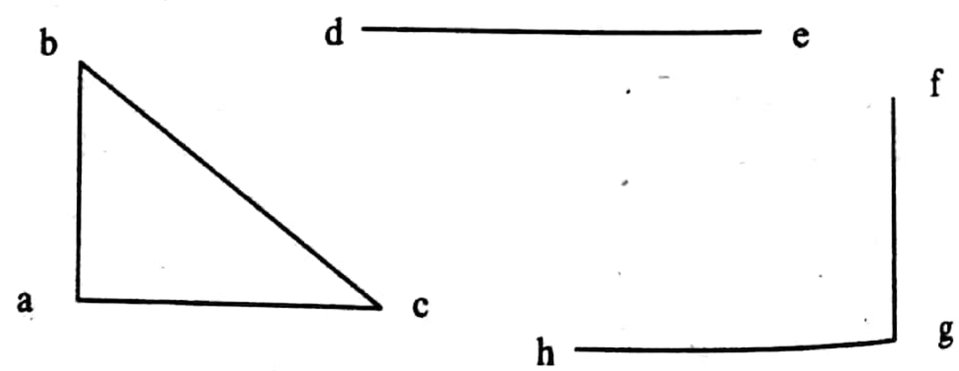


Graph G is connected but graph H is not connected.

- A graph that is not connected is the union of more than one connected graphs that do not share the common vertex. These disjoint connected sub graphs are called connected components of a graph.

For example:

What are the connected components of the following graph?



Connected components are {a, b, c,}, {d, e}, {f, g, h}

4.7 Cut Sets and Cut Vertices

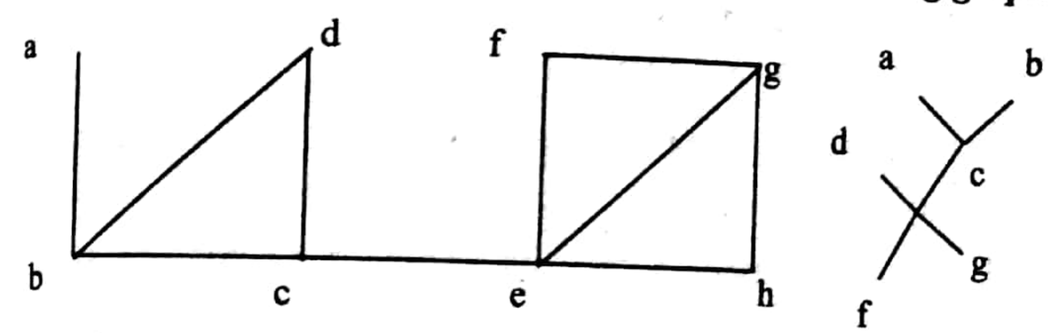
Cut Edges and Vertices

A cut vertex is a vertex, by removing which one can partition the graph to more connected components.

A cut edge is an edge by removing which one can partition the graph to more connected components.

If multiple edges need to be removed to partition the graph, then the minimal set of such edges are calledcut set.

Q. Find the cut vertices and cut edges in the following graph:



Cut vertices : c and e

Cut edge ; {c, e}

Vertex V is cut vertex of G iff deg v>2

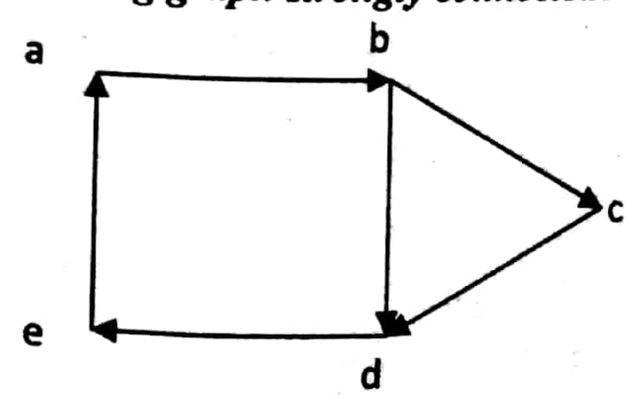
Connectedness in Directed graphs

A directed graph is strongly connected if there is a directed path between every pair of vertices.

A directed graph is weakly connected if there is a path between every pair of vertices in the underlying undirected graph.

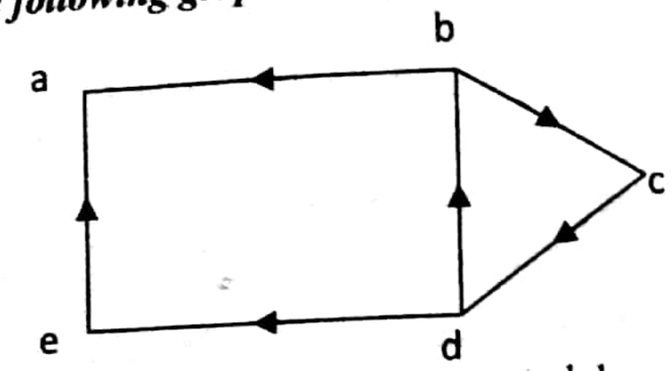
For example:

Q.N. Is the following graph strongly connected?



The graph is strongly connected because there is a directed path between every pair of vertices. If a directed graph is strongly connected, then it must also be weakly connected.

2. ***Is the following graph strongly connected?***



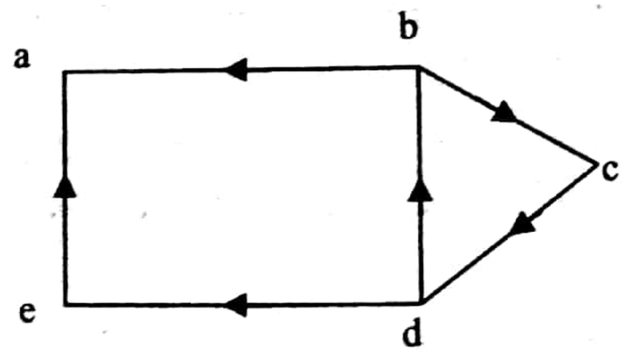
This graph is not strongly connected because there is no directed path between a and b, a and e etc.

However, it is weakly connected (imagine the graph as an undirected graph)

The subgraphs of a directed graph of G that are strongly connected but not contained in larger strongly connected subgraphs (the maximal strongly connected subgraphs) are called the strongly connected components or strong components of G.

For example:

What are strongly connected components of following graphs:



This graph has three strongly connected components

- The vertex a
- The vertex e
- The graph consists of

  V = { b, c, d} and

  E = {{b, c}, {c, d}, {d, b}}

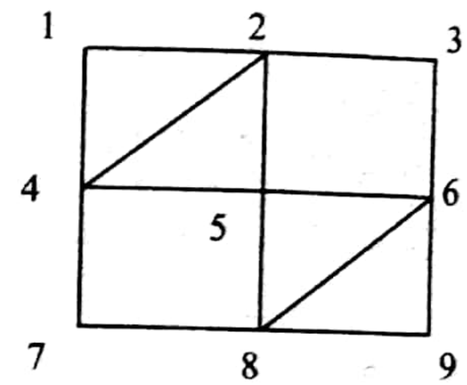
4.8 Eulerian Graph

Euler path and circuits

A circuit C in a graph G is called Eulerian circuits if C contains every edge of G. Since no edge is repeated in a circuit, every edge appears exactly once in Eulerian Circuits.

A connected graph that contains as Eulerian circuit is called Eulerian graph.

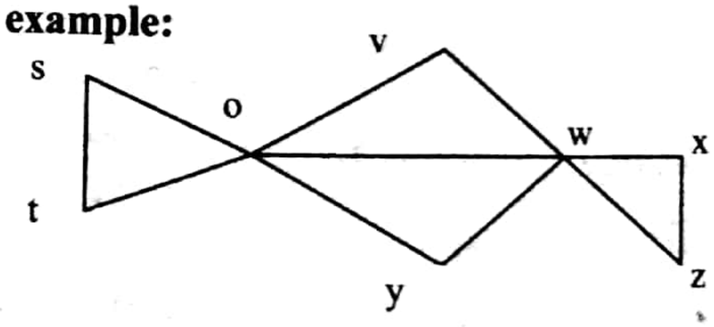
For example:



Euler circuit is 1, 2, 3, 6, 9, 8, 7, 4, 5, 8, 6, 5, 2

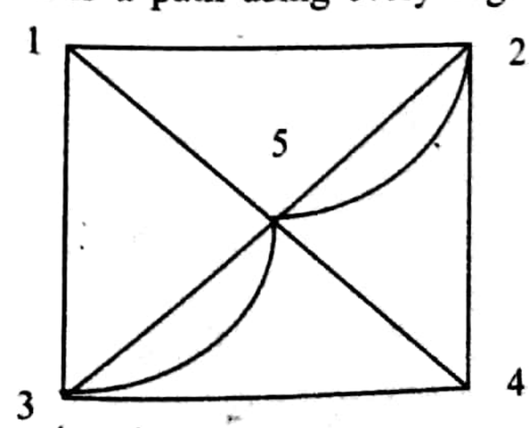
For connected graph G, we refer to an open trail that contains every edge of G as an Eulerian trail

For example:



T = u, s, t, u, v, w, y, u, w, x, z, w

An euler path is a path using every edge the graph exactly once.



No Euler circuit exist

Necessary and sufficient conditions

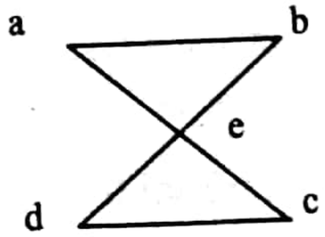
A connected mutligraph has a euler circuit iff each of its vertices has an even degree.

A connected multigraph has a eulerian trail iff exactly two vertices of G have odd degree .

For example:

Q. Which of the following graphs is a eularian circuit?

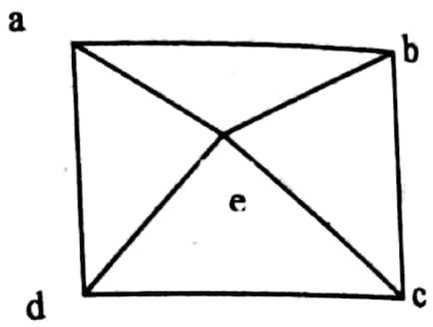
1.



Yes

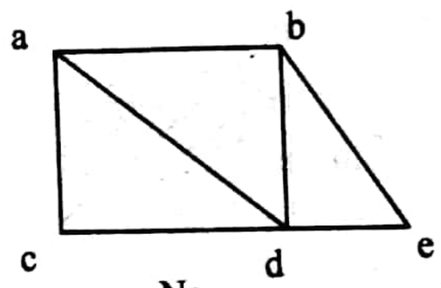
(a, e, c, d, e, b, a)

2.



No

3.

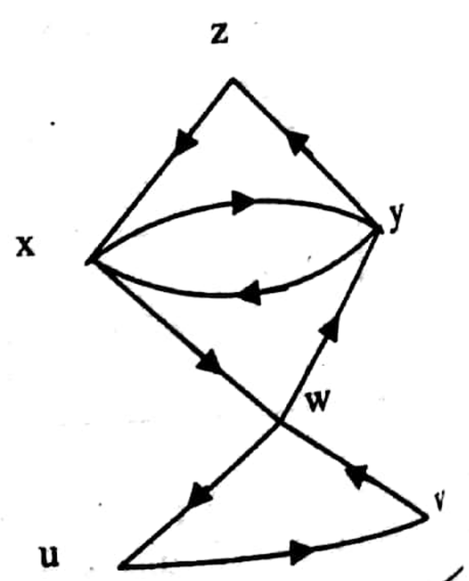


No

Euler trail : (a, c, d, e, b, d, a, b)

Eulerian for Directed graph

A non trivial connected digraph D is eulerian if and only if od V = id v for every vertex

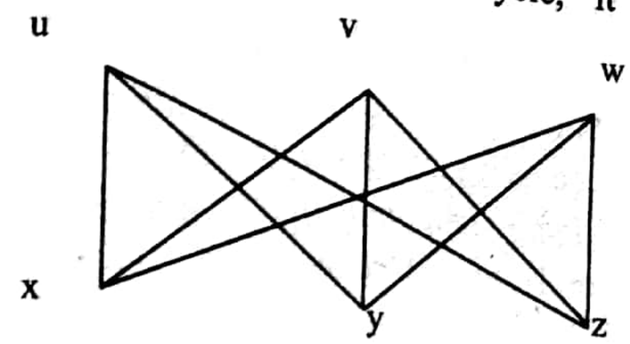


C: u, v, w, y, z, x, y, x, w, u, is eulerian circuit.

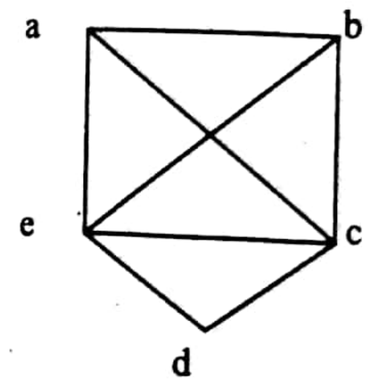
4.9 Hamiltonian Graph

Hamilton Paths and Circuits

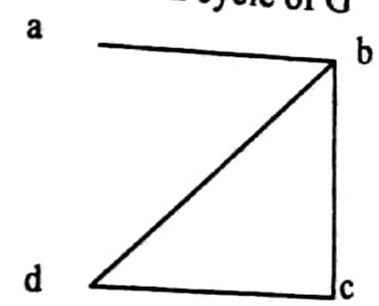
A cycle in graph G that contains every vertex of G is called a Hamiltonian cycle. Certainly a graph G ($n \geq 3$) is Hamiltonian path in G. If a graph contains Hamiltonian cycle, it contains Hamiltonian path.



c: u, x, v, y, w, z, u is a Hamiltonian cycle of G



(a, b, c, d, e, a)



(a, b, c, d)

Hamilton circuit

For Hamilton circuit

No vertex of degree 1

If a node has degree 2, then both edges incident to it must be in any Hamilton circuit incident to it must be in any Hamilton circuit.

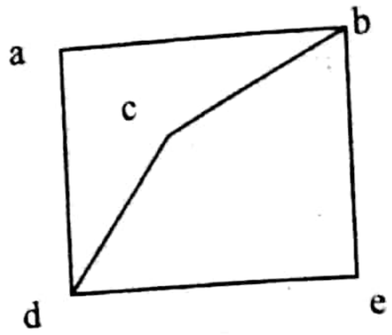
No smaller circuits contained in any Hamilton circuit (the start/end point of any smaller circuit would have to be visited twice).

A sufficient condition

Let G be a connected simple graph with a vertices with $n \geq 3$.

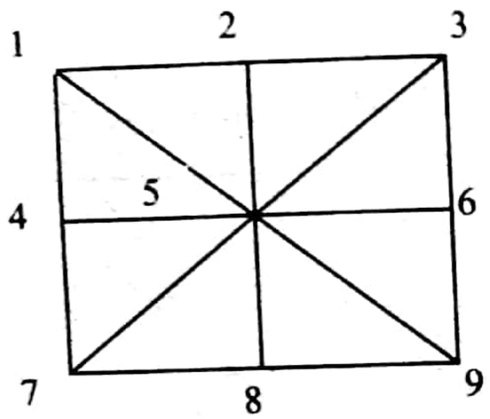
G has Hamilton circuit if the degree of each vertex is $\geq n/2$.

For example 1:



In this graph, there is no hamilton circuit since node has degree 2 and both edges from it must be in Hamilton circuit which is not possible one of the hamilton path is a, b, c, d, e.

Example 2 :



In graph, we can have hamilton circuit, the circuit can be 1, 2, 3, 5, 6, 9, 8, 7, 4, 1. Since, there is circuit. We can have path also.

Dirac's theorem:

If G is a simple graph with n vertices with $n \geq 3$ such that the degree of every vertex in G is at least $\frac{n}{2}$, then G has a hamilton circuit.

Ore's Theorem:

If G is a simple graph with n vertices with ≥ 3 such that deg (u) + deg(v)$\geq$n for every pair of non adjacent vertices u and v in G, then G has a Hamilton path.

4.10 A Shortest Path Algorithm

1. Weighted Graphs

Graphs that have a number assigned to each edge (u, v) has a weight (u, v). Each weight is a real number.

Weight can represent distance, cost, time, capacity etc.

The length of a path in a weighted graph is the sum of the weights on the edges.

2. Shortest Path

There are several algorithm that find a shortest path between two vertices in a weighted graph. Dijkstra's algorithm is one of them.

Dijkstra's Algorithm

Procedure Dijkstra (G: weighted connected simple graph with all weights positive)

G has vertices a = v_0, v_1,....., v_n = 2 and weighs (v_i, v_j) is not an edge in G.

For i = 1 to x

$L(v_i) = \infty$

L(a) = 0

s :: = ϕ

{ the labels are now initialized so that the label of a is 0 and all other labels are ∞ and s is an empty set}

While z ϵs

Begin

u: a vertex not in s with L(u) minima

s: s {u}

for all vertices u not in S.

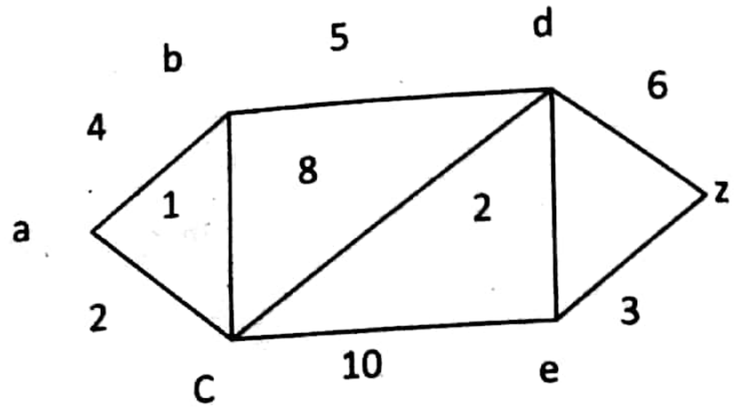
if L(u) + w (u, v) < L(v) then L(V) : = L(u) + w(u, v)

{this adds a vertex to s with minimal label and updates the labels of vertices not in s}

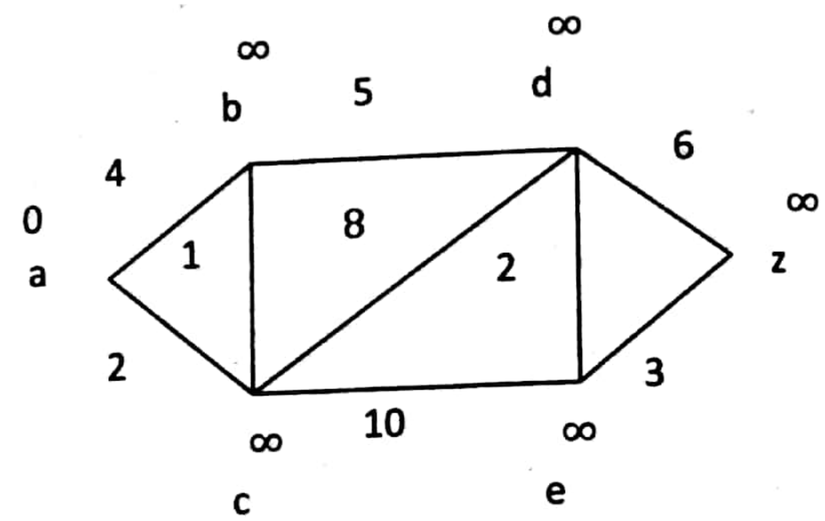
End {L(Z) ={ length of a shorter path from a to z}

QN.1. Use Dijkstra's algorithm to find the shortest path from the vertices a to z in the following weighted graph given below.

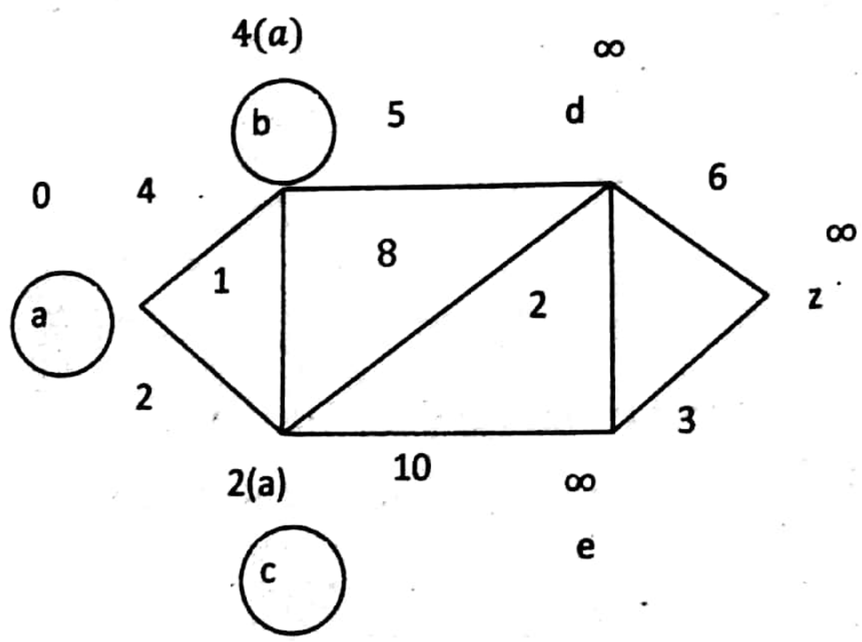
1)



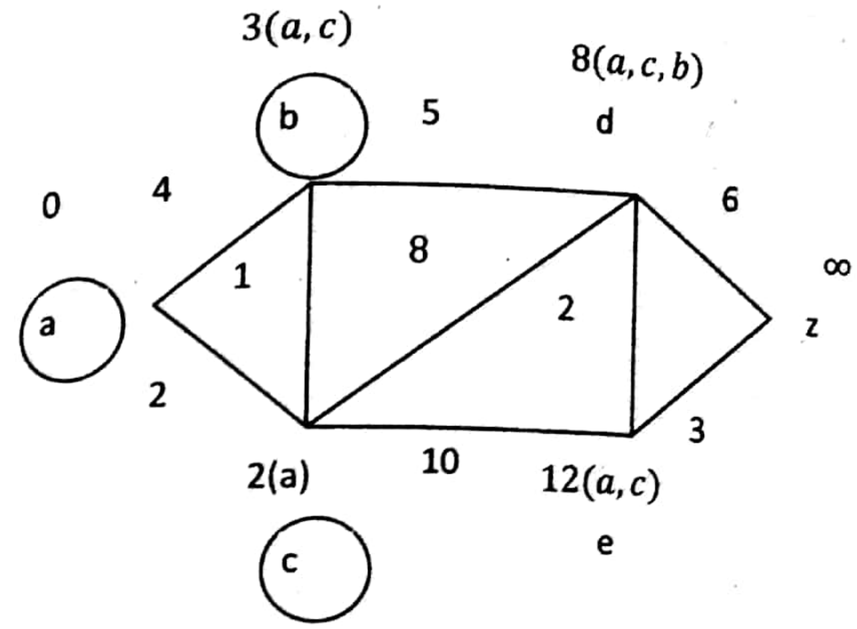
2)



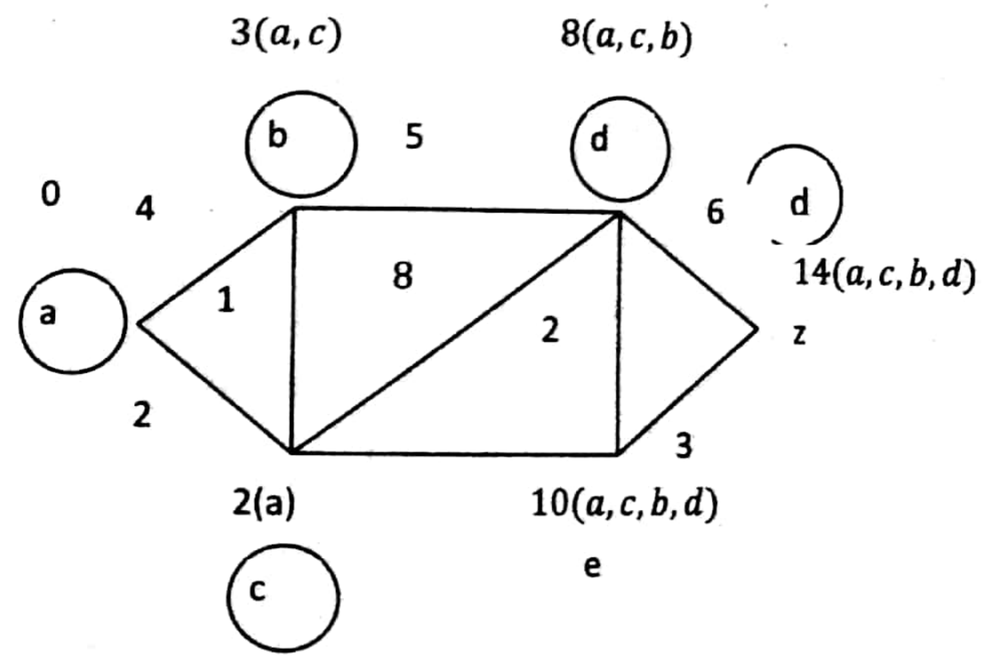
3)



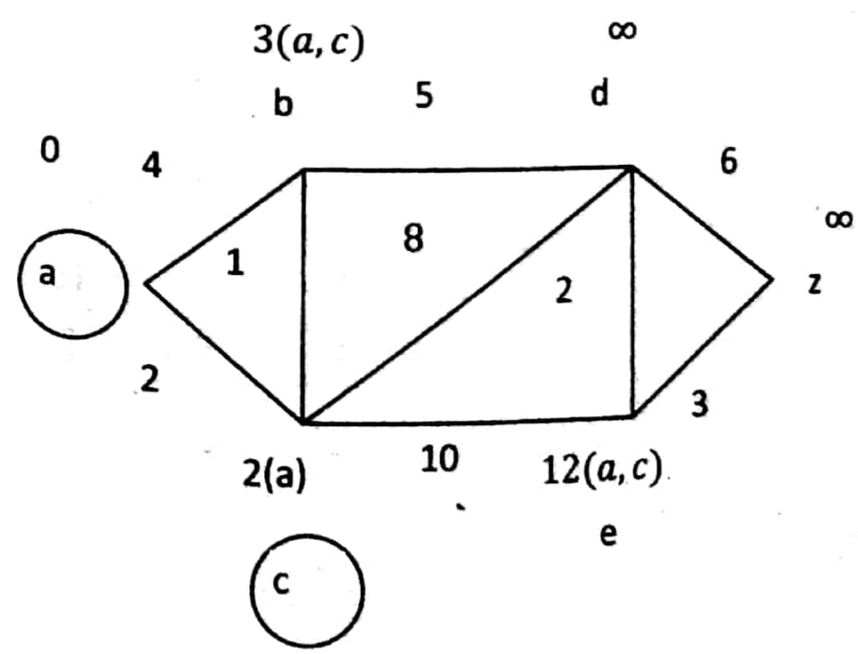
4)



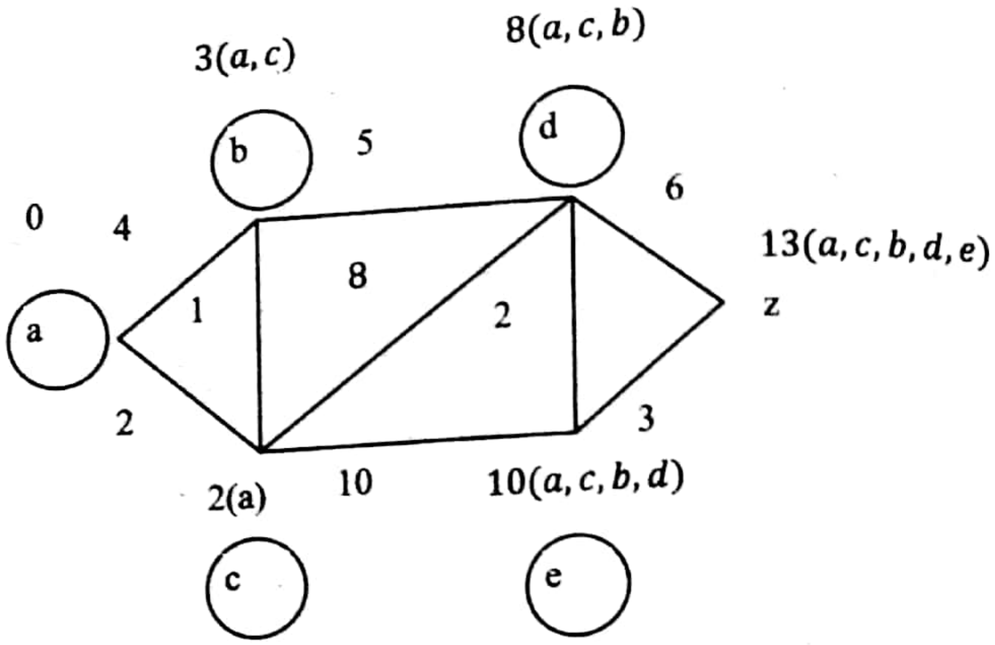
5)

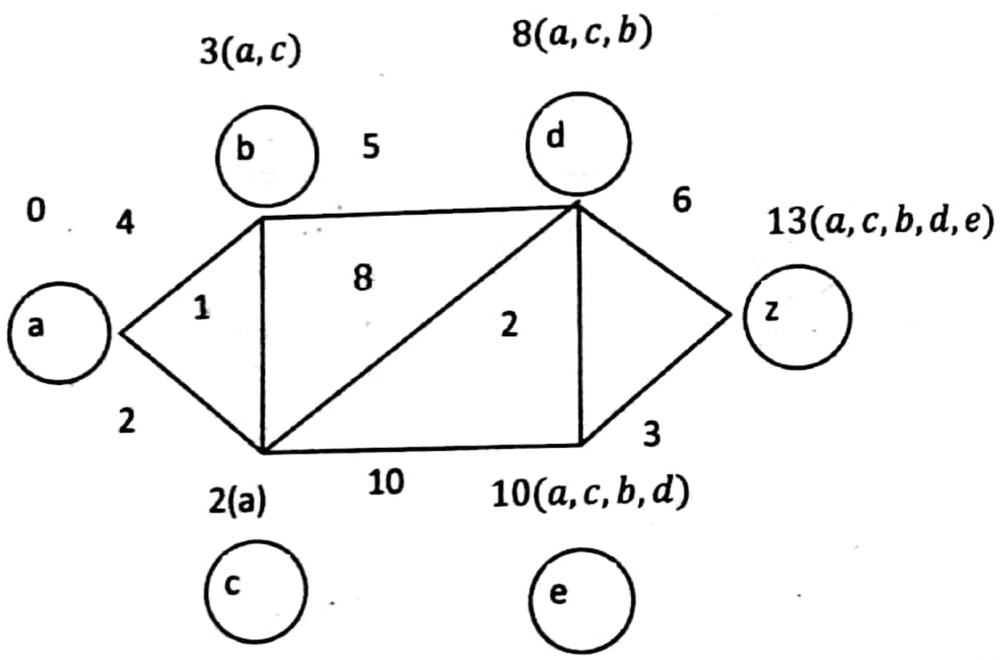


6)



7)



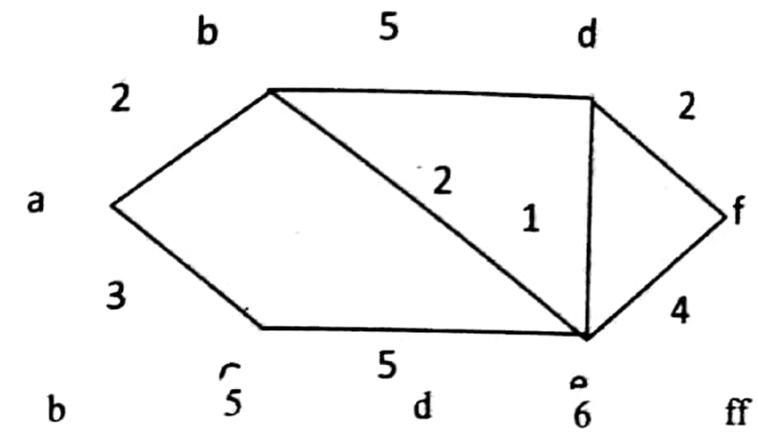


w(u,v)	A	b	c	D	e	z	
	A	0	4	2	∞	∞	∞
	B	4	0	1	5	∞	∞
	C	2	1	0	8	10	∞
	D	∞	5	8	0	2	6
	E	∞	∞	10	2	0	3
	2	∞	∞	∞	6	3	0

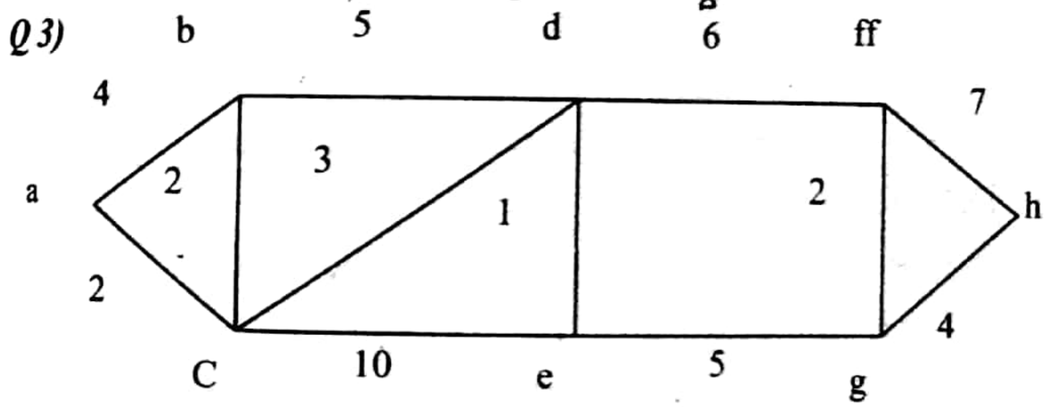
s	ϕ	{a}	{a,c}	{a,c,b}	{a,c,b,d}	{a,c,b,d,e}	{a,c,b,d,e,z}
L(a)	0	0	0	0	0	0	0
L(b)	∞	4	3	3	3	3	3
L(c)	∞	2	2	2	2	2	2
L(d)	∞	∞	10	8	8	8	8
L(e)	∞	∞	12	12	10	10	10
L(f)	∞	∞	∞	∞	14	13	13

Hence, the shortest path from a to z has length L(z) = 13

Q 2)



Q 3)



The travelling salesman problem

The travelling salesman is one of the classical problems in computer science. A travelling salesman wants to visit a number of cities and then return to his strarring point of course he wants to save time and energy. So he wants to determine the shortest cycle for his trip.

We can represent the cities and the distances between them by a weighted, complete, undirected graph.

The problem then is to find the shortest cycle of minimal total weight that visits each vertex exactly one.

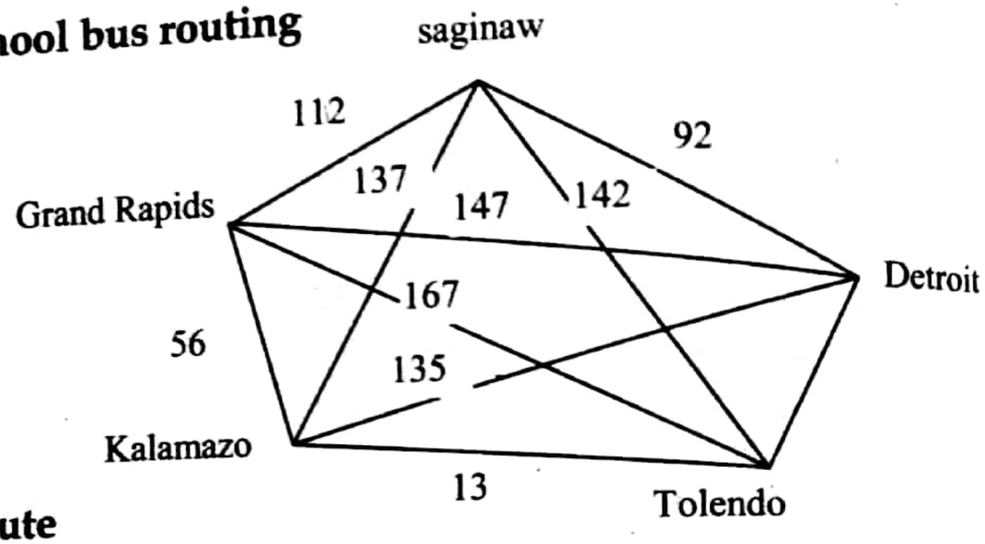
Importance

Variety of scheduling application can be solved as a traveling salesman problem.

Examples:

Ordering drill position on a drill press.

School bus routing



Route

- Detroit, Toledo, Grand - Rapids - saginaw - kala - mazoo - Detroit 610
- Detroit Toledo - Grand Rapids - kalamazoo - saginav - Detroit 546
- Detroit Toledo - Grand Rapids - kalamazoo - saginaw - Grand - Raphids - Detroit 588
- Detroit - Toledo - kalamazoo - Grand Rapids - saginaw - Detroit 458
- Detroit - Toledo - saginaw - kalamazoo - Grand Rapids - Detroit 540
- Detroit - Toledo - saginaw - Grand Rapids - kalamazoo - Detroit- 504
- Detroit - saginaw - Tolede - Grand Rapids - kalamazoo - Detroit - 598
- Detroit - saginaw - Toledo - kalamazoo - Grand Rapids - Detroit - 576
- Detroit - saginaw - kalamazoo - Toledo - Grand Rapids - Detroit - 682

- Detroit - saginaw - Grand Rapids - Toledo - kalamazoo - Detroit - 646
- Detroit - Grand Rapids - saginaw - Toledo - kalamazoo - Detroit - 670
- Detroit - GrandRapids - Toledo - saginaw - kalamazoo - Detroit - 728

4.11 Tree as a Directed Graph

Tree

A tree is a connected undirected simple graph with no simple circuit. A tree is a particular type of graph.

Properties

1. There is a unique simple path between any 2 of its vertices.
2. No loops
3. No multiple edges

For example:

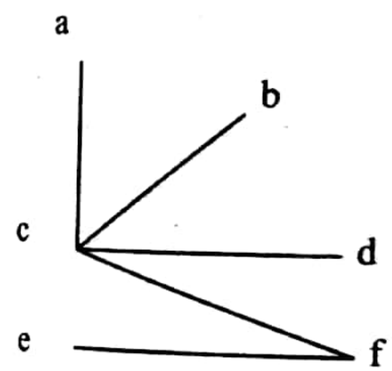


Fig: G1

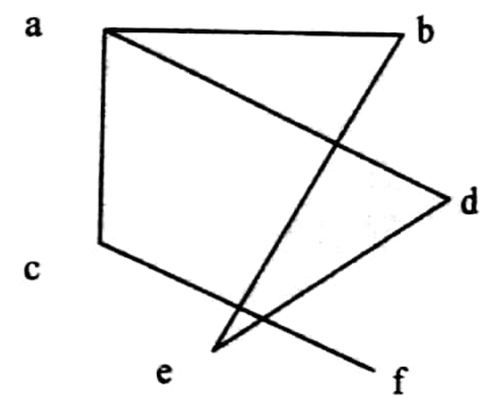


Fig: G2

G1: It is tree because it is connected graph with no simple circuit.

G2: It is not a free because there is a cycle a, b, e, d, a

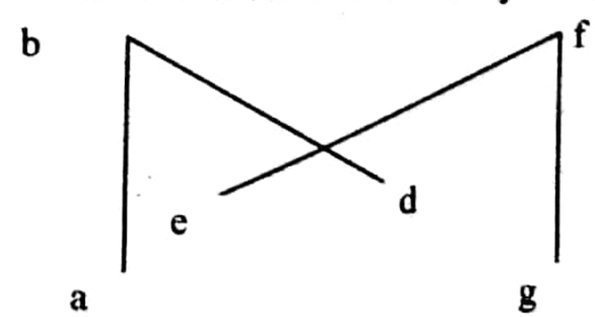


Fig: G3

An undirected graph having no simple circuit and is not connected is called forest. The forest has each of its connected components an tree G3 displays a forest.

Theorem 1:

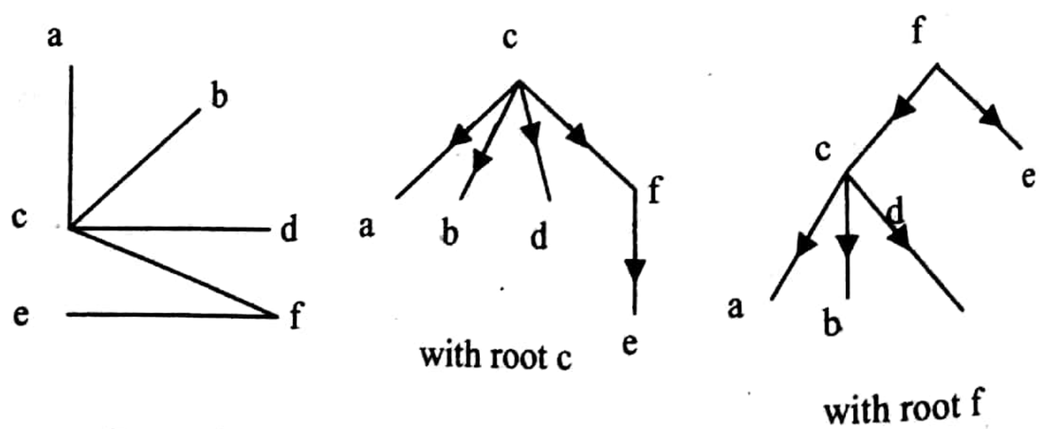
An undirected is a tree if and only if there is a unique simple path between any two of its vertices.

Proof:

Assume that T is a tree. Since T is a tree. It is a connected simple graph with no simple circuits. Let x and y be two vertices of T. We know that every connected graph has a simple path between every pair of vertices. So, there is a simple path from x to y. This path must be unique because, if the path between x and y is not unique then there is another path between x and y that uses edges different from the path, then reversing the path i.e, going to from x to y from first path and going from y to x through second path forms a circuit. This is a contradiction that is a tree. Hence, there is a unique simple path between any two vertices of a tree.

Rooted (Directed) Tree

In many applications of trees, a particular vertex of a tree is designated as the root. A rooted tree is a tree in which one vertex has been designated as the root and every edge is directed away from the root. We can change an unrooted true into a rooted by choosing any vertex as the root. The tree in which root is defined produces a directed graph.



- Root :Vertex with in degree O.
- In a rooted tree, if V is the vertex in T other than root, then the **Parent** of V is a vertex of u in T such that there is directed edge from u to v.

- In this scenario v is called **child** of u.
- Vertices with same parents are called **Siblings**
- All the vertices that appear in the path from root to some vertex v in T, including root are called **ancestors** of V.
- The **descendents** of a vertex V are those vertices that have V as their ancestor.
- All the vertices that have children are called internal vertices (root is also an internal vertex if the tree has more than one vertices.

For example:

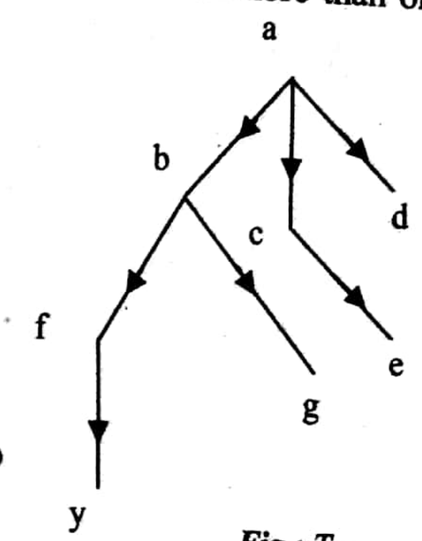


Fig.: Tree

Parent : b is parent of f and g.

Child : g and f are children of b

Siblings: f and g are siblings.

Ancestors : Ancestors of g are b, a.

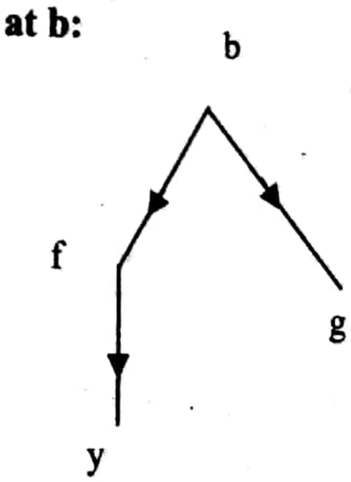
Descendents : Descendents of b : f, g, y

Leaf : Vertices with no children (y, g, e, d)

Internal vertices : a, b, c, f

Sub tree : Subgraphs consisting of V and its descentants and their incident edges

Subtree rooted at b:



- Level (of v) is length of unique path from root to v (level of root = 0, level of b = 1, level of g = 2)
- Height is the maximum of vertices level (height : 3)

m-ary trees

- A rooted tree is called m-ary if every internal vertex has no more than m children.
- It is full m-ary if every internal vertex has exactly m children.
- A 2-ary tree is called a binary tree.

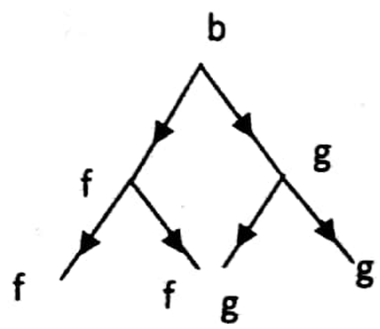


Fig.: Full binary tree

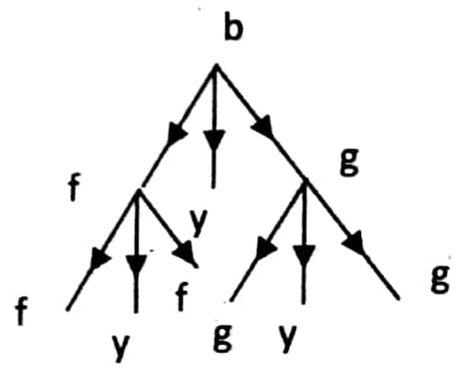


Fig.: Full 3-ary tree

Ordered Rooted Tree

- An rooted tree where the children of each interval node are ordered.
- In ordered binary tree, we can define :
  - o left child, right child
  - o left subtree, right subtree
- For m-ary trees with m>2, we can use terms like "left most", "right most" etc.

Properties of trees:

Theorems 1 :

A tree with n vertices has n-1 edges.

Proof :

Basis step: When n = 1, a tree with n = 1 vertex has no edge. It follows that theorem is true for n = 1.

Inductive hypothesis: Assume that the tree with K vertices has K-1 edges, where k is positive integer.

Inductive step: Suppose that a tree T has K + 1 vertices and that V is a leaf of T. Removing vertex V and the associated edge from T produces a tree T1 with K vertices, since the resulting graph is still connected and has no simple circuit. By the induction hypothesis, T1 has k-1 edges. Hence T has K edges since it has one more edge than the edge connecting V to its parent.

Theorem 2 :

A full m-ary tree with i-internal vertices contains n = mi + 1 vertices.

A level(depth) of a vertex v in a rooted tree is the length of the unique path form the root to vertex. The level of the root is zero. The height of the rooted tree is the length of the longest path from the root to any vertex. A rooted m-ary tree of height h is balanced if all leaves are at levels h or h-1.

For example:

Vertices = 13

Internal vertices (i) = 6

Leaves L = 7

$13 = 6 \times 2 + 1$

A full m-ary tree with

i. n vertices has I = (n-1)/m internal vertices and L = ((m-1) * n + 1)/m leaves

ii. I internal vertices has n = m*I + 1 vertices and L = (m-1)*I + 1 leaves

iii. L leaves has n =(m*1-1)/(m-1) vertices and I = (L-1)/(m-1) internal vertices.

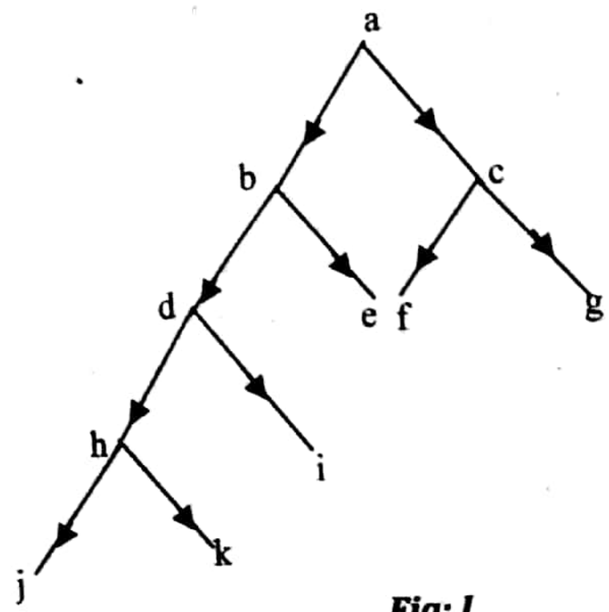


Fig: I

For example:

m = 2, n = 13, I = 6, L = 7 from fig (I)

i. I = (n - 1)/m = (13 - 1)/2 = 6
and L = ((m - 1)*n + 1)/m = ((2 - 1)*13 + 1)/2 = 7

ii. n = m*I + 1 = 2*6 + 1 = 13
and L = (m - 1) *1 = (2 - 1)*6 + 1 = 7

iii. n = (m*L - 1)/(m - 1) = (2*7 - 1)/(2 - 1) = 13
and I = (L - 1)/(m - 1) = (7 - 1)/(2 - 1) = 6

The level of a vertex in a rooted tree is the length of the path from the root to the vertex (level of the root is O)

The height of the rooted tree is the maximum of the levels of vertices (length of the longest path from the root to any vertex).

Balanced Tree:

A rooted m-ary tree of height h is balanced if all leaves are at levels h or h-1

For example

Balanced Balanced Not Balanced

Applications of trees:

- How should items in a list be stored so that an item can be easily located? For this, we use the concept of binary search trees.
- What series of decisions should be made to find an object with a certain property in a collection of objects of a certain type? For this, we use the concepts of decision trees.
- How should a set of characters be efficiency coded by,bit. Strings? Here, we use the concept of the prefix codes.

4.12 Binary Search Tree

Searching for items in a list is one of the most important task that a rises in computer science. Our primary goal is to implement a searching algorithm that finds items efficiently when the items are totally ordered. This can be accomplished through the used binary search tree.

A binary search tree (BST) is a binary tree in which each child of a vertex is designated as a right or left child, no vertex has more than one right child or left child and each vertex is labeled with key, which is one of the items. Further more vertices are assigned keys so that the key of a vertex is both larger than the keys of all vertices in its left sub tree and smaller than keys of all vertices in its right subtree.

We use recursive procedure to form a binary search tree for a list of items. We start with a tree containing just one vertex, namely, the root. The first item in the list is assigned as the key of the root. To add a new item, first compare it with the keys of vertices already in the tree, starting at the root and moving to the left if the item is less than the key of respective vertex if this vertex has a left child or moving to the right if the item is greater than the key of the respective vertex if this vertex has a right child. When the item is less than respective vertex and this vertex has no left

child, then a new vertex with this item as its key is inserted as a new left child. Similarly, when the item is greater than the respective vertex and this vertex has no right child, then a new vertex with this item as its key is inserted as a new right child.

For example:

Form a BST for the words mathematics, physics, geography, zoology, metrology, geology, psychology and chemistry.

Decision Trees:

Rooted trees can be used to model problems in which a series of decision leads to a solution. For instance, a binary search tree can be used to locate items based on a series of comparisions, where each comparison tell us whether we have located the item or whether we should go left or right in a substance. A rooted tree in which each internal vertex corresponds to a decision with a subtree at these vertices for each possible outcome of these vertices for each possible outcome of the decision is called a decision tree. The possible solution of the problem correspond to the paths to the leaves of the rooted tree.

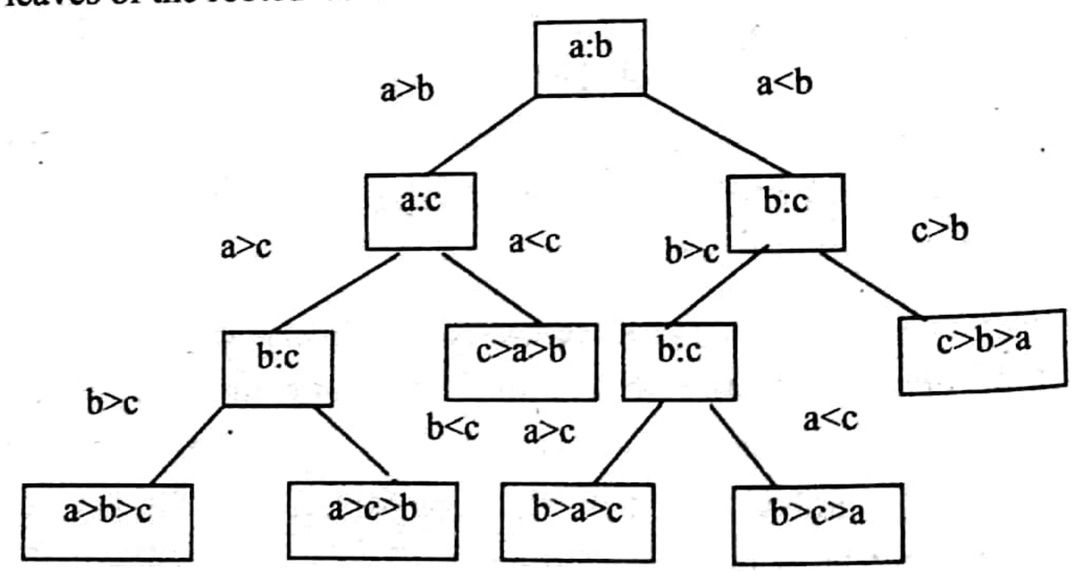


Fig.: *A decision tree for sorting three distinct elements*

4.13 Prefix Codes

Consider a problem using bit string to encode the letters of the English alphabet. We can represent each letter with a bit string of length five as there are only 26 letters and there are 32 bit strings of length five.

Consider using bit strings of different length to encode letters. Letters that occur more frequently should be encoded using short bit strings. When letters are encoded using varying number of bits, some method must be used to determine where the bits of a each character start and end.

For instance, if we were encoded e with 0, a with 1 and t with 01, then the bit string 0110 could represent to eat or tae or bit string 0101 could correspond to eat, tea, eaea or tt.

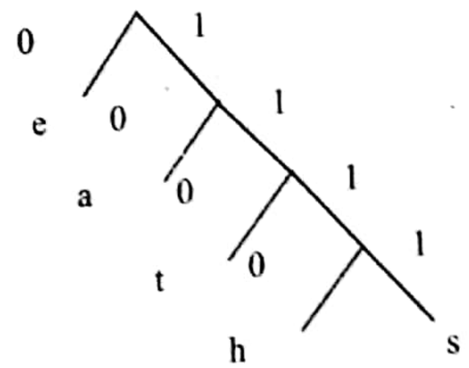
One way to ensure that no bit string corresponds to more than one sequence of letters is to encode letters so that the bit string for a letter never occurs as the first part of the bit string for another letter codes with this property are called prefix codes.

For example:

Encoding of e as o, a as 10 and t as 11 is a prefix code. A word can be recovered from unique bit string that encodes its letters.

For example: 10110 is the encoding of ate.

A prefix code can be represented using binary tree, where the characters are the labels of the leaves of tree. The edge of tree are labeled so that an edge leading to a left child is assigned a o and an edge 1. We construct prefix code from any binary tree where the left edge at each internal vertex is labeled by 0 and right edge by 1 and leaves are labeled by characters.



The tree in the figure represents encoding of e by 0, a by 10 t by 110, m by 1110 and s by 1111.

For example: 11111011100

s a n e

Huffman coding

Here is an algorithm that takes input the frequencies of symbols in a string and produces as output a prefix code that encodes the string using fewest possible bits. This algorithm is called Huftman coding. It is used in data compression. The subject devoted to reducing no. of bits required to represent information.

Procedure

F: Forest of n rooted trees.

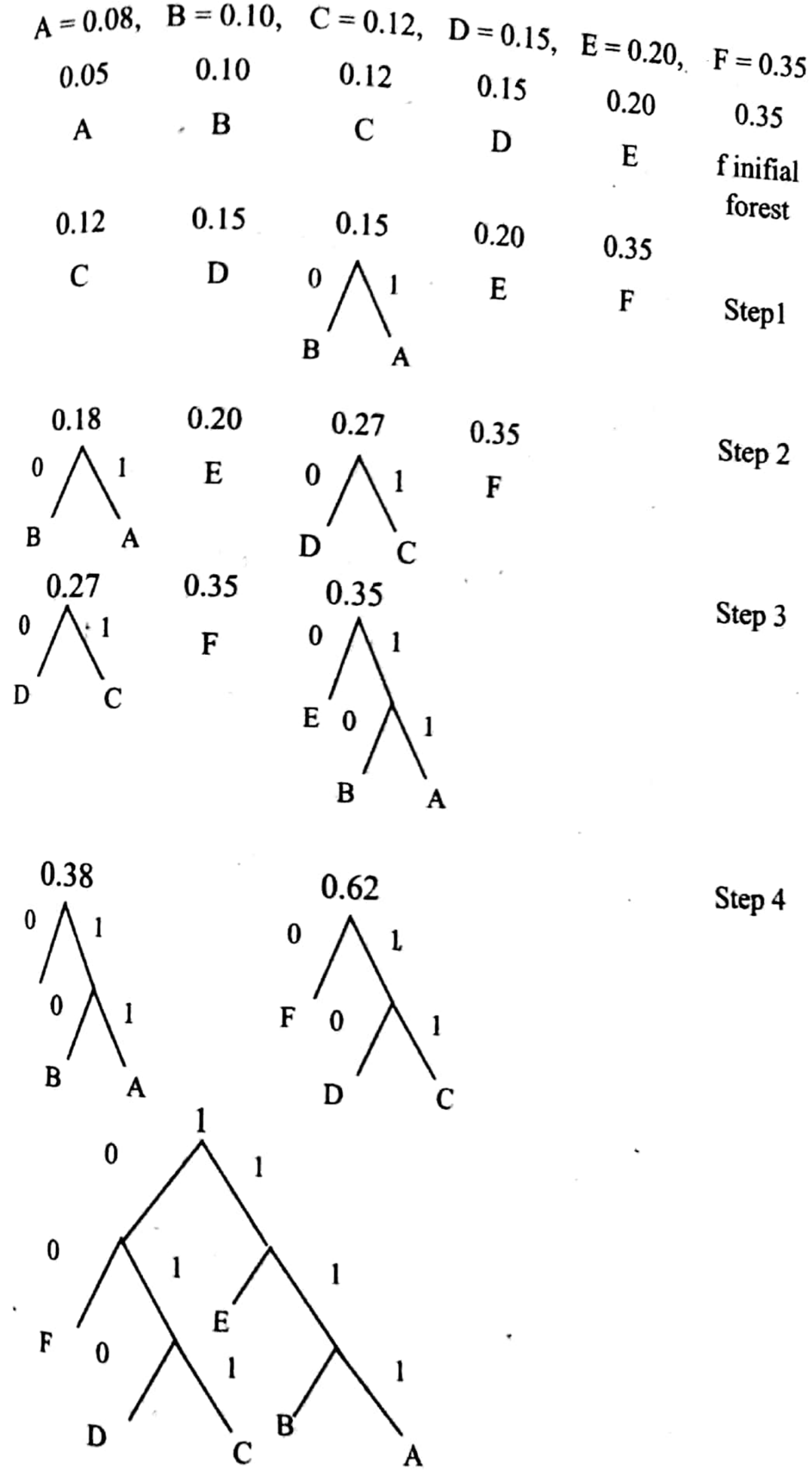
While F is no a tree.

begin:

Replace the rooted trees T and T' of least weight from F with W (T) ≥ W (T') with a tree having a new root that has T as its left subtree and T' as its right subtree. Label the new edge to T with O and new edge to T' with 1.

Again W (T) + W (T') as weight of new tree end.

Use Huffman coding to encode following symbols with the frequencies listed what is the average no. of bits used to encode a character.



From the graph, we can encode A by 111, B by 110, C by 011, D by 010, E by 10 , and F by 00.

The average number of bits used to encode a symbol using this encoding is

$P_k l_k$ (P_k : Probability, l_k : length)

$= 0.08\times3 + 0.10\times3 + 0.12\times3 + 0.15\times3 + 0.2\times2 + 0.35\times2$

$= 2.45$

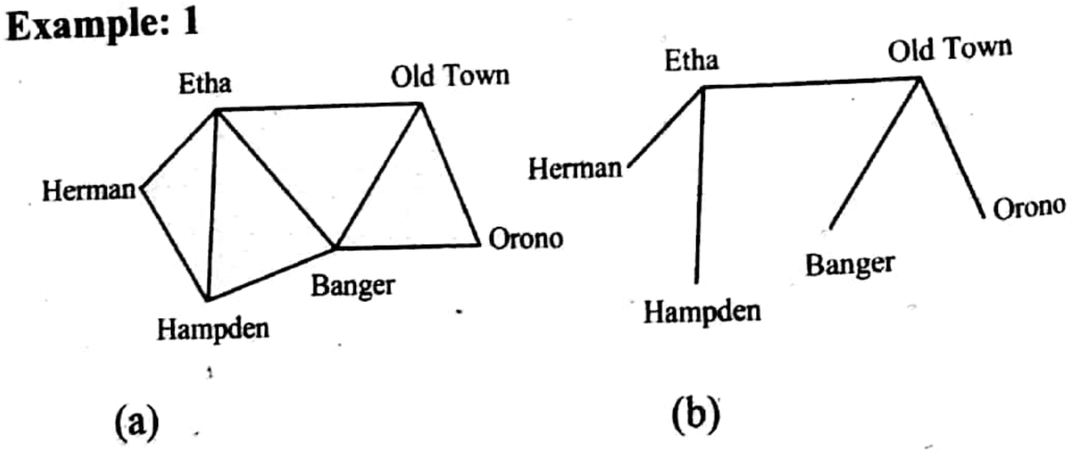
4.14 Spanning Tree

Let G be a simple graph. A spanning tree is a sub graph of G that is a tree containing every vertex of G.

Consider a system of roads in Maine. The only way the roads can be kept open in the winter is by frequently plowing them. The highway department wants to plow the fewest roads so that there will always be cleared roads connecting any two towns. At least 5 roads must be plowed to ensure that there is a path between any two towns. Fig (b) represents one such set of roads. Note that the subgraph representing these roads in a tree because it is connected and contains six vertices and five edges.

This problem was solved with a connected subgraph with the minimum number of edges containing all vertices of the original simple graph such a graph must be a tree.

Example: 1



Fig(a) : Road system and (b) sets of roads to plow

Example: 2

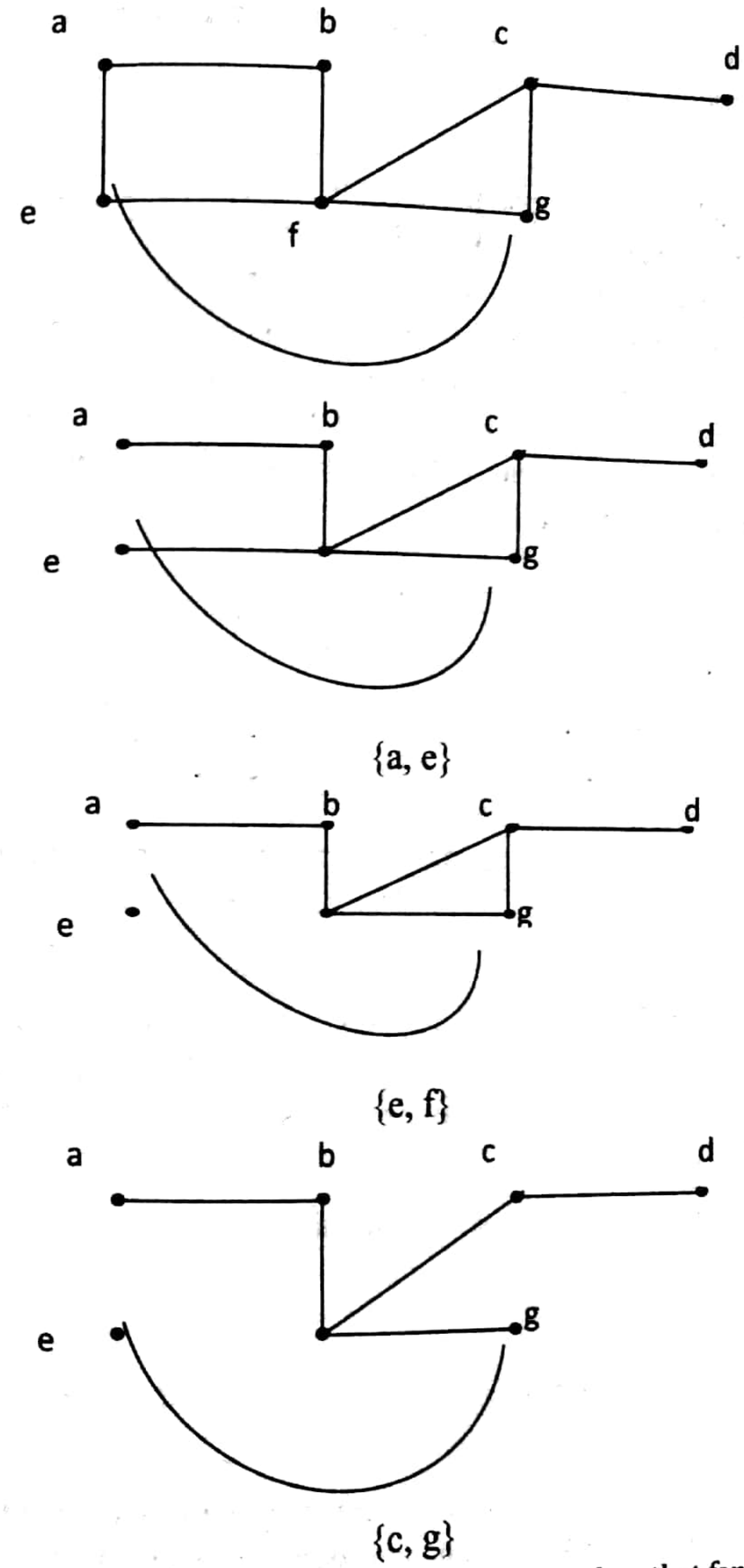


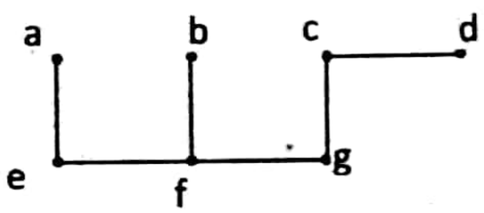
***Fig:** Producing spanning tree for G by removing edges that form simple circuits.*

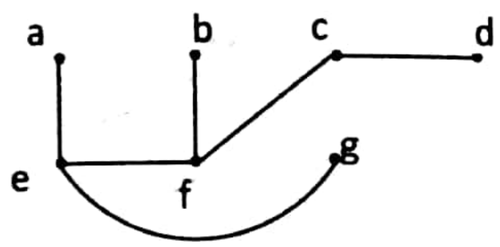
Theorem 1:

A simple graph is connected if and only if it has a spanning tree.

Proof:

Firstly let a simple graph G has a spanning Tree T. T contains every vertex of G. Therefore, there is a path T between any two of its vertices. Because T is a subgraph of G, there is a path in G between any two of its vertices. Hence, G is connected. Now suppose G is connected, If G is not a tree, it must contain a simple circuit. Remove an edge from one of those simple circuits. The resulting graph has one fewer edge but still contains all the vertices of G is connected.





spanning trees are important in data networking such as IP multicasting.

The proof by theorem 1 gives an algorithm to finding spanning trees by removing edges from simple circuits. This algorithm is insufficient because it requires that simple circuits be identified.

We can build spanning tree for a connected simple graph using depth first search.

For this,

Firstly, We will arbitrarily choose a vertex of the graph as a root.

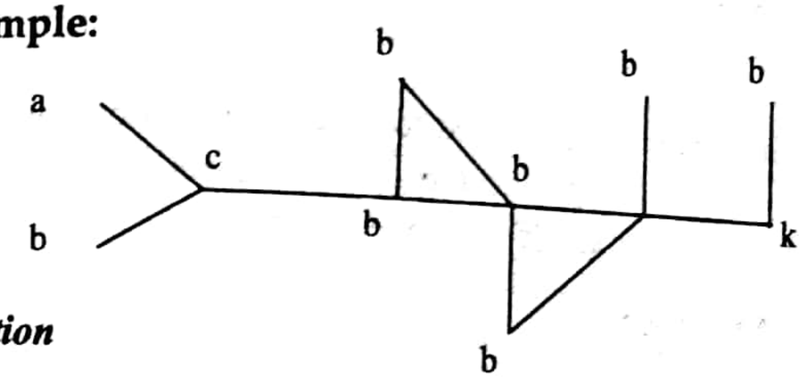
Form a path starting at this vertex by successively adding vertices and edges, where each new edges is incident with last vertex in path and a vertex not already in the path.

If path goes through all vertices of graph, the tree consisting of this path is spanning tree.

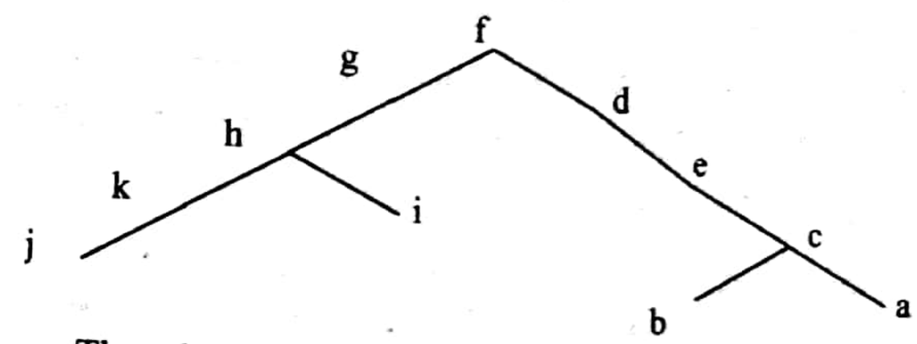
If path does not go through all vertices, more vertices and edges must be added.

Move back to the next to last vertex in the path and if possible, form a new path starting all this vertex passing through vertices that were not already visited. If this can not be done, move back another vertex in the path, that is two vertices back in the path and try again.

Example:

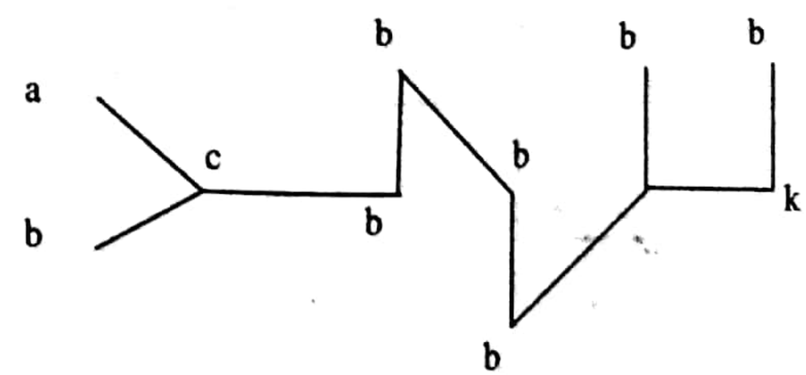


Solution



The edges selected by depth first search of a graph are called free edges.

All other edges of the graph must connect a vertex to an ancestor or descendant of this vertex in the tree. These edges are called back edges.



Depth first search is also called back tracking Algorithm

Procedure DFS (G: connected graph with vertices V_1, V_2, ...V_n)

T: tree consisting only of the vertex V_1 visit (V_1)

Procedure visit (Vertex of G)

for each vertex w adjacent to and not yet in T.

begin,

add vertex w and edge {v, w} to T visit (w)

end.

Depth first search can be used to find paths and circuits in a graph, can be used to determine the connected components of a graph and to find cut vertices of a connected graph.

Breadth-First Search:

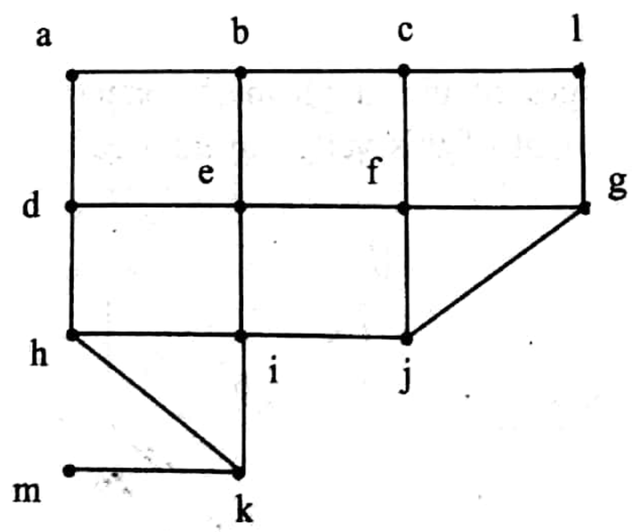
Choose a root to the vertex from the vertices of the graph.

Then, add all edges incident to this vertex. The new vertices added at this stage become the vertices at level 1 in a spanning tree.

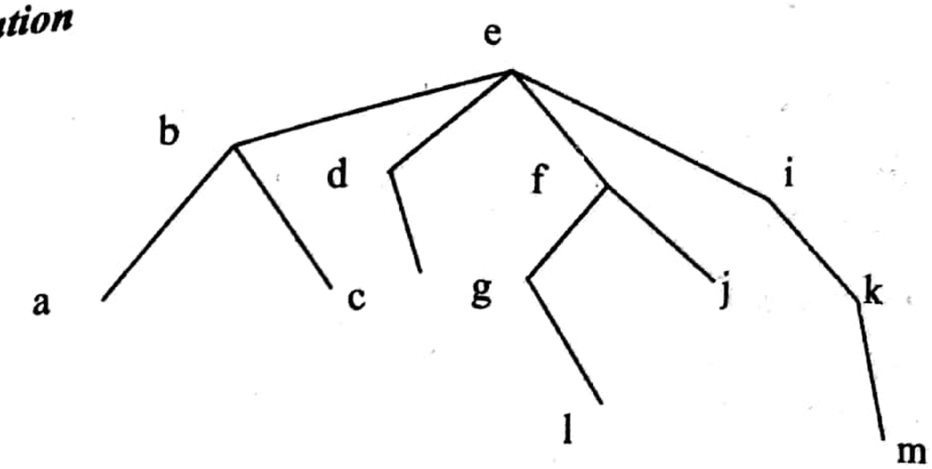
Arbitrarily order them.

For each vertex at level 1, visited in order, add each edge incident to this vertex to the tree as long as it doesnot produce a simple circuits.

Example:

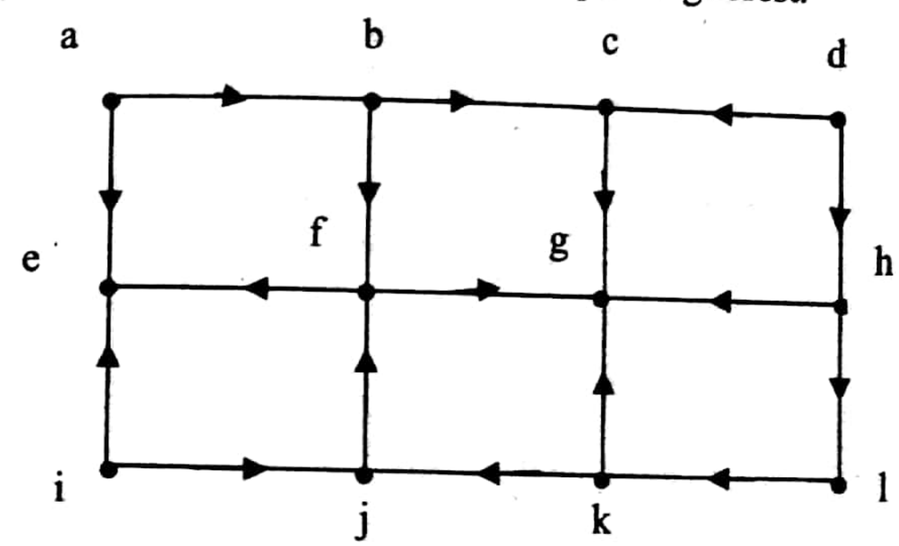


Solution

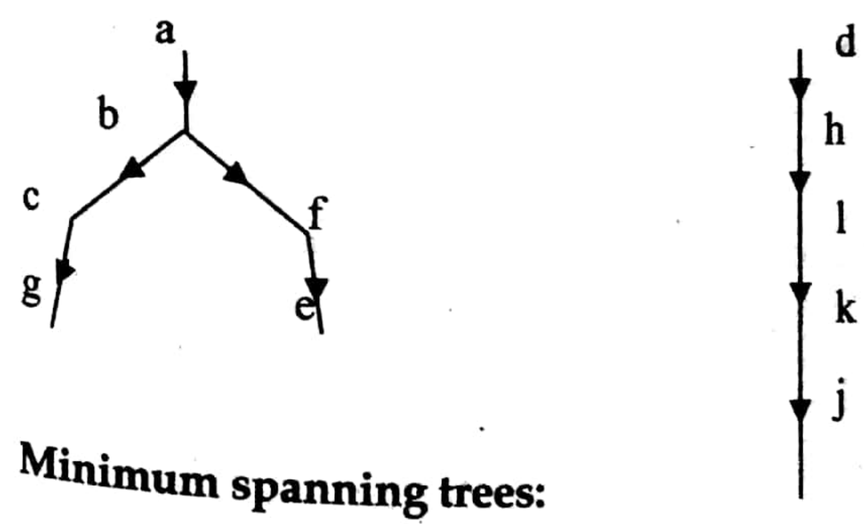


Depth first search in Directed Graphs

In directed graphs, there can be spanning forest.



Solution:



Minimum spanning trees:

A minimum spanning tree in a connected weighted graph is a spanning tree that has the smallest possible sum of weights of its edges.

We will use 2 algorithms for constructing minimum spanning trees.

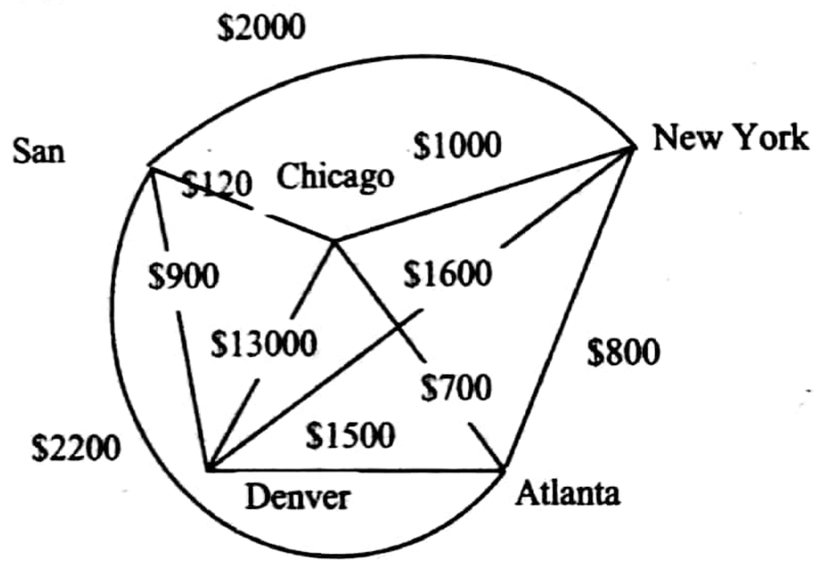
Prim's Algorithm

The algorithm was given by Robert Prim in 1957.

To carry Prism's algorithm

Begin by choosing any edge with smallest weight, putting it into the spanning tree.

Successively add to the tree edges of minimum weight that are incident to a vertex already in the tree and not forming a simple circuit with those edges already in the tree. Stop when n-1 edges have been added.



{Chicago, Atlanta}= $700
{Atlanta, New York}= $800
{Chicago, San Fransico}= $1200
{San Fransico, Denver} = $900
Total $3600

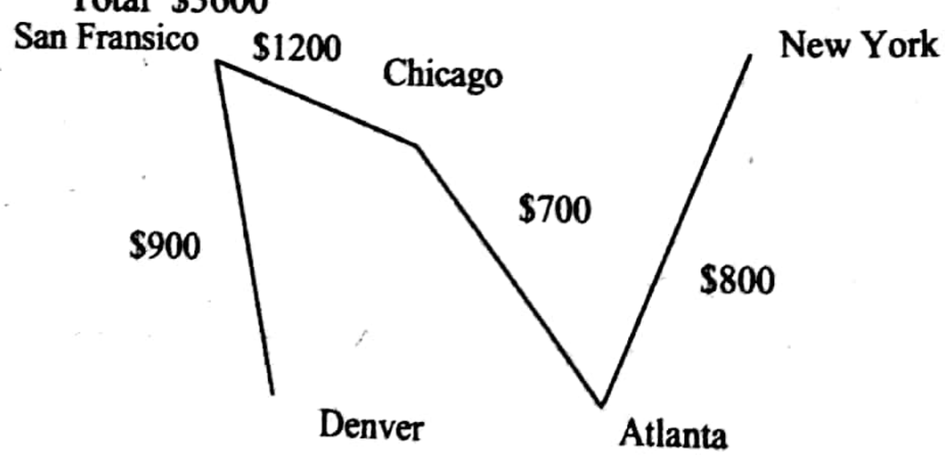


Fig.: Minimum Spanning tree

Example

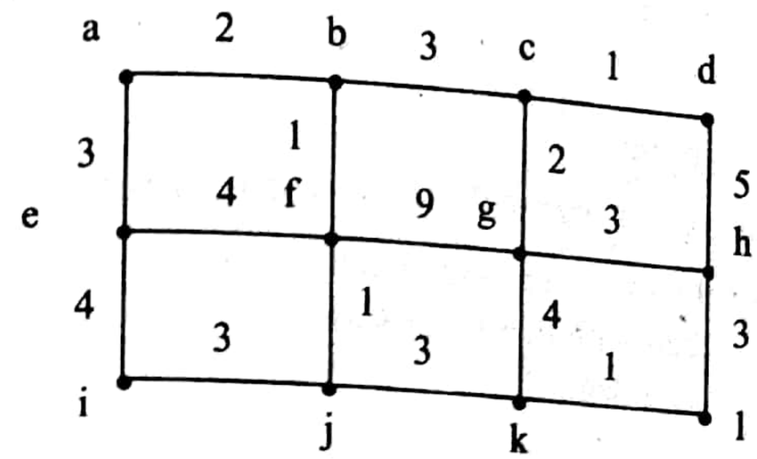


Fig: II

{b, f}= 1
{f, i} = 2
{a, b} = 2
{a, e} = 3
{i, j} = 3
{f, g} = 3
{c, g} = 2
{c, d} = 1
{g, h} = 3
{h, l} = 3
{k, l} = 1
24

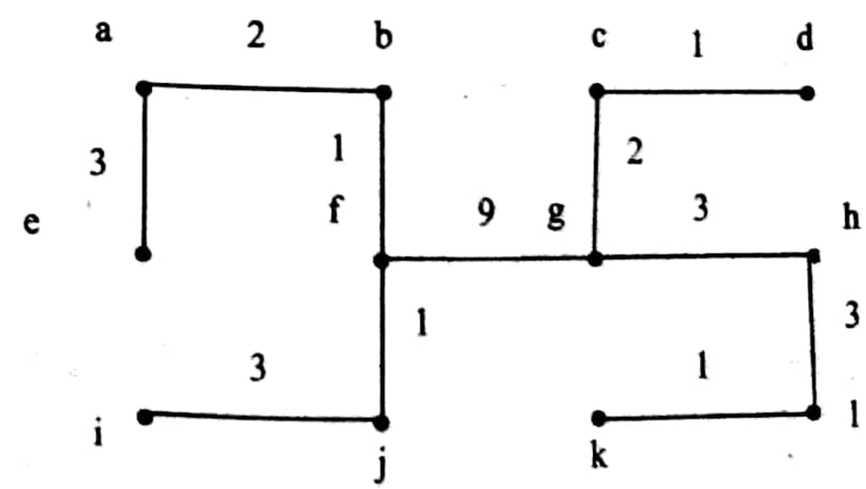


Fig.: minimum spanning tree

Krushal's Algorithm

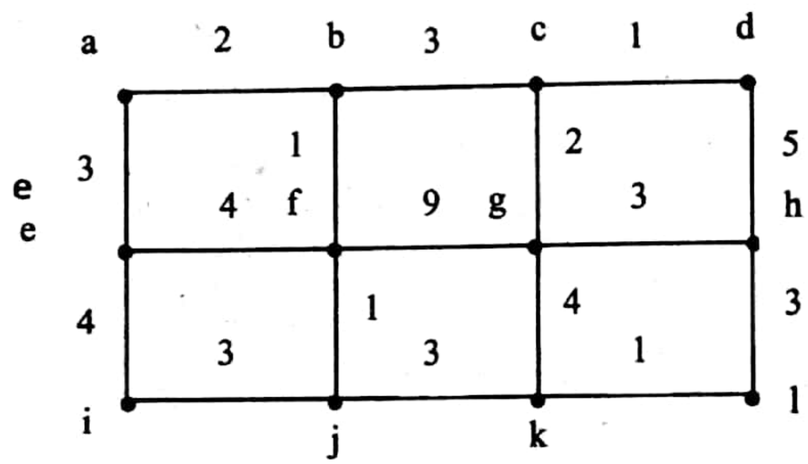
This algorithm is given by Jospeh Krushal

For this algorithm,

Firstly choose an edge in the graph with minimum weight.

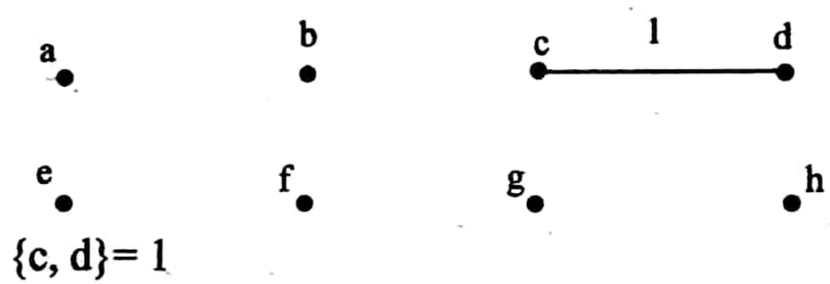
Successively add edge with minimum weight that do not form a simple circuit with those edges already choosen. Stop after n-1 edges havè been selected.

For Fig IIv

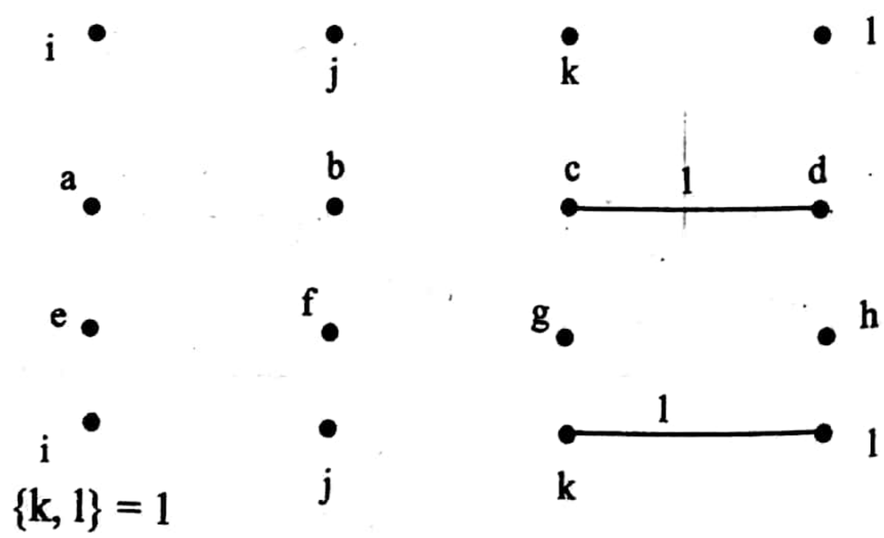


Solution

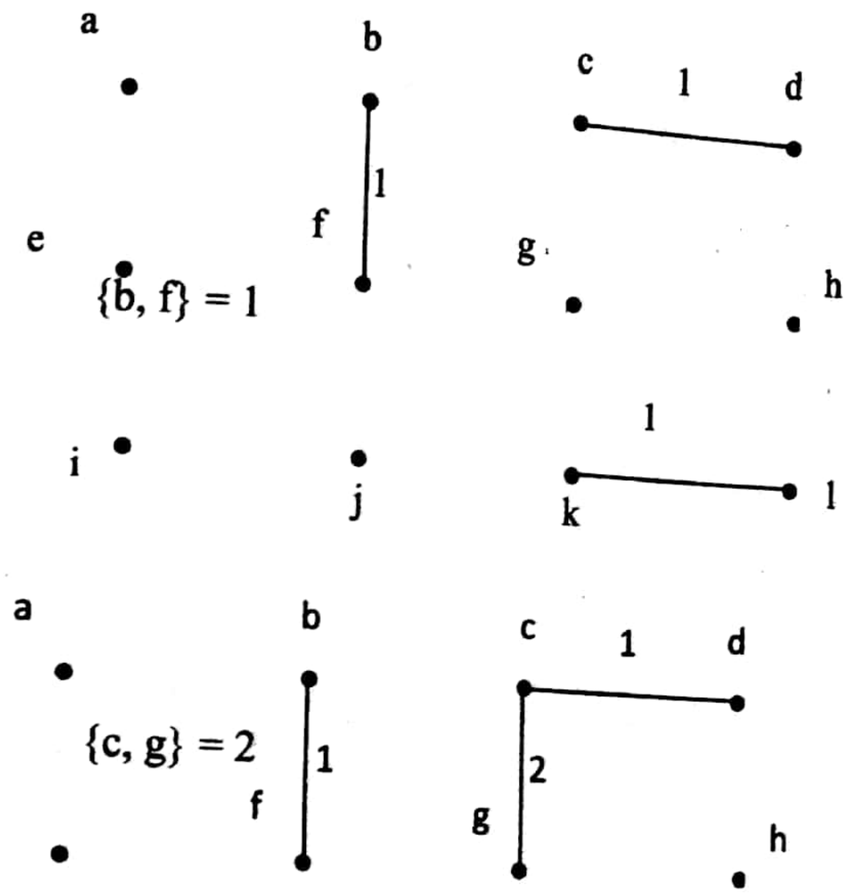
(a)



(b)

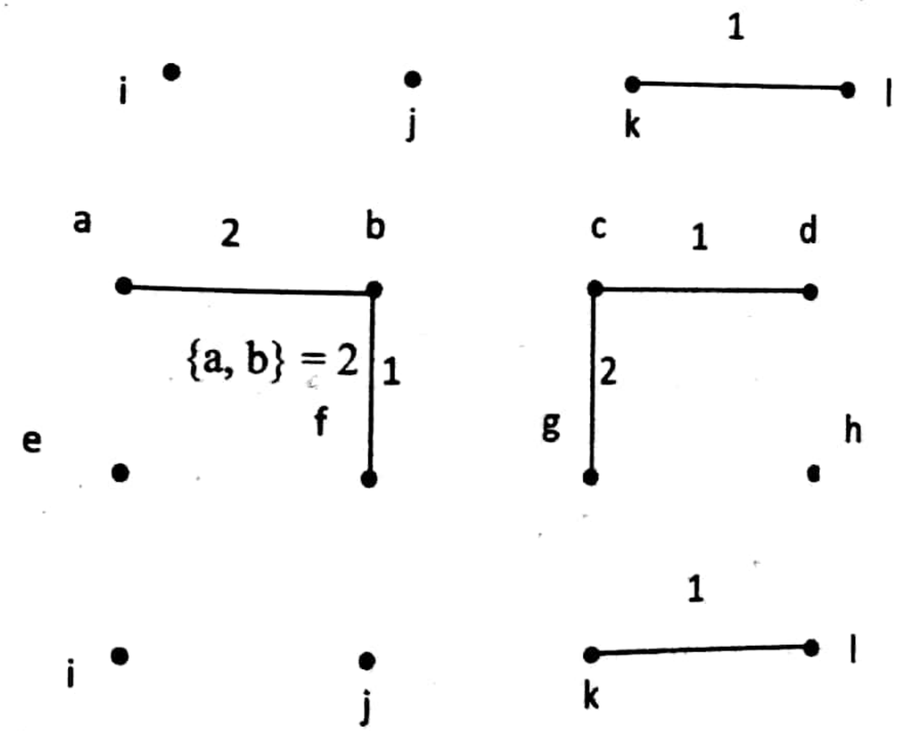


(c)

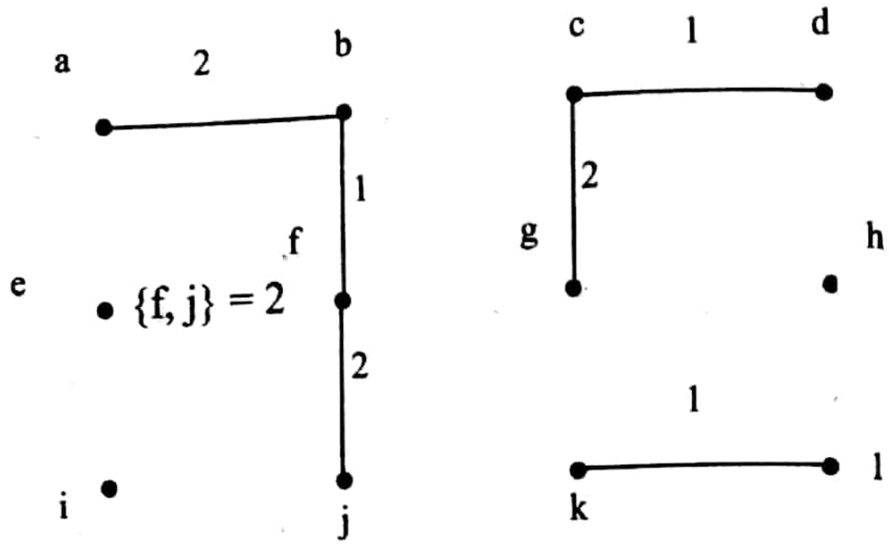


(d)

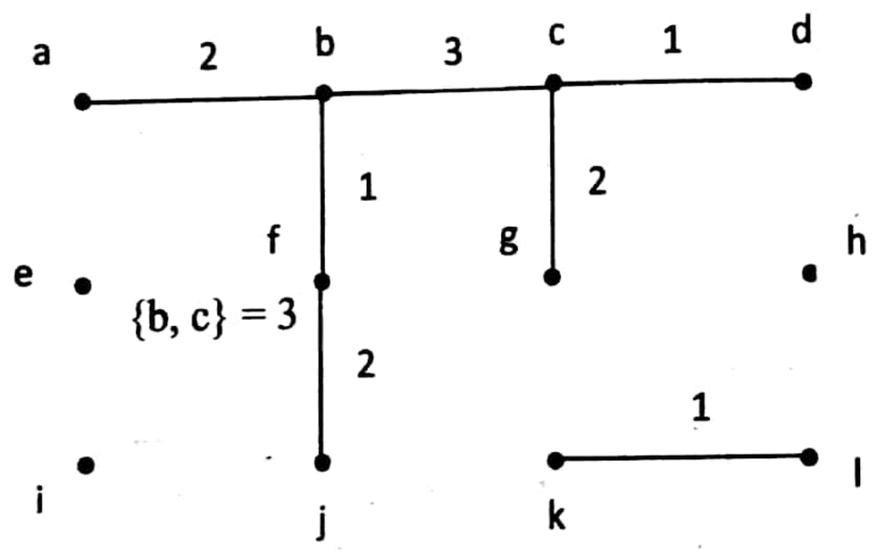
(e)



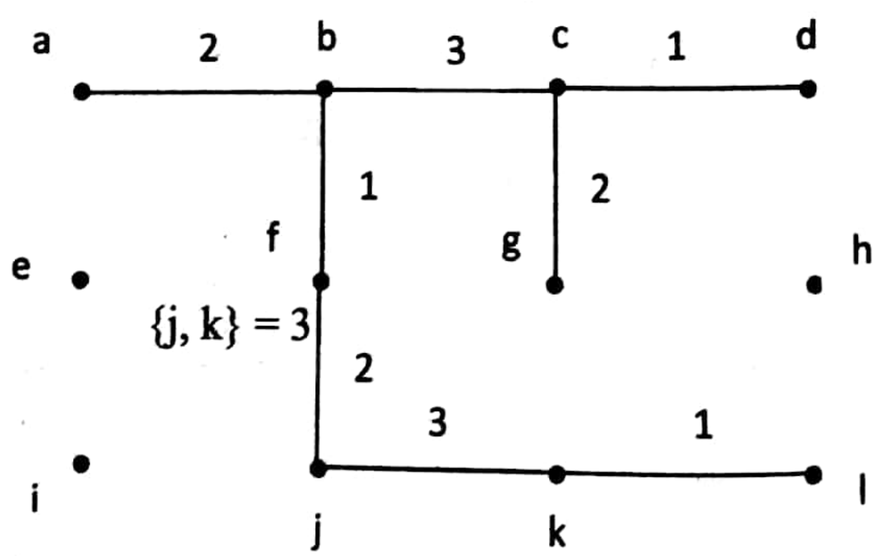
(f)



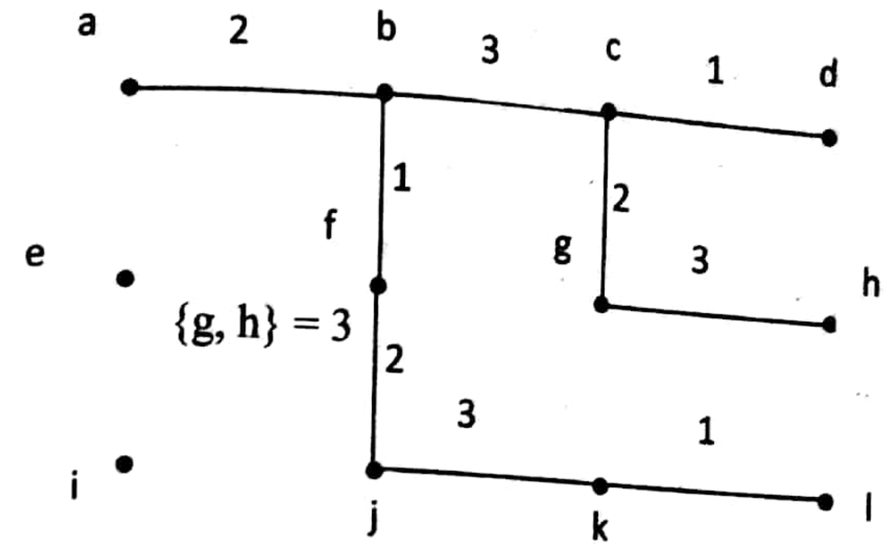
(f)



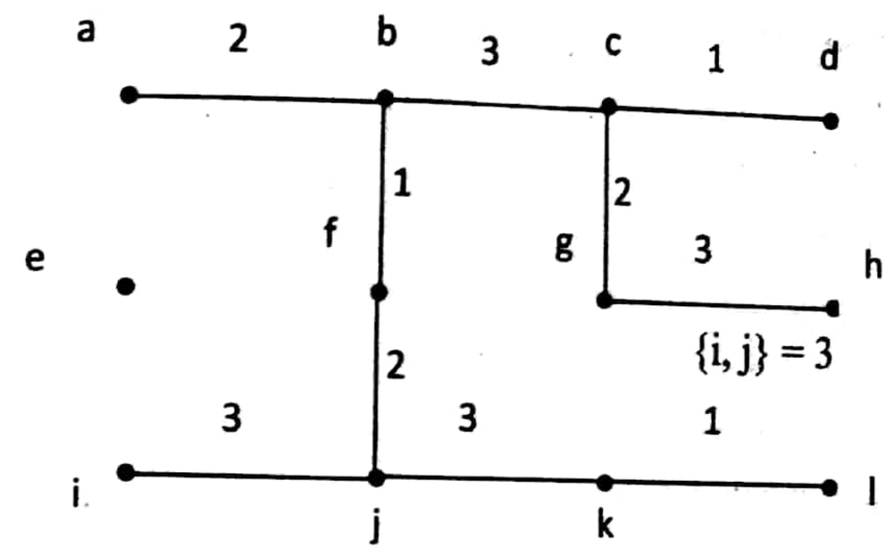
(g)



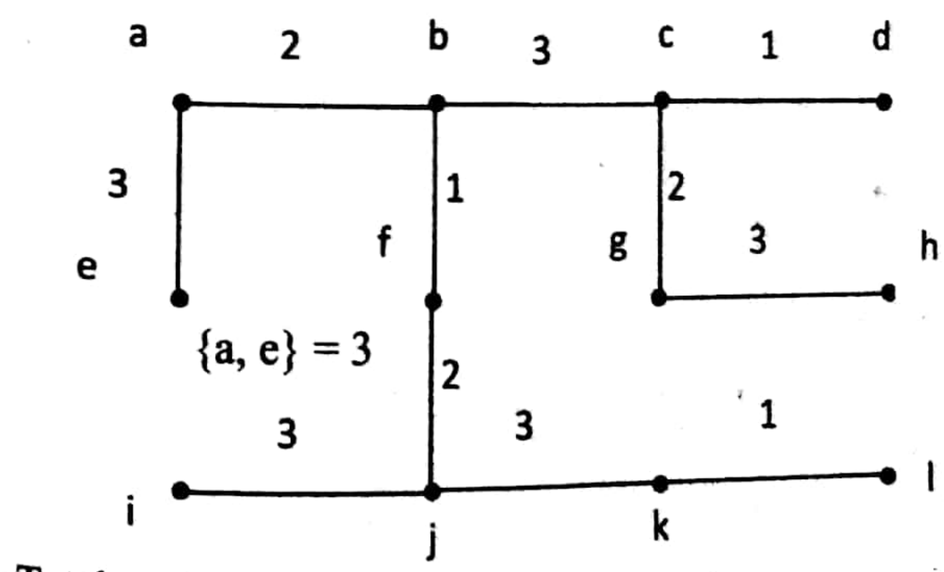
(h)



(i)



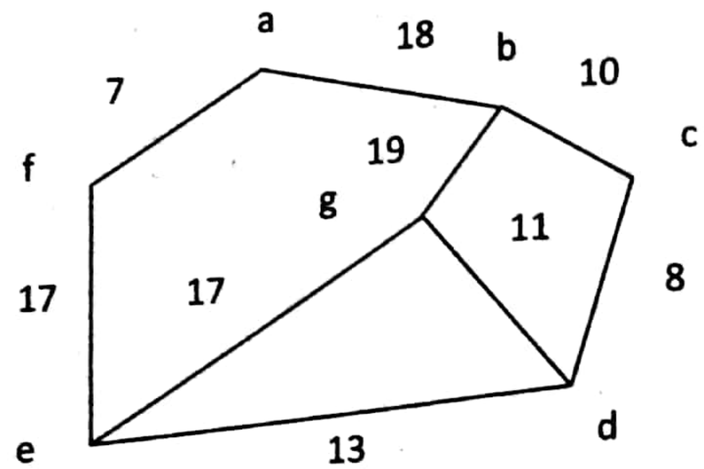
(j)



Total = 24

minimum spanning tree

Q. ***Graph given below, find the minimum spanning tree***



4.15 Graph Coloring

A coloring of a simple graph is the assignment of a color to each vertex of the graph so that no two adjacent vertices are assigned the same color.

Smallest number of colors must be used instead of just assigning every region its own color.

A chromatic number of a graph is the least no. of colour needed for coloring graph.

4 color Map Theorem:

The chromatic number of a plannar graph is no greater than four.

It can be shown that any two dimensional map can be painted using four color in such a way that adjacent regions are different colors.

Dual graph

Each map in a plane can be represented' by a graph .Each region is represented by a vertex. Edges connect to vertices if the regions represented by these vertices have a common border.Two regions that touch at only one point are not considered adjacent.

Example 1:

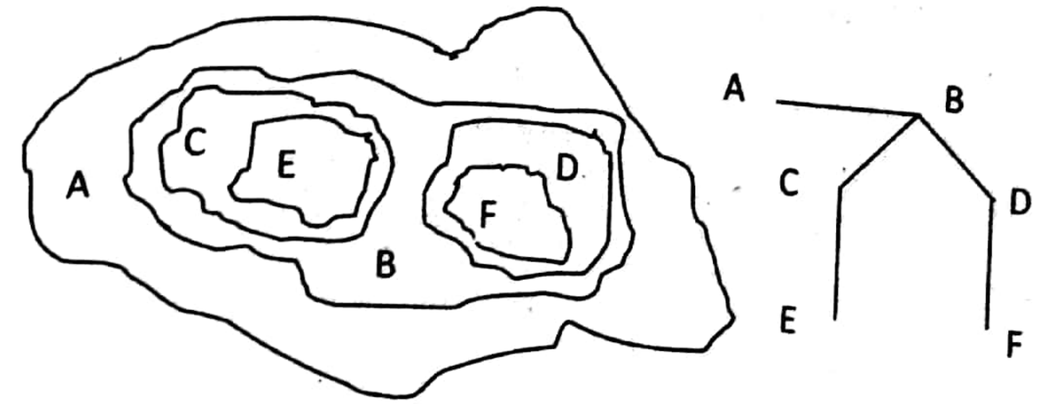
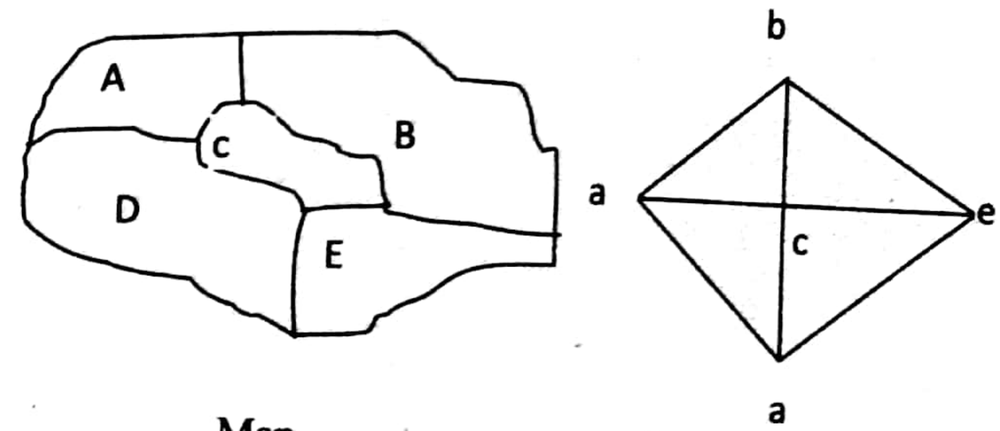


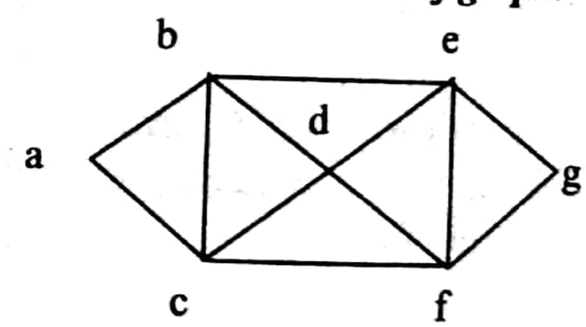
Fig.: *Dual graph*

Example 2:

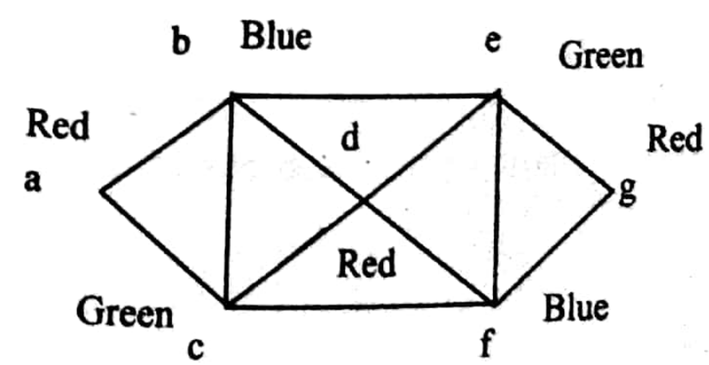


Map Dual graph

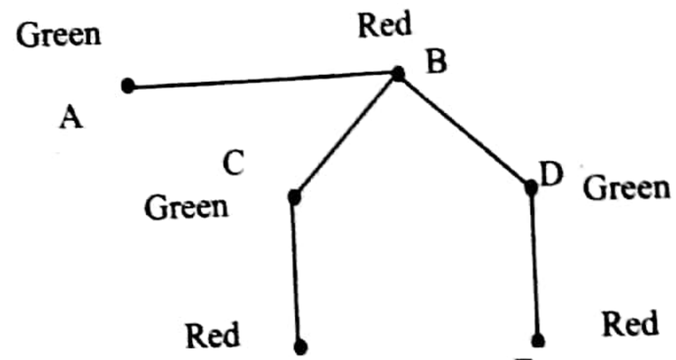
Q1. ***What is the chromatic no. of graph.***



Chromatic no. must be at least 3 since a, b, c must be assigned different colors

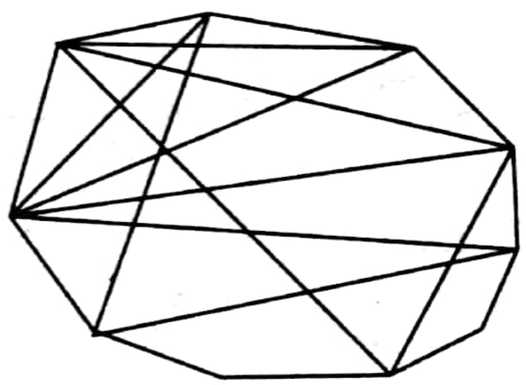


QN.2 What is the chromatic no. of the fig(A)



We can color the g with at mc colors as shown in the graph red and Green.

Q.N.3. Find the chromatic no. of a graph.



4.16 Max Flow and Min Cut

- In network or transport network the flow is the amount of commodity (number of car in a road, gallon of oil in pipe, bits of formation in channel etc.) transported from one place to another for an instant of time.
- The maximum amount of-low is called the capacity of the line arc or edge.
- The flow f(e)=f(v, w) and the capacity c(e)=c(v, w) are non-negative real numbers.
- $f(e) \leq c(e)$.
- The unique starting vertex that has in degree = 0 is called **source** s $\in V$
- The unique ending vertex rat has out degree = 0 is called **sink**, $t \in V$

- In a network there is one source and one sink. Any other vertex : ca.ied intermediate vertex.
- **Flow conservation :** For any intermediate vertex, total flow into x equals to the flow out of x .

  $\sum_{w \in V} f(w,x) = \sum_{v \in V} f(x,v)$
- Sum of flow from the source = Sum of flow to the sink.
- **Saturated flow:** The flow along an edge e(v, w) is said to be saturated if f(e)=c(e).
- If f(e)<c(e), then the flow is unsaturated, then residual capacity or slack is s(e)=c(e)-f(e).

Flow augmenting path

- The edge e_i is called forward edge if: is directed to x_j to x_{i+1} and transportation flow from x_i to $x_{,+1}$, it is backward edge if transportation flow from x_{i+1} to x_i .
- If a flow f is given by path p:s = $x_1, x_2 x_{k-1}, x_k$ = t is called a flow augmenting path (i)every forward edge of path has excess capacity that is f(e)<c(e). (ii) every backward edge has f(e)>0.

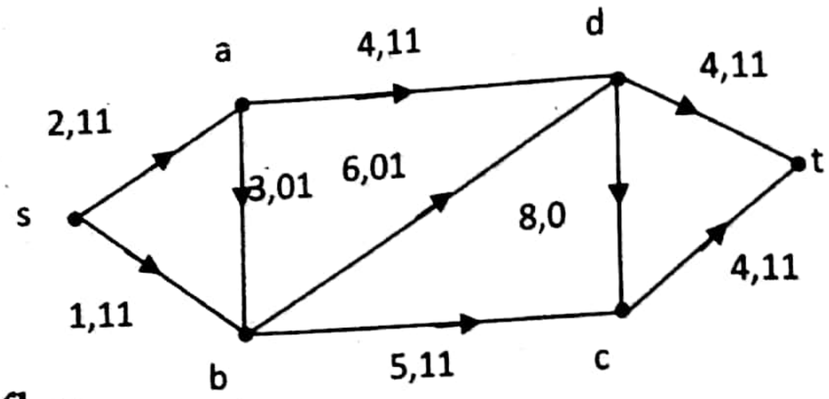
Max flow

- The flow that achieves the largest possible value is called the maximum flow or maximal flow in a network.
- To calculate max flow calculate the unused capacity c(e)-f(e) for each forward arc and f(e) in each backward arc.
- In a f- augmenting path from source to sink, for the forward edge increase the flow with the minimum slack(unused capacity) followed in the path.
- For backward edge in the f- augmenting path from source to sink, decrease the flow with the minimum slack.

  $\Delta = \min(c(e) - f(e))$

  $f^* = f + \Delta$ (for forward edge)

  $f^* = f - \Delta$ (for backward edge)



Max flow

Here the unused capacity = slack in f- augmenting path s, a, d, t are

S(s, a) = 2-l = 1

S(a, d) = 4-l = 1

S(d, t) = 4-l = 3

The minimum value is 1 in above slack value. So, now add the value 1 in the flow of s, a, d, t.

f(s, a) = 1 + 1 = 2

f(a, d) = 1 + 1 = 2

f(d, t) = 1 + 1 = 2

Now s(s, a) = 2-2 = 0. So, there is not necessity for further calculation of slack f-augmenting path containing s(s, a) edge.(remember for f-augmenting path (c(e)-f(e))>0).

Again, in s(s, b) = 1-1 = 0, So, there is no necessity for further calculation of slack in the augmented path containing edge (s, b)

The max flow = flow from source s = 2 + 1 = 3

Min cut

- In min cut we calculate the capacity by separating the vertices into two subsets p and p'
- p contains s only or s and other vertices which are not in p'.
- p' contains t only or t and other vertices which are not in p.
- If there are 4 vertices other than source and sink, then there are $2^4 = 16$ s - t cuts.

- The capacity of a cut denoted by c(p, p') is defined to be the s-m of the capacities of those edges directed from the vertices in set p to the vertices in p'.
- If the edge is backward edge put the value of the backward edge = 0.
- A cut is called minimum cut if its capacity not exceed the capacity of any other cut of the network.

In above network,

If p = {s} and p' = {a, b, c, d, t} then c(p, p') = c(s, a) + c(s, b) = 2 + 1 = 3.

If p = {s, a} and p'={b, c, d, t} then c(p, p') = c(s, b) + c(a, b) + c(a, d) = 1 + 3 + 4 = 8.

Similarly,

Table for possible s-t cut

p	p'	Capacity c(p, p')
{s}	{a, b, c,,d, t}	3
{s, a}	{b, c, d, t}	8
{s, b}	{a, c, d, t}	13
{s, c}	{a, b, d, t}	7
{s, d}	{a, b, c, t}	15
{s, a, b}	{c, d, t}	15
{s, a, c}	{b, d, t}	20
{s, a, d}	{b/C, t}	16
{s, b, c}	{a, d, t}	12
{s, b, d}	{a, c, t}	19
{s, c, d}	{a, b, t}	11
{s, a, b, c}	{d/t}	10
{s, a, b, d}	{c, t}	17
{s, a, c, d}	{b, t}	12
{s, b, c, d}	{a, t}	10
{s, a, b, c, d}	{t}	8

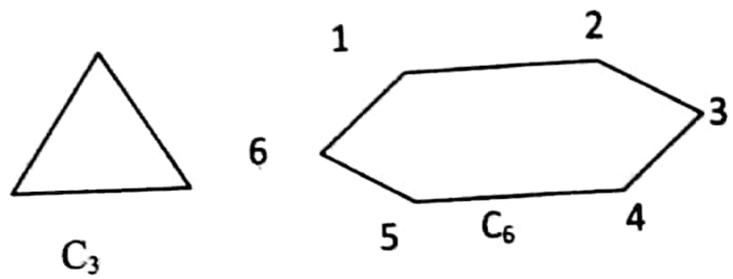
Min cut = 3

Hence, in the above example max flow = min cut = 3.

SOLVED NUMERICALS

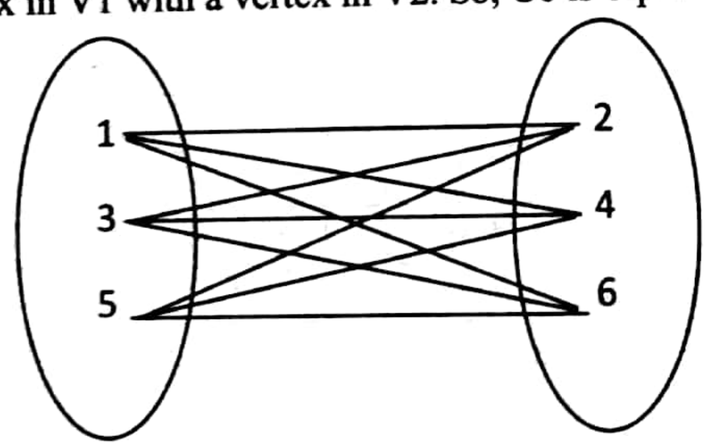
1. ***Are C_3 and C_6 bipartite, explain with figures. If G is a connected planar simple with E edges and V vertices, where $V \geq 3$, then prove that $E \leq 3V-6$.***

[2076 Baisakh, Back]



In C3 its vertex can not be partitioned into two disjoint subsets V1 and V2 such that every edge in the graph connects a vertex from the set V1 to the vertex of set V2 and no vertices of the same set are connected by an edge. So C3 is not bipartite.

But in C6, its vetex can be partitioned into two sets V1 ={1,3,5} and V2 ={2,4,6}. Every edge of C6 connects a vertex in V1 with a vertex in V2. So, C6 is bipartite.



Suppose a connected planar graph has $v \geq 3$ vertices and e edges. Then we have to prove $e \leq 3v-6$

By definition, a connected graph is planar iff it has a planar embedding.

In graph theory, a planar graph is a graph that can be embedded in the plane, i.e., it can be drawn on the plane in such a way that its edges intersect only at their endpoints. ... Such a drawing is called a plane graph or planar embedding

of the graph. So suppose a connected graph with v vertices and e edges has a planar embedding with f faces.

In a planar embedding of a connected graph, each edge is traversed once by each of two different faces, or is traversed exactly twice by one face.

So, every edge is traversed exactly twice by the face boundaries. So the sum of the lengths of the face boundaries is exactly 2e. Also, in a planar embedding of a connected graph with at least three vertices, each face is of length at least three.

when $v \geq 3$, each face boundary is of length at least three, so this sum is at least 3f. This implies that

$3f \leq 2e$

But $f = e - v + 2$ by Euler's formula, and substituting into the above gives

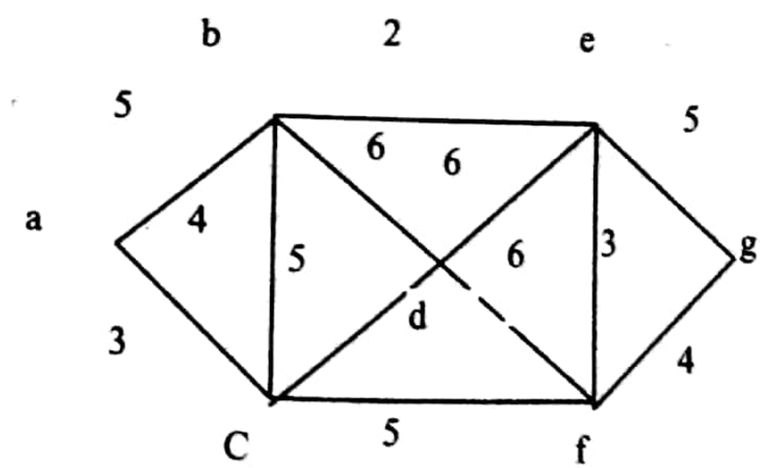
$3(e - v + 2) \leq 2e$

$= e - 3v + 6 \leq 2e$

$= e \leq 3v - 6$

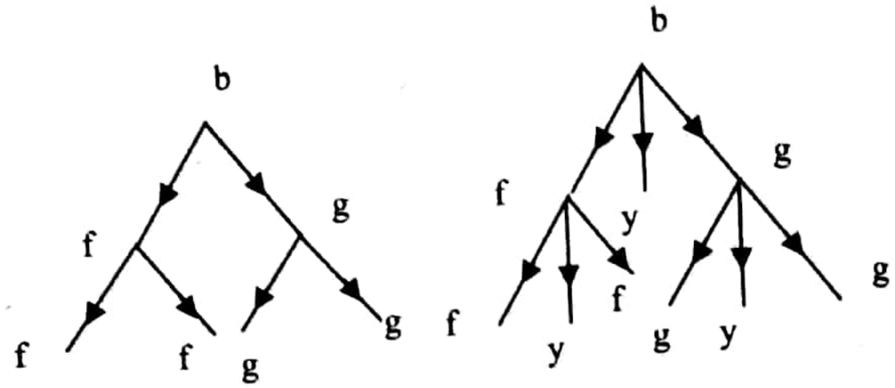
2. ***Define Binary tree, M-ary tree and Spanning tree. Find the minimal spanning tree from the graph given below.***

[2076 Baisakh, Back]



m-ary tree

A rooted tree is called m-ary if every internal vertex has no more than m children. It is full m-ary if every internal vertex has exactly m children. A 2-ary tree is called a binary tree.

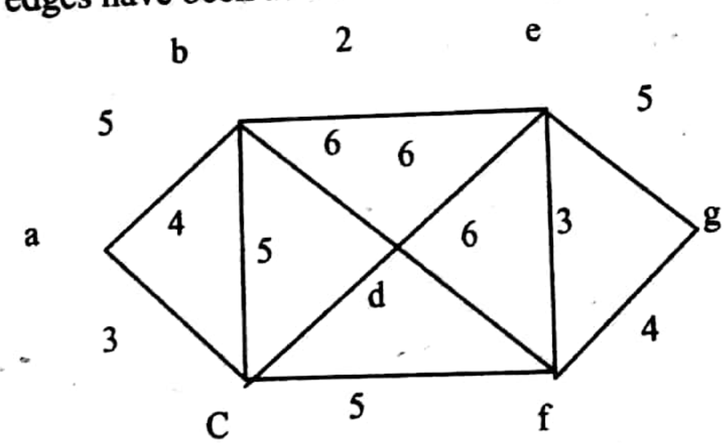


A minimum spanning tree in a connected weighted graph is a spanning tree that has the smallest possible sum of weights of its edges.

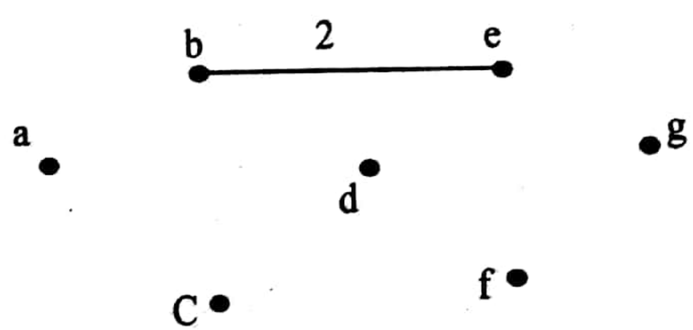
We use two algorithms, Prism's algorithm or Krushkal algorithm, for constructing minimum spanning trees.

Here we use Prism's algorithm. In this algorithm we begin by choosing any edge with smallest weight, putting it into the spanning tree.

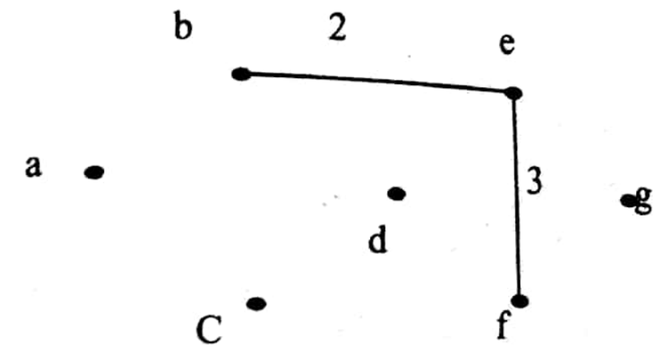
Successively add to the tree edges of minimum weight that are incident to a vertex already in the tree and not forming a simple circuit with those edges already in the tree. Stop when n-1 edges have been added.



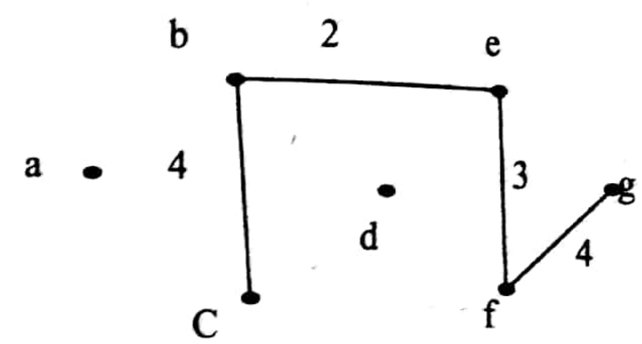
(1)



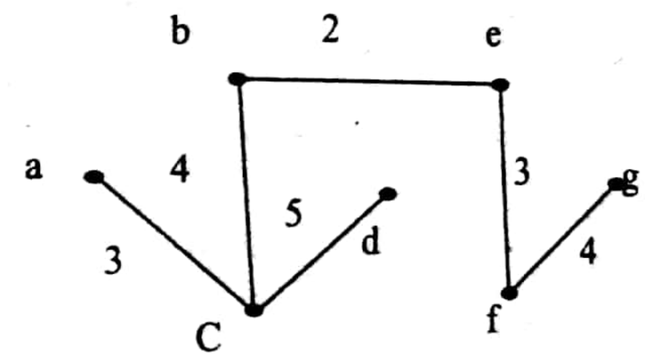
(2)



(3)

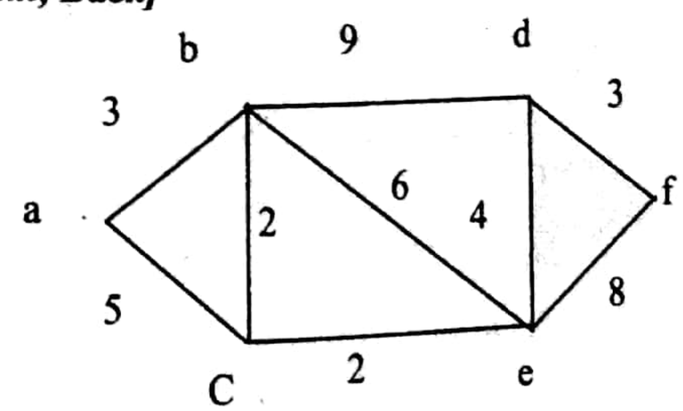


(4)

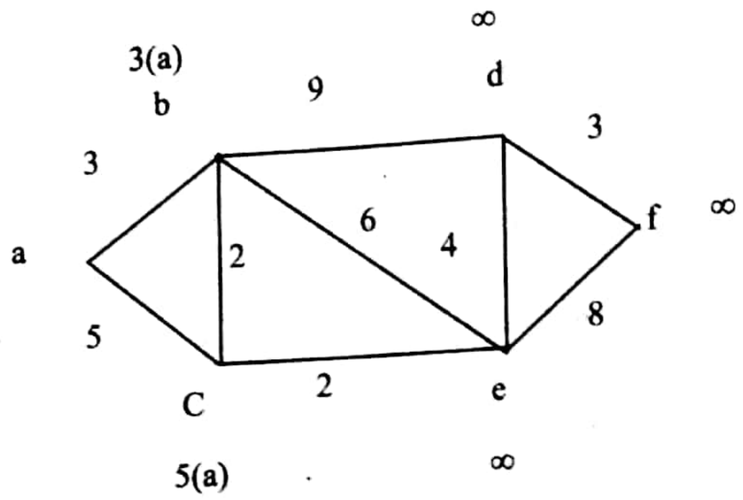


The minimal spanning tree from the given graph

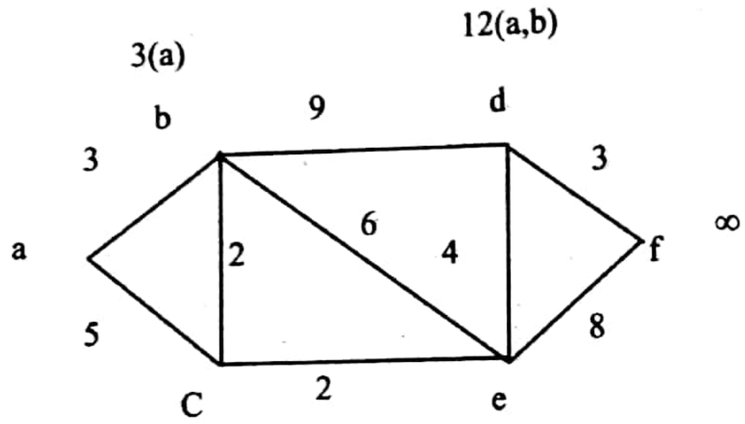
3. ***Use Dijkstra's Algorithm to find the length of shortest path from vertex a to vertex f in the following weighted graph. Also highlight the shortest path/paths in graph. [2076 Baisakh, Back]***



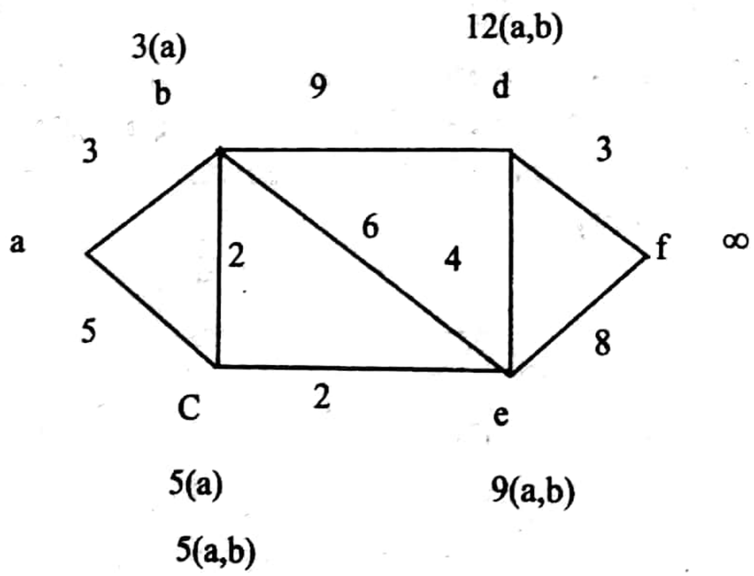
(1)



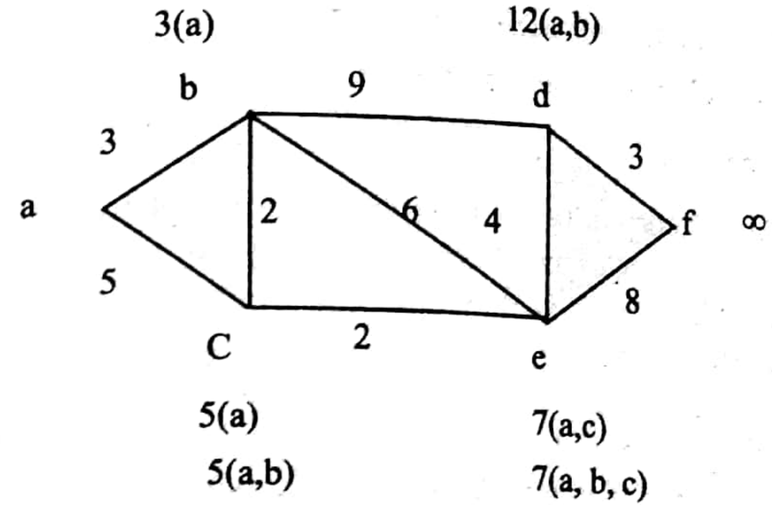
(2)



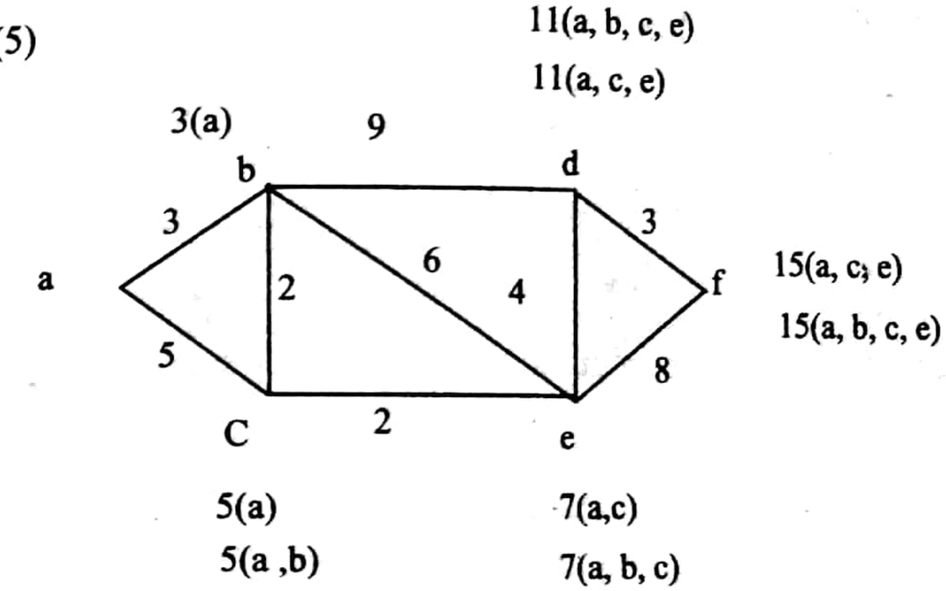
(3)



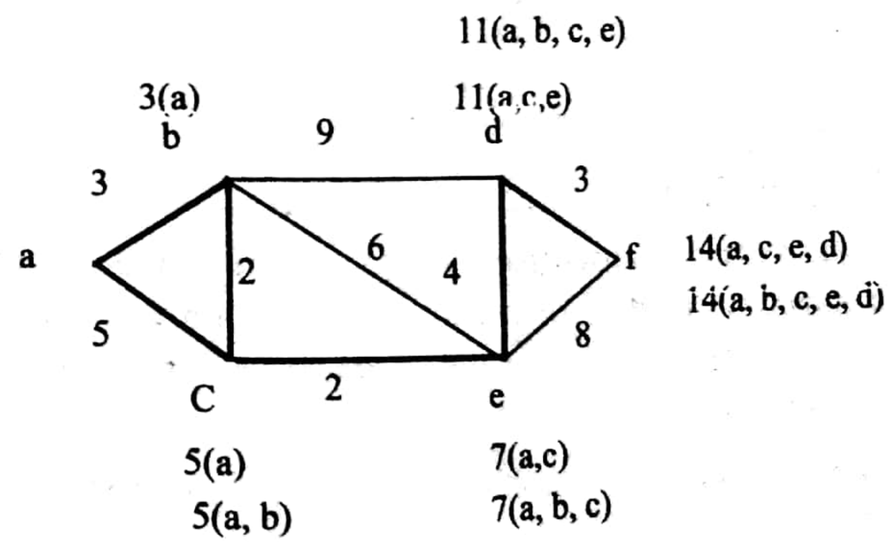
(4)



(5)



(6)



Hence, the shortest path from a to f is:

(a, c, e, d) or (a, b, c, e, d)

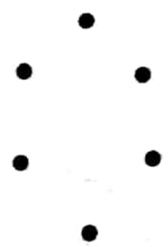
4. ***Write short notes on:*** ***[2076 Baisakh, Back]***

   a. ***Regular Graph with example***

   b. ***Max Flow and Min Cut Theorem***

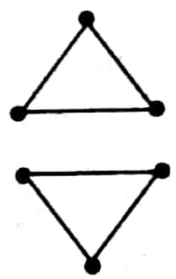
Regular Graphs:

In graph theory, a regular graph is a graph where each vertex has the same number of neighbors i.e., every vertex has the same degree. A regular graph with vertices of degree k is called a k-regular graph or regular graph of degree k. Regular graph of degree at most 2 are easy to classify. A o-regular graph consists of disconnected vertices a 1-regular graph consists of disconnected edges, and a 2-regular graph consists of disconnected cycles. A 3-regular graph is known as cubic graph.

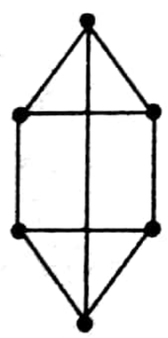


o-regular graph

1- regular graph



2-regular graph



3-regular graph

Max flow and Min cut

- In network or transport network the flow is the amount of commodity (number of car in a road, gallon of oil in pipe, bits of formation in channel etc.) transported from one place to another for an instant of time.
- The maximum amount of-low is called the **capacity** of the line arc or edge.
- The flow f(e) = f(v, w) and the capacity c(e) = c(v, w) are non-negative real numbers.

- $f(e) \leq c(e)$.
- The unique starting vertex that has in degree = 0 is called **source** $s \in V$
- The unique ending vertex rat has out degree = 0 is called **sink**, $t \in V$
- In a network there is one source and one sink. Any other vertex : ca.ied intermediate vertex.
- **Flow conservation :** For any intermediate vertex, total flow into x equals to the flow out of x .

  $\sum_{w \in V} f(w,x) = \sum_{v \in V} f(x,v)$
- Sum of flow from the source = Sum of flow to the sink.
- **Saturated flow:** The flow along an edge e(v, w) is said to be saturated if f(e) = c(e).
- If f(e) < c(e), then the flow is unsaturated, then residual capacity or slack is s(e) = c(e) - f(e).

Flow augmenting path

- The edge e_i is called forward edge if: is directed to x_j to x_{i+1} and transportation flow from x_i to $x_{,+1}$, it is backward edge if transportation flow from x_{i+1} to x_i .
- If a flow f is given by path p:s = x_1, x_2, x_{k-1}, x_k = t is called a flow augmenting path (i) every forward edge of path has excess capacity that is f(e) < c(e). (ii) every backward edge has f(e)>0.

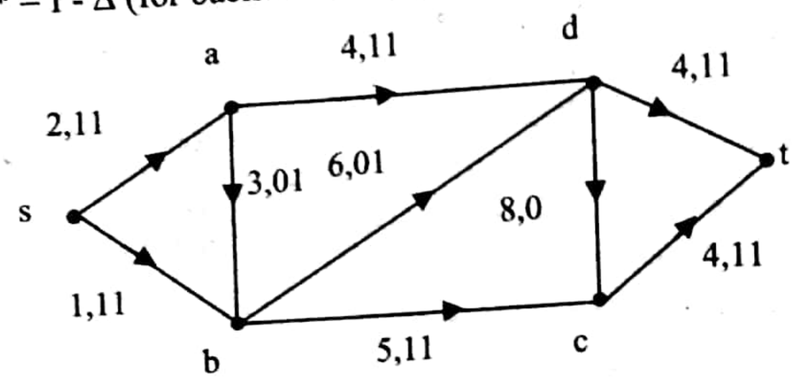
Max flow

- The flow that achieves the largest possible value is called the maximum flow or maximal flow in a network.
- To calculate max flow calculate the unused capacity c(e)-f(e) for each forward arc and f(e) in each backward arc.
- In a f- augmenting path from source to sink, for the forward edge increase the flow with the minimum slack(unused capacity) followed in the path.
- For backward edge in the f- augmenting path from source to sink, decrease the flow with the minimum slack.

$\Delta = \min(c(e)-f(e))$

$f^* = f + \Delta$ (for forward edge)

$f^* = f - \Delta$ (for backward edge)



Max flow

Here the unused capacity = slack in f- augmenting path s, a, d, t are

$S(s, a) = 2 - 1 = 1$

$S(a, d) = 4 - 1 = 1$

$S(d, t) = 4 - 1 = 3$

The minimum value is 1 in above slack value. So, now add the value 1 in the flow of s, a, d, t.

$f(s, a) = 1 + 1 = 2$

$f(a, d) = 1 + 1 = 2$

$f(d, t) = 1 + 1 = 2$

Now s(s, a) = 2-2 = 0. So, there is not necessity for further calculation of slack f-augmenting path containing s(s, a) edge.(remember for f-augmenting path (c(e)-f(e))>0).

- Again, in s(s, b) = 1-1 = 0, So, there is no necessity for further calculation of slack in the augmented path containing edge (s, b)
- The max flow = flow from source s = 2 + 1 = 3

Min cut

- In min cut we calculate the capacity by separating the vertices into two subsets p and p'

- p contains s only or s and other vertices which are not in p'.
- p' contains t only or t and other vertices which are not in p.
- If there are 4 vertices other than source and sink, then there are $2^4 = 16$ s-t cuts.
- The capacity of a cut denoted by c(p, p') is defined to be the s-m of the capacities of those edges directed from the vertices in set p to the vertices in p'.
- If the edge is backward edge put the value of the backward edge = 0.
- A cut is called minimum cut if its capacity not exceed the capacity of any other cut of the network.

In above network,

If p = {s} and p'={a, b, c, d, t} then c(p, p') = c(s, a) + c(s, b) = 2 + 1 = 3.

If p = {s, a} and p' = {b, c, d, t} then c(p, p') = c(s, b) + c(a, b) + c(a, d) = 1 + 3 + 4 = 8.

Similarly,

Table for possible s-t cut

P	p'	Capacity c(p, p')
{s}	{a, b, c,,d, t}	3
{s, a}	{b, c, d, t}	8
{s, b}	{a, c, d, t}	13
{s, c}	{a, b, d, t}	7
{s, d}	{a, b, c, t}	15
{s, a, b}	{c, d, t}	15
{s, a, c}	{b, d, t}	20
{s,,a, d}	{b/C, t}	16
{s, b, c}	{a, d, t}	12
{s, b, d}	{a, c, t}	19
{s, c, d}	{a, b, t}	11
{s, a, b, c}	{d/t}	10

P	p'	Capacity c(p, p')
{s, a, b, d}	{c, t}	17
{s, a, c, d}	{b, t}	12
{s, b, c, d}	{a, t}	10
{s, a, b, c, d}	{t}	8

Min cut = 3

Hence, in the above example max flow = min cut = 3.

5. ***Explain the Euler path and Euler circuit with the help of a diagram. State the necessary and the sufficient conditions for Euler circuits and paths.*** ***[072 Ashwin]***

Necessary and Sufficient Conditions for Euler circuits.

- A connected multigraph has an Euler circuit if and only if each of its vertices has even degree.

  Proof (why 'only if')

- Assume the graph has aeuler circuit. Observe that every time the circuit passes through a vertex, it contributes 2 to the vertex's degree, since the circuit enters via an edge incident with this vertex and leaves via another such edge.

Necessary and Sufficient Conditions for Euler paths

A connected multigraph has an Euler path but not an Euler circuit if and only if it has exactly two vertices of odd degree.

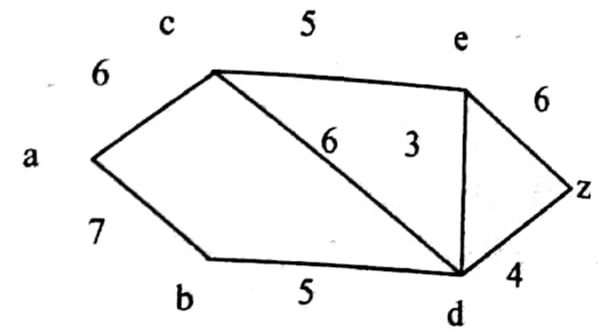
Proof:

("only if") assume the graph has an Euler path but not a circuit. Notice that every time the path passes through a vertex, it contributes 2 to the degree of the vertex (1 when it enters, 1 when it leaves). Obviously the first and the last vertices will have odd degree and all the other vertices even degree.

(if): Assume exactly two vertices, u and v have odd degree. If we connect then two vertices, then every vertex will have even degree. By definition of Eular circuit there is an Eular circuit in such a graph.

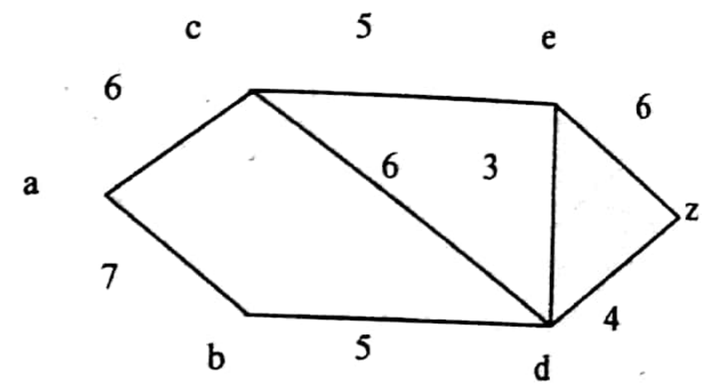
If we remove the added edge {u, v} from this circuit, we will get an Eular path for the original graph.

6. ***Using Dijkstra's algorithm to find the length of the shortest path between the vertices a and z in the weighted graph displayed below.[071 Magh]***



Solution:

Here, we can use dijkstra's algorithm to calculate the shortest path from a to z.

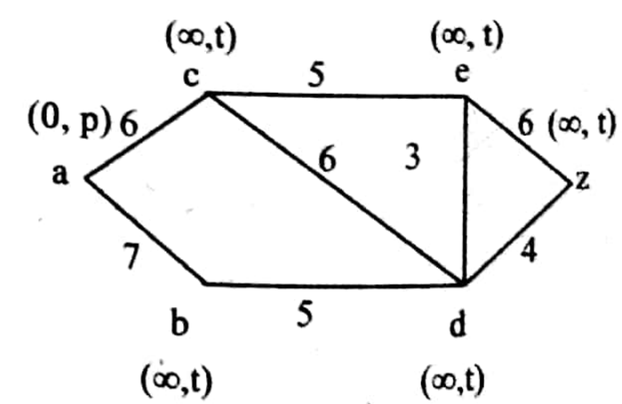


Source = a

destination = z

Step 1

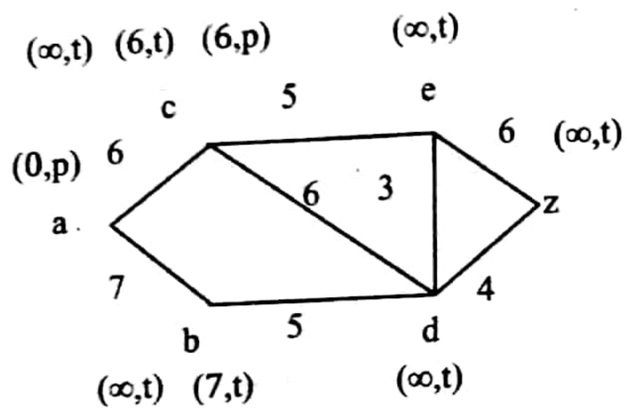
- node a is designated as the current node
- the state of node a is (o, p)
- every other node has state (∞ , t)



Step 2

Calculate possible adjacent node distance from node 'a'.

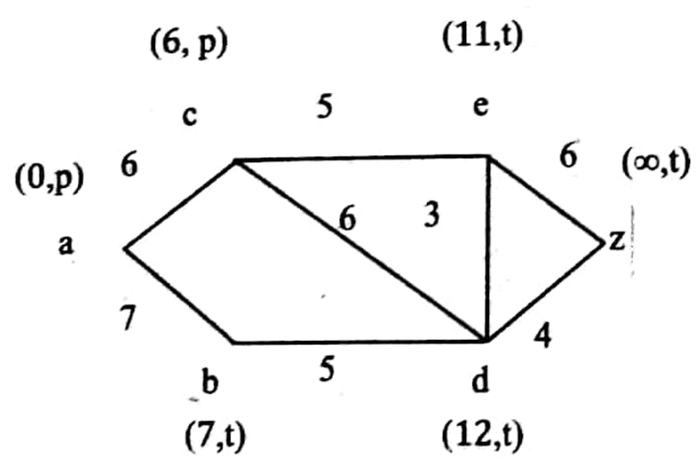
- node b and c can be reached from the current node a.
- update distance values for these node
- $d_b = \min\{\infty, 0+7\} = 7(a)$
- $d_c = \min\{\infty, 0+6\} = 6(a)$
- now, among the node b and c changes to permanent, so, its state is (6, p) while the status of b remain temporary.
- Node c becomes the current node.



Step 3

Calculate possible adjacent node from node c.

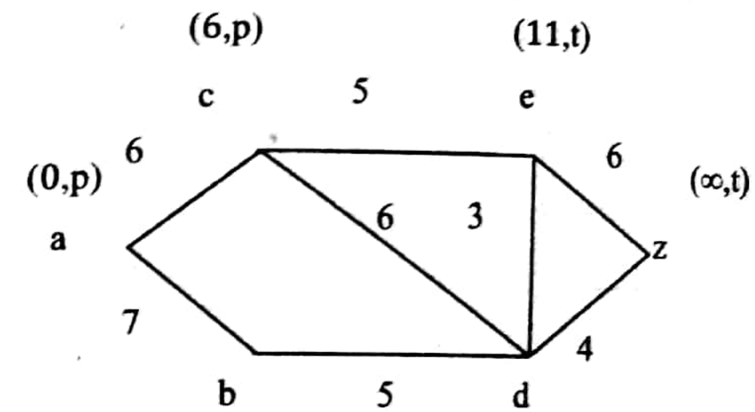
- Node d and e can be reached from the current node c
- Update distance values these nodes
- $d_d = \min\{\infty, 6+6\} = 12(a, c)$
- $d_e = \min\{\infty, 6+5\} = 11(a, c)$



Step 4

Calculate possible adjacent node from node b

- Node d can be reached from node b
- Update the distance value $dd = \{12, 7+5\} = 12(a, b)$
- Now, the status lable of node b becomes permanent
- Now, between the node d and node e, node e hass the smallest distance value,
- The status label of node e changes to permanent, while status of d remains temporary.
- Node e becomes the current node.



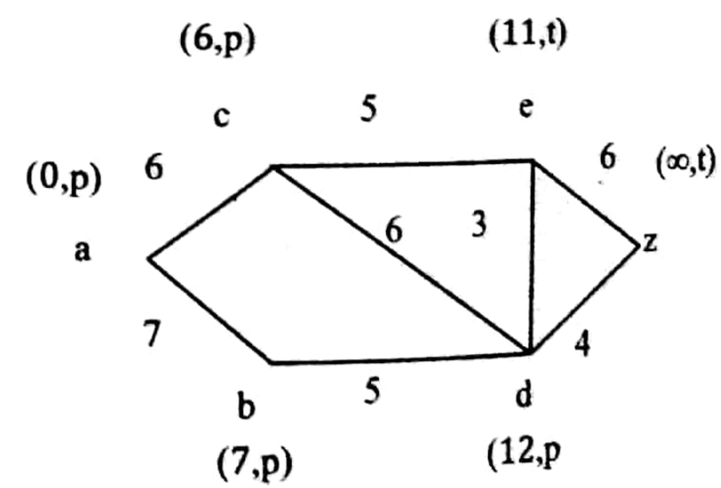
Step 5

Calculate possible adjacent node from node e

- update d and z can be reached from node e

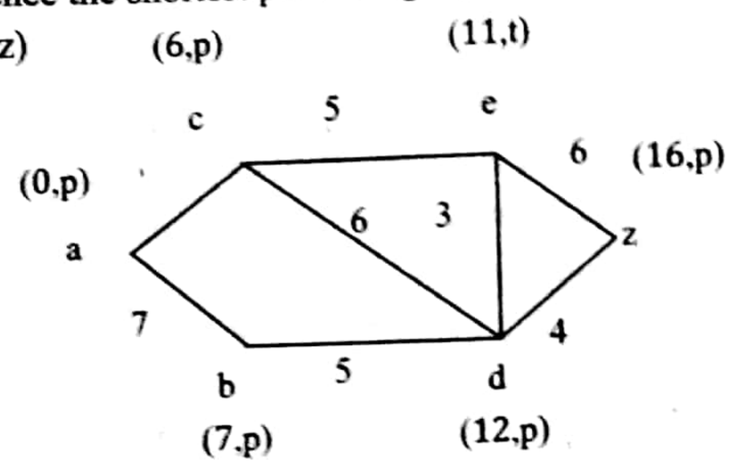
  $d_d = \min\{12, 11+3\} = 12(a, c)$

  $d_z = \min\{\infty, 11+6\} = 17(a, c, e)$
- The status of node d changes to permanent.



Step 6:

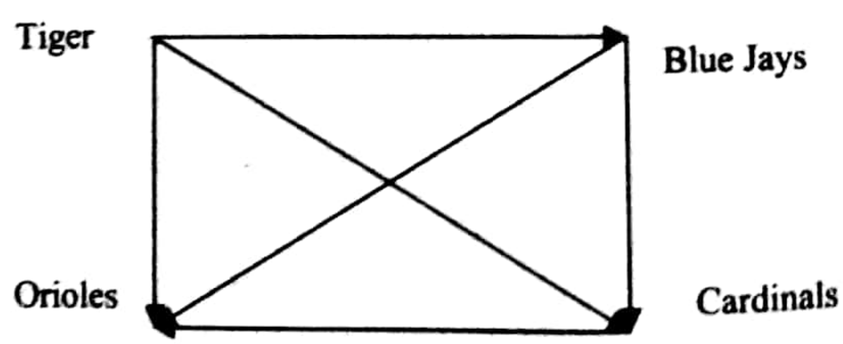
- The next possible adjacent node from node d os node z.
- Update the distance value
- $d_z = \min\{17, 12 + 4\} = 16(a, c, d)$
- Hence the shortest path using dijktra's algorithm is = 16(a, c, d, z)



The iterations of dijkstra's algorithm are described is the following table.

S	L(a)	L(b)	L(c)	L(d)	L(e)	L(z)
Initial	0	∞	∞	∞	∞	∞
{a}	0	7	6	∞	∞	∞
{a, c}				12	11	∞
{a, c, d}						16
{a, c, d, z}						16

7. ***In a round robin tournament the tigers beat the blue jays. The tigers beat the cardinals, the tigers beat the orioles, the blue jays beat the orioles, and the cardinals beat the orioles. Model this outcome with a directed graph.[071 Magh]***



8. ***Draw the figure for the complete bipartite graph k3,4 and the cycle graph with 5 vertex (this is usually denoted by (5). What is the chromatic number of the drawn complete bipartite graph k3,4 and the cycle graph c5? [071 Magh]***

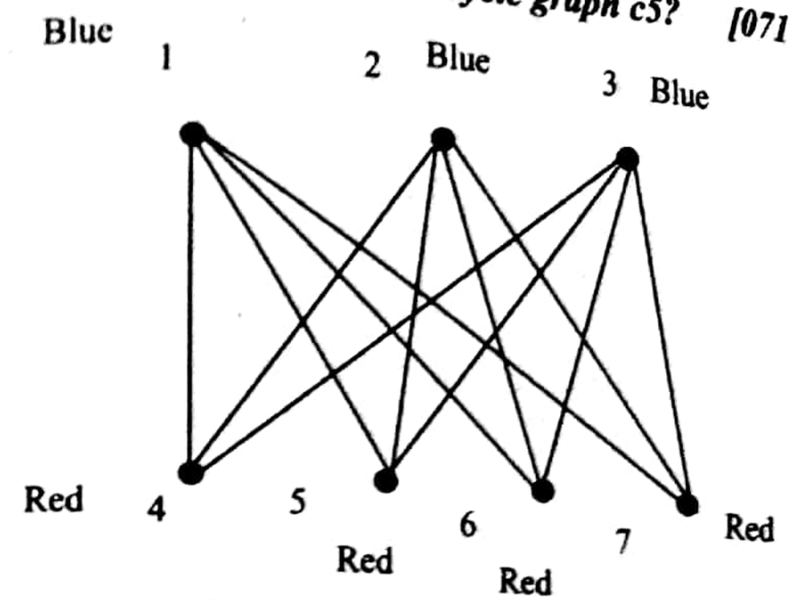


Fig: Bipartite graph k3,4

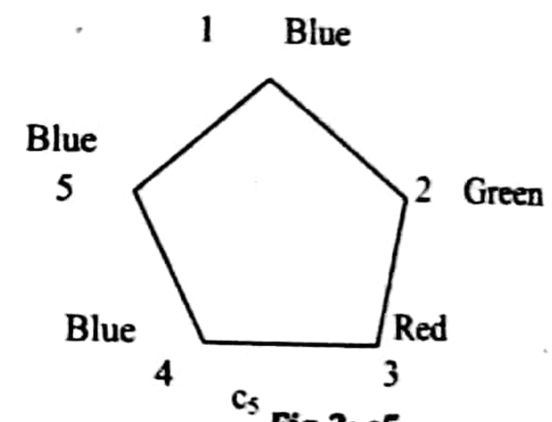


Fig 2: c5

The chromatic number of complete bipartite graph k3,4 is two

The chromatic number of cycle graph c5 is four.

9. ***State necessary and sufficient conditions for euler circuits and paths. [071 Mgh]***

Solution:

Necessary and Sufficient Conditions for Euler circuits.

- A connected multigraph has an Euler circuit if and only if each of its vertices has even degree.

  Proof (why 'only if')

- Assume the graph has aeuler circuit. Observe that every time the circuit passes through a vertex, it contributes 2 to the vertex's degree, since the circuit enters via an edge incident with this vertex and leaves via another such edge.

Necessary and Sufficient Conditions for Euler paths

- A connected multigraph has an Euler path but not an Euler circuit if and only if it has exactly two vertices of odd degree.

Proof:

("only if") assume the graph has an Euler path but not a circuit. Notice that every time the path passes through a vertex, it contributes 2 to the degree of the vertex (1 when it enters, 1 when it leaves). Obviously the first and the last vertices will have odd degree and all the other vertices even degree.

(if): assume exactly two vertices, u and v have odd degree. If we connect then two vertices, then every vertex will have even degree. By definition of Eular circuit there is an Eular circuit in such a graph.

If we remove the added edge {u, v} from this circuit, we will get an eula path for the original graph.

10. Write short notes on: ***[071 Magh]***

a. Hamiltonian path and Hamiltonian circuit

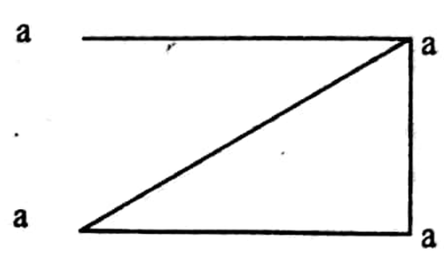
b. Max flow and min cut theorem

c. Planar graph

Solution:

a. Hamiltonian path:

- A path in a graph G that contain every vertex of G is called a Hamiltonian path in G.



Hamiltonian circuit:

- A circuit in a graph G that contains every vertex of G is called a Hamiltonian circuit in G.

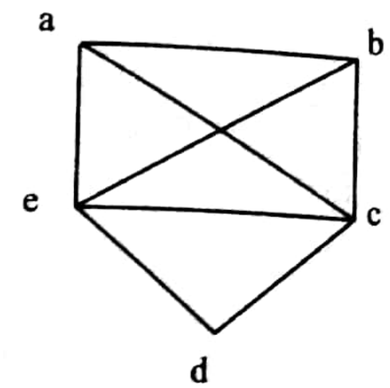


Fig 2: Hamiltonian circuit a, b, c, d, e,

b. Max flow and min cut theorem

Max flow theorem:

- In network or transport network the flow is the amount of commodity (number of car in a road, gallon of oil in pipe, bits of information in channel etc.) transported from one place to another for an instant of time.
- The maximum amount of flow is called the capacity of the line or arc or edge.
- The flow that achieves the largest possible value is called the maximum flow or maximal flow in a network.
- To calculate max flow calculate the unused capacity c(e)-f(e) for each forward arc and f(e) in each backward arc.
- In a f-augmenting path from source to sink, for the forward edge increase the flow with minimum slack (unused capacity) followed in the path.
- For backward edge in the f-augementing path from source to sink, decrease the flow with the minimum slack

  = min (c(e)-f(e))

  f* = f + (for forward edge)

  f* = f - (for backward edge)

Min cut theorem:

- In min cut we calculate the capacity by separating vertices into two subsets p and p'.

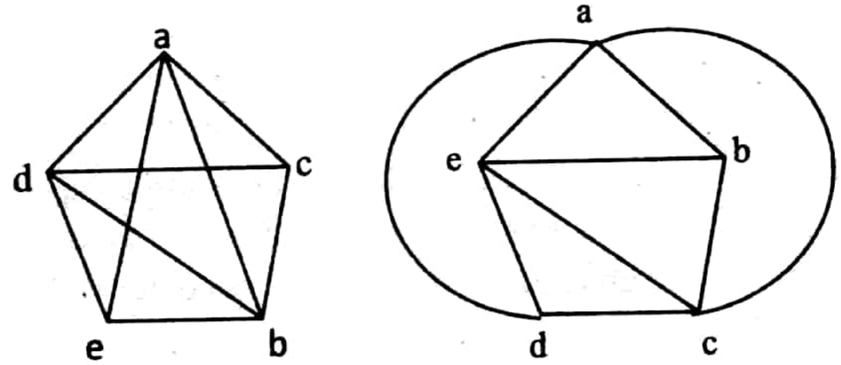
- P contains s only or s and other vertices which are not in p'.
- If there are 4 vertices other than source and sink, then there are 24= 16 s-t cuts.
- The capacity of cut denoted by c(p, p') is defined to the sum of the capacity of those edge directed from the vertices in set p to the vertices in p'.
- If the edge is backward edge put the value of the backward edge = 0.
- A cut is called minimum cut if its capacity not exceed the capacity of any cut of the network.
- In a network, max flow = min cut.

c. **Planar graph:**

A graph is called planar if it can be drawn in the plane without any edges crossing. Such drawing is called a planar representation of the graph.

For example,

The planar representation of the graph G1 is G2.



G1 G2

Euler's formula

Let G be a connected planar simple graph with e edges and v vertices. Let r be the number of regions in a planer representation of G. then r = e-v + 2

Corollary 1:

If G is a connected planar simple graph with e edges and v vertices where $v \geq 3$, then $e \leq 3v-6$.

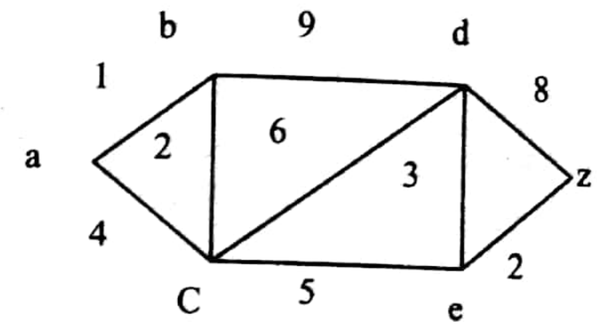
Corollary 2:

If G is a connected planar simple graph the G has a vertex of degree not exceeding 5.

Corollary 3:

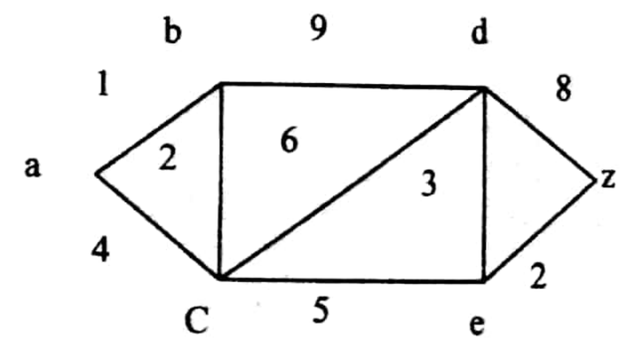
If a connected planar simple graph has e edges and v vertices with $v \geq 3$ and no circuits of length 3, then $e \leq 2v-4$.

Q. ***Use Dijkstra's algorithm to find the length of the shortest path between the vertices a and z in the weighted graph display below.*** ***[071 Bhadra]***



Solution:

we use Dijkstras algorithm to calculate the shortest path from a to z.

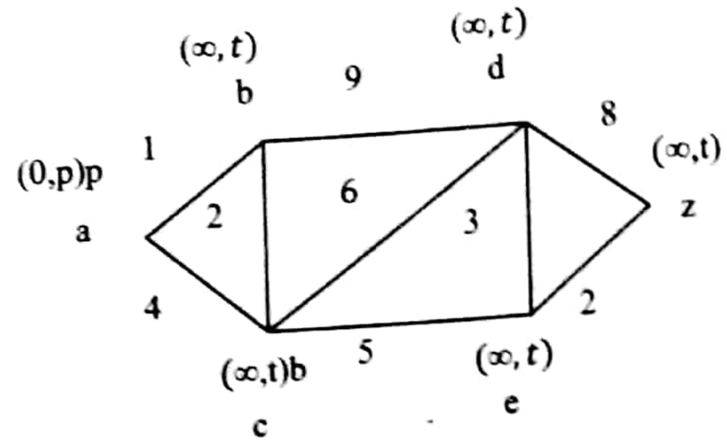


Source = a

destination = z

Step 1:

- Node a is designated as the current node
- The state of node a is (0, p)
- Every other node has state (α , t)

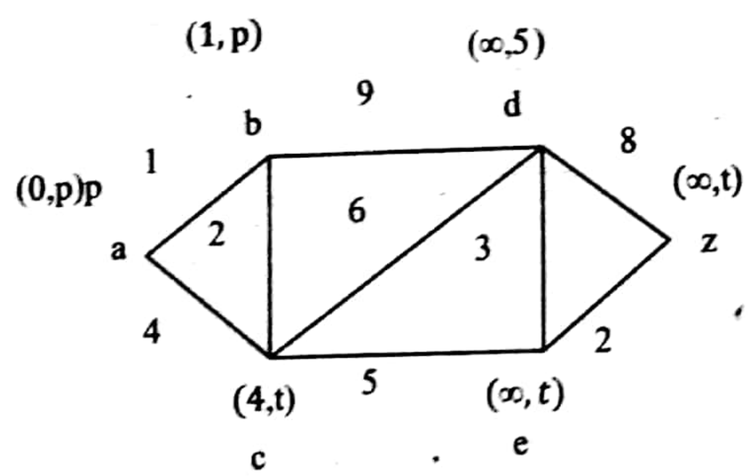


Step 2:

- Node b and c can be reached from the current node a.
- Update distance values for these node.

  $d_b = \min\{\alpha, 0 + 1\} = 1(a)$

  $d_c = \min\{\alpha, 0 + 4\} = 4(a)$
- Now, among the node b and c, node b has the smallest distance value
- The status label of node b changes to permanent so state is (1, p) while the status of c remains temporary.
- Node b becomes the current node.



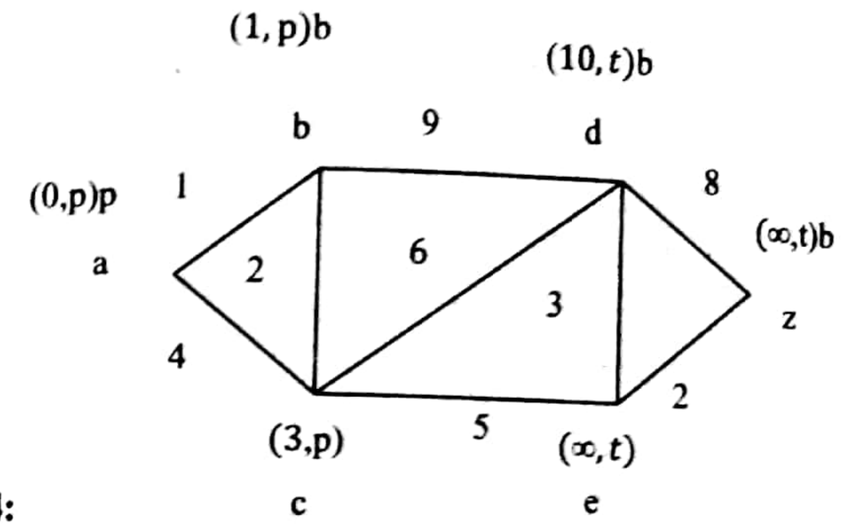
Step 3:

- Node c and d can be reached from the current node b.
- update distance values for these node.

  $d_d = \min\{\alpha, 1 + 9\} = 10(a, b)$

  $d_c = \min\{4, 1 + 2\} = 3(a, b)$

- Now, among the node c and d c has the smallest distance value
- The status label of node c changes to permanent, so its state is (3, p).
- node c become the current node.

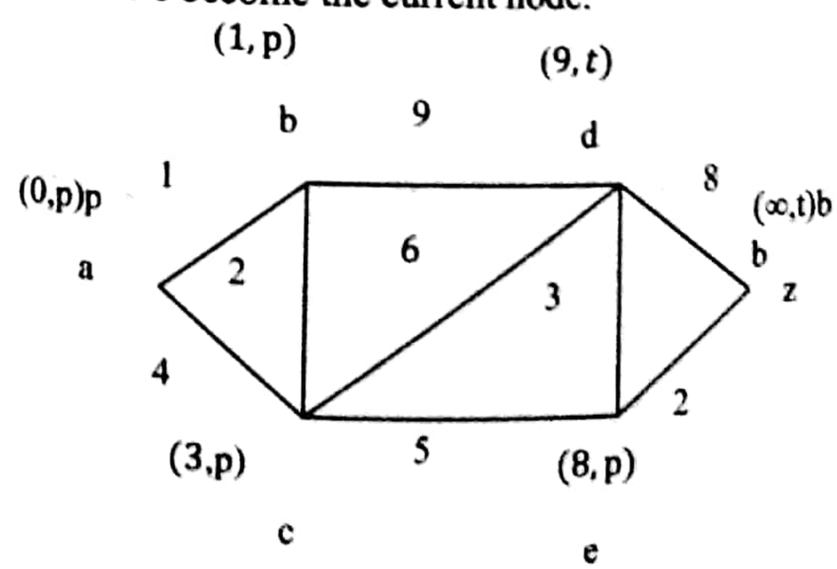


Step 4:

- Node d and e can be reached from node c.
- Update the distance values

  $d_d = \min\{10, 3 + 6\} = 9(a, b, c)$

  $d_e = \min\{\alpha, 3 + 5\} = 8(a, b, c)$
- Now, among node d and e, node e has smallest distance value.
- The status lable of e changes to permanent. So, its state is (8, p).
- Node e become the current node.

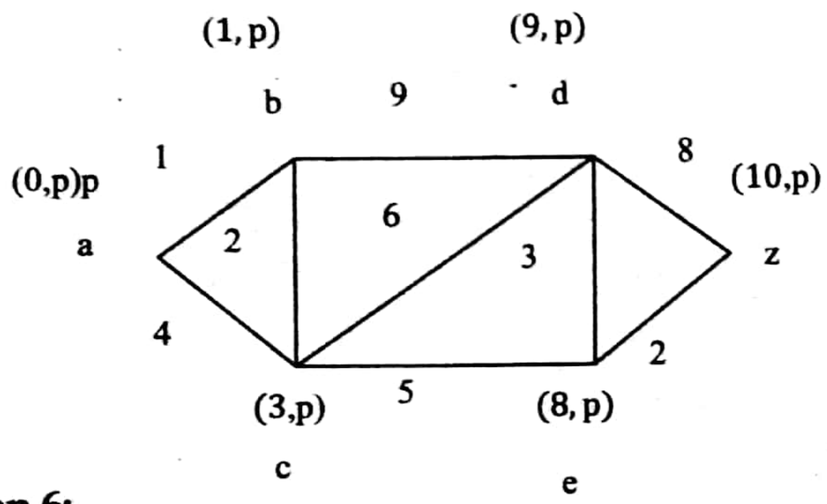


Step 5:

- Node d and z can be reached from node e.
- Update the distance values

  $d_d = \min\{9, 8 + 3\} = 9(a, b, c)$

  $d_z = \min\{\alpha, 8 + 2\} = 10(a, b, c, e)$
- Node z can be reached from node d.
- Node d becomes the current node



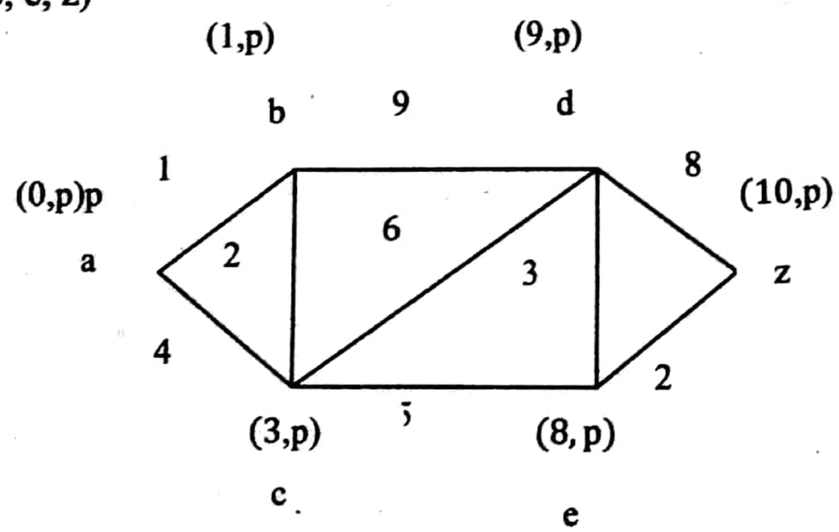
Step 6:

- Node e and z can be reached from node d
- Update the distance value

  $d_e = \min\{8, 9 + 3\} = 8(a, b, c)$

  $d_z = \min\{10, 9 + 8\} = 10(a, b, c, e)$

  Hence, the shortest path using dijkstra's algorithm is 10(a, b, c, e, z)



11. ***Draw the figure for the complete graph with 6 vertices (this is usually denoted by k_6). Define the term graph coloring and the chromatic number of a graph coloring. What is the chromatic number of the complete graph k6?[071 Bhadra]***

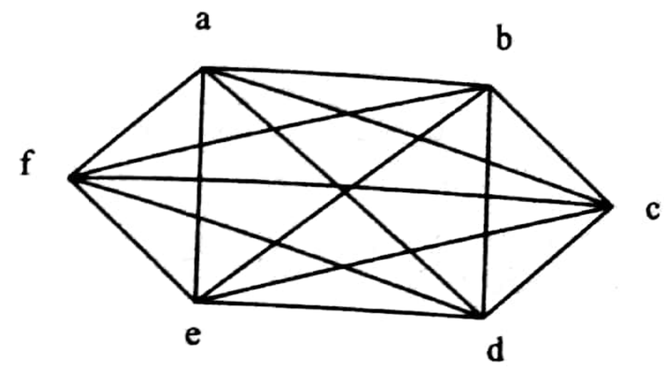
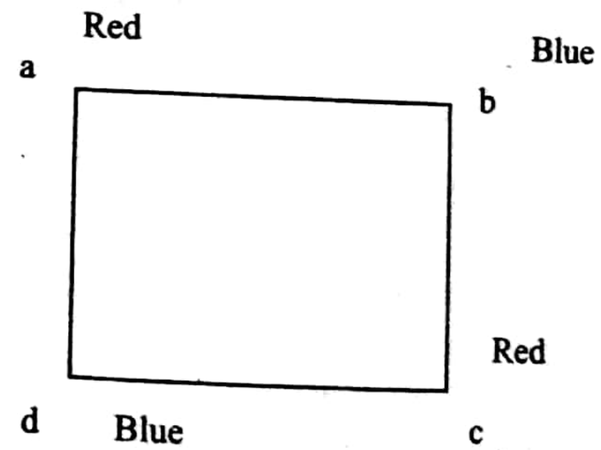


Fig: complete graph with 6 vertices(k6)

Graph coloring:

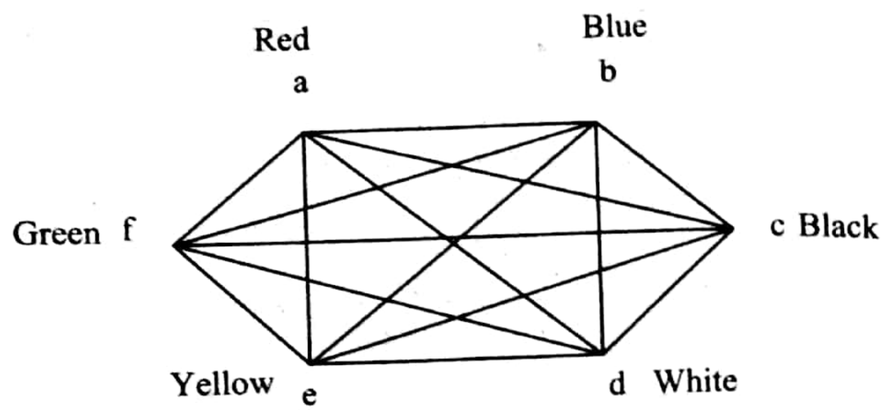
A coloring of a simple graph is the assignment of a color to each vertex of the graph so that no two adjacent vertices are assigned the same color.

Smallest number of the colors must be used instead of just assigning every region its own color.



Chromatic number:

- A chromatic number of a graph is the least no. of color needed for coloring graph.
- Chromatic number of the complete graph k_6



Chromatic number of the complete graph k6 = 6

12. ***Explain the Hamiltonian path and Hamiltonian circuit with the help of a diagram. State the necessary and sufficient conditions for Euler circuits and paths. How is Euler circuit different from the Hamiltonian circuit? [071 Bhadra]***

Hamiltonian path and Hamiltonian ciruit with the help of a diagram [2071 Magh Q.11(a)]

Necessary and sufficient conditions for Euler circuits and paths. [2071 Magh Q.10]

How is Euler circuit different from Hamiltonian circuits

For Euler circuit, a circuit c in a graph G must contain every edge of G. with no edge repetition.

But for Hamiltonian circuits, a circuit in a graph G contain every vertex of G.

13. ***Write short notes on:***

   a. ***Spanning tree***
   b. ***Cursets and cut vertices***
   c. ***Application of trees`[071 Bhadra]***

Solution:

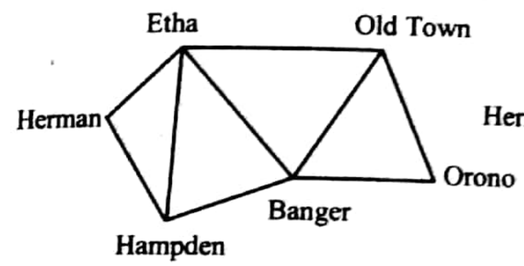
a) **Spanning tree:**

Let G be a simple graph. A spanning tree is a sub graph of G that is a tree containing every vertex of G.

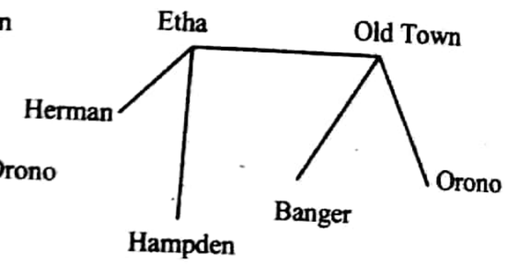
Consider system of roads in Maine. The only way the roads can be kept open in the winter is by frequently plowing them.

The highway department wants to plow the fewest roads so that there will always be cleared roads connecting any two towns. At least 5 roads must be plowed to ensure that there is a path between any two towns. Fig(b) represents one such set of roads. Note that the sub graph representing these roads is a tree because it is connected and contains six vertices and five edges.

This problem was solved with a connected sub graph with the minimum number of edges containing all vertices of the original simple graph such a graph must be a tree.



Fig(a)



Fig(b)

Fig(a): Road system

Fig(b): Sets of road to plow

b. ***Cut sets and cut vertices***

Solution:

Cut sets:

A set of edges of a graph which, if removed or cut, disconnects the graph, i.e. form a disconnected graph. A graph G is said to be disconnected if it is not connected i.e. if there exist two nodes in G such that no path in G has those nodes as endpoints.

c. **Application of trees:**

Tree, a special type of connected simple graph with no circuit, is important in both theory and application. Many powerful algorithms in computer science and software engineering are tree based algorithms.

How should items in a list be stored so that an item can be easily located? For this, we use the concept of binary search tree.

What series of decisions should be made to find an object with a certain property in a collection of objects of a certain type? For this we use the concept of decision trees.

How should a set of characters be efficiency coded by bit strings? Here, we use the concept of prefix codes.