LECTURE 1 Propositional Logic

November / 2013

Introduction

Formal logic is the study of validity and invalidity. it was first developed by the ancient Greeks who wanted to be able to reasoning carefully about sentences in natural language. However, they quickly realized that logical reasoning is difficult and unreliable when using natural language. All sorts of ambiguities arise, and it's hard to keep reasoning without getting confused.

To make such natural language sentences precise, they can be translated into the language of logic. Once we have translated sentences from English into logical expressions we can analyze these logical expressions to determine their truth values and manipulate them. This can be accomplished by **propositional logic**.

1.1. Lecture Outlines:

- What is Proposition.
- Compound Proposition.
- Truth Value.
- How to Built a Truth Table.
- Logical Operators.
- Translation English Sentences into Propositional Logic and Vice Versa.

1.2. What is Proposition

A proposition is a declarative sentence (sentence that declares a fact) that is either "true" or "false", it cannot be "maybe", or "sometimes". The following declarative sentences are propositions.

- Baghdad is the capital of Iraq.
- 1+1=3.

There are many sentences that cannot be represented by propositions because they are not declarative sentences or require preceding by other sentences in order to make their meaning clear. e.g: the following sentences are not proposition:

- What time is it?
- Read this carefully.
- x + 1 is odd.

The sentences "x+1 is odd" is true under the condition that x is an even integer.

These declarative sentences are represented by letters called **propositional variable**, e.g p, q, r but any letter could be used, e.g : P = "Baghdad is the capital of Iraq".

1.3. Compound Proposition

The compound proposition is a statement that constructed by combining more than one proposition using logical operators.

The following statement is a compound proposition: "The sun is shining and I feel happy"

p = "The sun is shining".

q = "I feel happy".

p and q = "The sun is shining and I feel happy".

The word (and) called logical operator,

Logical operator is a symbol, correspond to English words like (and, or, not,....), used to build compound Propositions.

1.4. Truth Values

Every proposition must have a truth value. The truth value for a true proposition is **true** and denoted by T, while the truth value for a false proposition is **false** and denoted by F.

For example

- p : "Baghdad is the capital of Iraq". the truth value of p is true.
- q: "13*7*11=1000". the truth value of q is false.
- r and s: "The sun is shining and I feel happy". truth value depend on value of r, s according the operator (and).

1.5. How to Build a Truth Table

The truth table for a given compound proposition displays the truth values that correspond to all possible combinations of truth values for its component proposition variables.

The truth table requires one row for each combination of propositional variable truth values. If there are k variables, this means there are 2^k rows in the table. For one variable, you get $2^1 = 2$ rows (one for true and other for false). The truth value founds in the last column.

Example

e.g: "The sun is shining and I feel happy" = p and q

- This compound proposition has 2 propositional variables(p , q), so k=2.
- Every variable has two different possible truth values (T, F), so the base is 2.
- Number of columns in the truth table is 3 for p, q, and (p and q) respectively.
- Number of rows in the truth table is 2^2 =4.

Truth Table

P	q	p ∧ q
T	T	?
T	F	?
F	T	?
F	F	?

1.6. Logical Operators

Given a statement p and q, the logical operators are as in table below:

Table1:
Logical Operators (Logical connectors)

English Word	Denoted by	Written as	Read as	Called
Not	¬ (sometimes ~)	$\neg p$	Not p	Negation
And	$\wedge$	$p \wedge q$	p and q	Conjunction
Or	$\vee$	$p \vee q$	p or q	Disjunction
Xor	$\oplus$	$p \oplus q$	p xor q	Exclusive or
If, then	$\rightarrow$	$p \rightarrow q$	If p, then q p implies q	Conditional statement (implication)
If and only if	$\leftrightarrow$	$p \leftrightarrow q$	p if and only if q p iff q	Biconditional statement (bi-implication)

1.6.1. Logical Operator - Not (Negation)

suppose the proposition: It is sunny = p

the negation is: It is not sunny = ¬p

The truth value of the negation of p, (¬p), is the opposite of the truth value of p.

Table 2:
Negation Truth Table

P	¬p
T	F
F	T

1.6.2. Logical Operator – And (Conjunction)

Suppose the propositions

p: It is hot.

q: It is sunny.

The compound proposition using conjunction is:

It is hot **and** it is sunny = $p \wedge q$.

The conjunction of p and q is true when, and only when, both p and q are true. If either p or q is false, or if both are false, $p \wedge q$ is false.

Table 3:
Conjunction Truth Table

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

There are alternative way for conjunction

- p but q = $p \land q$
- Neither p nor q = $\neg p \land \neg q$

1.6.3. Logical Operator - Or (Disjunction)

Suppose the propositions

p: Students who have taken calculus can take this class.

q: Students who have taken computer science can take this class.

The compound proposition using disjunction is:

Students who have taken calculus **or** computer science can take this class = $p \lor q$

the disjunction of p and q is true when either p is true, or q is true, or both p and q are true; it is false only when both p and q are false.

Table 4:
Disjunction Truth Table

P	q	$p \lor q$
T	T	T
T	F	T
F	T	T
F	F	F

1.6.4. Logical Operator - Xor (exclusive or)

Suppose the propositions:

p: Students who have taken calculus can take this class.

q: Students who have taken computer science can take this class.

The compound proposition using exclusive or is:

Students who have taken calculus **or** computer science, but not both, can take this class = $p \oplus q$.

The exclusive or of p and q is the proposition that is true when exactly one of p and q is true and is false otherwise.

Table 5:
Exclusive or Truth Table

P	Q	$p \oplus q$
T	T	F
T	F	T
F	T	T
F	F	F

1.6.5. Logical Operator – if, then (Conditional Statement)

Suppose the propositions:

p: It is sunny.

q: I will take you for a picnic.

The compound proposition using conditional statement or is:

If it is sunny **then** I will take you for a picnic = if p, then q = p → q

The conditional of q by p is false when p is true and q is false; otherwise it is true. p is called hypothesis and q is called conclusion.

Table 6: Conditional Statement Truth Table		
p	**q**	**p → q**
T	T	T
T	F	F
F	T	T
F	F	T

1.6.6. Logical Operator – if and only if (Biconditional Statement)

Suppose the propositions:

p: You can take the flight.

q: You buy a ticket .

The compound proposition using biconditional statement or is:

You can take the flight **if and only if** you buy a ticket = p **iff** q = p ↔ q

The biconditional statement p ↔ q is true when p and q have the same truth values, and is false otherwise. it is written by combining p→q and p←q.

Table 7: Biconditional Statement Truth Table		
P	**q**	**p ↔ q**
T	T	T
T	F	F
F	T	F
F	F	T

1.7. Translation English Sentences into Propositional Logic and Vice Versa.

The translation contains nothing but propositional variables and logical operators, represent a proposition by letters (any letter could be used), and use the logical operator symbols correspond to English words like (and, or, not,....).

The translation of proposition into English sentences is accomplished by replacing the propositional variable by the sentences and converting the logical operator to the corresponding English word. For example

- p = “A password must have at least three digits”
- q = “A password must be at least eight characters long.”

$p \wedge q$ convert to:

"A password must have at least three digits and it must be at least eight characters long.”

$p \vee q$ convert to:

"A password must have at least three digits and it must be at least eight characters long.”

1.8. Homework

HW 1:

Let p, and q be the propositions

p : It is below freezing.

q : It is snowing.

Express each of these propositions as an English sentence

1. $p \wedge \neg q$.
2. $p \rightarrow q$.
3. $(p \vee q) \wedge (p \rightarrow \neg q)$.

HW 2:

what is the truth table of the following proposition statement: $(\neg p \rightarrow q) \wedge (r \vee \neg q)$

HW 3:

Let h = “John is healthy,”
w = “John is wealthy,”
s = “John is wise.”

Write these propositions using p and q and logical connectives

1. John is healthy and wealthy but not wise.
2. John is not wealthy but he is healthy and wise.
3. John is neither healthy nor wise.

LECTURE 2

Logical Equivalence .... November / 2013

Introduction

The statements
" 6 is greater than 2 " and " 2 is less than 6 "
are two different ways of saying the same thing

A compound proposition can be replaced by another compound proposition that is logically equivalent to it without changing the truth value of the original one.

Logical equivalence is important in the design of digital circuits. Several circuits may be logically equivalent, in that they all have identical truth tables. The goal of the engineer is to find the circuit that performs the desired logical function using the least possible number of gates. This will result in optimal operating efficiency, and speed.

1. Lecture Outlines

- Logical Equivalences.
- Logical Equivalences Testing.
- Laws of Algebra of Propositions
- Compound Propositions Classification
- Precedence of logical operators.
- Homework.

2. Logical Equivalence

Compound propositions that have the same truth values in all possible cases are called **logically equivalent**. The symbol ≡ is used to denote logical equivalence, but it is not a logical connective. The symbol ⇔ is sometimes used instead of ≡ .

3. Logical Equivalence Testing

One way to determine whether two compound propositions are equivalent or not is to use a **truth table**. To show that p → q and ¬p ∨ q are logically equivalent, construct the truth table and check the truth values. If in each row the truth value of p → q is the same as the truth value of ¬p ∨ q, then they are logically equivalent, otherwise they are not logically equivalent.

Table 1:
A Demonstration That p → q and ¬p ∨ q are Logically Equivalent.

p	Q	¬ p	p→q	¬p ∨ q
T	T	F	**T**	**T**
T	F	F	**F**	**F**
F	T	T	**T**	**T**
F	F	T	**T**	**T**

They are logically equivalen

Ex:

Show if the following compound proposition are logically equivalent or not:

¬(p ∧ q) → r and (¬p ∨¬q) → ¬r

Sol:

p	q	R	¬p	¬q	¬r	p ∧ q	¬ (p ∧ q)	¬(p ∧ q) → r	(¬p ∨¬q)	(¬p ∨¬q) → ¬r
T	T	T	F	F	F	T	F	T	F	T
T	T	F	F	F	T	T	F	T	F	T
T	F	T	F	T	F	F	T	T	T	F
T	F	F	F	T	T	F	T	F	T	T
F	T	T	T	F	F	F	T	T	T	F
F	T	F	T	F	T	F	T	F	T	T
F	F	T	T	T	F	F	T	T	T	F
F	F	F	T	T	T	F	T	F	T	T

They are not logically equivalent

Sometime it is important to establish equivalences of compound propositions with a large number of variables, whereas it would take quite a bit of calculating to show their equivalence using truth tables.

So, we will establish this equivalence by developing a **series of logical equivalences**.

4. Laws of Algebra of Propositions

Given any statement variables p, q, and r, Propositions satisfy various laws which are listed in Table 2. These laws are considered a series of logical equivalences.

Table 2:
Laws of Algebra of Propositions (common logical equivalence)

p ∧ q ≡ q ∧ p	p ∨ q ≡ q ∨ p	Commutative laws
(p ∧ q) ∧ r ≡ p ∧ (q ∧ r)	(p ∨ q) ∨ r ≡ p ∨ (q ∨ r)	Associative laws
p ∧ (q ∨ r) ≡ (p ∧ q) ∨ (p ∧ r)	p ∨ (q ∧ r) ≡ (p ∨ q) ∧ (p ∨ r)	Distributive laws
p ∧ T ≡ p p ∧ F ≡ F	p ∨ F ≡ p p ∨ T ≡ T	Identity laws
p ∧¬p ≡ F ¬T≡ F	p ∨¬p ≡ T ¬F≡ T	Complement laws
¬(¬p) ≡ p		Double negative law
p ∧ p ≡ p	p ∨ p ≡ p	Idempotent laws
¬(p ∧ q) ≡¬p ∨¬q	¬(p ∨ q) ≡¬p ∧¬q	De Morgan’s laws
p ∧ (p ∨ q) ≡ p	p ∨ (p ∧ q) ≡ p	Absorption laws

Table 3: Logical equivalences involving conditional statements.	
1.	$p \rightarrow q \equiv \neg p \lor q$
	$\neg p \rightarrow q \equiv p \lor q$
2.	$\neg(p \rightarrow \neg q) \equiv p \land q$
	$\neg(p \rightarrow q) \equiv p \land \neg q$
3.	$(p \rightarrow q) \land (p \rightarrow r) \equiv p \rightarrow (q \land r)$
	$(p \rightarrow q) \lor (p \rightarrow r) \equiv p \rightarrow (q \lor r)$
4.	$(p \rightarrow r) \land (q \rightarrow r) \equiv (p \lor q) \rightarrow r$
	$(p \rightarrow r) \lor (q \rightarrow r) \equiv (p \land q) \rightarrow r$

Table 4: Logical equivalences involving biconditional statement	
1.	$p \leftrightarrow q \equiv (p \rightarrow q) \land (q \rightarrow p)$
2.	$p \leftrightarrow q \equiv \neg p \leftrightarrow \neg q$
3.	$p \leftrightarrow q \equiv (p \land q) \lor (\neg p \land \neg q)$
4.	$\neg(p \leftrightarrow q) \equiv p \leftrightarrow \neg q$

The equivalences of Tables 2, 3 and 4 are general laws that can be used to simplify more complicated compound proposition.

NOTE

Although the properties in Tables 2, 3 and 4 can be used to prove the logical equivalence of two propositions, they cannot be used to prove that propositions are not logically equivalent. On the other hand, truth tables can always be used to determine both equivalence and nonequivalence

Example

Ex:

Simplify the following compound proposition using the laws of propositions algebra :

$\neg(\neg p \land q) \land (p \lor q) \equiv p$.

Sol:

$\neg(\neg p \land q) \land (p \lor q) \equiv (p \lor \neg q) \land (p \lor q)$, by De Morgan's laws.

$\equiv p \lor (\neg q \land q)$, by distributive law.

$\equiv p \lor F$, by complement laws.

$\equiv p$, by identity laws.

<u>Ex:</u>

Show that $\neg(p \lor (\neg p \land q))$ and $\neg p \land \neg q$ are logically equivalent by developing a series of logical equivalences.

<u>Sol:</u>

$\neg(p \lor (\neg p \land q)) \equiv \neg p \land \neg(\neg p \land q)$, by De Morgan law.

$\equiv \neg p \land (p \lor \neg q)$, by De Morgan law.

$\equiv (\neg p \land p) \lor (\neg p \land \neg q)$, by distributive law.

$\equiv F \lor (\neg p \land \neg q)$, by complement law.

$\equiv (\neg p \land \neg q)$, by identity law.

<u>Ex:</u>

Show that $(\neg p \rightarrow q) \land (q \rightarrow \neg p)$ and $\neg(p \leftrightarrow q)$ are logically equivalent using the laws of propositions algebra

<u>Sol:</u>

$(\neg p \rightarrow q) \land (q \rightarrow \neg p) \equiv (\neg(\neg p) \lor q) \land (\neg q \lor \neg p)$,by conditional statement law.

$\equiv (p \lor q) \land (\neg q \lor \neg p)$,by Double negation law.

$\equiv [(p \lor q) \land \neg q] \lor [(p \lor q) \land \neg p]$,by Distributive law.

$\equiv [(p \land \neg q) \lor (q \land \neg q)] \lor [(p \land \neg p) \lor (q \land \neg p)]$,by Distributive law.

$\equiv [(p \land \neg q) \lor F] \lor [F \lor (q \land \neg p)]$,by complement law.

$\equiv (p \land \neg q) \lor (q \land \neg p)$,by identity law.

$\equiv \neg(p \rightarrow q) \lor \neg(q \rightarrow p)$,by conditional statement law.

$\equiv \neg[(p \rightarrow q) \land (q \rightarrow p)]$,by De Morgan law.

$\equiv \neg[p \leftrightarrow q]$,by biconditional statement law.

5. Compound Proposition Classification

The compound propositions can be classified according to their all possible truth values into the following categories:

- **Tautology**: A compound proposition that is always true.
- **Contradiction**: A compound proposition that is always false.
- **Contingency**: A compound proposition that is neither a tautology nor a contradiction.

<u>Ex:</u>

Show that $(p \land q) \rightarrow p$ is a tautology.

<u>Sol:</u>

$(p \land q) \rightarrow p \equiv \neg(p \land q) \lor p$,by conditional statement law.

$\equiv (\neg p \lor \neg q) \lor p$,by the De Morgan law.

$\equiv (\neg q \lor \neg p) \lor p$,by Commutative law.

$\equiv \neg q \lor (\neg p \lor p)$,by Associative law.

$\equiv \neg q \lor T$,by complement law.

$\equiv T$,by identity law.

6. Precedence of Logical Operators

We will generally use parentheses to specify the order in which logical operators are to be applied. For instance, ¬p ∨ q is the disjunction of ¬p and q, namely, (¬p) ∨ q, not the negation of the disjunction of p and q, namely ¬(p ∨ q). However, to reduce the number of parentheses, we specify the general rules of precedence as listed in table 5.

Table 5:
Precedence of Logical Operation

Operator	Precedence
¬	1
∧	2
∨	3
→	4
↔	5

- The First highest precedence operator is ¬. For example, ¬P ∧ Q means (¬P) ∧ Q.
- The second highest precedence is the ∧ operator. In expressions combining ∧ and ∨,the ∧ operations come first. For example, P ∨ Q ∧ R means P ∨ (Q∧R). If there are several ∧ operations in a sequence, they are performed left to right; for example, P ∧ Q ∧ R ∧ S means (((P ∧ Q) ∧ R) ∧ S).
- The ∨ operator has the next level of precedence, and it associates to the left. For example, P ∧ Q ∨ R ∨ U ∧ V means (((P ∧ Q) ∨ R) ∨ (U ∧ V)).
- The → operator has the next lower level of precedence. For example, P ∧ Q → P ∨ Q means (P ∧ Q) → (P ∨ Q). The → operator associates to the right; thus P → Q → R → S means (P → (Q → (R → S))).
- The ↔ operator has the lowest level of precedence, and it associates to the right.

7. Homework

HW 1:

Verify the absorption laws by truth table.

HW 2:

Show that ¬ (p⊕q) and p ↔ q are logically equivalent by developing a series of logical equivalences.

HW 3:

Use the laws of propositions algebra to verify the logical equivalence for the following compound propositions: ¬ q → ¬ p ≡ p → q

LECTURE 3

Predicate and Quantifiers

November /2013

Introduction

The sentence “x + y is greater than 0” is not a proposition because its truth value depends on the values of the variables x and y. In this lecture, we will discuss the ways that propositions can be produced from such statements.

1. Lecture Outlines

- Definition of Predicate.
- Truth Value of Predicate.
- Definition of Quantifier.
    - Universal Quantifier.
    - Existential Quantifier.
- Nested Quantifier.
- Translation from English Language into Predicate Logic and Vice Versa.
- Homework.

2. Definition of Predicate

In grammar, the word predicate refers to the part of a sentence that gives information about the noun. In the sentence “James is a student at Bedford College,” the word "James" is the noun and the phrase "is a student at Bedford College" is the predicate. The predicate is the part of the sentence from which the subject has been removed.

In logic, the statement “x is greater than 3” has two parts. The first part is the variable x. The second part—the predicate, “is greater than 3”—refers to a property that the x have. We can denote the statement “x is greater than 3” by P(x), where P denotes “ x is greater than 3” and x is the variable.

The statement P(x) is also said to be the value of the propositional function P at x. Once a value has been assigned to the variable x, the statement P(x) becomes a proposition and has a truth value.

A predicate may contain more than one variable, for instance, G(x, y), where G denotes " x > y"

Let P(x) denote the statement “x > 3.”What are the truth values of P(4) and P(2)?

Sol:

P(4) = True.

P(2) = False.

Let Q(x, y) denote the statement “x = y + 3.” What are the truth values of the propositions Q(1, 2) and Q(3, 0)?

Sol:

Q(1,2) = False.

Q(3,0) = True.

The set of all possible values that the variables can have is called domain.

3. Truth Value of Predicate

When an element in the domain of the variable of a one-variable predicate is substituted for the variable, the resulting statement is either true or false. The set of all such elements that make the predicate true is called the **truth set** of the predicate. The values for variable for which predicate is false is called a **counterexample**

4. Definition of Quantifier

One sure way to change predicates into proposition is to assign specific values to all their variables. Another way is to add quantifiers.

Quantifiers are words that refer to quantities such as “some” or “all” and tell for how many elements a given predicate is true. The types of quantification are:

- **Universal Quantification:** which tells us that a predicate is true for every element in domain
- **Existential Quantification:** which tells us that there is at least one element in the domain for which the predicate is true.

4.1. Universal Quantifier

Let Q(x) be a predicate and D the domain of x. A universal statement is a statement of the form “$\forall x \in D$, Q(x)” It is defined to be true if, and only if, Q(x) is true for every x in D. It is defined to be false if, and only if, Q(x) is false for at least one x in D.

Let the sentences “All human beings are mortal”, the universal statement can be written more formally as

$\forall x \in H$ Q(x), "x is mortal", where H:is set of human beings.

which is read as “For all x in the set of all human beings, x is mortal.”

The upside-down A symbol (∀) is intended to remind you of word (All)

Besides "for all" and "for every", universal quantification can be expressed in many other ways, including "all of," or "for each,".

To determine whether ∀x Q(x) is true, we can loop through all n values of x to see whether Q(x) is always true. If we encounter a value x for which Q(x) is false, then we have shown that ∀x Q(x) is false. Otherwise, ∀x Q(x) is true.

Ex: What is the truth value of the quantification ∀x Q(x), "x< 2", where the domain consists of all real numbers?

Sol:

x = 3 is a counterexample for the statement ∀x Q(x). Thus ∀x Q(x) is false.

Universal quantification statements are generalizations of ∧ statements. If Q(x) is a predicate and the domain D of x is the set $\{x_1; x_2; \ldots ; x_n\}$, then the statements ∀x ∈ D, Q(x) is the same as the statement $Q(x_1) \wedge Q(x_2) \wedge \ldots \wedge Q(x_n)$.

4.2. Existential Quantifier

Let Q(x) be a predicate and D the domain of x. An existential statement is a statement of the form "∃x ∈ D such that Q(x)." It is defined to be true if, and only if, Q(x) is true for at least one x in D. It is false if, and only if, Q(x) is false for all x in D.

Let the sentences "There exsits a real number which is greater than 5", the existential statement can be written more formally as

∃x ∈ R, Q(x), "x is greater than 5".

Which is read as "there is at least one x in the set of real numbers which is greater than 5".

The backwards E symbol (∃) is intended to remind you of word (Exist)

Besides the phrase "there exists" we can also express existential quantification in many other ways, such as by using the words "for some", "for at least one", "exactly", or "there is."

To see whether ∃x Q(x) is true, we loop through the value of x searching for a value for which Q(x) is true. If we find one, then ∃x Q(x) is true. If we never find such an x, then we have determined that ∃x Q(x) is false.

Ex: Let P(x) denote the statement "x> 3" What is the truth value of the quantification ∃x P(x), where the domain consists of all real numbers?

Sol:

True, Because "x> 3" is sometimes true—for instance, when x = 4.

Existential statements are generalizations of ⋁ statements. if Q(x) is a predicate and the domain D = { x_1; x_2;… ; x_n}, then ∃x ∈ D, Q(x) is the same as the statement $Q(x_1) \vee Q(x_2) \vee \ldots \vee Q(x_n)$.

NOTE

The rules for negating quantified statements are exactly the same as De Morgan's laws. This is why these rules are called De Morgan's laws for quantifiers. When the domain has n elements $x_1, x_2, \ldots, x_n$, it follows that ¬∀xP(x) is the same as $\neg(P(x_1) \wedge P(x_2) \wedge \cdots \wedge P(x_n))$, which is equivalent to $\neg P(x_1) \vee \neg P(x_2) \vee \cdots \vee \neg P(x_n)$ by De Morgan's laws, and this is the same as ∃x¬P(x). Similarly, ¬∃xP(x) is the same as $\neg(P(x_1) \vee P(x_2) \vee \cdots \vee P(x_n))$, which by De Morgan's laws is equivalent to $\neg P(x_1) \wedge \neg P(x_2) \wedge \cdots \wedge \neg P(x_n)$, and this is the same as ∀x¬P(x).

Example

Ex: Negate the following statements informally: "All cars are red"

Sol: "There exists a car which is not red".

If we translate it to predicate logic , the statement "All cars are red" is ∀x ∈ C, P(x), "x is red", where C is set of cars

The negation become ∃x ∈ C, ¬P(x), "x is red", where C is set of cars.

5. Nested Quantifier

Where one quantifier is within the scope of another, this case is called nested quantifiers, such as: ∀x ∈ R ∃y ∈ R, A(x,y), "x + y = 0".

To see whether a quantification with more than one variable is true or not, see table 1.

Table 1:
Truth Value for a quantification with two variables.

	True value	False value
∀x∀y P(x,y)	If P(x,y) is true for every pair x,y.	If there is a pair x,y for which P(x,y) is false.
∀x ∃y P(x,y)	If for every x there is a y for which P(x,y) is true.	If there is an x such that P(x,y) is false for every y.
∃x ∀y P(x,y)	If there is an x for which P(x,y) is true for every y.	If for every x there is a y for which P(x,y) is false.
∃x ∃y P(x,y)	If there is a pair x,y for which P(x,y) is true.	If P(x,y) is false for every pair x,y.

Example

Ex: Let S = {0, 2, 4, 6} and R = {0, 1, 2, 3}. What is the truth value of

$$\forall x \in S, \exists y \in R, x = 2 \times y$$

Sol: The first step is to expand the outer quantifier:

$= (\exists y \in R.\ 0=2 \times y)$

$\wedge (\exists y \in R.\ 2=2 \times y)$

$\wedge (\exists y \in R.\ 4=2 \times y)$

$\wedge (\exists y \in R.\ 6=2 \times y)$

The second step is to expand all four of the remaining quantifiers:

$= (0 = 2 \times 0) \vee (0 = 2 \times 1) \vee (0 = 2 \times 2) \vee (0 = 2 \times 3)$

$\wedge (2 = 2 \times 0) \vee (2 = 2 \times 1) \vee (2 = 2 \times 2) \vee (2 = 2 \times 3)$

$\wedge (4 = 2 \times 0) \vee (4 = 2 \times 1) \vee (4 = 2 \times 2) \vee (4 = 2 \times 3)$

$\wedge (6 = 2 \times 0) \vee (6 = 2 \times 1) \vee (6 = 2 \times 2) \vee (6 = 2 \times 3)$

$= (T \vee F \vee F \vee F) \wedge (F \vee T \vee F \vee F) \wedge (F \vee F \vee T \vee F) \wedge (F \vee F \vee F \vee T) = T \wedge T \wedge T \wedge T =$ True.

To negate a nested quantifier, we can move the negation inside all the quantifiers. For instance, to negate $\neg\forall x\ \exists y(xy = 1)$. We find that $\neg(\forall x\ \exists y(xy = 1))$ is equivalent to $\exists x \neg(\exists y(xy = 1))$, which is equivalent to $\exists x\ \ \forall y \neg(xy = 1)$. Because $\neg(xy = 1)$ can be expressed more simply as $xy \neq 1$, we conclude that our negated statement can be expressed as $\exists x\ \forall y(xy \neq 1)$.

6. Translation from English Language into Predicate Logic and Vice Versa.

There is no “cookbook” approach that can be followed step by step for translation. But it is useful when translate to logic to identify the appropriate quantifiers, identify the noun, identify the predicate, and the domain.

Ex: Translate the following statement into a logical expression:

1. "Every student in class1 has studied discrete structure".
2. "The sum of two positive integers is always positive".
3. "There is an integer number that is prime".
4. "Everyone has exactly one best friend".
5. " Every number is either even or odd".

Sol:

1. $\forall x \in S$, P(x), "x has studied discrete structure", where S is the set of student in class1.
2. $\forall x \in Z^+\ \forall y \in Z^+$, Q(x,y), "x+y > 0" where Z^+ is the set of positive integer number.
3. $\exists x \in Z^+$, P(x), "x is prime".
4. $\forall x \in P\ \exists y \in P$, F(x,y), "y is the best friend of x", where P is the set of people.
5. $\forall x \in Z$, $O(x) \vee E(x)$, "x is odd or x is even".

Ex: Translate the following statement into English language:

1. $\forall x \in T$, P(x), "x has three sides", where T is the set of triangles.
2. $\exists m \in Z^+$, Q(m), "$m^2=m$".
3. $\exists x \in R\ \exists y \in R$, S(x,y), "xy=6".
4. $\forall x \in R\ \exists y \in R$, T(x,y), "x+y=10".
5. $\exists n \in Z^+$, Prime(n) $\wedge$ Even(n), "n is prime and n is even".

Sol:

1. All triangles have three sides.
2. There exist a positive integer number whose square is equal to itself.
3. There is a pair of real number x and y for which xy=6.
4. For all real number x there exist a real number y such that x+y=10.
5. There is an integer number which is both prime and even.

7. Homework

Homework 1 Express the following statement in predicate logic

1. "Everybody must take a discrete structure course or be a computer science student".
2. "For every integer number x there is a number y such that x+y>10".

Homework 2 Rewrite the following expression: $\neg\forall x(\exists y\forall x\ P(x, y) \wedge \exists y\ P(x, y))$

Homework 3 Express the following statement in English language, negate it ,and then express it again to predicate logic: $\exists b \in B$, Bird(b),"b cannot fly", where B is set or birds.

LECTURE 4 Sets

November /2013

Introduction

The set is the fundamental discrete structure on which all other discrete structures are built. Sets are used to group objects together. For instance, all the students currently taking a course in discrete structure at your college make up a set.

1. Lecture Outlines

- Definition of set.
- The size of set.
- Empty set.
- Subsets.
- Power Set.
- Set Equality.
- Cartesian Products of Set.
- Venn Diagram.
- Operation on set.
- Examples on Venn Diagram for Different Sets Operations
- Lecture Summary.
- Homework.

2. Definition of Set

The set is a collection of objects called members or elements. One way to describe a set is to write down all its members inside braces {}. For instance, the notation {a, b, c, d} represents the set with the four elements a, b, c, and d. This way of describing a set is known as the **roster method**. Sometimes set can be described without listing all its members. For instance, The set of positive integers less than 100 can be denoted by {1, 2, 3, . ., 99}.

Another way to describe a set is to use **set builder notation**. For instance, the set O of all odd positive integers less than 10 can be written as

$$O = \{x \in Z^+ \mid x \text{ is odd} \land x < 10\}.$$

which is read “x belongs to Z^+ such that x is odd and x < 10.” . This type of notation is used to describe sets when it is impossible to list all the elements of the set. The expression $x \in S$, where S is any set, is read as "x is in S" or "x belongs to S", while $x \notin S$ is read "x is not in S" or "x does not belong to S".

Some of common number sets are listed below:

1. N ={0, 1, 2, 3,...}, the set of natural numbers
2. Z ={...,−2,−1, 0, 1, 2,...}, the set of integers
3. R: the set of real numbers

3. The Size of Set

Let S be a set. If there are exactly n elements in S where n is a nonnegative integer, we say that n is the cardinality of S. The cardinality of S is denoted by |S|. Let A be the set of odd positive integers less than 10. Then |A|= 5.

4. Empty Set and Universal Set

The **empty set** is a special set that has no elements. It is also called null set, and is denoted by ∅. The empty set can also be denoted by {}. For instance, the set of all positive integers that are greater than their squares is the null set.

A common error is to confuse the empty set Ø with the set {Ø}. The single element of the set {Ø} is the empty set itself

The **universal set** (denoted by U), is a set that contains all the objects under consideration. The universal set varies depending on which objects are of interest For instance, the universal set of vowels in the English is the set of the 26 letters of the English alphabet.

5. Subset

The set A is a subset of B if every element of A is also an element of B. This can be expressed by

$$\forall x\ (x \in A \rightarrow x \in B).$$

The notation $A \subseteq B$ is used to indicate that A is a subset of the set B. For example, The set of all odd positive integers less than 10 is a subset of the set of all positive integers less than 10.

To show that A is not a subset of B we need only to find one element $x \in A$ with $x \notin B$. Such an x is a counterexample to the claim that $x \in A$ implies $x \in B$.

Ex: Let $S =\{x \in N \mid x > 3\}$ and $T =\{x \in N \mid x^2 > 4\}$. Determine whether $S \subseteq T$ or $T \subseteq S$.

Sol:

S={4,5,6,…..}

T={3,4,5,6,…..}

To show if S ⊆ T, we must find that the statement ∀x (x ∈ S → x ∈ T) is true. To show if T ⊆ S, we must find that the statement ∀x (x ∈ T → x ∈ S) is true

So that S ⊆ T because ∀x (x ∈ S → x ∈ T) is true but ∀x (x ∈ T → x ∈ S) is false

6. Power Set

Given a set A, the **power set** of A is the set of all subsets of A, it is denoted by $\mathscr{P}(A)$. Any set is a subset of itself, also ∅ is a subset of every set. If a set has n elements, then its power set has 2^n sets.

Example: Suppose A={x, y} what is the power set of A?

Sol: $\mathscr{P}(A)$: { ∅, {x}, {y}, {x, y} }.

7. Set Equality

Given sets A and B, A equals B, if and only if, every element of A is in B and every element of B is in A. it is written symbolically as:

$$A = B \Leftrightarrow A \subseteq B \text{ and } B \subseteq A.$$

The sets {1, 3, 5} and {3, 5, 1} are equal, because they have the same elements. Note that the order in which the elements of a set are listed does not matter. Note also that it does not matter if an element of a set is listed more than once, so {1, 3, 3, 3, 5, 5, 5, 5} is the same as the set {1, 3, 5} because they have the same elements.

8. Cartesian Products of Set

Let A and B be sets. The Cartesian product of A and B, denoted by A × B, is the set of all ordered pairs (a, b), where a ∈ A and b ∈ B. Hence,

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

Example: Ex: Let A1 ={x, y}, A2 ={1, 2}, and A3 ={a, b}, Find A1 × A2, (A1 × A2) × A3, A1 × A2 × A3.

Sol:

A1×A2 ={ (x, 1), (x, 2), (y, 1), (y, 2) }

(A1×A2)×A3 ={ ((x, 1), a), ((x, 2), a), ((y, 1), a), ((y, 2), a), ((x, 1), b), ((x, 2), b), ((y, 1), b), ((y, 2), b) }

A1×A2×A3 ={ (x, 1, a), (x, 1, b), (x, 2, a), (x, 2, b), (y, 1, a), (y, 1, b), (y, 2, a),(y, 2, b) }

9. Venn Diagram

Sets can be represented graphically using Venn diagrams. In Venn diagrams the universal set U, is represented by a rectangle. Inside this rectangle, circles are used to represent sets. Sometimes points are used to represent the particular elements of the set. Venn diagrams are often used to represent the relationships between sets.

EX: Draw a Venn diagram that represents set V, the set of vowels in the English alphabet.

Sol:

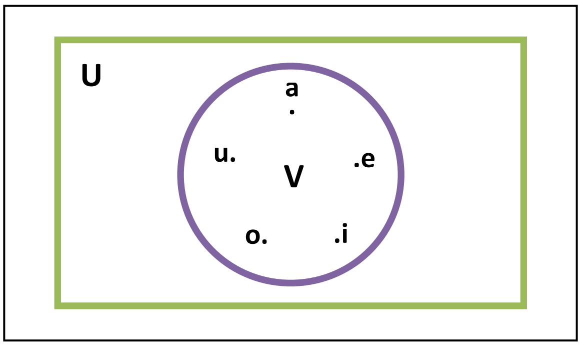


Figure1: Venn Diagram for Vowels

If A and B are two sets, the relationship A ⊆ B can be pictured as shown in figure 2.

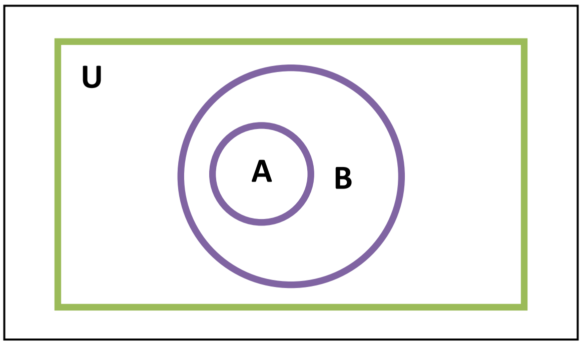


Figure2: Venn Diagram for A⊆B

The relationship A ⊈ B can be represented in different ways with Venn diagrams, as shown in Figure 3.

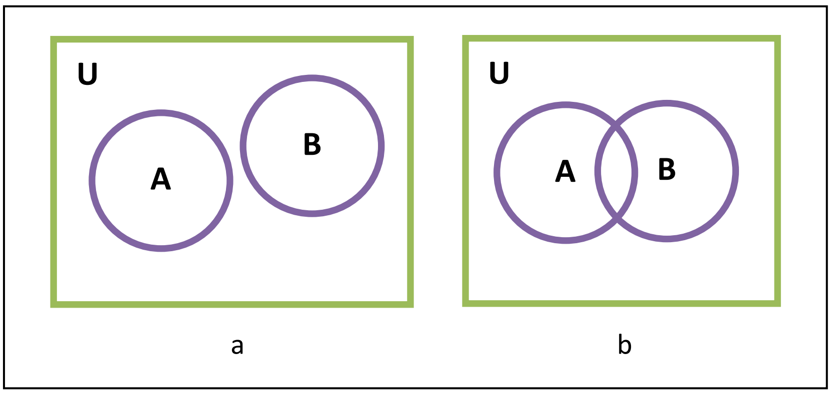


Figure3: Venn Diagram for A⊈B

The two sets A and B are called **disjoint**, if A∩B=∅. They can be represented as shown in figure 3b.

Ex: Let A = {1, 3, 5}, and B = {2, 4, 6}. Show if the two sets are disjoint or not?

Sol: A∩B = {1, 3, 5} ∩ {2, 4, 6} = ∅.

10. Operations on Set

Let A and B be subsets of a universal set U, these two sets can be combined in many different ways as follows:

1. **The union of A and B**: it is denoted by $A \cup B$. It is equal the set of all elements that are in at least one of A or B. The shaded region represent $A \cup B$.
   $A \cup B = \{x \mid x \in A \vee x \in B\}$

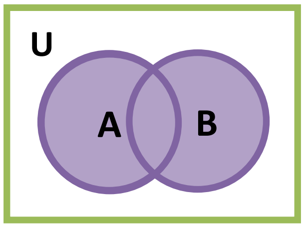


Figure 4: Venn Diagram For $A \cup B$

2. **The intersection of A and B:** it is denoted by $A \cap B$. It is equal the set of all elements that are common to both A and B. The shaded region represent $A \cap B$.
   $A \cap B = \{x \mid x \in A \wedge x \in B\}$

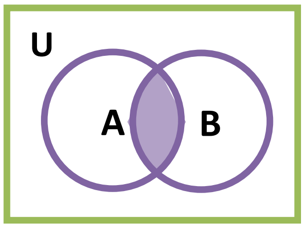


Figure 5: Venn Diagram For $A \cap B$

3. **The difference of B minus A**: it is denoted by $B - A$. It is the set of all elements that are in B and not A. The shaded region represent B - A.
   $A - B = \{x \mid x \in A \wedge x \notin B\}$
   $B - A = \{x \mid x \notin A \wedge x \in B\}$

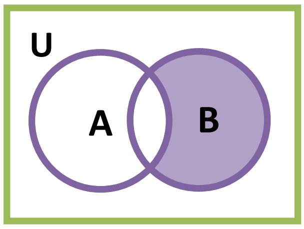


Figure 6: Venn Diagram For B - A

4. **The complement of A**: It is denoted by A^c (sometimes denoted by $\bar{A}$). is the set of all elements in U that are not in A. The shaded region represent A^c.
   $A^c = \{x \mid x \in U \wedge x \notin A\}$

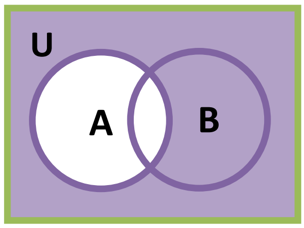


Figure 7: Venn Diagram For A^c

Ex: Let the universal set be U={a, b, c, d, e, f, g} and A ={a, c, e, g} and B ={d, e, f, g},
$A \cup B$ ={a, c, d, e, f, g}.
$A \cap B$ ={e, g}.
$B - A$ ={d, f }.
A^c ={b, d, f }.

Ex: Let $S=\{x \in N \mid 2 < x < 9\}$, $T=\{x \in N \mid 5 \le x < 14\}$
$S \cap T = \{x \in N \mid 5 \le x \le 8\}$.
$S \cup T = \{x \in N \mid 2 < x < 14\}$.

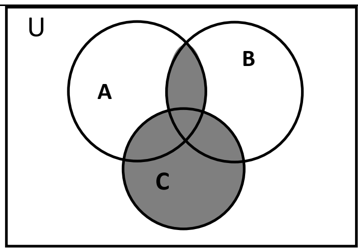
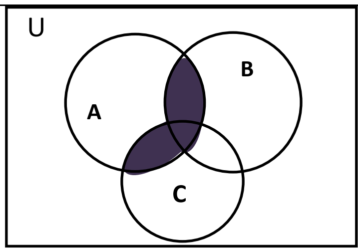
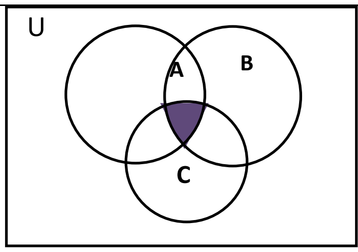
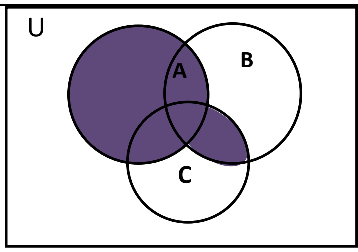
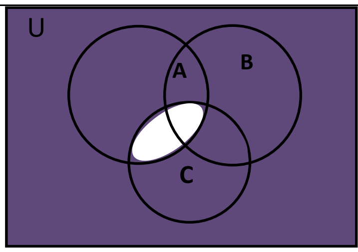
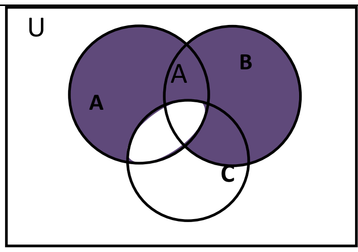
Ex: Let S={x∈N|1≤ x ≤ 5}, T={x∈N|8 < x ≤ 12}

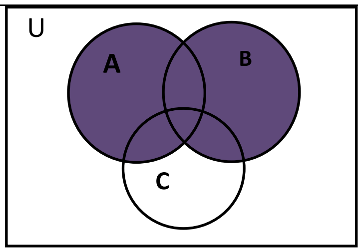
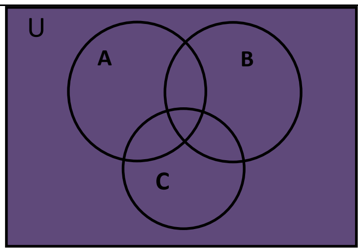
S⋂T = ∅.

S⋃T = {x∈N|1≤ x ≤ 5 ⋁ 8 < x ≤ 12}.

11. Examples on Venn Diagram for Different Sets Operations

Ex1:

(A∩B) ∪ C	
(A∩B) ∪ (A∩C)	
(A∩B) ∩ (A∩C)	
(A∪C)∩(A∪B)	
$(A∩C)^c$	
$(A∪B) ∩ (A∩C)^c$	

(A∪B) ∩ U	
(A∪B) ∪ U	

12. Homework

1 Let U={1,…,10}, A ={1, 2, 3, 4, 5}, B ={3, 4, 5, 7, 8, 9}, C ={1, 2, 3, 4, 9}, D ={2, 4, 6, 8}.
Calculate each of the following:

1. (A ∪ D) ∩ C.
2. (A ∩ B) ∪ C.
3. B^c.
4. C-D.
5. $(B-A)^c$.

2 Draw a Venn diagrams for $(A \cup B)^c \cap C$, where A, B, and C are not disjoint sets.

3 Answer the following:

1. Let S={1, 2, 3}, and T={a, b}. Find S × T. What is the cardinality of S × T?
2. Find 𝒫(A), where A={l, m, n}. how many sets are in 𝒫(A)?
3. Let E={2,4,6,8,…20}, write E in set builder notation.

LECTURE 5

Introduction to Graphs..................................... December /2013

Introduction

Graph is a discrete structure used as models in a variety of area. For instance, we can determine whether it is possible to walk down all the streets in a city without going down a street twice. We can determine whether two computers are connected by a communications link using graph models.

1. Lecture Outlines

- Definition of Graph.
- Directed Graph (Digraph).
- Basic Terminology.
- Representation of Graph.
    - Adjacency List.
    - Adjacency Matrix.
- Homework.

2. Definition of Graph

A graph G = (V ,E) consists of V , a nonempty set of vertices (or nodes) and E, a set of edges. Each edge has either one or two vertices associated with it, called its endpoints.

In general, we visualize graphs by using points to represent vertices and line segments, possibly curved, to represent edges. The edges should either connect one vertex to another or a vertex to itself. See figure 1.

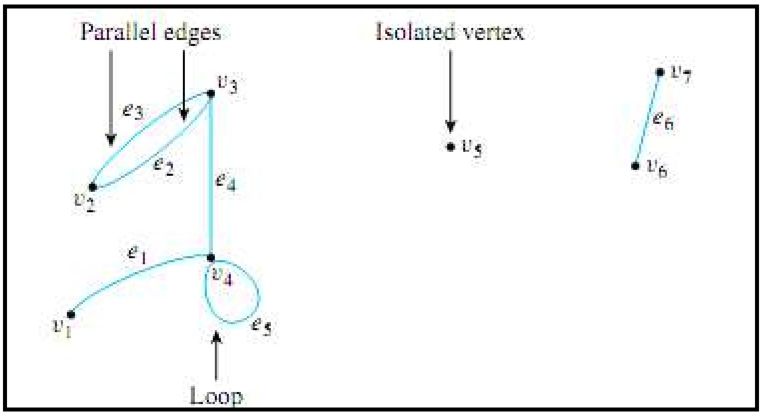


Figure 1: a graph

From the figure 1, we notice the following:

- The vertices have been labeled with (v) and the edges with (e).
- An edge connects a vertex to itself (as e5 does), it is called a loop.
- Two edges connect the same pair of vertices (as e2 and e3 do), they are said to be parallel (or multiple).
- The vertex that is unconnected by an edge to any other vertex in the graph (as v5 is), is called isolated.

The graph can be defined formally by specifying its vertex set, its edge set, and the edge-endpoint table. The description of graph shown in figure1 is as following:

- vertex set = $\{v_1, v_2, v_3, v_4, v_5, v_6, v_7\}$.
- edge set = $\{e_1, e_2, e_3, e_4, e_5, e_6\}$.
- edge-endpoint table:

edge	endpoints
e_1	$\{v_1, v_4\}$
e_2	$\{v_2, v_3\}$
e_3	$\{v_2, v_3\}$
e_4	$\{v_3, v_4\}$
e_5	$\{v_4, v_4\}$
e_6	$\{v_6, v_7\}$

3. Directed Graph (Digraph)

A directed graph (or digraph) G= (V, E) consists of a nonempty set of vertices V and a set of directed edges (or arcs) E. Each directed edge is represented by an ordered pair of vertices. The directed edge represented by (u, v) is said to start at u and end at v.

An arrow pointing from u to v used to indicate the direction of an edge that starts at u and ends at v. The vertex u is called the initial vertex of (u, v), and v is called the terminal or end vertex of (u, v).

A directed graph may contain

- Loops.
- Multiple directed edges that start and end at the same vertices.
- Directed edges that connect vertices u and v in both directions.

4. Basic Terminology

Adjacent Vertices (Neighbors):

Two vertices u and v in an undirected graph G are called adjacent (or neighbors) if u and v are endpoints of an edge.

When (u, v) is an edge of the graph G with directed edges, u is said to be adjacent to v, but v is not adjacent to u.

Degree:

In an undirected graph, The degree of a vertex (denoted by deg(v)) is the number of edges incident with it, except that a loop at a vertex contributes twice to the degree of that vertex.

In a digraph, the degree of a vertex is the sum of its in-degree and out-degree.

The in-degree of a vertex (denoted by $deg^{-}(v)$) is the number of edges coming to the vertex. The out-degree of a vertex (denoted by $deg^{+}(v)$) is the number of edges leaving the vertex.

(Note that a loop at a vertex contributes 1 to both the in-degree and the out-degree of this vertex).

$$deg(v) = deg^{-}(v) + deg^{+}(v)$$

The degree of the graph is the sum of the degrees of all vertices of that graph.

Path:

The path is a sequence of edges that begins at a vertex of a graph and travels from vertex to vertex along edges of the graph. The length of a path is the number of edges on the path.

Circuit:

The path is a circuit if it begins and ends at the same vertex, that is, if u = v, and has length greater than zero.

Simple Path:

A path or circuit is simple if it does not contain the same edge more than once.

Simple Graph:

A graph in which each edge connects two different vertices. it does not have any loops or parallel edges.

Multigraphs:

A graph that have loops, multiple edges connecting the same vertices.

From the simple graph shown in Figure 2, notice the following:

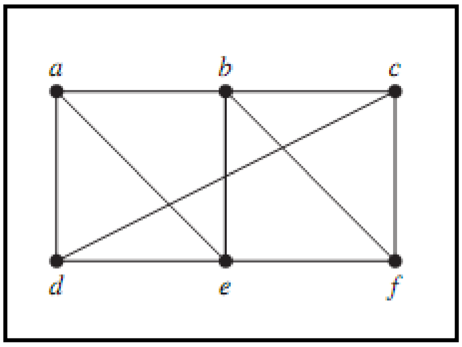
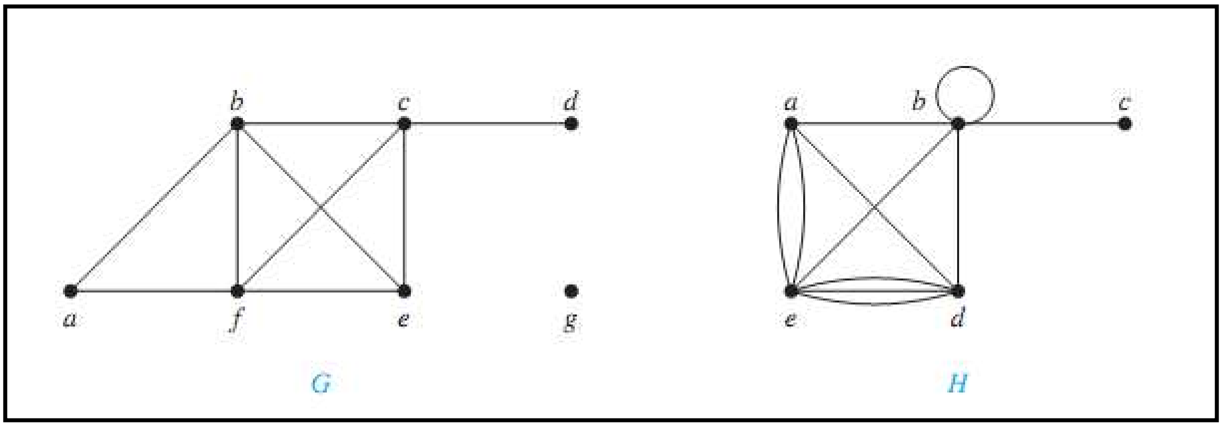


Figure 2 : A Simple Graph

- a, d, c, f, e: is a simple path of length 4, because {a,d}, {d,c}, {c, f}, and {f, e} are all edges.
- d, e, c, a: is not a path, because {e, c} is not an edge.
- b, c, f, e, b: is a circuit of length 4 because {b, c}, {c, f}, {f, e}, and {e, b} are edges and this path begins and ends at b.
- a, b, e, d, a, b:is a path of length 5, is not simple because it contains the edge {a,b} twice.

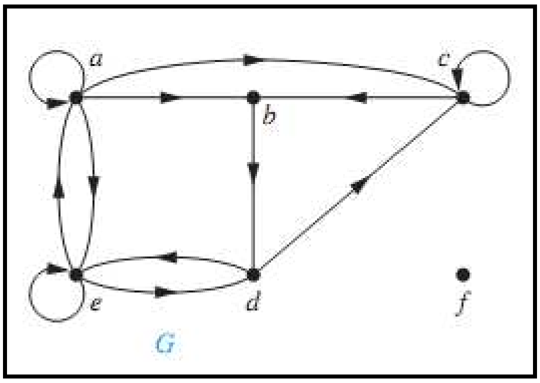
Ex1:
What are the degrees and neighbors of the vertices in the graphs G and H displayed below.



Sol:
Graph G
Degree: deg(a) = 2, deg(b) = 4, deg(c) = 4, deg(d) = 1, deg(e) = 3, deg(f) = 4, deg(g) =0;
Neighbors: N(a) = {b, f} , N(b) = {a, f, e, c}, N(c) = {b, f, e}, N(d) = {c}, N(e) = {c, b, f}, N(f) = {e, c, b, a}, N(g)= {}

Graph H
Degree: deg(a) = 4, deg(b) = 6, deg(c) = 1, deg(d) = 5, deg(e) = 6.
Neighbors: N(a) = {e, d, b} , N(b) = {a, b, c, d, e}, N(c) = {b}, N(d) = {b, a, e}, N(e) = {a, b, d}.
Graph G is simple graph.
Graph H is multigraph.

Ex 2: Find the in-degree and out-degree of each vertex in the digraph G shown below.



Sol:

in-degrees in G: $deg^{-}(a) = 2$, $deg^{-}(b) = 2$, $deg^{-}(c) = 3$, $deg^{-}(d) = 2$, $deg^{-}(e) = 3$, $deg^{-}(f) = 0$.
out-degrees in G: $deg^{+}(a) = 4$, $deg^{+}(b) = 1$, $deg^{+}(c) = 2$, $deg^{+}(d) = 2$, $deg^{+}(e) = 3$, $deg^{-}(f) = 0$.

5. Representation of Graph

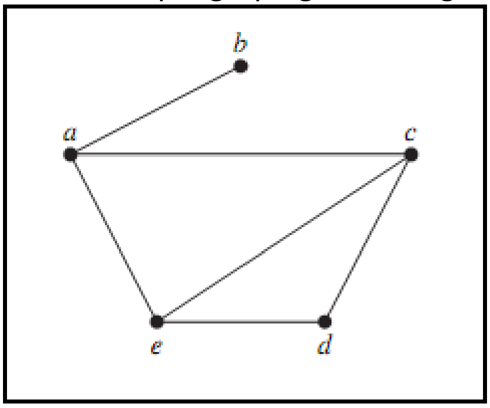
The two common ways to represent graphs are:

1- Adjacency lists.
2- Adjacency matrix.

5.1. Adjacency List

The adjacency list represent the graph by specify the vertices that are adjacent to each vertex of the graph. Directed graph represented by listing all the vertices that are the terminal vertices of all vertices of the graph.

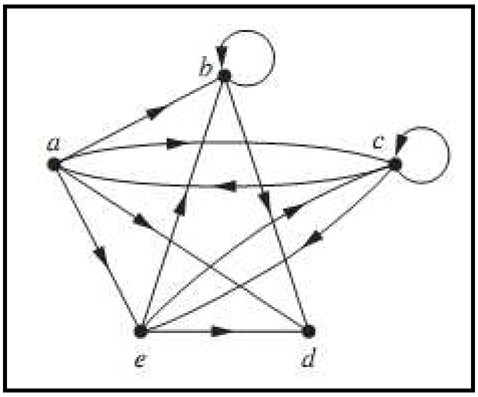
Ex 3:
Use adjacency lists to describe the simple graph given in Figure below:



Sol:

Vertex	Adjacent Vertices
a	b, c, e
b	A
c	a, e, d
d	c, e
e	a, c, d

Ex 4:
Use adjacency lists to describe the digraph given in Figure below:



Sol:

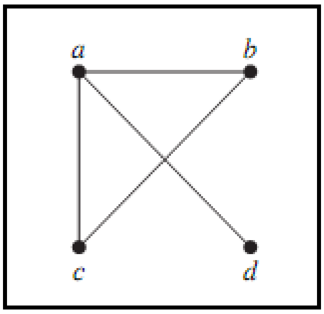
Initial Vertex	Terminal Vertices
a	b, c, d, e
b	b, d
c	a, c, e
d	
e	b, c, d

5.2. Adjacency Matrix

The adjacency matrix A of a graph is the n x n zero–one matrix with 1 as its (i, j)th entry when v_i and v_j are adjacent, and 0 as its (i, j)th entry when they are not adjacent. In other words, if its adjacency matrix is A =[aij], then

$$a_{ij} = \begin{cases} 1 & \text{if } \{v_i, v_j\} \text{ is an edge of } G, \\ 0 & \text{otherwise.} \end{cases}$$

Ex 5: Use an adjacency matrix to represent the graph shown below.



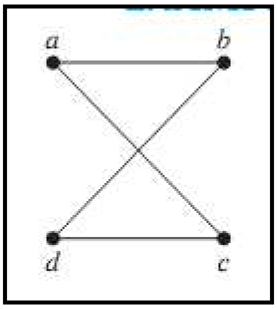
Sol:

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

Ex 6: Draw a graph with the following adjacency matrix, with respect to the ordering of vertices a, b, c, d.

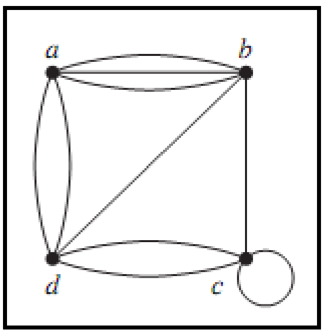
$$A = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix}$$

Sol:



In the undirected graph, when multiple edges connecting the same pair of vertices v_i and v_j ,or multiple loops at the same vertex, are present, the adjacency matrix is no longer a zero–one matrix, because the (i, j)th entry of this matrix equals the number of edges that are associated to $\{v_i , v_j\}$.

Ex 7: Use an adjacency matrix to represent the graph shown below



Sol:

$$A = \begin{bmatrix} 0 & 3 & 0 & 2 \\ 3 & 0 & 1 & 1 \\ 0 & 1 & 1 & 2 \\ 2 & 1 & 2 & 0 \end{bmatrix}$$

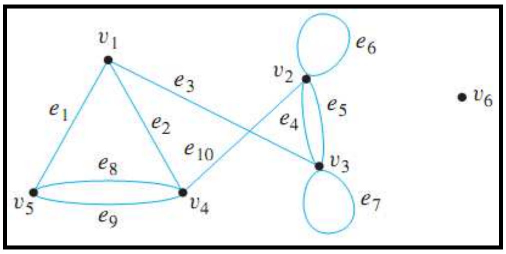
Ex 8: Use an adjacency matrix to represent the digraph shown in example 4.

Sol:

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

6. Homework

1 For the graph shown below, answer the following:



1. Find all vertices that are adjacent to v3.
2. Find all loops.
3. Find all parallel edges.
4. Find all isolated vertices.
5. Find the degree of v3.
6. Find the total degree of the graph
7. Is v1 v3 v2 v2 v4 is a path or circuit or simple path and why
8. Is v5 v4 v2 v3 path or simple path, make it a circuit.
9. Represent the graph by adjacency matrix.
10. Define this graph formally

LECTURE 6

Relations

December /2013

Introduction

There are many kinds of relationship that occur in everyday life. Some of these describe how the members of a family are related to each other: parent, child, brother, sister. We could also have a relation called is in for cities and countries: for example, London is in Great Britain, and Paris is in France. Relations are used to describe how two numbers are related to each other.

1. Lecture Outlines

- Definition of Relation.
- Binary Relation.
- Relations on a Set.
- Graphical Representation of Relation.
- Properties of Relations.
- Homework.

2. Definition of Relation

Relations are used to describe how two numbers are related to each other; for example expressions like x < y use the relations <. It is a subset of a Cartesian product

We use the notation **aRb** to denote that (a, b) ∈ R and **a**$\not R$**b** to denote that (a, b) ∉ R. Moreover, when (a, b) belongs to R, a is said to be related to b by R.

<u>Ex 1:</u>
Consider R1 is relation on the set of integers:

R1 ={(a, b) | a ≤ b},

(1, 1) ∈ R1

(1, 2) ∈ R1

(2, 1) ∉ R1

(1,−1)∉ R1

3. Binary Relation

When we use the term relation by itself, we will mean **binary relation**, because it is a subset of a Cartesian product of two sets.

A **binary relation** from A to B is a set R of ordered pairs where the first element of each ordered pair comes from A and the second element comes from B.

Ex 2:

Let A = {0, 1, 2}, B = {a, b} and E = {(0, a), (0, b), (1, a), (2,b)}

This means that 0 E a, 0 E b, 1 E a, 1 $\not E$ b, 2 $\not E$ a, 2 E b.

Ex 3:

Let A = {2, 4}, B = {6, 8, 10}, define the relation S = {(a, b) | b – 4 = a}.

Sol:

S= { (2, 6), (4, 8) }

4. Relations on a Set

A relation on a set A is a relation from A to A. In other words, a relation on a set A is a subset of A × A.

Ex 4:

Let A ={1, 2, 3, 4}, define the relation D = {(a, b) | a divides b}.

Sol:

D ={ (1, 1), (1, 2), (1, 3), (1, 4), (2, 2), (2, 4), (3, 3), (4, 4)}.

5. Graphical Representation of Relation

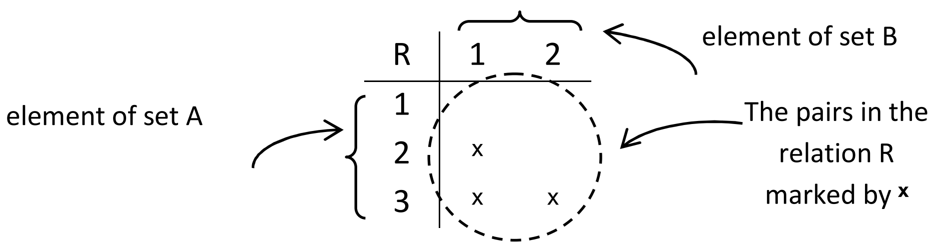
The common ways to represent a relation between sets graphically are:

- Table.
- Arrow Diagram.
- Zero–One Matrices.
- Digraph.

Suppose that A ={1, 2, 3} and B ={1, 2}. Let the relation R= { (a, b) |a > b} where a ∈ A, b ∈ B. The ordered pair of relation R is { (2,1), (3,1), (3,2) }. The relation R can be represented by these ways as follows:

5.1. Tables

Relation R can be represented by table as follows:

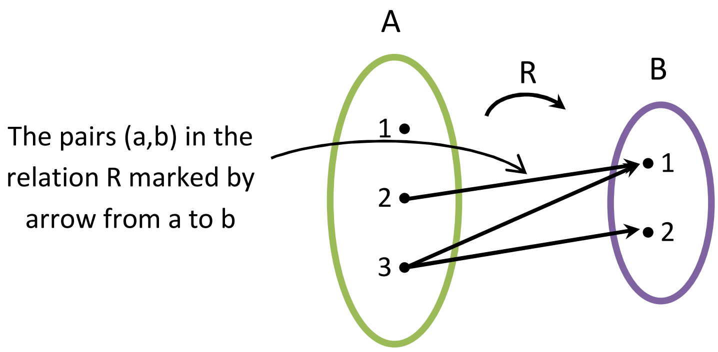


The table representation of D in example 4 is as follows:

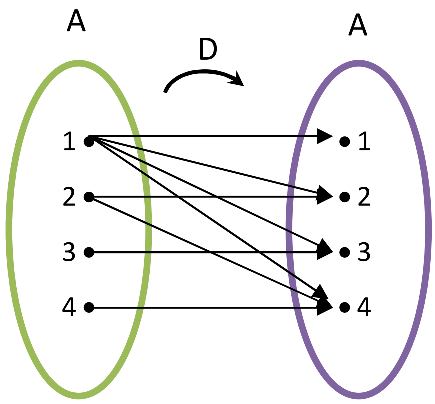
D	1	2	3	4
1	x	x	x	x
2		x		x
3			x	
4				x

5. 2. Arrow Diagram Representation

Relation R can be represented by arrow diagram as follows:



The arrow diagram representation of D in example 4 is as follows:



5. 3. Zero - One Matrix Representation

Relation can be represented by the matrix M =[m_{ij}], where

$$mij = \begin{cases} 1 & \text{if } (ai, bj) \in R, \\ 0 & \text{if } (ai, bj) \notin R \end{cases}$$

The zero-one matrix representation of relation R is as follows:

$$m_{ij} = \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 1 & 1 \end{bmatrix}$$

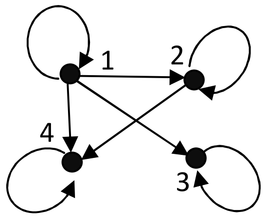
The zero-one matrix representation of D in example 4 is as follows:

$$m_{ij} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

5. 4.Digraph Representation

When a relation R is defined on a set A, the relation can be represented as directed graph. Each element of a set is represented by a vertex, and each ordered pair in the relation is represented using an edges with its direction indicated by an arrow.

The diagraph representation of D in example 4 is as follows:



6. Properties of Relation

There are several properties that are used to classify relations on a set A. We will introduce the most important of these here.

1. **Reflexive**: every element is related to itself.

$$\forall x \in A, xRx$$

2. **Symmetric**: If any one element is related to any other element, then the second element is related to the first.

$$\forall\ x, y \in A, xRy \rightarrow yRx$$

3. **Transitive**: If any one element is related to a second and that second element is related to a third, then the first element is related to the third.

$$\forall\ x, y, z \in A,\ (xRy \wedge yRz) \rightarrow xRz.$$

Ex 5:

Consider the following relations on set B={1, 2, 3, 4}:

R1 ={(1, 1), (1, 2), (2, 1), (2, 2), (3, 4), (4, 1), (4, 4)},

R2 ={(1, 1), (1, 2), (2, 1)},

R3 ={(1, 1), (1, 2), (1, 4), (2, 1), (2, 2), (3, 3), (4, 1), (4, 4)},

R4 ={(2, 1), (3, 1), (3, 2), (4, 1), (4, 2), (4, 3)},

R5 ={(1, 1), (1, 2), (1, 3), (1, 4), (2, 2), (2, 3), (2, 4), (3, 3), (3, 4), (4, 4)},

R6 ={(3, 4)}.

Determine which of the above relations are reflexive, which of them are symmetric, and which transitive.

Sol:

Reflexive:

R3 and R5 is reflexive, because they both contain all pairs of the form (a, a). The other relations are not reflexive because they do not contain all of these ordered pairs.

Symmetric:

R2 and R3 is symmetric, because in each case (b, a) belongs to the relation whenever (a, b) does.

Transitive:

R4, R5, and R6 are transitive. For each of these relations,

R4 is transitive, because (3, 2), (2, 1), and (3, 1) belongs to R4. (4, 2),(2, 1), and (4, 1) belongs to R4. (4, 3), (3, 2), and (4, 2) belong to R4.

R5 is transitive, because (1, 2), (2, 4), and (1, 4) belongs to R5. (2, 3),(3, 4), and (2, 4) belongs to R5. (1, 3), (3, 4), and (1, 4) belong to R5.

R6 is transitive because $\forall$ x, y, z $\in$ A, (x R6 y $\wedge$ y R6 z) $\rightarrow$ x R6 z is true.

Notice that (3 **R6** 4 $\wedge$ 4 **R6** z) $\rightarrow$ 3 **R6** z where z ia any element is true, because (T$\wedge$ F)$\rightarrow$F

Ex 6:

Let A ={0, 1, 2, 3} and the relations R, and S on A as follows:

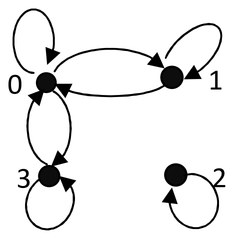
R ={(0, 0), (0, 1), (0, 3), (1, 0), (1, 1), (2, 2), (3, 0), (3, 3)},

S ={(0, 1), (2, 3)}.

What are the properties these relation have?

Sol:

from the R relation diagraph

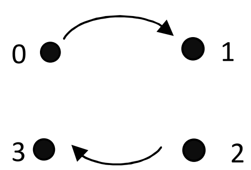


R is reflexive: There is a loop at each point of the directed graph. This means that each element of A is related to itself.

R is symmetric: In each case where there is an arrow going from one point of the graph to a second, there is an arrow going from the second point back to the first. This means that whenever one element of A is related by R to a second, then the second is related to the first.

R is not transitive: There is an arrow going from 1 to 0 and an arrow going from 0 to 3, but there is no arrow going from 1 to 3.

from the S relation diagraph



S is not reflexive: There is no loop at 0

S is not symmetric: There is an arrow from 0 to 1 but not from 1 to 0.

S is transitive: because the statement if ((x,y) ∈ S Λ (y, z)) ∉ S then (x, z) ∈ S. is always True in the case of relation S.

Ex 7:

What is the properties of Taller relation T?

Sol:

Let x, y and z be three person in a set A:

T is not reflexive because (x, x) ∉ T.

T is not symmetric because ∀ x, y ∈ A, xTy → yTx is false.

T is transitive because ∀ x, y, z ∈ A, (xTy ∧ yTz) → xTz is true.

7. Homework

Homework 1 What is the properties of Father relation F?

Homework 2 Let A ={0, 1, 2, 3} and relations R on A as follows:
R ={(0, 0), (0, 2), (0, 3), (2, 3)},
a. Is R reflexive? symmetric? transitive?
b. represent R by table, Arrow Diagram, and zero-one matrix methods.

Homework 3 List all the ordered pairs in the relation R ={(a, b) | a > b} on the set {1, 2, 3, 4, 5, 6}.

LECTURE 7

Introduction to Tree

December /2013

Introduction

There are many situations in which information has a hierarchical or nested structure like that found in family trees or organization charts. The abstraction that models hierarchical structure is called a tree and this data model is among the most fundamental in computer science. It is the model that underlies several programming language , including Lisp.

1. Lecture Outlines

- Definition of Tree.
- Tree Terminologies.
- An m-ary Tree.
- Ordered Rooted Tree.
- Tree Traversal.
- Homework.

2. Definition of Tree

A **tree** is an undirected connected graph with no simple circuits, no multiple edges, no loops. Therefore any tree must be a simple graph. An undirected graph is a tree if and only if there is a unique simple path between any two of its vertices.

Ex 1: Which of the graphs shown in Figure 1 are trees?

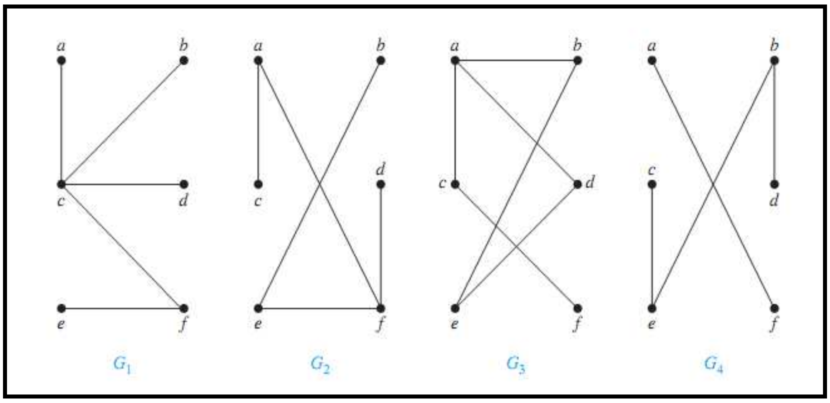


Figure1: Examples of Trees and Graphs That Are Not Trees.

Sol:

G1 and G2 are trees, because both are connected graphs with no simple circuits.

G3 is not a tree because e, b, a, d, e is a simple circuit in this graph.

G4 is not a tree because it is not connected.

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.

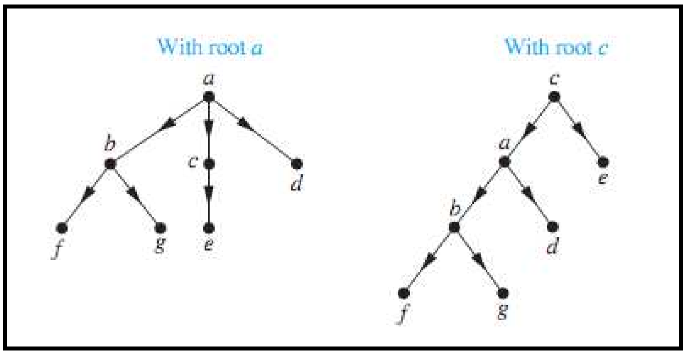


Figure 2: Examples of Rooted Trees.

3. Tree Terminologies

Suppose the following rooted tree. The common tree terminologies are as follows:

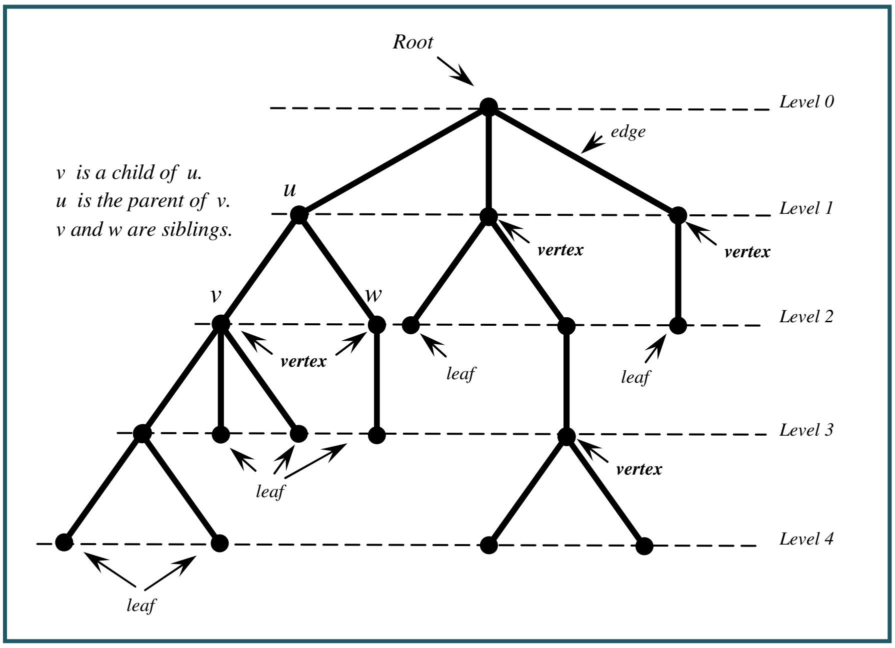


Figure 3: Tree

- **Parent:** the parent of v is the unique vertex u such that there is a directed edge from u to v.
- **Child:** When u is the parent of v, v is called a child of u.
- **Siblings:** siblings are vertices with the same parent.
- **Ancestors:** the ancestors of a vertex other than the root are the vertices in the path from the root to this vertex, excluding the vertex itself and including the root.
- **Descendants:** The descendants of a vertex v are those vertices that have v as an ancestor.
- **Leaf:** A vertex that has no children.
- **Internal Vertices:** they are the vertices that have children.
- **Level of a Vertex:** the level of vertex in a rooted tree is the length of the unique path from the root to this vertex. The level of the root is defined to be zero.
- **The Height of a Rooted Tree**: the height is the maximum level of vertices.

Ex: In the rooted tree T shown in the figure below, find the root, the parent of c, the children of g, the siblings of h, all ancestors of e, all descendants of b , all internal vertices, and all leaves.

Sol:

The root is a.

The parent of c is b.

The children of g are h, i, and j.

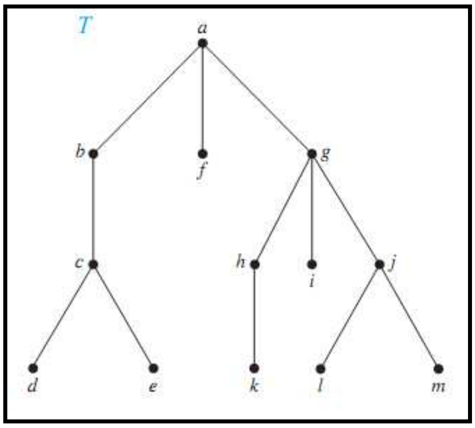
The siblings of h are i and j.

The ancestors of e are c, b, and a.

The descendants of b are c, d, and e.

The internal vertices are a, b, c, g, h, and j.

The leaves are d, e, f , i, k, l, and m.



4. An m-ary Tree

A rooted tree is called an m-ary tree if every internal vertex has no more than m children. The tree is called a full m-ary tree if every internal vertex has exactly m children.

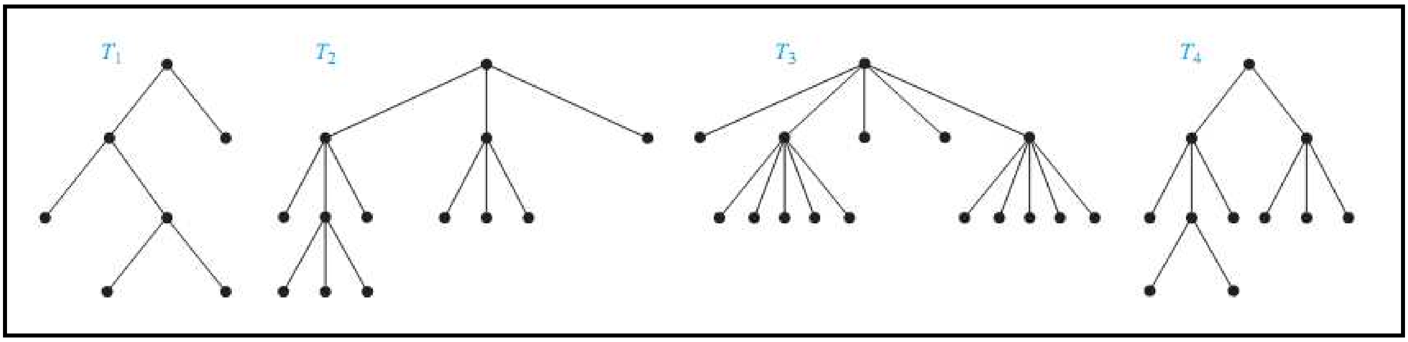


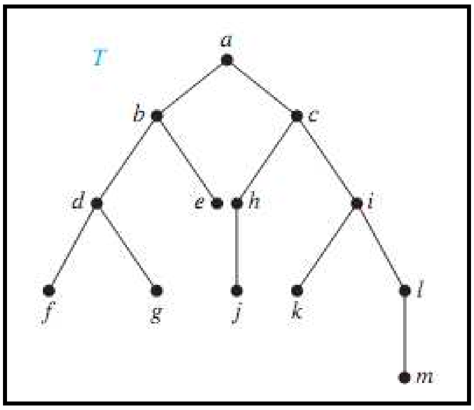
Figure 4: Examples on m-ary trees

From figure 4, we notice that it includes different m-ary trees as follows:

- T1: is a full 2-ary tree; because each of its internal vertices has two children (m=2).
- T2: is a full 3-ary tree; because each of its internal vertices has three children(m=3).
- T3: is a full 5-ary tree; because each internal vertex has five children (m=5).
- T4: is not a full m-ary tree for any m; because some of its internal vertices have two children and others have three children. it is just 3-ary tree.

5. Ordered rooted Tree

An ordered rooted tree is a rooted tree where the children of each internal vertex are ordered from left to right.



6. Tree Traversal

Tree Traversal is a procedures for visiting each vertex of an ordered rooted tree in order to process the data in that vertices and building a list of the results.

Ordered rooted trees can also be used to represent various types of expressions, such as arithmetic expressions involving numbers, variables, and operations.

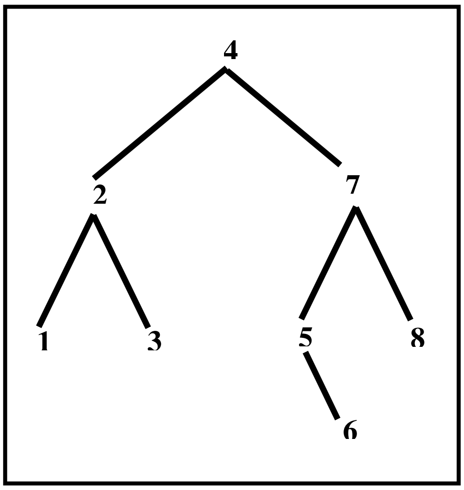
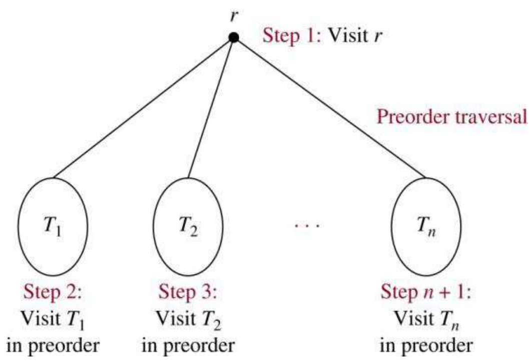


Figure 5: An ordered Tree

The three common traversal Algorithms are as follows:

- **Preorder traversal**:

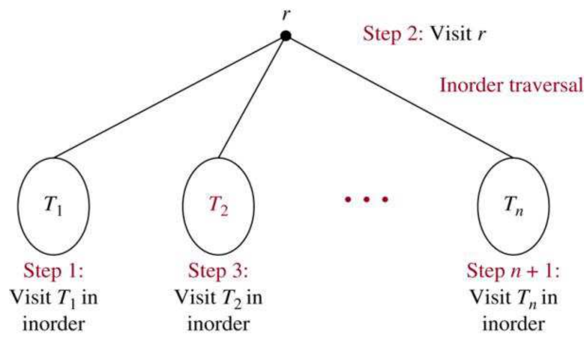
  In this algorithm, the root is firstly visited, then traverse the left subtree, then traverse the right subtree.



A preorder traversal of the tree in figure 5 is: 4, 2, 1, 3, 7, 5, 6, 8.

- **Inorder traversal:**

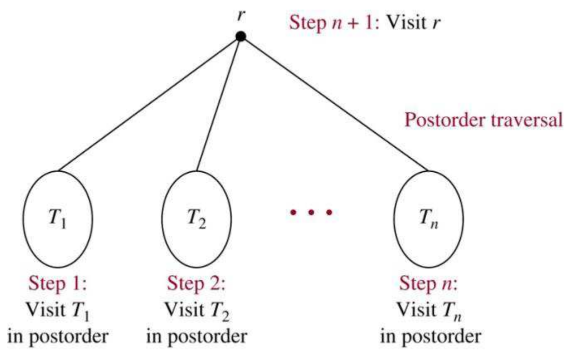
In this algorithm, the left subtree is firstly visited, then the root, then the right subtree.



An inorder traversal of the tree in the figure 5 is: 1, 2, 3, 4, 5, 6, 7, 8.

- **Postorder traversal:**

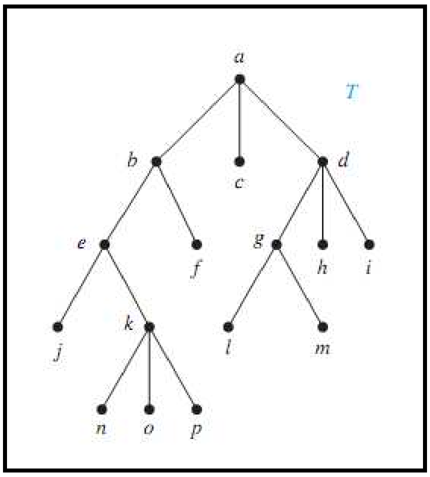
In this algorithm, traverse the left subtree, then the right subtree, and finally visit the root.



A postorder traversal of the tree in figure 5 is: 1, 3, 2, 6, 5, 8, 7, 4.

Tree Traversal
Converts hierarchical data into a linear sequence
Preorder: root, left, right
Inorder: left, root, right
Postorder: left, right, root

Ex: Show the preorder, inorder, postorder traversal for the tree shown below.



Sol:

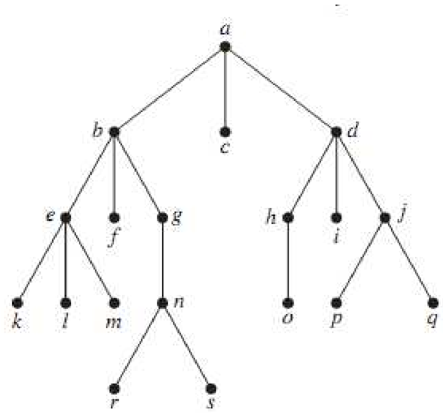
Preorder: a b e j k n o p f c d g l m h i.

Inorder: j e n k o p b f a c l g m d h i.

Postorder: j n o p k e f b c l m g h I d a.

7. Homework

Homework **1** For the following tree, Answer the following:



1. Show the preorder, inorder, postorder traversal.
2. What is the root?
3. What is the parent of m?
4. What are the children of j?
5. What are the siblings of h?
6. What are the ancestors of n?
7. What are the descendants of d?
8. What are the leaves?
9. What is the level of I vertex?
10. What is the height of the tree?
11. What is the type of tree?
12. Is the tree is ordered tree or not? Justify your answer.