SYLLABUS

Analysis and Design of Algorithms - 2150703

1. Basics of Algorithms and Mathematics (Chapter - 1)

What is an algorithm ?, Mathematics for algorithmic sets, Functions and relations, Vectors and matrices, Linear inequalities and linear equations.

2. Analysis of Algorithm (Chapter - 2)

The efficient algorithm, Average, best and worst case analysis, Amortized analysis, Asymptotic notations, Analyzing control statement, Loop invariant and the correctness of the algorithm, Sorting algorithms and analysis : Bubble sort, Selection sort, Insertion sort, Shell sort, Heap sort, Sorting in linear time : Bucket sort, Radix sort and Counting sort.

3. Divide and Conquer Algorithm (Chapter - 3)

Introduction, Recurrence and different methods to solve recurrence, Multiplying large integers problem, Problem solving using divide and conquer algorithm - Binary search, Max-Min problem, Sorting (Merge sort, Quick sort), Matrix multiplication, Exponential.

4. Dynamic Programming (Chapter - 4)

Introduction, The principle of optimality, Problem solving using dynamic programming - Calculating the binomial coefficient, Making change problem, Assembly line-scheduling, Knapsack problem, All points shortest path, Matrix chain multiplication, Longest common subsequence.

5. Greedy Algorithm (Chapter - 5)

General characteristics of greedy algorithms, Problem solving using Greedy algorithm

- Activity selection problem, Elements of greedy strategy, Minimum spanning trees (Kruskal's algorithm, Prim's algorithm), Graphs : Shortest paths, The knapsack problem, Job scheduling problem, Huffman code.

6. Exploring Graphs (Chapter - 6)

An introduction using graphs and games, Undirected graph, Directed graph, Traversing graphs, Depth first search, Breadth first search, Topological sort, Connected components.

7. Backtracking and Branch and Bound (Chapter - 7)

Introduction, The eight queens problem , Knapsack problem, Travelling salesman problem, Minimax principle.

8. String Matching (Chapter - 8)

Introduction, The naive string matching algorithm, The Rabin-Karp algorithm, String matching with finite automata, The Knuth-Morris-Pratt algorithm.

9. Introduction to NP-Completeness (Chapter - 9)

The class P and NP, Polynomial reduction, NP-completeness problem, NP-hard problems, Travelling salesman problem, Hamiltonian problem, Approximation algorithms.

1 Basics of Algorithms and Mathematics

Syllabus

What is an algorithm ?, Mathematics for algorithmic sets, Functions and relations, Vectors and matrices, Linear inequalities and linear equations.

Contents

1.1 Introduction

An algorithm, named for the ninth century Persian mathematician **al-Khowarizmi**, is simply a set of rules used to perform some calculations, either by hand or more usually on a machine. In this subject we will refer algorithm as a method that can be used by the computer for solution of a problem. For instance performing the addition, multiplication, division or subtraction is an algorithm. Use of algorithm for some computations is a common practice. Even ancient Greek has used an algorithm which is popularly known as Euclid's algorithm for calculating the greatest common divisor of two numbers.

Basically algorithm is a finite set of instructions that can be used to perform certain task. In this chapter we will learn some basic concepts of algorithm. We will start our discussion by understanding the notion of algorithm. And for understanding how to write an algorithm we will discuss some examples.

1.2 What is an Algorithm ?

GTU : Dec.-10, Winter-15, Marks 3

In this section we will first understand *"What is algorithm?"* and *"When it is required ?"*

Definition of algorithm : The algorithm is defined as a collection of unambiguous instructions occurring in some specific sequence and such an algorithm should produce output for given set of input in finite amount of time.

This definition of algorithm is represented in Fig. 1.2.1.

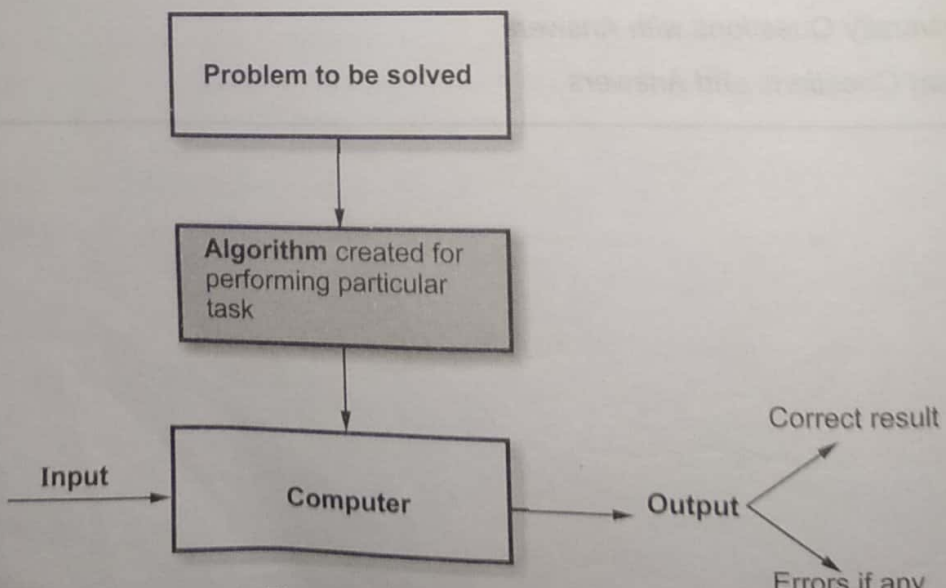


Fig. 1.2.1 Notion of algorithm

After understanding the problem statement we have to create an algorithm carefully for the given problem. The algorithm is then converted into some programming language and then given to some computing device (computer). The computer then executes this algorithm which is actually submitted in the form of source program. During the process of execution it requires certain set of input. With the help of algorithm (in the form of program) and input set, the result is produced as an output. If the given input is invalid then it should raise appropriate error message ; otherwise correct result will be produced as an output.

1.2.1 Properties of Algorithm

Simply writing the sequence of instructions as an algorithm is not sufficient to accomplish certain task. It is necessary to have following properties associated with an algorithm :

1. **Non-ambiguity :** Each step in an algorithm should be non-ambiguous. That means each instruction should be clear and precise. The instruction in an algorithm should not denote any conflicting meaning. This property also indicate the *effectiveness* of algorithm.
2. **Range of input :** The range of input should be specified. This is because normally the algorithm is input driven and if the range of the input is not been specified then algorithm can go in an infinite state.
3. **Multiplicity :** The same algorithm can be represented in several different ways. That means we can write in simple English the sequence of instructions or we can write it in the form of pseudo code. Similarly for solving the same problem we can write several different algorithms. For instance : for searching a number from the given list we can use sequential search or a binary search method. Here *"searching"* is a task and use of either a *"sequential search method"* or *"binary search method"* is an algorithm.
4. **Speed :** The algorithms are written using some specific ideas (which is popularly known as logic of algorithm). But such algorithms should be efficient and should produce the output with fast speed.
5. **Finiteness :** The algorithm should be finite. That means after performing required operations it should terminate.

1.2.2 Issues in Writing an Algorithm

There are various issues in the study of algorithms and those are :

1. How to devise algorithms ?

The creation of an algorithm is a logical activity and one cannot automate it. But there are certain **algorithmic design strategies** and using these strategies one can create many useful algorithms. Hence mastering of such design strategies is an important activity in study of design and analysis of algorithms.

2. How to validate algorithms ?

The next step after creation of algorithms is to validate algorithms. The process of checking whether an algorithm computes the correct answer for all possible legal inputs is called **algorithm validation**. The purpose of validation of algorithm is to find whether algorithm works properly without being dependant upon programming languages. Once validation of algorithm is done a program can be written using corresponding algorithm.

3. How to analyze algorithms ?

Analysis of algorithm is a task of determining how much **computing time** and **storage** is required by an algorithm. Analysis of algorithms is also called **performance analysis**. This analysis is based on mathematics. And a judgment is often needed about better algorithm when two algorithms get compared. The behaviour of algorithm in best case, worst case and average case needs to be obtained.

4. How to test a program ?

After finding an efficient algorithm it is necessary to test that the program written using the efficient algorithm behaves properly or not. Testing of a program is an activity that can be carried out in two phases - **i) debugging** and **ii) performance measuring** (profiling). While debugging a program, it is checked whether program produces faulty results for valid set of input, and if it is found then the program has to be corrected. Thus by debugging thoroughly the program is corrected.

Profiling is a process of measuring time and space required by a corrected program for valid set of inputs.

1.2.3 How to Write an Algorithm ?

Algorithm is basically a sequence of instructions written in simple English language. The algorithm is broadly divided into two sections -

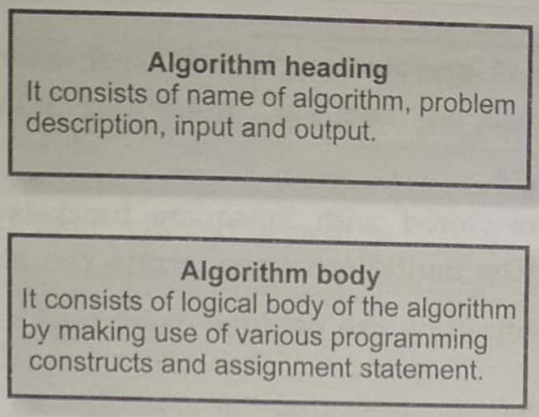
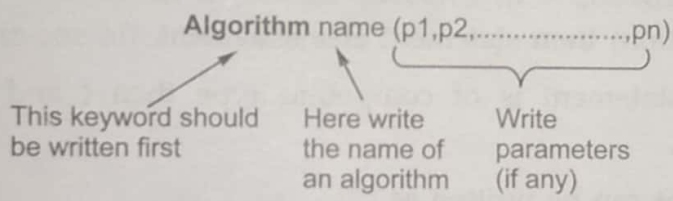


Fig. 1.2.2 Structure of algorithm

Let us understand some rules for writing the algorithm.

1. Algorithm is a procedure consisting of heading and body. The heading consists of keyword **Algorithm** and name of the algorithm and parameter list. The syntax is



2. Then in the heading section we should write following things :

```
//Problem Description :
//Input :
//Output :
```

3. Then body of an algorithm is written, in which various programming constructs like if, for, while or some assignment statements may be written.
4. The compound statements should be enclosed within { and } brackets.
5. Single line comments are written using // as beginning of comment.
6. The **identifier** should begin by letter and not by digit. An identifier can be a combination of alphanumeric string.

   It is not necessary to write data types explicitly for identifiers. It will be represented by the context itself. Basic data types used are integer, float, char, Boolean and so on. The pointer type is also used to point memory location. The compound data type such as structure or record can also be used.
7. Using assignment operator ← an assignment statement can be given.

   For instance :

```
Variable ← expression
```

8. There are other types of operators such as Boolean operators such as true or fals Logical operators such as **and**, **or**, **not**. And relational operators such a
<, <=, >, >=, =, ≠

9. The array indices are stored with in square brackets '[' ']'. The index of arra usually start at zero. The multidimensional arrays can also be used in algorithm.

10. The inputting and outputting can be done using **read** and **write**.

For example :

write("This message will be displayed on console") ;

read(val) ;

11. The conditional statements such as if-then or if-then-else are written in followin form :

if (condition) **then** statement

if (condition) **then** statement **else** statement

If the **if-then** statement is of compound type then { and } should be used f enclosing block.

12. **while** statement can be written as :

```
while (condition) do
{
    statement 1
    statement 2
        :
        :
    statement n
}
```

While the condition is true the block enclosed with { } gets executed otherwi statement after } will be executed.

13. The general form for writing for loop is :

```
for variable ← value1 to valuen do
{
    statement 1
    statement 2
        :
        :
    statement n
}
```

Here $\mathbf{value_1}$ is initialization condition and $\mathbf{value_n}$ is a terminating condition.

Sometime a keyword **step** is used to denote increment or decrement the value variable. **For example** :


```
for i←1 to n ste
{
        Wri
}
```

14. The **rep**

```
rep
unt
```

15. The **bre** to return contr

Note that s order as they a

Some Examp

Example 1

```
Algorithm sum
//Problem Desc
//sum of given
//Input : 1 to n
//Output : The
    result ←
    for i ← 1
        res
return result
```

Example 2

```
Algorithm eve
//Problem Des
//number is ev
//Input : the n
//Output : App
if (val%2=0) th
        wr
else
        wr
```

Example 3

```
Algorithm sor
//Problem Des
//Input : An ar
//is total numb
```

```
for i←1 to n step 1
{
        Write (i)
}
```

Here variable i is incremented by 1 at each iteration

14. The **repeat - until** statement can be written as :

```
repeat
        statement 1
        statement 2
            :
            :
        statement n
until (condition)
```

15. The **break** statement is used to exit from inner loop. The **return** statement is used to return control from one point to another. Generally used while exiting from function.

Note that statements in an algorithm executes in sequential order i.e. in the same order as they appear-one after the other.

Some Examples

Example 1 : Write an algorithm to count the sum of n numbers.

```
Algorithm sum (1, n)
//Problem Description : This algorithm is for finding the
//sum of given n numbers
//Input : 1 to n numbers
//Output : The sum of n numbers
    result ← 0
    for i ← 1 to n do i ← i+1
        result ← result+i
return result
```

Example 2 : Write an algorithm to check whether given number is even or odd.

```
Algorithm eventest (val)
//Problem Description : This algorithm test whether given
//number is even or odd
//Input : the number to be tested i.e. val
//Output : Appropriate messages indicating even or oddness
if (val%2=0) then
        write ("Given number is even")
else
        write("Given number is odd")
```

Example 3 : Write an algorithm for sorting the elements.

```
Algorithm sort (a,n)
//Problem Description : sorting the elements in ascending order
//Input : An array a in which the elements are stored and n
//is total number of elements in the array
```

```
//Output : The sorted array
for i ← 1 to n do
for j ← i+1 to n – 1 do
{
 if(a[i]>a[j]) then
 {
            temp ← a[i]
            a[i] ← a[j]
            a[j] ← temp
 }
}
write ("List is sorted")
```

Example 4 : Write an algorithm to find factorial of n number.

```
Algorithm fact (n)
//Problem Description : This algorithm finds the factorial
//of given number n
//Input : The number n of which the factorial is to be
//calculated.
//Output : factorial value of given n number.
if(n ← 1) then
            return 1
else
            return n*fact(n – 1)
```

Example 5 : Write an algorithm to perform multiplication of two matrices.

```
Algorithm Mul(A,B,n)
//Problem Description : This algorithm is for computing
//multiplication of two matrices
//Input : The two matrices A,B and order of them as n
//Output : The multiplication result will be in matrix C
for i ← 1 to n do
  for j ← 1 to n do
    C[i,j] ← 0
            for k ← 1 to n do
      C[i,j] ← C[i,j]+A[i,k]B[k,j]
```

Example for Practice

Example 1.2.1 : *Design recursive algorithm for computing 2^n for non-negative integer using the formula $2^n = 2^{n-1} + 2^{n-1}$*

Review Questions

1. *What is an algorithm ? Explain various properties of an algorithm.* **GTU : Dec.-10, Marks 3**
2. *Define the term - algorithm.* **GTU : Winter-15, Marks 2**

1.3 Designing of an Algorithm

GTU : May-12, Marks 6

In computer science, developing an algorithm is an art or a skill. And we can have mastery on algorithm development process only when we follow certain method. Before actual implementation of the program, designing an algorithm is very important step.

Suppose, if we want to build a house we do not directly start constructing the house. Instead we consult an architect, we put our ideas and suggestions, accordingly he draws a plan of the house, and he discusses it with us. If we have some suggestion, the architect notes it down and makes the necessary changes accordingly in the plan. This process continues till we are happy. Finally the blue print of house gets ready. Once design process is over actual construction activity starts. Now it becomes very easy and systematic for construction of desired house. In this example, you will find that all designing is just a paper work and at that instance if we want some changes to be done then those can be easily carried out on the paper. After a satisfactory design the construction activities start. Same is a program development process.

If we could follow same kind of approach while designing and analyzing the algorithm then we can have successful implementation for complex problems also. Let us list the **"What are the steps that need to be followed ?"** while designing an algorithm.

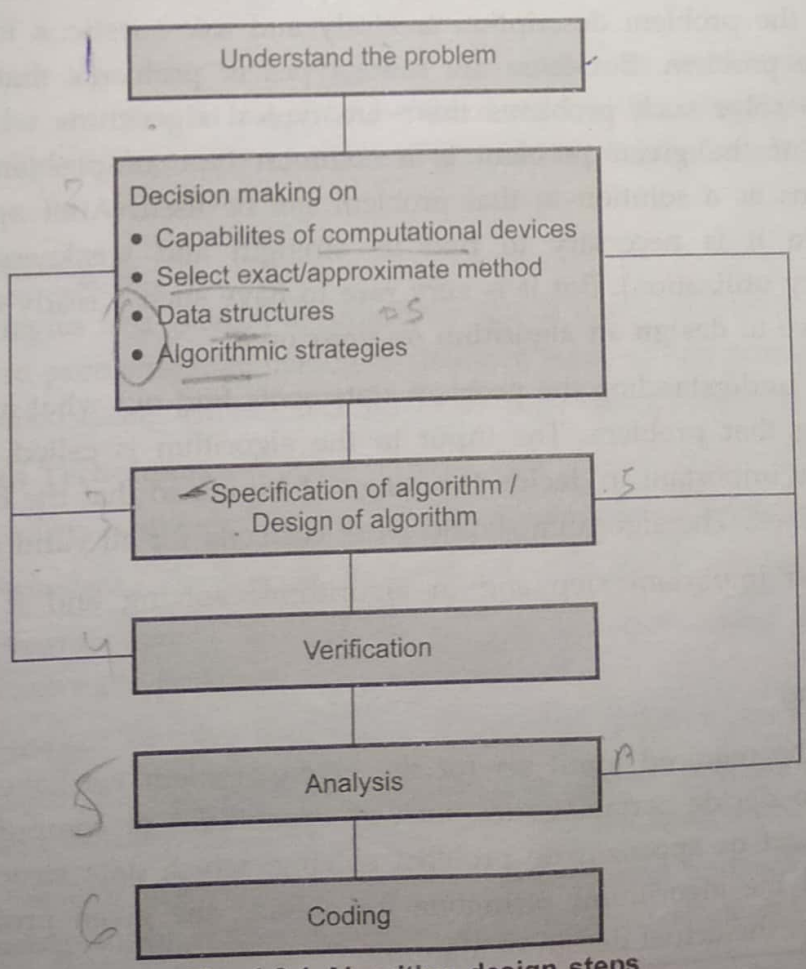


Fig. 1.3.1 Algorithm design steps

These steps are -

1. Understanding the problem.
2. Decision making on,
   a. Capabilities of computational devices
   b. Choice for either exact or approximate problem solving method.
   c. Data structures.
   d. Algorithmic strategies.
3. Specification of algorithm.
4. Algorithmic verification.
5. Analysis of algorithm.
6. Implementation or coding of algorithm.

Let us now discuss each step in detail

1. Understanding the problem

This is the very first step in designing of algorithm. In this step first of all you need to **understand the problem** statement completely. While understanding the problem statements, read the problem description carefully and ask questions for clarifying the doubts about the problem. But there are some types of problems that are commonly occurring and to solve such problems there are typical algorithms which are already available. Hence if the given problem is a common type of problem, then already existing algorithms as a solution to that problem can be used. After applying such an existing algorithm it is necessary to find its strength and weakness (For example, efficiency, memory utilization). But it is very rare to have such a ready-made algorithm. Normally you have to **design an algorithm on your own**.

After carefully understanding the problem statements find out what are the necessary inputs for solving that problem. The input to the algorithm is called **instance** of the problem. It is very important to **decide the range of inputs** so that the boundary values of algorithm get fixed. The algorithm should work correctly for all **valid inputs**.

This step is an important step and in algorithmic solving and it should not be skipped at all.

2. Decision making

After finding the required input set for the given problem we have to analyze the input and need to decide certain issues such as capabilities of computational devices, whether to use exact or approximate problem solving, which data structures has to be used, and to find the algorithmic technique for solving the given problem. This step serves as a base for the actual design of algorithm.

a. Capabilities of computational devices

It is necessary to know the computational capabilities of devices on which the algorithm will be running. Globally we can classify an algorithm from execution point of view as **sequential algorithm** and **parallel algorithm**. The sequential algorithm specifically runs on the machine in which the instructions are executed one after another. Such a machine is called as Random Access Machine (RAM). And the parallel algorithms are run on the machine in which the instructions are executed in parallel.

There are certain complex problems which require huge amount of memory or the problems for which execution time is an important factor. For solving such problems it is essential to have proper choice of a **computational device** which is **space and time efficient**.

b. Choice for either exact or approximate problem solving method

The next important decision is to decide whether the problem is to be solved exactly or approximately. If the problem needs to be solved correctly then we need **exact algorithm**. Otherwise if the problem is so complex that we won't get the exact solution then in that situation we need to choose **approximation algorithm.** The typical example of approximation algorithm is travelling Salesperson Problem.

c. Data structures

Data structure and algorithm work together and these are interdependent. Hence choice of proper data structure is required before designing the actual algorithm. The implementation of algorithm (program) is possible with the help of algorithm and data structure.

d. Algorithmic strategies

Algorithmic strategies is a general approach by which many problems can be solved algorithmically. These problems may belong to different areas of computing. Algorithmic strategies are also called as algorithmic techniques or algorithmic paradigm.

Algorithm Design Techniques -

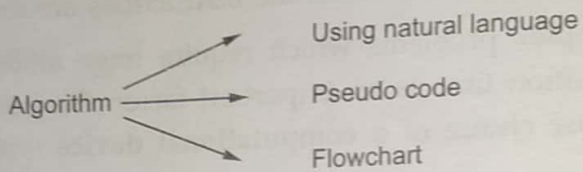
- ***Brute force :*** This is straightforward technique with naive approach.
- ***Divide-and-conquer*** **:** The problem is divided into smaller instances.
- ***Dynamic programming :*** The results of smaller, reoccurring instances are obtained to solve the problem.
- ***Greedy technique :*** To solve the problem locally optimal decisions are made.
- ***Back tracking :*** This method is based on the trial and error. If we want to solve some problem then desired solution is chosen from the finite set S.

In this subject, we will discuss these algorithmic strategies and see how to solve certain problems using these strategies. Various algorithmic strategies are classified

according the design idea that the particular algorithm adopts. And if using certa design idea the particular problem is getting solved then that problem belongs corresponding algorithmic strategy.

3. Specification of algorithm

There are various ways by which we can specify an algorithm



It is very simple to specify an algorithm using **natural language**. But many tim specification of algorithm by using natural language is not clear, and there by we g brief specification.

For example : *write an algorithm to perform addition of two numbers.*

Step 1 : Read the first number say a.

Step 2 : Read the second number say b.

Step 3 : Add the two numbers and store the result in a variable c.

Step 4 : Display the result.

Such a specification creates difficulty while actually implementing it. Hence ma programmers prefer to have specification of algorithm by means of **pseudo code**.

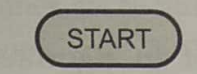
Pseudo code is nothing but a combination of natural language and programmi language constructs. A pseudo code is usually more precise than a natural language.

For example : *Write an algorithm for performing addition of two numbers.*

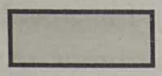
```
Algorithm sum(a,b)
//Problem Description This algorithm performs addition of
//two integers
//Input : two integers a and b
//Output : addition of two integers
c ← a+b
write (c)
```

This specification is more useful from implementation point of view.

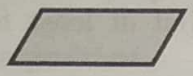
Another way of representing the algorithm is by **flowchart.** Flowchart is a graphi representation of an algorithm. Typical symbols used in flowchart are -



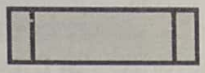
Start state



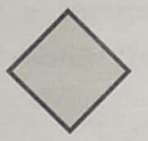
Transition



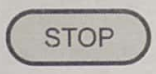
Processing or assignment statements



Input-output statement

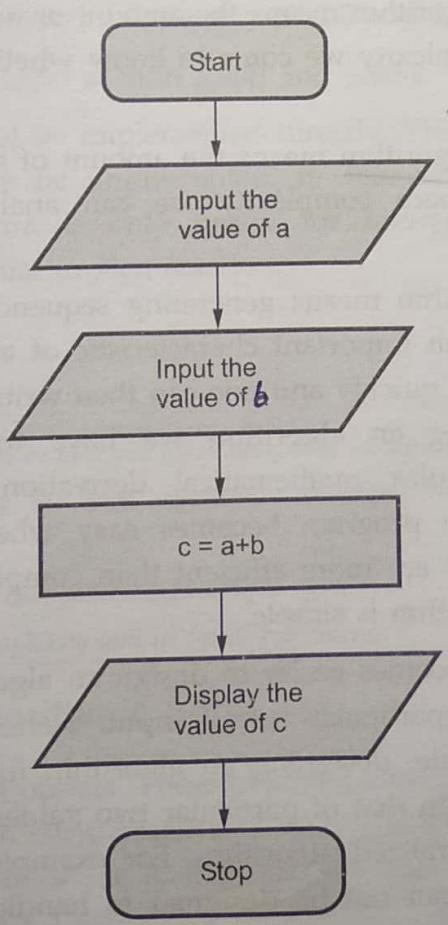


Conditional statement



Stop state

For example



4. Algorithmic verification

Algorithmic verification means checking correctness of an algorithm. After specifying an algorithm we go for checking its correctness. We normally check whether the algorithm gives correct output in finite amount of time for a valid set of input. The proof of correctness of an algorithm can be complex sometimes. A common method of proving the correctness of an algorithm is by using mathematical induction. But to show that an algorithm works incorrectly we have to show that at least for one instance of valid input the algorithm gives wrong result.

5. Analysis of algorithm

While analyzing an algorithm we should consider following factors -

- Time efficiency of an algorithm
- Space efficiency of an algorithm
- Simplicity of an algorithm
- Generality of an algorithm
- Range of input.

Time complexity of an algorithm means the amount of time taken by an algorithm to run. By computing time complexity we come to know whether the algorithm is slow or fast.

Space complexity of an algorithm means the amount of space (memory) taken by an algorithm. By computing space complexity we can analyze whether an algorithm requires more or less space.

Simplicity is of an algorithm means generating sequence of instructions which are easy to understand. This is an important characteristic of an algorithm because simple algorithms can be understood quickly and one can then write simpler programs for such algorithms. While simplifying an algorithm we have to compute any predefined computations or some complex mathematical derivation. Finding out bugs from algorithms or debugging the program becomes easy when an algorithm is simple. Sometimes simpler algorithms are more efficient than complex algorithms. But it is not always possible that the algorithm is simple.

Generality sometimes it becomes easier to design an algorithm in more general way rather than designing it for particular set of input. Hence we should write general algorithms always. For example, designing an algorithm for finding GCD of any two numbers is more appealing than that of particular two values. But sometimes it is not at all required to design a generalized algorithm. For example, an algorithm for finding roots of quadratic equations can not be designed to handle a polynomial of arbitrary degree.

Range of inputs comes in picture when we execute an algorithm. The design of an algorithm should be such that it should handle the range of input which is the most natural to corresponding problem.

Analysis of algorithm means checking the characteristics such as : time complexity, space complexity, simplicity, generality and range of input. If these factors are not satisfactory then we must redesign the algorithm.

6. Implementation of algorithm

The implementation of an algorithm is done by suitable programming language. For example, if an algorithm consists of objects and related methods then it will be better to implement such algorithm using some object oriented programming language like C++ or JAVA. While writing a program for given algorithm it is essential to write an optimized code. This will reduce the burden on compiler.

Example 1.3.1 *Define an algorithm. What features does one look for in a good computer algorithm ? When an algorithm is said to be correct ? When an algorithm is said to efficient ? Can an abstract algorithm be directly implemented ? If yes, how ? If not, what is its use ?* **GTU : May-12, Marks 6**

Solution : Algorithm and its features : Refer section 1.2 and 1.2.1.

Correctness and efficiency : Refer section 1.3(4) and 1.3(5).

An abstract algorithm can not be implemented directly. This kind of algorithm gives the abstract view of what is to be implemented. It does not specify how to do the things. Hence abstract algorithm is only useful for knowing the structure of the implementation, and not the actual implementation.

Review Questions

1. *What are the steps that need to be followed while designing an algorithm ?*
2. *What do you understand by the term algorithmic strategies ? Name some commonly used strategies.*
3. *Explain the concept of pseudo code with some example.*
4. *Define the terms - i) Time complexity and ii) Space complexity.*

1.4 Mathematics for Algorithmic Sets

Set is defined as collection of objects. These objects are called **elements of the set**. All the elements are enclosed within curly brackets '{' and '}' and every element is separated by commas. If 'a' is an element of set A then we say that $a \in A$ and if 'a' is not an element of A then we say that $a \notin A$.

Typically set is denoted by a **capital letter**. The set can be represented using three methods.

1) Listing method

The elements are listed in the set.

For example : A set of element which are less than 5. Then,

$$A = \{0,1, 2, 3, 4\}$$

2) Describing properties

The properties of elements of a set define the set.

For example : A set of vowels. Hence here vowel is a property of the set which defines the set as :

$$A = \{a, e, i, o, u\}$$

3) Recursion method

The recursion occurs to define the elements of the set.

For example : $A = \{ x | x \text{ is square of n}\}$ where $n \leq 10$.

This defines the set as :

$$A = \{0,1, 4, 9, 16,..... 100\}$$

Subset : The subset A is called subset of set B if every element of set A is present in set B but reverse is not true. It is denoted by $A \subseteq B$. For example $A = \{1, 2, 3\}$ and $B = \{ 1, 2, 3, 4, 5 \}$ then $A \subseteq B$.

Empty set : The set having no element in it is called empty set. It is denoted by $A = \{ \}$ and it can be written as ϕ(phi).

Null string : The null element is denoted by ε or $\wedge$ character. Null element means no value character. But ε does mean ϕ.

Power set : The power set is a set of all the subsets of its elements.

For example : $A = \{1, 2, 3\}$

Then power set : $Q = \{\phi, \{1\}, \{2\}, \{3\}, \{1,2\}, \{1,3\}, \{3,2\}, \{2,3\}, \{1,2,3\}\}$

The number of elements are always equal to 2^n where n is number of elements in original set. As in set A there are 3 elements. Therefore in power set Q there are $2^3 = 8$ elements.

Equal set : The two sets are said to be equal (A = B) if $A \subseteq B$ and $B \subseteq A$ i.e. every element of set A is an element of B and every element of B is an element of A.

For example :

A = {1, 2, 3} and B = {1, 2, 3} then A = B.

|A| denotes the length of set A. i.e. number of elements in set A.

For example : If

A = {1, 2, 3, 4, 5} then

|A| = 5

1.4.1 Operations on Set

Various operations that can be carried out on set are -

i) Union ii) Intersection

iii) Difference iv) Complement.

i) $A \cup B$ is union operation - If

A = {1, 2, 3} B = {1, 2, 4} then

$A \cup B$ = {1, 2, 3, 4} i.e. combination of both the sets.

ii) $A \cap B$ is intersection operation - If

A = {1, 2, 3} and B = {2, 3, 4} then

$A \cap B$ = {2, 3} i.e. collection of common elements from both the sets.

iii) A – B is the difference operation - If

A = {1, 2, 3} and B = {2, 3, 4} then

A – B = {1} i.e. elements which are there in set A but not in set B.

iv) $\overline{A}$ is a complement operation - If

$\overline{A}$ = U – A where U is a universal set.

For example :

If U = {10, 20, 30, 40, 50}

A = {10, 20}

then $\overline{A}$ = U – A

= {30, 40, 50}

1.4.2 Cartesian Product of Two Sets

The cartesian product of two sets A and B is a set of all possible ordered pairs whose first component is member of A and whose second component is member of B. The cartesian product is denoted by $A \times B$.

$$\therefore \quad A \times B = \{\{a, b\} \mid a \in A \text{ and } b \in B\}$$

For example : Let $A = \{a, b\}$ and $B = \{0, 1, 2\}$

then the cartesian product of A and B is,

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

1.4.3 Cardinality of Sets

The cardinality of the set is nothing but the number of members in the set.

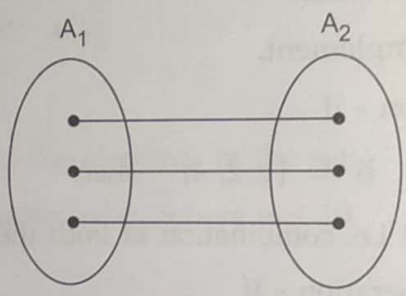


Fig. 1.4.1 Cardinality of two sets

These sets A_1 and A_2 have the same cardinality as there is one to one mapping of the elements of A_1 and A_2. The different cardinalities of set can be - one to one, one to many, many to one, many to many.

1.4.4 Sequence

The sequence means ordering of the elements of the set in some specific manner. For instance : {0, 2, 4, 6, 8} is a sequence which denotes the even numbers.

1.4.5 Tuple

An ordered pair of n elements is called tuple.

For example :

{10, 20} is a 2-tuple of pair

{10, 20, 30} is 3-tuple.

Review Questions

1. *Define the terms - i) Subset ii) Empty set iii) Power set iv) Tuple.*
2. *Explain four operations that can be performed on set.*
3. *What is cardinality of set ?*

1.5 Functions and Relations

GTU : Dec.-11, May-11, Winter-15, Marks 3

1.5.1 Functions

Function can be defined as the relationship between two sets. That means using function we can map one element of one set to some other element of another set. A function is a relation but a relation is not necessarily a function.

A function can be denoted using a letter f. If $f(x) = x^3$ then we say that f of x equals to x cube, then we can have,

$$f(2) = 8$$
$$f(5) = 125$$
$$f(-1) = -1$$

and so on.

1.5.1.1 Basic Terminologies

IF D is a set then we can define function as,

$$f(D) = \{ f(x) \mid x \in D \}$$

If we map some element to some other element then it can be denoted as ,

$$f : D \rightarrow R \qquad \text{i.e.} \qquad x \rightarrow x^3$$

If set D consists of all the real number then

D = {1, 2, ... } is a **domain**.

The functions are denoted in terms of its mappings.

For example Let, D = {1, 2, 3, 4} and $f(x) = x^3$.

Then the range of f will be R = { f (1), f (2), f (3), f (4) }

= { 1, 8, 27, 64 }

If we take Cartesian product of D and R then we obtain

F = { (1, 1), (2, 8), (3, 27), (4, 64) }

Thus a function can be represented using Cartesian product of its domain and range.

Example 1.5.1 *Find the domain and range of the following relation. Is this relation a function*

$\{(2, 9), (3, 14), (4, 21)\}$

Solution : In above given set a relationship between certain x's and y's is a relation. Hence all the x values denote the domain and all the y values denote range. Therefore -

Domain : $\{ 2, 3, 4 \}$

Range : $\{ 9, 14, 21 \}$

We can observe the x and y pair to get the relationship. Note that, if $x = 2$ then $y = x^2 + 5 = (2)^2 + 5 = 9$. This holds true for all the x and y pair in given set. Hence we can define a relation as a function as $f(x) = x^2 + 5$.

Example 1.5.2 *Determine the domain of the given function :*

$$y = \frac{x^2 + x - 5}{x^2 + 3x - 10}$$

Solution : Here domain means all the values of x that are allowed to take. For this function, the only care that has to be taken is that, the function should not produce divide by zero error. If the denominator equals to zero, then divide by zero error will occur, hence

$$x^2 + 3x - 10 = 0$$

$$(x+5)\ (x-2) = 0$$

$$\therefore \quad x = -5 \quad \text{or} \quad x = 2$$

Hence **domain will be all those values of x that are not equal to – 5 or 2.**

- **Quantifiers :**

Quantifiers - In predicate logic the quantifier is a kind of operator which is used to determine the quantity.

There are two types of quantifiers -

Universal quantifier and **extential quantifier.**

Universal Quantifier - Any quantifier that starts with "$\forall$" is **universal quantifier.** $\forall x$ means **for all x.** This quantifier is used as a prefix to a predicate.

For example : $\forall x\, Bx$ means for all x, Bx.

For example : "All girls are intelligent" could be transformed into the propositional form as $\forall x\ B(x)$ where

- B (x) is the predicate denoting x is intelligent.
- The universe of discourse is only populated by girls.

Extential Quantifier - Any quantifier that starts with "$\exists$" is an **extential quantifier.**

$\exists x$ means **there exists an x such that ...**

This quantifier is used as a prefix to a predicate.

For example : " Some cars are red in color" could be transformed into the proportional form an $\exists x B(x)$ where

- B (x) is the predicate denoting x is red in colour
- The universe of discourse is populated by cars of various colours.

1.5.2 Relations

Relationship is a major aspect between two objects, even this is true in our real life. One object can be related with the other object by a 'mother of' relation. Then those two objects form a pair based on this certain relationship.

Definition : The relation R is a collection for the set S which represents the pair of elements.

For example : (*a*, b) is in R. We can represent their relation as ***a* R b**. The first component of each pair is chosen from a set called domain and second component of each pair is chosen from a set called range.

Properties of Relations

A relation R on set S is,

1. Reflexive if iRi for all i in S.
2. Irreflexive if iRi is false for all i in S.
3. Transitive if iRj and jRk imply iRk.
4. Symmetric if iRj implies jRi.
5. Asymmetric if iRj implies that jRi is false.

Every asymmetric relation must be irreflexive.

For example : If A = { *a*, b } then

Reflexive relation R can be = { (*a*, *a*), (b, b) }

Irreflexive relation R can be = { (*a*, b) }

Transitive relation R can be = { (*a*, b), (b, *a*), (*a*, *a*) }

Symmetric relation R can be = { (*a*, b), (b, *a*) }

Asymmetric relation R can be = { (*a*, b) }

Equivalent Relation

A relation is said to be equivalence relation if it is **reflexive, symmetric and transitive**, over some set S.

Suppose R is a set of relations and S is the set of elements.

For example : S is the set of lines in a plane and R is the relation of lines intersecting to each other.

Example 1.5.3 *Determine whether R is equivalence relation or not where*

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

Solution : The R is reflexive because (0, 0), (1, 1), (2, 2) ∈ R.

Where as R is not symmetric because (0, 1) is not in R whereas (1, 0) is in R.

Hence the R is not a equivalence relation.

Closures of Relations

Sometimes when the relation R is given, it may not be reflexive or transitive.

By adding some pairs we make the relation as reflexive or transitive.

For example : Let { (a, b), (b, c), (a, a), (b, b) }

Now this relation is not transitive because (a, b), (b, c) is there in relation R but (a, c) is not there in R so we can make the transitive closure as

{ (a, a), (b, b), (a, b), (b, c), (a, c) }

We can even define reflexive closure and symmetric closure in the same way.

Review Questions

1. *What is relation ? Explain equivalence relation.* **GTU : Dec.-11, Marks 3**
2. *Explain the term quantifiers.* **GTU : May-11, Winter-15, Marks 2**
3. *Explain the concept of domain and range.*
4. *Give various properties of relations.*

1.6 Vectors

A vector A is a collection of n tuples.

For example $A = (a_1, a_2, a_3, \ldots, a_n)$

where a_i are called the **components** of A.

Equal vectors :

If there exists two vectors namely, A and B and if both the vector contain same number of components and if corresponding components are equal then vector A = B i.e. vector A and B are equal.

Zero vector :

Let $A = (a_1, a_2, a$
vector.

Addition of two vecto

For addition of tw
he same. The sum
rom A and B.

$A + B = (a$

$= (a$

Multiplication of vect

By multiplying eac
Let k be a scalar th

$k \cdot A = k$

$= k$

milarly, $-A = (-$

lso, $k(A + B) = k$

ot product :

The dot product
$= (b_1, b_2, b_3 \ldots, b_n$

$A \cdot B = a$

ngth of a vector :

A length or norm

$|| A || = -$

.7 Matrices

In mathematics, ar
For Example :

$A =$

Zero vector :

Let $A = (a_1, a_2, a_3, ..., a_n)$ be a vector and if $a_i = 0$ then vector A is called **Zero vector.**

Addition of two vectors :

For addition of two vectors, the number of components in both the vectores must be the same. The sum A + B is a vector obtained by adding corresponding components from A and B.

$$A + B = (a_1, a_2, ..., a_n) + (b_1, b_2, b_3 ...., b_n)$$
$$= (a_1 + b_2, a_2 + b_2, a_3 + b_3, ..., a_n + b_n)$$

Multiplication of vector by scalar :

By multiplying each component of vector by a scalar the product is obtained.

Let k be a scalar then the product can be

$$k \cdot A = k(a_1, a_2, ... a_n)$$
$$= k_{a_1}, k_{a_2} .... k_{a_n}$$

Similarly, $-A = (-1)A$ and $A - B = A + (-B)$

Also, $k(A + B) = kA + kB$ where A and B are vectors.

Dot product :

The dot product or inner product of vectors $A = (a_1, a_2, a_3 ...., a_n)$ and $B = (b_1, b_2, b_3 ...., b_n)$ is denoted by A·B. It can be defined as follows -

$$A \cdot B = a_1 b_1 + a_2 b_2 + a_3 b_3 + ...+ a_n b_n$$

Length of a vector :

A length or norm of the vector, A is denoted by $|| A ||$. It is

$$|| A || = \frac{\sqrt{A \cdot A}}{\sqrt{a_1^2 + a_2^2 + ... + a_n^2}}$$

1.7 Matrices

In mathematics, an **array** is rectangular representation of numbers.

For Example :

$$A = \begin{bmatrix} 11 & 12 & 13 \\ 8 & 7 & 5 \\ 15 & 20 & 25 \end{bmatrix}$$

Alternate representation is with the help of parenthesis instead of box brackets.

$$A = \begin{pmatrix} 11 & 12 & 13 \\ 8 & 7 & 5 \\ 15 & 20 & 25 \end{pmatrix}$$

The horizontal lines in a matrix are called **rows** and vertical lines are calle **columns.** The number in the matrix are called **entries** or **elements** of matrix.

The **size** of the matrix can be specified with m rows and n columns. It is denoted a **m × n** or **m-by-n**, where m and n are called its dimensions.

Row vector and column vector

A matrix with one row is called **row vector** and matrix with one column is calle **column vector.**

$$[10 \quad 20 \quad 30]$$

Row vector

$$\begin{bmatrix} 11 \\ 12 \\ 13 \end{bmatrix}$$

Column vector

1.7.1 Basic Terminologies

1) Zero Matrix :

A zero matrix is a matrix with all its entries being zero.

For example : $0_{2,2} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$

2) Identify Matrix :

The identity matrix or unit matrix of size n is a square matrix having one's on main diagonal, and zero's elsewhere.

The identity matrix is denoted by I.

For example :

$$I_1 = [1], \; I_2 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, I_3 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

If we multiply any matrix A by unit matrix then the resultant matrix is the sa matrix A.

3) Square Matrix

If the number that the matrix is

1.7.2 Operatio

Various operat

1. Addition

1) Addition of two

Let, A and B matrices will be ad

For example :

A =

B =

A + B =

) Multiplication of

For multiplicatio atrix. That means ze. Then the size

For example :

A =

B =

The multiplicatio

3) Square Matrix :

If the number of rows and number of columns of any matrix are same then we say that the matrix is square matrix.

1.7.2 Operations on Matrix

Various operations that can be performed on matrix are -

1. Addition 2. Multiplication 3. Transpose

1) Addition of two matrices

Let, A and B are the two matrices of same size. Then the addition of these two matrices will be addition of each corresponding element.

For example :

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$$

$$B = \begin{bmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \end{bmatrix}$$

$$A + B = \begin{bmatrix} a_{11} + b_{11} & a_{12} + b_{12} & a_{13} + b_{13} \\ a_{21} + b_{21} & a_{22} + b_{22} & a_{23} + b_{23} \\ a_{31} + b_{31} & a_{32} + b_{32} & a_{33} + b_{33} \end{bmatrix}$$

2) Multiplication of two matrices

For multiplication of two matrices, width of first matrix equals to height of second matrix. That means a matrix A with $\mathbf{m \times n}$ can be multiplied with matrix B having $\mathbf{n \times p}$ size. Then the size of resultant matrix is $\mathbf{m \times p}$.

For example :

$$A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix}_{3\times2}$$

$$B = \begin{bmatrix} 10 & 11 & 12 \\ 13 & 14 & 15 \end{bmatrix}_{2\times3}$$

The multiplication $A \times B$ can be done by multiplying a row by a column.

$$A \times B = \begin{bmatrix} 1*10+2*13 & 1*11+2*14 & 1*12+2*15 \\ 3*10+4*13 & 3*11+4*14 & 3*12+4*15 \\ 5*10+6*13 & 5*11+6*14 & 5*12+6*15 \end{bmatrix}$$

$$A \times B = \begin{bmatrix} 36 & 39 & 42 \\ 82 & 89 & 96 \\ 128 & 139 & 150 \end{bmatrix}$$

Note that the size of resultant matrix is $3 \times \boxed{2 \times 2} \times 3 = 3 \times 3$

Properties of matrix multiplication

Let, A, B and C are the matrices and k is the scalar then,

1. $A(B + C) = AB + AC$
2. $(B + C)A = BA + CA$
3. $(AB)C = A(BC)$
4. $k(AB) = A(kB)$

3) Transpose of matrix

The transpose of matrix A is obtained by interchanging row and column. transposed matrix is denoted by A^T. If matrix A is of size $m \times n$ then A^T is $n \times m$.

For example :

$$\text{If } A = \begin{bmatrix} 10 & 20 \\ 30 & 40 \\ 50 & 60 \end{bmatrix} \text{ then,}$$

$$A^T = \begin{bmatrix} 10 & 30 & 50 \\ 20 & 40 & 60 \end{bmatrix}$$

1.7.3 Determinant of Matrix

The determinant of matrix A is a specific number. It is denoted by $|A|$.

- The determinant of order one matrix is

  $|a_{11}| = a_{11}$

- The determinant of order two matrix is

$$\begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix} = a_{11}a_{22} - a_{12}a_{21}$$

- The determinant of order three matrix is

$$\begin{vmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{vmatrix}$$

$= a_{11}(a_{22}a_{33} - a_{23}a_{32}) - a_{12}(a_{21}a_{33} - a_{23}a_{31}) + a_{13}(a_{21}a_{32} - a_{22}a_{31})$

Following is an important property of determinant function

For any two n square matrices A and B

$$\det(AB) = \det(A) \cdot \det(B)"$$

Review Question

1. *Explain the method of obtaining the determinant of matrix.*

1.8 Linear Inequalities

In mathematics, linear inequality is a statement containing <, >, <= or >=

For example : $x + y > 14$ is a linear inequality.

1.8.1 Solution to Linear Inequality

The solution of linear inequality is based on number of variables.

The solution to one variable inequality :

$3x + 7 \leq 10$, the value of x gives true statement. For instance : $x = 1$ is a solution to the above given inequality.

The solution to two variable inequality :

$3x + 7y - 5 \leq 17$, the value of x and y should be such that the statement remains true. The solution to above inequality is (2, 2) because $3(2) + 7(2) - 5 = 15 \leq 17$.

The solution to three variable inequality :

$3x + 2y - 7z \leq 5$, the value of x, y and z should be such that the statement remains true. The solution to above inequality is (2, 2, 1). because $3(2) + 2(2) - 7(1) = 3 \leq 5$.

Key Point *Solution of an inequality is a solution that satisfies the inequality.*

1.8.2 Properties of Inequalities

1. If $x \leq y$ and $y \leq z$ then $x \leq z$.

2. If $x \leq y$ and c is some positive number then $x+c \leq y+c$.
3. If $x \leq y$ and c is negative, then $xc \geq y c$.
4. If $x \leq y$ and c is some positive, then $x+c \leq y+c$.

1.9 Linear Equations

GTU : Dec.-11, Marks 4

The linear equation is an equation containing n unknowns. A linear equation with one unknown can be given in standard form :

$$ax = b$$

Where x is unknown and a and b are constants. The solution of such equation will be,

$$x = \frac{b}{a}$$

The linear equation with two unknowns can be given in standard form as

$$ax + by = c.$$

Where x, y are unknowns and a, b and c are constants.

Example 1.9.1 *Solve $2x + 8 = 20$ for x.*

Solution :

$$2x + 8 = 20 \qquad \text{(Subtract 8 from both sides)}$$

$$2x + 8 - 8 = 20 - 8$$

$$\therefore \quad 2x = 12$$

$$\frac{2x}{2} = \frac{12}{2}$$

$$x = 6$$

Example 1.9.2 *If a number is increased by 9, the result is 25. Find the number.*

Solution : Assume that m be that number then we can write linear equation as -

$$m+9 = 25 \qquad \text{(subtract 9 from both sides)}$$

$$m+9-9 = 25-9$$

$$m = 16$$

Hence the number is **16**.

1.9.1 Two Linear Equations with Two Unknowns

The two linear equations with two unknowns is in following standard form :

$$a_1 x + b_1 y =$$

$$a_2 x + b_2 y =$$

To solve such consider following

Step 1 : Multipl coefficients of at le

Step 2 : Perform

For example :

$$5x+10y =$$

$$3x+2y =$$

Multiply equation

$$15x+30y =$$

$$-15x-10y =$$

Perform addition

$$20y =$$

$$y =$$

Now put $y = 1$ in

$$3(x)+2(1) =$$

$$3x =$$

$$x =$$

Thus we get a sol

1.9.2 Gauss E

Gaussian elimi augmented matri backward substitu

Example 1.9.3 *Sol*

$$b + c = 2$$

$$2a + 3c = 1$$

$$a + b + c = 3$$

Solution : We wil

$$a_1 x + b_1 y = c_1$$

$$a_2 x + b_2 y = c_2$$

To solve such linear equations in order to obtain solution to unknowns, we will consider following **algorithm -**

Step 1 : Multiply the two equation by any two numbers such that resulting coefficients of at least one term will be negative of each other.

Step 2 : Perform additions of two equations.

For example :

$$5x + 10y = 15 \quad (1.9.1)$$

$$3x + 2y = 5 \quad (1.9.2)$$

Multiply equation (1.9.1) by (1.9.3) and equation (1.9.2) by -5

$$15x + 30y = 45 \quad (1.9.3)$$

$$-15x - 10y = -25 \quad (1.9.4)$$

Perform addition of equations (1.9.3) and (1.9.4),

$$20y = 20$$

$$y = 1.$$

Now put $y = 1$ in equation (1.9.2), we will get,

$$3(x) + 2(1) = 5$$

$$3x = 3$$

$$x = 1$$

Thus we get a solution $x = 1$, $y = 1$ which is the unique solution.

1.9.2 Gauss Elimination Method

Gaussian elimination method is a method of solving linear system $Ax = b$ by bringing augmented matrix, to an upper triangular form and then obtaining a solution by backward substitution method.

Example 1.9.3 *Solve the linear equation by Gauss elimination method.*

$$b + c = 2$$

$$2a + 3c = 15$$

$$a + b + c = 3$$

Solution : We will write augmented matrix as :

$$\begin{bmatrix} 0 & 1 & 1 & 2 \\ 2 & 0 & 3 & 15 \\ 1 & 1 & 1 & 3 \end{bmatrix}$$

Step 1 : Now interchange 1st and 2nd equation.

$$\begin{array}{rcl} 2a + 3c = 15 \\ b + c = 2 \\ a + b + c = 3 \end{array} \Rightarrow \begin{bmatrix} 2 & 0 & 3 & 15 \\ 0 & 1 & 1 & 2 \\ 1 & 1 & 1 & 3 \end{bmatrix}$$

Step 2 : Divide first equation by 2.

$$\begin{array}{rcl} a + \frac{3}{2}c = \frac{15}{2} \\ b + c = 2 \\ a + b + c = 3 \end{array} \Rightarrow \begin{bmatrix} 1 & 0 & \frac{3}{2} & \frac{15}{2} \\ 0 & 1 & 1 & 2 \\ 1 & 1 & 1 & 3 \end{bmatrix}$$

Step 3 : Multiply first equation by −1 and add it to third equation.

$$-a - \frac{3}{2}c = \frac{-15}{2}$$

$$+ \quad a + b + c = 3$$

$$b - \frac{1}{2}c = \frac{-9}{2}$$ is the third equation

To summerize

$$\begin{array}{rcl} a + \frac{3}{2}c = \frac{15}{2} \\ b + c = 2 \\ b - \frac{1}{2}c = \frac{-9}{2} \end{array} \Rightarrow \begin{bmatrix} 1 & 0 & \frac{3}{2} & \frac{15}{2} \\ 0 & 1 & 1 & 2 \\ 0 & 1 & \frac{-1}{2} & \frac{-9}{2} \end{bmatrix}$$

Step 4 : Multiply 2nd equation by −1 and add it to 3rd equation.

$$\begin{array}{rcl} a + \frac{3}{2}c = \frac{15}{2} \\ b + c = 2 \\ b - \frac{3}{2}c = \frac{-13}{2} \end{array} \Rightarrow \begin{bmatrix} 1 & 0 & \frac{3}{2} & \frac{15}{2} \\ 0 & 1 & 1 & 2 \\ 0 & 0 & \frac{-3}{2} & \frac{-13}{2} \end{bmatrix}$$

Step 5 : Multiply 3rd equation by $\frac{-2}{3}$

$$\begin{aligned} a+\frac{3}{2}c &= \frac{15}{2} \\ b+c &= 2 \\ c &= \frac{13}{3} \end{aligned} \Rightarrow \begin{bmatrix} 1 & 0 & \frac{3}{2} & \frac{15}{2} \\ 0 & 1 & 1 & 2 \\ 0 & 0 & 1 & \frac{13}{3} \end{bmatrix}$$

Step 6 : The 3rd equation gives $c = \frac{13}{3}$, if we substitute this value in 2nd equation then

$$b + \frac{13}{3} = 2$$

$$\therefore \quad b = 2 - \frac{13}{3}$$

$$b = \frac{-7}{3}$$

Now if we put value of $c = \frac{13}{3}$ in 1st equation then

$$a + \frac{3}{2}c = \frac{15}{2}$$

$$a + \frac{3}{2}\left(\frac{13}{3}\right) = \frac{15}{2}$$

$$a + \frac{13}{2} = \frac{15}{2}$$

$$a = 1$$

Thus we get the solution for linear equations as $a = 1$, $b = \frac{-7}{3}$ and $c = \frac{13}{3}$

Review Questions

GTU : Dec.-11, Marks 4

1. *Explain linear inequality and equations.*
2. *Explain the solution to linear in equality.*
3. *Explain the Guass elimination method with an illustrative example.*

1.10 University Questions with Answers

(Regulation 2008)

Dec.-2010

Q.1 *What is an algorithm ? Explain various properties of an algorithm.* [3]

[Refer section 1.2]

May 2011

Q.2 *Explain the term quantifiers.* **[Refer section 1.5]** [2]

Dec. 2011

Q.3 *What is relation ? Explain equivalence relation.* **[Refer section 1.5]** [3]

(Regulation 2013)

Winter 2015

Q.4 *Define the term - Algorithm.* **[Refer section 1.2]** [2]

Q.5 *Explain the term quantifiers.* **[Refer section 1.5]** [2]

1.11 Short Questions and Answers

Q.1 Define the term algorithm.

Ans. : The algorithm is defined as a collection of unambiguous instructions occurring in some specific sequence and such an algorithm should produce output for given set of input in finite amount of time.

Q.2 Specify any two desirable properties of algorithm.

Ans. : 1. Non ambiguity 2. Finiteness

Q.3 What is algorithmic strategy?

Ans. : Algorithmic strategies is a general approach by which many problems can be solved algorithmically. These problems may belong to different areas of computing. Algorithmic strategies are also called as algorithmic techniques or algorithmic paradigm.

Q.4 Name five algorithm design techniques.

Ans. : 1. Brute Force 2. Divide and Conquer 3. Dynamic Programming
4. Greedy Technique 5. Backtracking.

Q.5 What is pseudo code?

Ans. : Pseudo code is a method of specifying an algorithm. It is basically a combination of natural language and programming construct.

Q.6 What are three ways of representing a set?

Ans. :

1. **Listing Method :** In this method the elements are listed in the set.
2. **Describing Properties :** In this method the properties of elements of a set are specified.

3. Recursion Method : The recursion occurs to define the elements of the set.

Q.7 List out various operations on the set.

Ans. : 1. Union 2. Intersection 3. Difference 4. Complement

Q.8 What is cardinality of set?

Ans. : The cardinality of set is nothing but the number of members in the set. The cardinality can be one to one, one to many, many to many.

Q.9 List out various properties of relations.

Ans. : Various properties of relations are - 1. Reflexive 2. Irreflexive 3. Transitive 4. Symmetric 5. Asymmetric

Q.10 What is Equivalent relation ?

Ans. : A relation is said to be equivalent relation if it is reflexive, symmetric and transitive.

Q.11 What is vector ?

Ans. : A vector A is a collection of n tuples.

For example A = (a1,a2,a3,...,an)

where ai are called the components of A.

Q.12 What is row vector and column vector ?

Ans. : A matrix with one row is called row vector and matrix with one column is called column vector.

[10 20 30] is a row vector.

$$\begin{bmatrix} 10 \\ 20 \\ 30 \end{bmatrix}$$

is called column vector.

Q.13 What is identity matrix ?

Ans. : The identity matrix or unit matrix of size n is a square matrix having one on main diagonal and zeros elsewhere.

Q.14 What is linear inequality ?

Ans. : In mathematics, linear inequality is a statement containing <, >, <=,>=. For example x+y>14 is a linear inequality.

Q.15 What is solution to linear in-equality ?

Ans. : The solution of an inequality is a solution that satisfies inequality.

Q.16 Give example of solution to linear in-equality.

Ans. : 3x+7y–5<=17, the value of x and y should be such that the statement remains true. The solution to above inequality is (2, 2) because 3(2)+7(2)–5=15<=17.

Q.17 What is linear equation ?

Ans. : Linear equation is an equation containing n unknowns.

Q.18 Specify the form of linear equation with one unknown.

Ans. : The standard form of linear equation with one unknown is

ax=b

Q.19 What is Gauss elimination method ?

Ans. : Gaussian elimination method is a method of solving linear system Ax=b by bringing augmented matrix, to an upper triangular form and then obtaining a solution by backward substitution method.

Q.20 Solve 5x+15=50 for x.

Ans. : 5x+15–15=50–15

5x=35

x=7

❑❑❑