Design and Analysis of Algorithms

Amanur Rehman

This reference book can be useful for

BBA, MBA, B.Com, BMS, M.Com, BCA, MCA

and many more courses for Various Universities

NEERAJ
PUBLICATIONS
www.neerajbooks.com

Published by:

NEERAJ PUBLICATIONS

(Publishers of Educational Books)

Sales Office : 1507, 1st Floor,
Nai Sarak, Delhi-110 006
E-mail: info@neerajbooks.com
Website: www.neerajbooks.com

Typesetting by: Competent Computers

Terms & Conditions for Buying E-Book

- The User must Read & Accept the Terms and Conditions (T&C) carefully before clicking on the accept option for Buying the Online Soft Copy of E-books. Under this Particular Facility you may buy only the Online Soft Copy of E-books, no Hard Copy or Printed Copy shall be provided under this facility.
- These E-Books are valid for 365 days online reading only (From the Date of Purchase) and no kind of Downloading, Printing, Copying, etc. are allowed in this facility as these products are just for Online Reading in your Mobile / Tablet / Computers.
- All the online soft copy E-books given in this website shall contain a diffused watermark on nearly every page to protect the material from being pirated / copy / misused, etc.
- This is a Chargeable Facility / Provision to Buy the Online Soft Copy of E-books available online through our Website Which a Subscriber / Buyer may Read Online on his or her Mobile / Tablet / Computer. The E-books content and their answer given in these Soft Copy provides you just the approximate pattern of the actual Answer. However, the actual Content / Study Material / Assignments / Question Papers might somewhat vary in its contents, distribution of marks and their level of difficulty.
- These E-Books are prepared by the author for the help, guidance and reference of the student to get an idea of how he/she can study easily in a short time duration. Content matter & Sample answers given in this E-Book may be Seen as the Guide/Reference Material only. Neither the publisher nor the author or seller will be responsible for any damage or loss due to any mistake, error or discrepancy as we do not claim the Accuracy of these solution / Answers. Any Omission or Error is highly regretted though every care has been taken while preparing these E-Books. Any mistake, error or discrepancy noted may be brought to the publishers notice which shall be taken care of in the next edition. Please consult your Teacher/Tutor or refer to the prescribed & recommended study material of the university / board / institute / Govt. of India Publication or notification if you have any doubts or confusions before you appear in the exam or Prepare your Assignments before submitting to the University/Board/Institute.
- Publisher / Study Badshah / shall remain the custodian of the Contents right / Copy Right of the Content of these reference E-books given / being offered at the website www.studybadshah.com.
- The User agrees Not to reproduce, duplicate, copy, sell, resell or exploit for any commercial purposes, any portion of these Services / Facilities, use of the Service / Facility, or access to the Service / Facility.
- The Price of these E-books may be Revised / Changed without any Prior Notice.
- The time duration of providing this online reading facility of 365 days may be alter or change by studybadshah.com without any Prior Notice.
- The Right to accept the order or reject the order of any E-books made by any customer is reserved with www.studybadshah.com only.
- All material prewritten or custom written is intended for the sole purpose of research and exemplary purposes only. We encourage you to use our material as a research and study aid only. Plagiarism is a crime, and we condone such behaviour. Please use our material responsibly.
- In any Dispute What so ever Maximum Anyone can Claim is the Cost of a particular E-book which he had paid to Study Badshah company / website.
- If In case any Reader/Student has paid for any E-Book and is unable to Access the same at our Website for Online Reading Due to any Technical Error/ Web Admin Issue / Server Blockage at our Website www.studybadshah.com then He will be send a New Link for that Particular E-Book to Access the same and if Still the Issue is Not Resolved Because of Technical Error/ Web Admin Issue / Server Blockage at our website then His Amount for that Particular Purchase will be refunded by our website via PayTM.
- All the Terms, Matters & Disputes are Subjected to "Delhi" Jurisdiction Only.

CONTENTS

Sample Preview of The Chapter

Published by:

NEERAJ PUBLICATIONS

www.neerajbooks.com

DESIGN AND ANALYSIS OF ALGORITHMS

INTRODUCTION TO ALGORITHMICS

1

Elementary Algorithmics

INTRODUCTION

Since the earth came into existence, living being are busy in solving problem. What do you mean by problem? Problem is any unacceptable/undesirable situation and it could be harmful for one and profitable for another e.g. enchroachment is profitable for public and problem for government. If there exist a solution for any problem then there would be a sequence of activities called process should be there. A solution of any problem is just merely a transition from undesirable to desirable state.

Technically, the set of instruction or description in a particular notation of the process is termed as algorithm. An algorithm is a finite step-by-step well defined instructions of the sequence of the activities that constitute a process of getting the desired outputs from the given inputs. The raw material needed at the time of beginning is referred to as input and the resulting entity is referred as output.

An algorithm, when designed in a fashion that can be understood and executed by a computer system is called a computer program/program.

CHAPTER AT A GLANCE

EXAMPLE OF AN ALGORITHMS

Take a example of well known algorithm for finding greatest common divisor (G.C.D.) of two natural numbers.

Euclid's algorithm for finding G.C.D. two natural numbers m & n:

E1: { Find remainder } Divide m by n and let r be the remainder (new) {e have $0 \leq r < n$}

E2: { Is r zero } If r = 0,the algorithm terminates and n is the answer otherwise.

E3: { Interchange } Let the new value of m be the current value of r. Go back to step E1.

The termination of the above method is sure, since m and n is used to reduce in each iteration and r must become zero in finite number of repetitions of steps E1, E2 and E3.

The Euclid's algorithm in a pseudo–code notation which is closer to a programming language.

Algorithm GCD Euclid (m, n)

```
{This algorithm computes the
greatest common divisor of the two
given positive integers.}
```

```
    begin {of algorithm}
        while n ≠ 0 do
        begin {of while loop}
            r ← m mod n;
            {a new variable is used to
store the remainder which is obtained
by dividing m by n ,with 0 < r, m}
        m ← n;
    { the value of n is assigned as new
value of m; but at this stage value of n
remains unchaged}
        m → r;
    { the value of r becomes the new value
of n and the value of r remains unchanged}
    end {of while loop}
        return (n).
    end;{of algorithm}
```

PROBLEMS AND INSTANCES

We know that the roots of a general quadratic equation

$$ax^2 + bx + c = 0 \qquad a \neq 0 \qquad ...(1)$$

are given by the equation

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2c} \qquad ...(2)$$

where a, b, c may be any real number except the restriction that a ≠ 0.

Now take a = 3, b = 4, c = 1

then the equation (1) becomes $3x^2 + 4x + 1=0$

using equation (2) x =–1/2 or –1

Considering the above discussion, finding out the roots of general quadratic equation $ax^2 + bx + c = 0$ is called a problem whereas finding out the roots of particular equation $3x^2 + 4x + 1= 0$ is called instance or question of the problem (general).

Depending upon the problem, any problem can have minimum one instance and maximum infinite instance. For some problems, there may be only one instance/question corresponding to each of the problems. For example, the problem of finding out the largest integer that can be stored or can be arithmetically operated on, in a given computer, is a single–instance problem.

CHARACTERISTICS OF AN ALGORITHM

There are five important characteristics of an algorithm that should be considered while designing any algorithm for any problem.

1. Finiteness: An algorithm should terminate in finite number of steps and each step must finish in finite amount of time.

2. Definiteness (no ambiguity): Each step of an algorithm should be clearly n precisely define n there should not be any ambiguity.

Example of definiteness:

A program fragment is given below:

x ← 1

toss a coin

if the result is head then x ← 3 else x ← 4

in the above program, all the steps would be carried out effectively but there is no definiteness since there are two possible values of x i.e., 1 and 3/4.

3. Inputs: An algorithm must have zero or more but must be finite number of inputs.

Example of zero input algorithm:

Print the ASCII code of each of the letter in the alphabet of the computer system.

4. Output: An algorithm must have at least have one desirable outcome i.e., output.

5. Effectiveness: An algorithm should be effective. Effective means that each step should be referred as principle and should be executing in finite time.

Example of not effectiveness: Find exact value of e using the following formula:

e = 1 + 1/(1!) +1/(2!) + 1/(3!)+…………

and add it to x.

It is not effective since it requires summation of infinite terms. Therefore it takes infinite time hence not effective.

PROBLEMS, AVAILABLE TOOLS AND ALGORITHM

To understand the available tools, we would consider some alternative algorithms for finding the product m*n of two natural numbers m and n.

First Algorithm

The usual method of multiplication, is to multiply each digit of one number to each digit of another number using multiple table as shown below:

$$\begin{array}{cccc} & 1 & 2 & 3 \\ & & 1 & 2 \\ \hline & 2 & 4 & 6 \\ 1 & 2 & 3 & \times \\ \hline 1 & 4 & 7 & 6 \end{array}$$

Second Algorithm

In this algorithm, we don't have multiplication table, we are having only arithmetic capability like:

(a) that of counting are

(b) that of comparing two integers w.r.t. "less than or equal to" relation with the above facilities. one possible algorithm would require two storage device, one of the storage device is used to accommodate marks upto n, the multiplier and other to accommodate marks m*n, the resultant product.

The algorithm are as follows:

Step 1: Initially make a mark of first portion on the paper.

Step 2: For each new mark on the first portion, make a new mark on the second portion.

Step 3: Count the number of marks in first portion. If the count equals n, then count the number of all marks in the second portion and return the last count as the result. However, if the count in the first portion is less than n, then make one more mark in the first portion and goto step 2.

There are several other logic to design the algorithm for multiplication of the two natural numbers depending upon the available tools.

BUILDING BLOCKS OF AN ALGORITHM

There are three basic actions and corresponding instructions form the basis of any imperative language.

Basic Actions and Instructions

(a) **Assignment of value:** Any value to a variable can be assigned as
Variable ← expression;

– Variable could be any combination of alphabets and numeric

– Expression could be any fixed value or the expression of calculation
e.g. x ← y + z

– Every time the statement execute, the new value assigned to the variable.

(b) Second basic action is to read value of variables. It can be done with the help of the command

Read (i. j,……)

(c) The third basic action is to write values of some variables. It can be done with the help of the command

Write (i, j…..)

If any sequence of character is given in the quotes inside the bracket of write command then it is print as it is, e.g.

Write ("xyz")

The outcome of algorithm would be xyz.

Control Mechanism and Control Structure

In order to understand algorithm, it is must to know and to understand the mechanism. There are three basic control mechanism.

***(a)* Direct Sequencing:** In this, the execution of instruction is same as the sequence of instruction is written in the program text. Unless instructions are given to the contrary, the modules are executed in the obvious sequence. The sequence may be presented explicitly, by means of numbered steps, or implicitly, by the order in which the modules are written as shown in figure.

Algorithm	Flowchart
	↓
Module A	Module A
	↓
Module B	Module B
	↓
Module C	Module C
	↓

***(b)* Selection:** Selection logic employs a number of conditions which lead to a selection of one out of several alternative modules. The structures which implement this logic are called conditional structures or If structures. For clarity, we will frequently the end of such a structure by the statement.

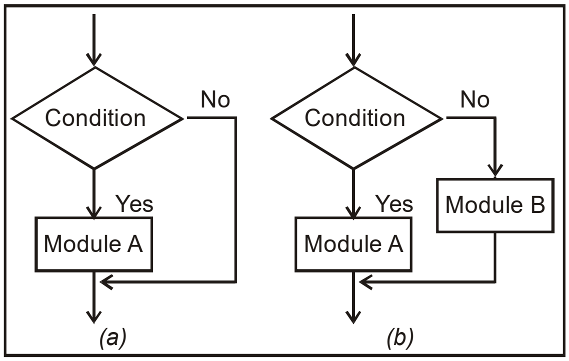
```
[End of If structure.]
```

or some equivalent. These conditional structures fall into three types, which are discussed separately.

***(a)* Single alternative: This structure has the form**

```
If condition, then:
[Module A]
[End of If structure]
```

The logic of this structure is shown in figure *(a)* below. If the condition holds, then Module A, which may consist of one or more statements, is executed; otherwise Module A is skipped and control transfers to the next step of the algorithm.



***(b)* Double alternative:** This structure has the form

```
If condition, then:
    [Module A]
Else:
    [Module B]
[End of If structure]
```

The logic of this structure is shown in figure (b) above. If the condition holds, then module A is executed; otherwise module B is executed.

***(c)* Multiple alternatives:** This structure has the form:

```
If condition(1), then:
   [Module $A_1$]
Else if condition(2), then:
   [Module $A_2$]
         .
         .
Else if condition(M), then:
   [Module $A_m$]
Else:
   [Module B]
[End of if structure.]
```

The logic of this structure allows only one of the modules to be executed specifically. Either the module which follows the first condition which holds is executed, or the module which follows the final. Else statement is executed. In practice, there will rarely be more than three alternatives.

Example: Suppose we are to write a program segment that converts % of marks to grades as follows:

% of marks(M)	**grade(G)**
$80 \le M$	A
$60 \le M < 80$	B
$50 \le M < 60$	C
$40 \le M < 50$	D
$M < 40$	F

Then the corresponding notation may be

Case M of
80…100: 'A'
60….79: 'B'
50….59: 'C'
40….49: 'D'
0….39: 'F'

Where M is an integer variable.

Example: The solutions of the quadratic equation $ax^2 + bx + c = 0$.

Where $a \neq 0$, are given by the quadratic formula.

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2c}.$$

The quantity $D = b^2 - 4ac$ is called the discriminant of the equation. If D is negative, then there are no real solutions. If D = 0, then there is only one (double) real solutions, $x \neq - b/2a$. If D is positive, the formula gives the two distinct real solutions. The following algorithm finds the solutions of a quadratic equation.

Algorithm: (Quadratic equation): This algorithm inputs the coefficients A, B, C of a quadratic equation and outputs the real solutions, if any.

```
Read: A, B, C.
Set D: $B^2$ -4AC.
If D > 0, then:
        Set $x_1$: =(-B+√D)/2A and $x_2$: (-B-
        √D)/2A.
  Write: $x_1$, $x_2$.
Else if D=0, then:
        Set x: = -B/2A.
        Write: 'UNIQUE SOLUTION', x.
Else:
        Write: 'NO REAL SOLUTIONS'
{end of if structure}
End.
```

***(c)* Repetition:** There are some situation occurs when there is a requirement of execution of the same task repeatedly. Then we use this technique. This technique refers to either of two types of structures involving loops. Each type begins with a Repeat statement and is followed by a module, called the body of the loop. For clarity, we will indicate the end of the structure by the statement

```
[End of loop.]
```

or some equivalent.

Each type of loop structure is discussed separately. The repeat–for loop uses an index variable, such as K, to control the loop. The loop will usually have the form:

```
Repeat for K=R to S by T:
        [Module]
     [End of loop.]
```

The logic of this structure is shown in figure *(a)* below. Here R is called the initial value, S the end value or test value, and T the increment. Observe that the body of the loop is executed first with K=R, then with K=R+T, then with K=R+2T, and so on. The cycling ends when K > S. The flowchart assumes that the increment T is positive; if T is negative, so that K decreases in value, then the cycling ends when K < S.