Design & Analysis of Algorithms [10CS43]

UNIT - I

Topics

Chapter 1:
(i) Notion of Algorithms
(ii) Algorithm Design & Analysis Process

Chapter 2:
(i) Analysis of Framework
(ii) Asymptotic Notations
(iii) Analysis of Non-recursive Algorithms
(iv) Analysis of Recursive Algorithms

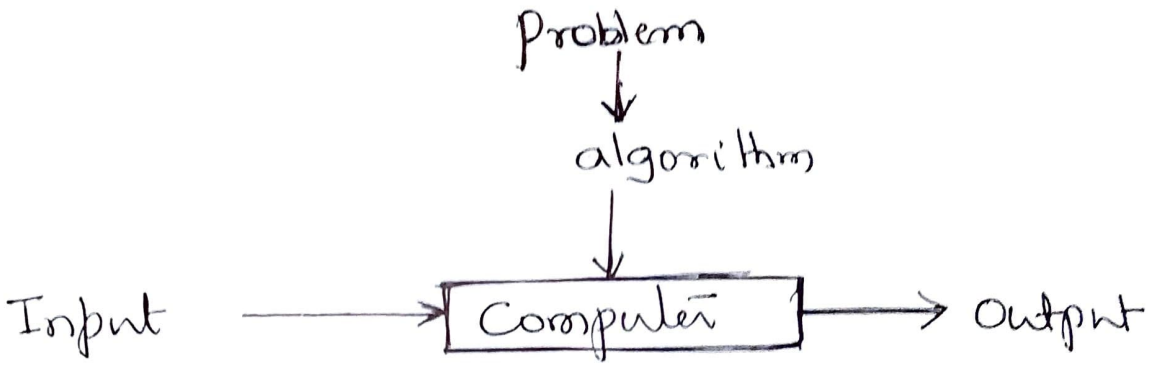
Chapter 3:
(i) Brute Force approaches, Introduction
(ii) Analysis of Bubble sort
(iii) Analysis of Selection Sort
(iv) Analysis of Sequential search
(v) Analysis of Brute force string Matching

CHAPTER 1

Definition of Algorithm

An algorithm is a sequence of unambiguous instruction for solving a problem for obtaining a required output for any legitimate input in a finite ammount of time.

The definition can be illustrated by a simple diagram



The important different properties of an algorithm are

1. Input
2. Output
3. Termination
4. Definiteness
5. Effectiveness

As an example, illustrating the concept of algorithm, consider three methods for solving same problem: Computing greatest common divisor of two integers. These examples will illustrate several important points.

1. The nonambiguity requirement for each step of algorithm
2. The range of inputs for which algorithm works
3. The same algorithm can be represented in different ways.
4. Many algorithms for solving the same problem may exists.
5. Algorithm for the same problem may be based on different ideas and solve problem with different speed.

Example: GCD of two integers.

The three different methods to compute GCD considered are.

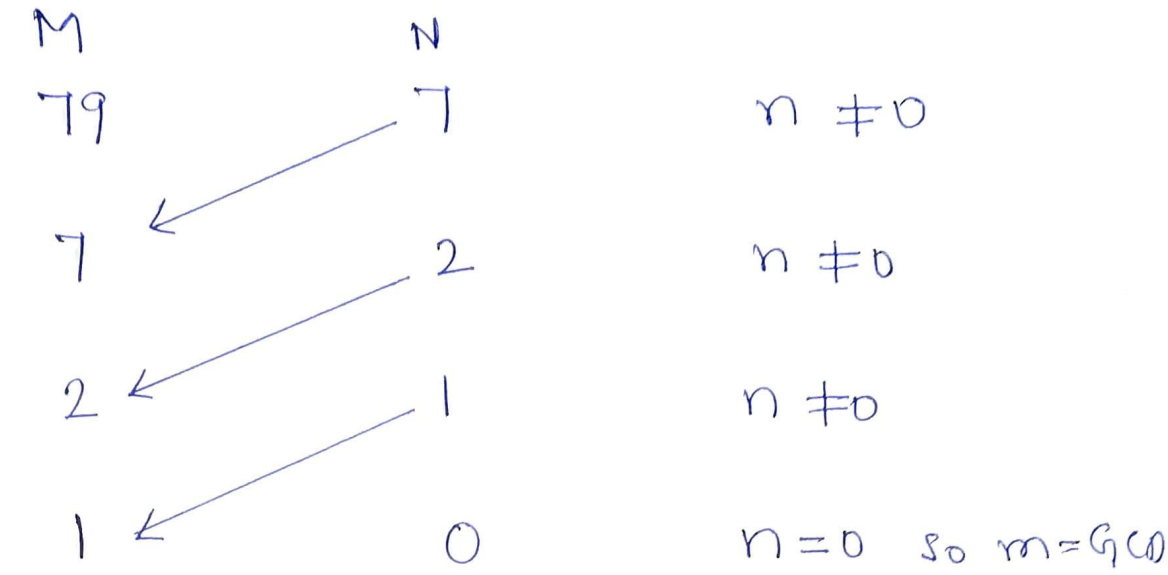
1. Euclids method
2. Consecutive integer checking method
3. Middle school method.

I. Euclid's Algorithm: accepts two nonzero & positive Integers. Euclid's algorithm based on applying repeatedly

$$m = n$$
$$n = m \bmod n$$

where $m \bmod n$ is remainder of division of m by n, untill $m \bmod n$ or n is equal to Zero. When n is Zero, the value of m is GCD of initial m and n.

For example, consider gcd (79, 7)



Algorithm.

Algorithm Euclid (m, n)

// Input: positive, non-zero integers m & n

// Output: Greatest common divisor of m and n

```
begin
    while (n ≠ 0)
    begin
        rem ← m mod n
        m ← n
        n ← rem
    End
    return m
End.
```

II. Consecutive Integer checking Algorithm

This algorithm is based on the applying

m mod t , n mod t

where t is minimum of m and n. If t divides both then t is GCD of (m, n) else decrement t by 1 and repeat

Consider the example to find gcd (60, 24)

m	n	t	
60	24	24	
60	24	23	
60	24	22	
.	.		
.	.		
.	.		
.	.		
60	24	12	stop.

12 divides both 60 & 24. So 12 is gcd

Algorithm:

```
Algorithm Integercheck (m, n)
// I/p: Two positive, nonzero Integers m, n
// O/p: gcd of m, n.

begin
        t = minimum (m, n)
while ( m mod t ≠ 0  or  n mod t ≠ 0)
t = t - 1

return t

End
```

III. Middle school Procedure

This method is based on finding

(i) prime factors of first number m

(ii) prime factors of second number n

(iii) Identify all the common prime factors in both numbers m & n

(iv) compute the product of all common factors.

Consider the example of finding gcd (60, 24)

factors of 60 = 2 · 2 · 3 · 5

factors of 24 = 2 · 2 · 2 · 3

common factors = 2 · 2 · 3.

= 12

So gcd (60, 24) = 12

ALGORITHM TO GENERATE PRIME NUMBERS

Sieve method: This algorithm starts by initializing a list of prime candidates or all integer numbers from 2 to n. Then on its first iteration eliminates from the list multiples of 2. Next iteration it eliminates multiples of 3. Next 4 and so on, till square root of n.

	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
I	2	3		5		7		9		11		13		15		17		19	
II	2	3		5		7				11		13				17		19	
III	2	3		5		7				11		13				17		19	

```
Algorithm Sieve (n)
  // Input: A positive integer n > 1
  // output: Prime numbers between 1 to n
begin
      for p← 2 to n
       A[p] ← p

      for p← 2 to √n
  begin
        if (A[p] ≠0) do
    begin
          j← P×p
              while (j<=n) do
        begin
            A[j]←0
             j←j+p
        End
    End
End
  begin for p← 2 to n do
        if(A[p]≠0) L[i]←A[p]
                   i←i+1
End
```

Design and Analysis of Algorithms Process

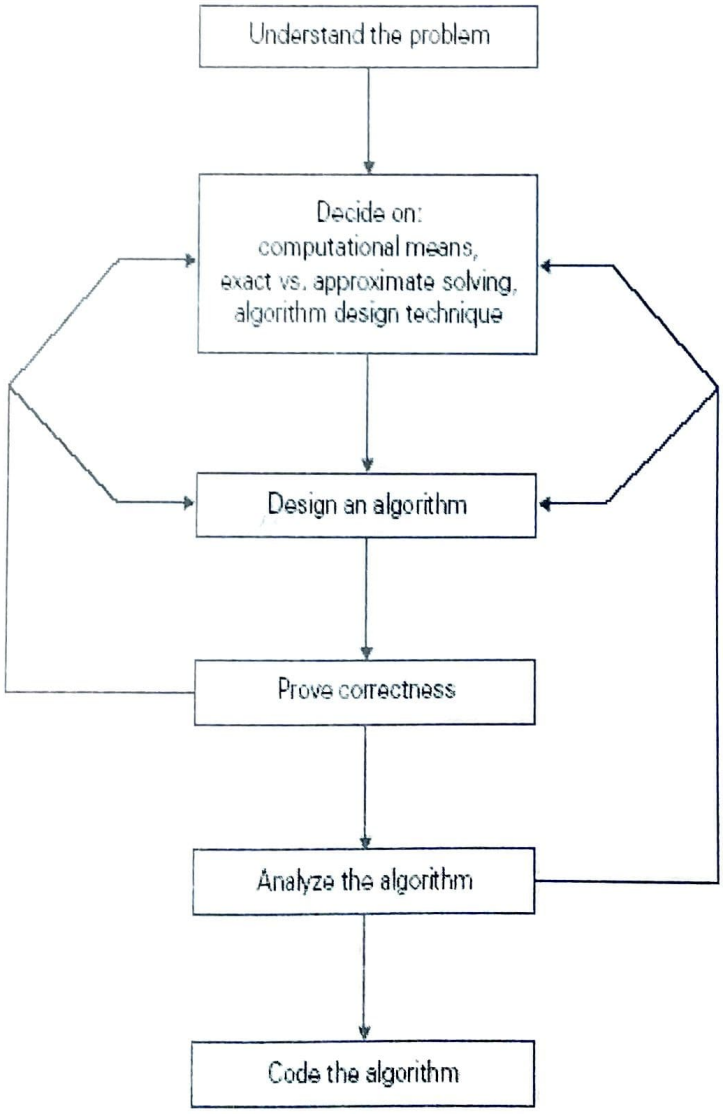


FIGURE 1.2 Algorithm design and analysis process.

The different steps in algorithm design & analysis are

1. Understanding the problem
2. Ascertaining the capabilities of computational devices
3. Choosing between Exact & Approximate problem solving
4. Algorithm design Techniques.
5. Designing an algorithm & Data structures
6. Methods of specifying an algorithm
7. Proving an algorithm's correctness
8. Analyzing an algorithm
9. Coding an algorithm.