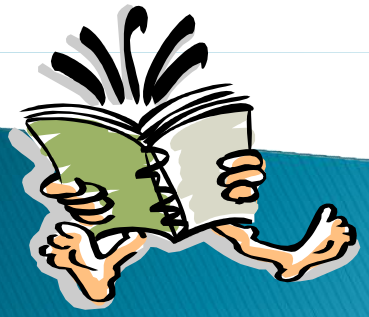
Algorithm Design-2

(CSE4131)

Introduction Lecture

Presented
By

Dr. Rajesh Purkait
Asst. Professor
Dept. of CSE ITER
(S`O'A Deemed To Be University)
E-mail: rajeshpurkait@soa.ac.in



Outline

- Course Structure (**Lesson plan discussion)**
- Overview of Algorithm Design-1
- Problem identification
- Different algorithm design approach
- What is Algorithms?
- Correctness verification
- Analysis of Algorithms (Input size)
  - Time
  - Memory Space
- Different type of analytical model
- Coding (Execution of the computer program)

Outline

- Course Structure (**Lesson plan discussion)**
- Overview of Algorithm Design-2
- Deterministic Algorithm
  - Dynamic Programming
  - Backtracking
  - Heuristic Approach
- Non-Deterministic Algorithm
  - Approximation Algorithm using different approach
  - Intractable Problems

COURSE STRUCTURE

Prerequisites

Fundamentals of Data structures

Fundamentals of Discrete Mathematics

Algorithm Design-I

Adequate programming skills (in C, C++, java, Python etc.)

Ability to work hard

Commitment to attend classes

Marks Breakup

Assignments and Project**: 20%

Programming as well as theoretical.

Attendance: 5%

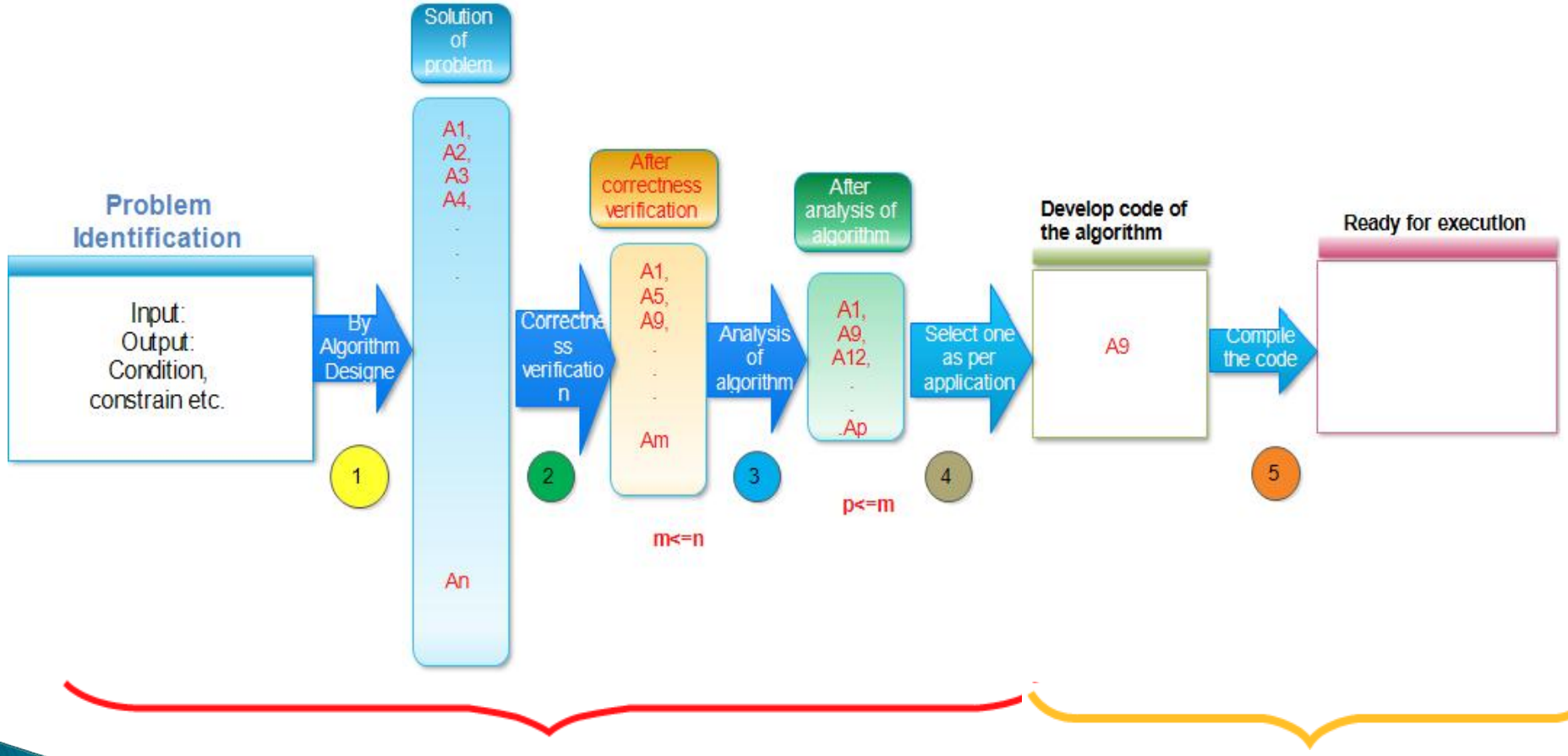
Mid Semester Exam: 15%

End Semester (Lab. Test/Quiz): 15%

End Semester Exam: 45%

Passing criteria:

Algorithm Design



Problem identification

- **Problem:** Given numbers m, n find their greatest common divisor.
- Algorithm 1:Simple school level

- Algorithm 2: Euclid

Problem identification

- Specification of what are valid input and what are acceptable outputs for each valid input.
- Input Instance:
  - A value x is an input instance for problem P, if x is a valid input as per the specification.
- **Example of problem:**
  - GCD of two numbers.
  - Finding the shortest path on a map
  - Finding a word in a dictionary
  - Given an X-ray is there a disease

Different algorithm design approach

- **Incremental approach**
  - Bubble sort
  - Selection sort
  - Insertion sort
- **Divide and conquer approach**
  - Binary search
  - Merge sort
  - Quick sort
- **Greedy method**
  - Prim's algorithm, Kruskal's algorithm etc
- **Dynamic programming method**
- **Branch and bound**
- **Backtracking**
- **Heuristic method**

What is Algorithms?

- An *algorithm* is a finite set of precise instructions for performing a computation or for solving a problem.
- The following problem are well defined and are considered to be computational problems-
  - Given an integer x, test whether x is prime or not
  - Given a program P, check whether P runs into an infinite loop.
- So an algorithm **must have an input** provide by the user and **must produce a correct output**.
- Computational instructions in the algorithm must involve only basic algebraic operations and it should be terminates after a finite number of steps.

What is Algorithms? (Cont'd)

- Finally, any algorithm must involve unambiguous instructions and produce the desired output.
- Mathematically, an algorithm can be represented as a function,
  - **F: I → O**, where I is the set of input and O is the set of output generated by the algorithms.
- The word algorithm comes from the name of Persian author "**Abu Jafar Mohammad ibn Musa al Khawarizmi**"

What is Algorithms? (Cont'd)

- Definition of algorithm spark nature fundamental questions?
  - How to design an algorithm for a given problem?
  - Is every problem algorithmically solvable?
    - If so, how many algorithms can a problem have and
    - how to find the efficient one
- We shall address this question in the lecture?

What is Algorithms? (Cont'd)

- Let us consider an example of finding a maximum element in an array of size n. An algorithm is typically described using pseudo as follows:
  - Finding a maximum element in an array

```
Algo Max-array(A,n)
{
        Max = A[1];
           for i = 2 to n do
            if ( A[i] > Max ) then Max = A[i];

        return Max;
}
```

  - Maximum can be computed by sorting the array in an increasing order(decreasing order) and pick the last element(fast).
  - There are at least five different algorithms to find a maximum elements, it is natural ask for an efficient algorithm.
  - This call for the study of correctness verification and analysis of algorithms?

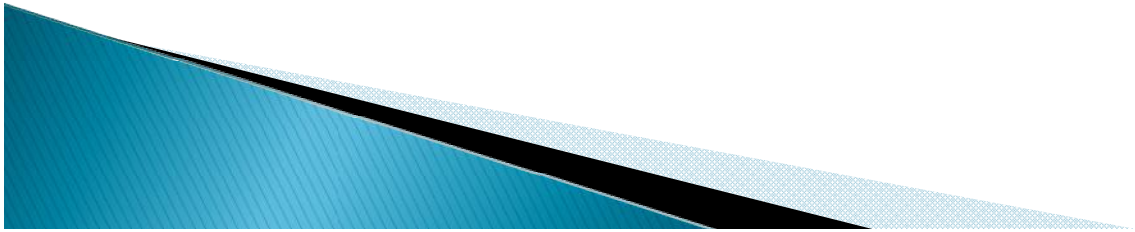
Correctness verification

- This means to verify if the algorithm leads to the solution of the problem (hopefully after a finite number of processing steps).
- Verify that algorithm produces correct o/p for the given correct i/p

Analysis of Algorithms

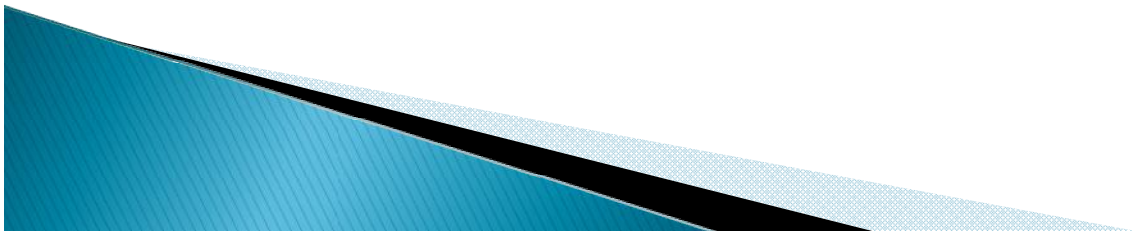
- **What is the goal of analysis of algorithms?**
  - To compare algorithms mainly in terms of running time but also in terms of other factors (e.g., memory requirements, programmer's effort etc.)

- What do we mean by *running time analysis*?
  - **Determine how running time increases as the size of the problem increases.**



Input Size

- Input size (number of elements in the input)
  - size of an array
  - polynomial degree
  - # of elements in a matrix
  - # of bits in the binary representation of the input
  - vertices and edges in a graph



Different type of analytical model

- RAM (Random Access Machine) Model
  - Mathematical model of a computer
  - How much time takes to execute an instruction in process and memory label.
  - Memory: Collection of locations
- Asymptotic Analysis
  - To compare f(n) and g(n) check how fast each function grows.

Types of Analysis

- Worst case
  - Provides an **upper bound** on running time
  - An absolute guarantee that the algorithm would not run longer, no matter what the inputs are
- Best case
  - Provides a **lower bound** on running time
  - Input is the one for which the algorithm runs the fastest

$$Lower\ Bound \leq Running\ Time \leq Upper\ Bound$$

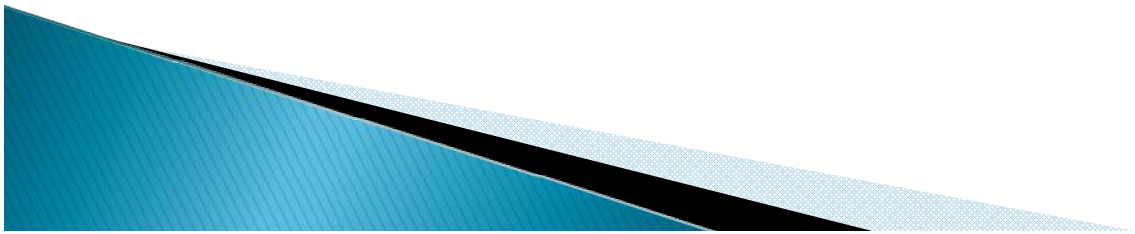
- Average case
  - Provides a prediction about the **running time**
  - Assumes that the input is random

Coding

- In this step programmer transform the algorithm into a program by using different programming language (PL).
  - Low level PL : Machine code etc.
  - Middle level PL: C, Basic, PASCEL etc.
  - High level PL: JAVA, C++, Python etc.

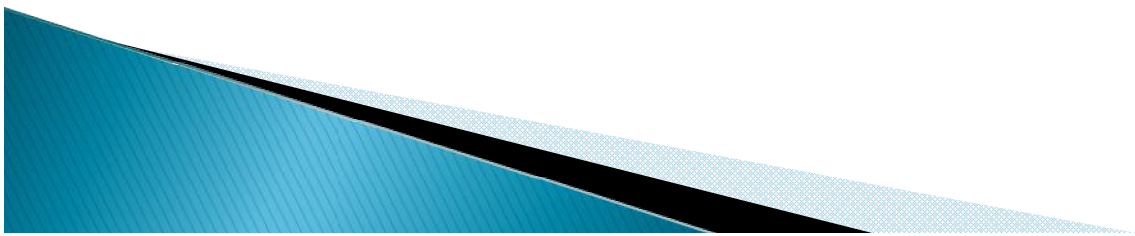
Factor influencing program efficiency

- Problem identification
- Problem being solved
- Programming language
- Compiler/Interpreter
- Computer hardware
- Programmer ability
- Programmer effectiveness
- Algorithm



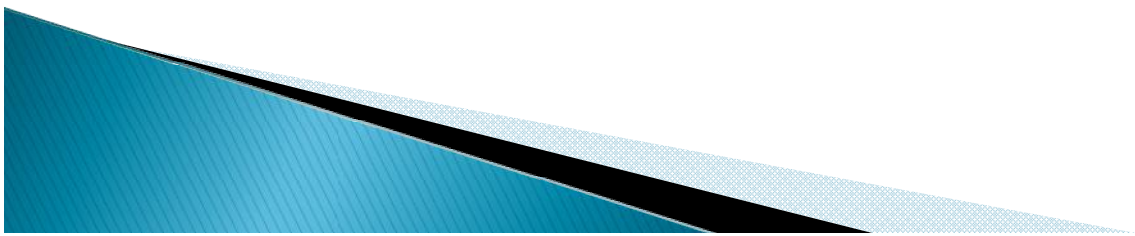
Why study this subject?

- Efficient algorithms shell better
- Efficient program make better use of hard wired.
- Programmers who write efficient programs are more marketable than those who don't.

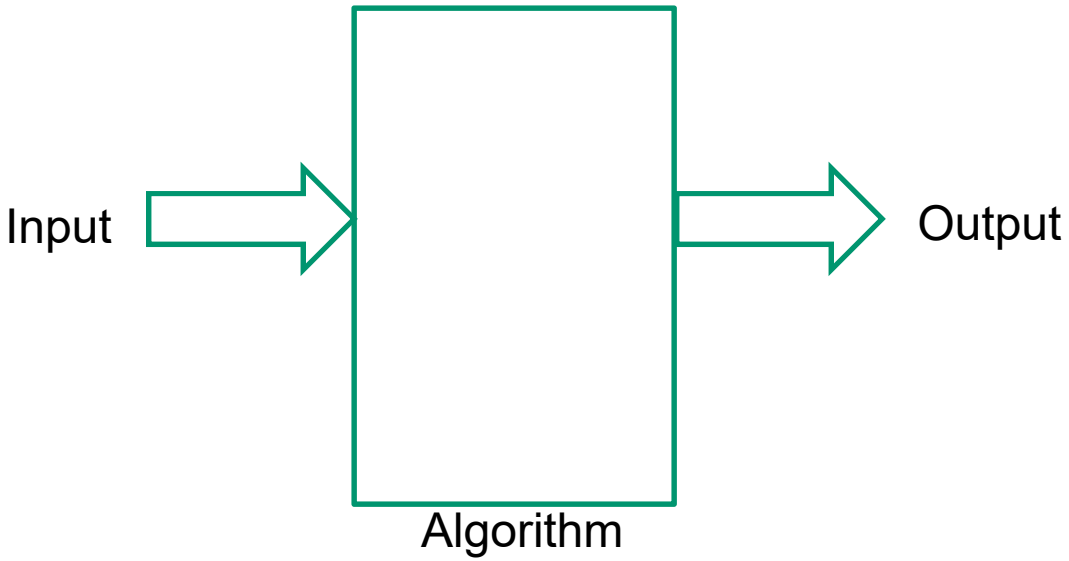


Objectives

- Method for analyzing algorithm efficiency.
- A set of standered algorithm technique.
- A set of standered algorithm



Deterministic Algorithm



- The **output** as well as the **running time** are **functions** only of the **input**.

Thank You and
WELCOME to All of You