LECTURE 1

DESIGN AND ANALYSIS OF ALGORITHMS

Introduction to Algorithms

ORIGIN OF WORD 'ALGORITHM'

The word algorithm comes from the name of a Persian author, Abu Ja'far Mohammed ibn Musa al Khowarizmi (c.825A.D.), who wrote a textbook on mathematics.

WHAT IS AN ALGORITHM?

In mathematics and computer science, an algorithm is a finite sequence of well-defined, computer-implementable instructions, typically to solve a class of problems or to perform a computation.

Or

Informally, for a problem-solving procedure, *algorithm*, the set of rules a machine (and especially a computer) follows to achieve a particular goal. It does not always apply to computer-mediated activity; however, the term may as accurately be used of the steps followed in making a cup of tea or solving a Rubik's Cube as for computer-powered data analysis.

Or

In its purest sense, an algorithm is a mathematical process to solve a problem using a finite number of steps.

In the world of computers, an algorithm is the set of instructions that defines not just what needs to be done but how to do it.

Or

An algorithm is a finite set of instructions that, if followed, accomplishes a particular task. In addition, all algorithms must satisfy the following criteria:

1. **Input.** Zero or more quantities are externally supplied.

2. **Output.** At least one quantity is produced.

3. **Definiteness.** Each instruction is clear and unambiguous.

4. **Finiteness.** If the instructions of an algorithm are trace out, then for all cases, the algorithm terminates after a finite number of steps.

5. **Effectiveness.** Every instruction must be very basic so that it can be carried out, in principle, by a person using only pencil and paper. It is not enough that each operation be definite as in criterion3; it also must be feasible.

WHAT ARE THE CHARACTERISTICS/FEATURES OF AN ALGORITHM?

- Well Defined Inputs
- Well Defined Outputs
- Clear and Unambiguous
- Finiteness
- Feasible
- Language Independent

- **Well-Defined Inputs:** If an algorithm says to take inputs, it should be well-defined inputs.
- **Well-Defined Outputs:** The algorithm must clearly define what output will be yielded and it should be well-defined as well.
- **Clear and Unambiguous:** Algorithm should be clear and unambiguous. Each of its steps should be clear in all aspects and must lead to only one meaning.
- **Finiteness:** The algorithm must be finite, i.e. it should not end up in an infinite loops or similar.
- **Feasible:** The algorithm must be simple, generic and practical; such that it can be executed upon all the available resources. It must not contain some future technology, or anything.
- **Language Independent:** The Algorithm designed must be language-independent, i.e. it must be just plain instructions that can be implemented in any language, and yet the output will be same, as expected.

WHAT IS A PROGRAM?

A computer program is a collection of instructions that can be executed by a computer to perform a specific task.

A computer program is usually written by a computer programmer in a programming language.

WHAT IS A FLOWCHART?

A **flowchart** is a type of diagram that represents a workflow or process. A flowchart can also be defined as a diagrammatic representation of an algorithm, a step-by-step approach to solving a task.

The flowchart shows the steps as boxes of various kinds, and their order by connecting the boxes with arrows. This diagrammatic representation illustrates a solution model to a given problem. Flowcharts are used in analyzing, designing, documenting or managing a process or program in various fields.

WHAT IS A PSEUDOCODE?

It is a **simpler version of a programming code** in plain English which uses short phrases to write code for a program before it is implemented in a specific programming language.

Or

Pseudocode is an artificial and informal language that helps programmers develops algorithms. Pseudocode is a "text-based" detail (algorithmic) design tool.

DIFFERENCE BETWEEN ALGORITHM AND FLOWCHART:

S.NO	ALGORITHM	FLOWCHART
1.	Algorithm is step by step procedure to solve the problem.	Flowchart is a diagram created by different shapes to show the flow of data.
2.	Algorithm is complex to understand.	Flowchart is easy to understand.
3.	In algorithm plain text are used.	In flowchart, symbols/shapes are used.
4.	Algorithm is easy to debug.	Flowchart it is hard to debug.
5.	Algorithm is difficult to construct.	Flowchart is simple to construct.
6.	Algorithm does not follow any rules.	Flowchart follows rules to be constructed.
7.	Algorithm is the pseudo code for the program.	Flowchart is just graphical representation of that logic.

DIFFERENCE BETWEEN PROGRAM AND ALGORITHM

Program is a set of structured activities in any programming language while algorithm is a precise step-by-step plan for a computational procedure that possibly begins with an input value and yields an output value in a finite number of steps.

AREA OF STUDY

The main areas of research studied by the algorithm group fall into five mutually interrelated categories, namely computational geometry, geometric graph algorithms, parallel, distributed and sequential graph algorithms, computational biology, searching and sorting etc.

The study of algorithms includes many important and active areas of research. There are four distinct areas of study one can identify:

1. **How to devise algorithms:**

Creating an algorithm is an art which may never be fully automated. By mastering various design strategies, like divide and conquer, greedy methods, dynamic programming etc., it will become easier to devise new and useful algorithms. Some of the techniques are especially useful in fields other than computer science such as operations research and electrical engineering.

2. **How to validate algorithms:**

Once an algorithm is devised, it is necessary to show that it computes the correct answer for all possible legal inputs. This process is refers to as **algorithm validation**. The algorithm need not as yet be expressed as a program. It is sufficient to state it in any precise way. The purpose of the validation is to assure us that this algorithm will work correctly independently of the issues concerning the programming language it will eventually be written in. Once the validity of the method has shown, a program can be written and a second phase begins. This phase is referred to as **program proving or program verification**. A proof of correctness requires that the solution be stated in two forms. One form is usually as a program which is annotated by a set of assertions about the input and output variables of the program. These assertions are often expressed in the predicate calculus. The second form is called a specification, and this may also be expressed in the predicate calculus. A proof consists of showing that these two forms are equivalent in that for every given legal input, they describe the same output. A complete proof of program correctness requires that each statement of the programming language be precisely defined and all basic operations be proved correct.

3. **How to analyze algorithms**

This field of study is called analysis of algorithms. As an algorithm is executed, it uses the computer's central processing unit (CPU) to perform operations and its memory (both immediate and auxiliary) to hold the program and data. Analysis of algorithms or performance analysis refers to the task of determining **how much computing time and storage an algorithm requires**. This is a challenging area which sometimes requires great mathematical skill. An important result of this study is that it allows you to make quantitative judgments about the value of one algorithm over another. Another result is that it allows you to predict whether the software will meet any efficiency constraints that exist.

Questions such as how well an algorithm performs in the best case, in the worst case, or on the average are typical.

4. **How to test a program**

Testing a program consists of two phases: debugging and profiling (or performance measurement). **Debugging** is the process of executing programs on sample data sets to determine whether faulty results occur and, if so, to correct them. However, as E. Dijkstra has pointed out, "Debugging can only point to the presence of errors, but not to their absence."

Profiling or performance measurement is the process of executing a correct program on data sets and measuring the time and space it takes to compute the results.

References:

1. Computer algorithms, Ellis Horowitz, Rajasekaran S. and Sartaj Sahni, 1997, Computer Science Press.
2. **www.wikipedia.org**
3. www.tutorialspoint.com
4. www.geeksforgeeks.org
5. Introduction to Algorithms, Thomas H Cormen, Charles E Leiserson and Ronald L Rivest: 1990, TMH