Design and Analysis of Algorithms

Dr. Chayan Halder
Assistant Professor
Ramakrishna Mission Vivekananda Centenary College, Kolkata

Programming

What is Programming?

The communicative technique by which one can instruct a computing device to perform some task.

Why Programming?

Automated solution to the problems that can be solved with the aid of computing devices.

- Less time consuming
- No errors
- Repetitive

Programming

A typical programming task can be divided into two phases:

Problem solving phase

produce an ordered sequence of steps that describe solution of problem

this sequence of steps is called an ***algorithm***

Implementation phase

implement the program in some programming language

Problem Solving

- Problem can be simple or complex
- Solutions of a problem can also be simple or complex.
- Solutions should have finite number of steps
- The steps should be sequential.

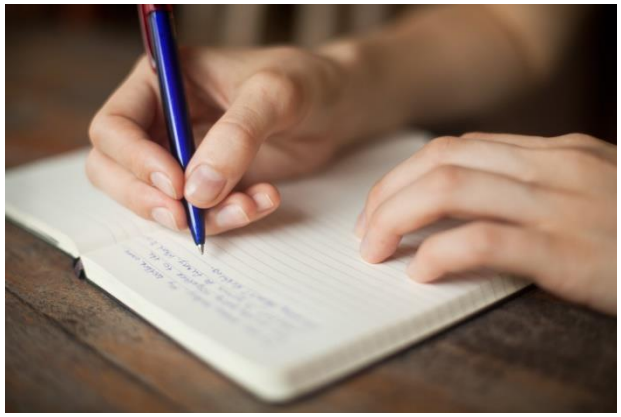
Problem Solving Example

- Writing

End result / output

Inputs

- Pen
- Paper
- Hand



- Boiling water

Inputs

- Container
- Water
- Heat / Fire

Algorithm

- An algorithm is a step by step recipe for solving an instance of a problem.
- Every single procedure that a computer performs is an outcome of some sort of algorithm.
- An algorithm is a precise procedure for solving a problem in finite number of steps.
- An algorithm states the actions to be executed and the order in which these actions are to be executed.
- An algorithm is a well ordered collection of clear and simple instructions of definite and effectively computable operations that when executed produces a result and stops executing at some point in a finite amount of time rather than just going on and on infinitely.

Algorithm

Definition

An algorithm is a well ordered collection of clear and simple instructions of definite and effectively computable operations that when executed produces a result and stops executing at some point in a finite amount of time rather than just going on and on infinitely.

Various steps in developing Algorithms

- Devising the Algorithm:

  It’s a method for solving a problem. Each step of an algorithm must be precisely defined and no vague statements should be used. Pseudo code is used to describe the algorithm , in less formal language than a programming language.

- Validating the Algorithm:

  The proof of correctness of the algorithm. A human must be able to perform each step using paper and pencil by giving the required input , use the algorithm and get the required output in a finite amount of time.

Various steps in developing Algorithms

- Expressing the algorithm:

  To implement the algorithm in a programming language.

  The algorithm used should terminate after a finite number of steps.

Algorithm Example

- **Example 1:** Write an algorithm to determine a student's final grade and indicate whether it is passing or failing. The final grade is calculated as the average of four marks.

Algorithm Example

Step1: Input 4 marks of a student

Step2: Calculate their average by summing and dividing by 4

Step3: If the average is below 50 then “FAIL” otherwise “PASS”

Algorithm Example

Algorithm

Step 1: Input M1,M2,M3,M4

Step 2: GRADE ← (M1+M2+M3+M4)/4

Step 3: if (GRADE < 50) then

Print “FAIL”

else

Print “PASS”

Step 4: End

Algorithm	Program
Design part to solve a Problem	Implementation part of the algorithm
Domain Knowledge	Programming language Knowledge
Sequential idea of the whole solution	The in depth implementation of the whole solution
Any natural languages can be used	Only programming languages are used
Analyse through pen and paper for input ranges belong to the domain.	Execute the program to get the output.
Algorithm is not Hardware or OS dependent.	Program is Hardware or OS dependent.

Efficiency of an algorithm

- Writing efficient programs is what every programmer hopes to be able to do. But what kinds of programs are efficient? The question leads to the concept of generalization of programs.
- Algorithms are programs in a general form. An algorithm is an idea upon which a program is built. An algorithm should meet three things:

  It should be independent of the programming language in which the idea is realized

  Every programmer having enough knowledge and experience to understand it

  It should be applicable to inputs of all sizes

Efficiency of an algorithm

- Efficiency of an algorithm denotes the rate at which an algorithm solves a problem of size n.
- It is measured by the amount of resources it uses, the time and the space.
- The time refers to the number of steps the algorithm executes while the space refers to the number of unit memory storage it requires.
- An algorithm’s complexity is measured by calculating the time taken and space required for performing the algorithm.
- The input size, denoted by n, is one parameter , used to characterize the instance of the problem.
- The input size m is the number of registers needed to hold the input (data segment size).

Time Complexity

- Time Complexity of an algorithm is the amount of time(or the number of steps) needed by a program to complete its task (to execute a particular algorithm)
- The way in which the number of steps required by an algorithm varies with the size of the problem it is solving. The time taken for an algorithm is comprised of two times

  Compilation Time

  Run Time

- Compilation time is the time taken to compile an algorithm. While compiling it checks for the syntax and semantic errors in the program and links it with the standard libraries , your program has asked to.

Time Complexity

- Run Time: It is the time to execute the compiled program. The run time of an algorithm depend upon the number of instructions present in the algorithm. Usually we consider, one unit for executing one instruction.
- The run time is in the control of the programmer , as the compiler is going to compile only the same number of statements , irrespective of the types of the compiler used.
- Note that run time is calculated only for executable statements and not for declaration statements
- Time complexity is normally expressed as an order of magnitude, eg $O(n^2)$ means that if the size of the problem n doubles then the algorithm will take four times as many steps to complete.

Time Complexity of an Algorithm

- Time complexity of a given algorithm can be defined for computation of function f() as a total number of statements that are executed for computing the value of f(n).
- Time complexity is a function dependent from the value of n. In practice it is often more convenient to consider it as a function from |n|
- Time complexity of an algorithm is generally classified as three types.

  (i) Worst case

  (ii) Average Case

  (iii) Best Case

Time Complexity

- Worst Case: It is the longest time that an algorithm will use over all instances of size n for a given problem to produce a desired result.

- Average Case: It is the average time(or average space) that the algorithm will use over all instances of size n for a given problem to produce a desired result. It depends on the probability distribution of instances of the problem.

- Best Case: It is the shortest time (or least space) that the algorithm will use over all instances of size n for a given problem to produce a desired result.

Space Complexity

- Space Complexity of a program is the amount of memory consumed by the algorithm (apart from input and output, if required by specification) until it completes its execution.
- The way in which the amount of storage space required by an algorithm varies with the size of the problem to be solved.
- The space occupied by the program is generally by the following:

  A fixed amount of memory occupied by the space for the program code and space occupied by the variables used in the program.

  A variable amount of memory occupied by the component variable whose size is dependent on the problem being solved. This space increases or decreases depending upon whether the program uses iterative or recursive procedures.

Space Complexity

- The memory taken by the instructions is not in the control of the programmer as its totally dependent upon the compiler to assign this memory.
- But the memory space taken by the variables is in the control of a programmer. More the number of variables used, more will be the space taken by them in the memory.
- Space complexity is normally expressed as an order of magnitude, eg $O(n^2)$means that if the size of the problem n doubles then four times as much working storage will be needed.
- There are three different spaces considered for determining the amount of memory used by the algorithm.

Space Complexity

- Instruction Space is the space in memory occupied by the compiled version of the program. We consider this space as a constant space for any value of n. We normally ignore this value , but remember that is there. The instruction space is independent of the size of the problem
- Data Space is the space in memory , which used to hold the variables , data structures, allocated memory and other data elements. The data space is related to the size of the problem.
- Environment Space is the space in memory used on the run time stack for each function call. This is related to the run time stack and holds the returning address of the previous function. The memory each function utilises on the stack is a constant as each item on the stack has a return value and pointer on it.

Algorithm Analysis

- Code each algorithm and run them to see how long they take.

- Problem: How will you know if there is a better program or whether there is no better program?

- What will happen when the number of inputs are high.

Algorithm Analysis

- Develop a model of the way computers work and compare how the algorithms behave in the model.

- Goal: To be able to predict performance at a coarse level. That is, to be able to distinguish between good and bad algorithms.

- Another benefit: when assumptions change, we can predict the effects of those changes.

Algorithm Analysis

- As computers get faster and problem sizes get bigger
  Analysis will become *more* important.

- Why?
  The difference between good and bad algorithms will get bigger.

Algorithm Analysis

How to measure the Algorithm Performance

- What metric should be used to judge algorithms?
  - Length of the program (lines of code)
  - Memory required

- Running time
  - **Running time is the dominant standard.**
    - Quantifiable and easy to compare
    - Often the critical bottleneck