BSE202L Data Structures and Algorithms

Module-1: Algorithms, Analysis, Space/time complexity

Why Data Structures and Algorithms?

- To solve computational problems
- DS + Algo = Code
- Efficiency of algorithms is critical
- How to compute efficiency?
- By analysing the algorithms

Data structures

- Data is the key in any computational task
  - Examples?
- Tools for Organized storage, manipulation and retrieval of data
  - Primitive- Integer, floating point/double, character, pointers etc.
  - Non primitive
    - Linear/sequential- Array, List, Stack, Queue
    - Non-sequential/ random- Trees and Graphs
- Operations to access and modify data can be performed on data structures
  - Eg.:- Insert, delete, pop, enqueue, preorder, BFS
- Algorithms in this course are used to perform these operations on data structures

Algorithm History

- The word Algorithm has its roots on the Persian mathematician, Abdullah Jafar Muhammad ibn Musa Al-khowarizmi in ninth century, who defined algorithm as:
- An algorithm is a set of rules for carrying out calculation either by hand or on a machine
- An algorithm is a well-defined computational procedure that takes input and produces output
- An algorithm is a finite sequence of instructions or steps to achieve a particular output

Algorithms

- Step-by-step procedure for solving a (well-specified computational) problem
- Input to output transformation
- Written in simple English (On paper, even!!)
- Algorithm has no fixed syntax
- { } to enclose block of statements or *begin* and *end*
- For assignment, := or = or ←
- Pseudocode is widespread, which borrows syntax from popular programming languages

Characteristics of Algorithm

- Input: It generally requires finite no. of inputs (can be 0 as well)
- Output: It must produce at least one output
- Definiteness: Each instruction should be clear and unambiguous
- Finiteness: It must terminate after a finite no. of steps
- Correctness: For every input instance, it must halt with the correct output
- Effectiveness:

Analysis of Algorithms

- Measuring the efficiency of an algorithm
- Two parameters
  - Time: How long the algorithm takes to run (running time, of course as a program)
  - Space: Memory requirement

Time and Space

- Time depends on processing speed
  - Impossible to change for a given hardware
- Space is a function of available memory
  - Easier to reconfigure, augment
- Typically we focus more on time less on space

Measuring running time

- Analysis independent of underlying hardware
  - Don’t use actual units time (clock time)
  - Measure in terms of basic operations
- Typical basic operations
  - Compare two values
  - Assign a value to a variable
- Some operations are basic depending on the context
  - Exchange values of a pair of variables

Analysis -- Complexity of the Algorithm

- Time Complexity:- Number of computational steps/operations to be performed to complete the algorithm
- Obtained as time function
- Check whether it depends on input size (Very important!!) and how the dependence is
- Typically written as T(n) where n is the input size

Dependence on Input size- Example

- Sorting an array with n elements:
  - Naïve algorithms: time proportional to n^2
  - Best algorithms: time proportional to n logn
- How important this distinction?
- Current day, a typical CPU processes 10^8 operations per second

Impact of input size

- Sorting all mobile phone numbers in India
- India has about 10^9 phones
- Naïve n^2 algorithm requires $(10^9)^2 = 10^{18}$ operations
  - 10^8 operations per second ➔ $10^{18}/10^8 = 10^{10}$ seconds
  - 2778000 hours
  - 115700 days
  - 300 years

Impact on input size

- Smarter n log n algorithm takes $10^9 \log 10^9$ operations
- 10^{18} operations per second → $(10^9 * 30)/(10^8)$ seconds
- 300 seconds
- 5 minutes

Thank you..