TEG

By-Debajyoti Dutta

DATA STRUCTURES AND ALGORITHM

INTRODUCTION

LEARNING OBJECTIVE

In this chapter, we are going to discuss common data structures and algorithms which serve as building blocks for creating efficient programs. We will also discuss different approaches to designing algorithms and different notations for evaluating the performance of algorithms.

WHAT IS DATA STRUCTURE?

A data structure is a particular way of storing and organizing data either in a computer's memory or on the disk storage so that it can be used efficiently.

USE CASES

- Compiler design
- Operating system
- Statistical analysis package
- DBMS
- Numerical analysis
- Simulation
- Artificial intelligence
- Graphics

TYPES OF DATA STRUCTURE

- Primitive
  - int
  - float
  - double
  - pointer
- Non-primitive
  - Linear
    - Array
    - List
      - Single linked list
      - Double linked list
      - Circular linked list
      - Doubly circular linked list
    - Stack
    - Queue
  - Non-Linear
    - Tree
      - Binary tree
      - Binary search tree
      - Expression tree
      - AVL tree

        - Threaded Binary tree
        - B tree
        - B+ tree
    - Graph
        - Directed graph
        - Undirected graph

OPERATIONS ON DATA STRUCTURES

1. **Traversing** It means to access each data item exactly once so that it can be processed.
2. **Searching** It is used to find the location of one or more data items that satisfy the given constraint.
3. **Inserting** It is used to add new data items to the given list of data items.
4. **Deleting** It means to remove (delete) a particular data item from the given collection of data items.
5. **Sorting** Data items can be arranged in some order like ascending order or descending order depending on the type of application.
6. **Merging** Lists of two sorted data items can be combined to form a single list of sorted data items.

ABSTRACT DATA TYPE

An abstract data type (ADT) is the way we look at a data structure, focusing on what it does and ignoring how it does its job.

WHAT IS ALGORITHM?

An algorithm is basically a set of instructions that solve a problem.

The typical definition of algorithm is ‘a formally defined procedure for performing some calculation’.It is considered to be an effective procedure for solving a problem in finite number of steps.Algorithms are mainly used to achieve software reuse.

ALGORITHM PROPERTIES

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

DIFFERENT APPROACHES TO DESIGNING AN ALGORITHM

- **Top-down approach** Top-down design method is a form of stepwise refinement where we begin with the topmost module and incrementally add modules that it calls.
- **Bottom-up approach** A bottom-up approach is just the reverse of top-down approach. In the bottom-up design, we start with designing the most basic or concrete modules and then proceed towards designing higher level modules.

COMPLEXITY

Analysing an algorithm means determining the amount of resources (such as time and memory) needed to execute it.The efficiency or complexity of an algorithm is stated in terms of time and space complexity.

Time complexity :- The time complexity of an algorithm is basically the running time of a program as a function of the input size.

Space complexity :- The space complexity of an algorithm is the amount of computer memory that is required during the program execution as a function of the input size.

Generally, the space needed by a program depends on the following two parts:

- **Fixed part:** It varies from problem to problem. It includes the space needed for storing instructions, constants, variables, and structured variables (like arrays and structures).

- **Variable part:** It varies from program to program. It includes the space needed for recursion stack, and for structured variables that are allocated space dynamically during the runtime of a program.

Worst-case, Average-case, and Best-case Time Complexity

- **Worst-case :-** This denotes the behaviour of an algorithm with respect to the worst possible case of the input instance. The worst-case running time of an algorithm is an upper bound on the running time for any input.It is expressed as O(n) [Big O notation]
- **Best-case :-** The term 'best-case performance' is used to analyse an algorithm under optimal conditions.It is expressed as Ω(n) [Omega notation]
- **Average-case :-** The average-case running time of an algorithm is an estimate of the running time for an 'average' input.It is expressed as Θ(n) [Theta notation]