Data Structures and Algorithms

Dr. L. Rajya Lakshmi

Data Modelling/Data Structuring

- Representation of data elements and relationship between them
- Data Structure: data representation and its associated operations
- Examples: Integers, float point numbers
- A data structure meant to be an organization or structuring for a collection of data items
- Example: An ordered list of integers stored in an array
  - Search
  - Print and process
  - Modification

Data Structures

- Data Structure: A scheme/particular way of organizing data in the memory of a computer
- Different kinds of data structures are suited for different applications
  - For databases B-Trees
  - For Compilers Hash Tables
- Data structures are used in almost every program or software systems
- These are essential components of many efficient algorithms and make management of huge amounts of data possible
  - Large databases
  - Internet indexing

Data Structures

- The logical or mathematical model of a particular organization of data is called a data structure
- A data model/data structure depends on two things:
  - It must mirror the actual relationship of the data items in the real world
  - It must be simple to process the data when necessary

Motivation

- Consider a program that reads a triangle and outputs the area
- No variable of type triangle can be defined and read
- This program needs new data structures, since there is no data structure for triangle
- A triangle is a set of three points in the plane and a point in the plane can be represented by its x and y coordinates
- Declare a structure for point which consists of x and y coordinates of a point as its fields
- Triangle is a structure that consists of three fields for its three corners
- Using built-in types defined more complex types

Motivation

- There are no functions to read in points and triangles
- Define functions for reading in points and triangles
- There is no function for computing area of a triangle
- Define a function for the same
- Using built-in types we may have to write more complicated types or structures according to the requirement
- If required, we may write functions for printing points and triangles

Motivation

- While solving a programming problem, we need to identify the data structures and algorithms that are required to solve the problem
- In the triangle area problem, two new data structures and an algorithm to find the area of the triangle are required
- Different data structures may be defined to solve the same problem and different implementations are possible for the same data structure or algorithm
- For example, we can use “r” theta representation to represent point
- A triangle can be represented by its sides instead of its corners
- Using the data structure triangle, we can write many other functions that solve various problems related to triangles

Selection of a data structure

- Nature of the data and the operations/processes that need to be performed on the data decide the selection
- The steps to be follow while selecting a data structure are:
  - Problem analysis and identification of the operations that need to be supported
  - Quantify of resource constraints for each operation
  - Select the data structure that meets these requirements in the best possible way

Selection of a data structure

- Examples of basic operations: inserting a data item into a data structure, deleting a data item from a data structure, and finding a specific data item
- Resource constraints on certain key operations such as search, inserting data records, and deletion data records, drive the data structure selection process
- Many issues relating to relative importance of these operations are addressed by the following questions:
    - Are all data items are inserted into the data structure at the beginning or are the insertions interspersed with other operations?
    - Can data items be deleted?
    - Are all data items processed in some well-defined order, or is search for specific items allowed?

Overview

- The study of a data structure consists of three steps:
  1. Logical or mathematical description of the structure
  2. Implementation of the structure on a computer
  3. Quantitative analysis of the structure, which includes determining the amount of memory needed to store the structure and the time required to process the structure and perform the required operations

Overview

- Linear data structures such as: Lists, restricted access lists (stacks and queues)
- Non-linear data structures: Trees, heaps, tries, graphs
- Dictionaries: Sorting, searching, hashing, hash tables, Bloom filters

Overview

- Implementation issues:
  - recursive and iterative implementation, dynamic allocation
  - Ex: Arrays or linked lists

Overview

- Performance analysis: A study of efficiency of a data structure and an algorithm
- Efficiency in terms of space analysis and time analysis
- Efficiency of algorithms as a function of the input size
  - The running will increases with the size of the input
  - Study the growth of the running time with the input size
- How to measure the running time?
  - Can be done using an experimental study
  - Implement your algorithm in certain platform and measure its running time (using a system clock) on different types of inputs
  - Based on this analysis make conclusions about the running time

Overview

- There are few limitations
  - Have to implement the algorithm
  - Study can only be done on the subset of input, which may not give a proper indication of the running time of your algorithm
  - To make a choice between two algorithms, your experimental study should be based on the same platform
- It is advantages to have a general methodology to analyse the running time of an algorithm
- How this methodology should work?
  - Take a high level description (pseudo-code) of an algorithm
  - Take all possible inputs into consideration
  - Allow the analysis independent of the software and hardware platform

Overview: Pseudo-code

- What is pseudo-code?
  - A mixture of natural language and high-level programming concepts that describe the main ideas behind a general implementation of any data structure or algorithm
  - Example:

```
Algorithm arrayMax(A, n)
Input: An array A storing n integers and the size
Output: The maximum element in A
currentMax = A[0];
for i=1 to n-1 do
        if currentMax < A[i] then
                currentMax = A[i];
return currentMax
```

Abstraction

- An abstract data type is a logical description of how we view the data and the set of operations allowed on that data without regard to how they will be implemented
- An abstract data type (ADT) is a set of objects together with a set of operations
- It is a method to bring flexibility to your logical data model
- We are concerned with what the data is representing and not with how it will eventually be constructed
- With this level of abstraction we are creating an encapsulation around the data
- By encapsulating implementation details, we are hiding them from the user’s view

Abstraction

- To implementation an ADT (also referred as a data structure), we have to provide a physical view of the data using a collection of programming constructs and primitive data types
- We provide a separation between logical view and physical view of the data
- Provides implementation-independent view of the data
- Provides freedom to the programmer to switch the implementation details without changing the way interaction for the data
- The user can focus on the problem solving process

Abstraction

- The implementation details of the methods are not mentioned in an ADT’s definition
- Objects such as lists, sets, and graphs along with their operations can be viewed as ADTs
- Operations to be supported by an ADT is a design decision