Unit 8 Overview of Data Structure

Structure:

8.1 Introduction

A data structure is the organization of data in a computer's memory or in a file. The proper choice of a data structure can lead to more efficient programs.

In computer science, a **data structure** is a way of storing data in a computer so that it can be used efficiently. It is an organization of mathematical and logical concepts of data. Often a carefully chosen data structure will allow the most efficient algorithm to be used. The choice of the data structure often begins from the choice of an abstract data type. A well-designed data structure allows a variety of critical operations to be performed, using as few resources, both execution time and memory space, as possible. Data structures are implemented by a programming language as data types and the references and operations they provide.

At the lowest level, there are data structures supplied and supported by the CPU (or computer chip), itself.

At the second level of the data structures spectrum are the data structures supported by particular programming languages. These vary a lot from language to language.

Definition: A data structure is a representation of data and the operations allowed on that data.

A specification, application and implementation are the view of a collection of one or more items of data, and the operations necessary and sufficient to interact with the collection.

The specification is the definition of the data structure as an abstract data type. The specification forms the programming interface for the data structure.

The application level is a way of modeling real-life data in a specific context. It is happening especially for the memory management.

Objectives:

At the end of this unit, you will be able to:

- present a brief idea of Data Structure.
- explain Abstract Data Type .
- explain different types of Data Structures.
- describe the applications of Data Structures.

8.2 Examples of Data Structure

The following is a list of data structures used in application systems:

a) array
b) stack
c) queue
d) linked list
e) binary tree

f) hash table
g) heap
h) graph

Data structures are often used to build databases. Typically, data structures are manipulated using various algorithms.

8.3 Abstract Data Types (ADT)

In computing, an **abstract data type** (**ADT**) is a specification of a set of data and the set of operations that can be performed on the data. Such a data type is abstract in the sense that it is independent of various concrete implementations. The definition can be mathematical, or it can be programmed as an interface. A first class ADT supports the creation of multiple instances of the ADT, and the interface normally provides a *constructor*, which returns an abstract handle to new data, and several *operations*, which are functions accepting the abstract handle as an argument.

When primitive data types are not sufficient to handle the data, we have to define some data type along with its activities, storage structures and algorithms. The ADT is the data type which has the following components:

1) **Operations:** Specifications of external appearance of a data structure.
2) **Storage Structures:** Organizations of data implemented in lower-level data structures.
3) **Algorithms:** Description on how to manipulate information in the storage structures to obtain the results defined for the operations.

8.4 Levels of Data Structure

The Abstract Level

The abstract (or logical) level is the specification of the data structure - the "what" but not the "how". At this level, the user or data structure designer is

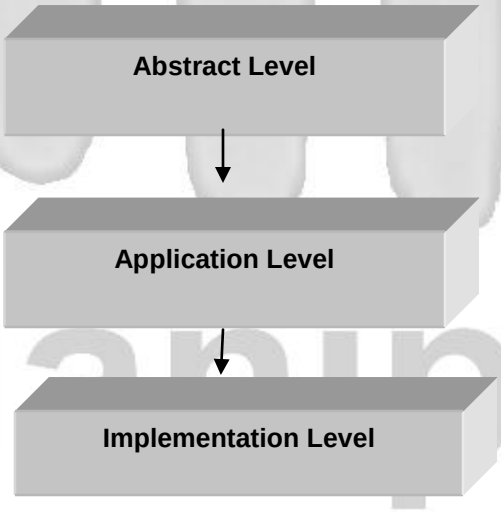
free to think outside the bounds of any programming language. For instance, a linear list type would consist of a collection of list nodes such that they formed a sequence. The operations defined for this list might be insert, delete, sort and retrieve.

The Application Level

At the application or user level, the user is modeling real-life data in a specific context. In our list example we might specify what kinds of items were stored in the list and how long the list is. The context will determine the definitions of the operations. For example, if the list was a list of character data, the operations would have a different meaning than if we were talking about a grocery list.

The Implementation Level

The implementation level is where the model becomes compiling and generates executable code. We need to determine where the data will reside and allocate space in that storage area. We also need to create the sequence of instructions that will cause the operations to perform as specified.



Data Structure Level Diagram

Fig. 8.1

8.5 Types of Data Structure

There are two types of data structures

a) Linear Data Structure

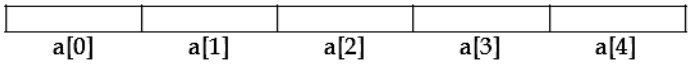
A data structure is said to be *linear if its elements form a* sequence or a linear list.

Examples:

- Arrays
- Linked Lists
- Stacks
- Queues

Pictorial Representation of Array

One Dimension



Two Dimension

a[0][0]	a[0][1]	a[0][2]	a[0][3]	a[0][4]
a[1][0]	a[1][1]	a[1][2]	a[1][3]	a[1][4]
a[2][0]	a[2][1]	a[2][2]	a[2][3]	a[2][4]

Fig. 8.2

Manipal

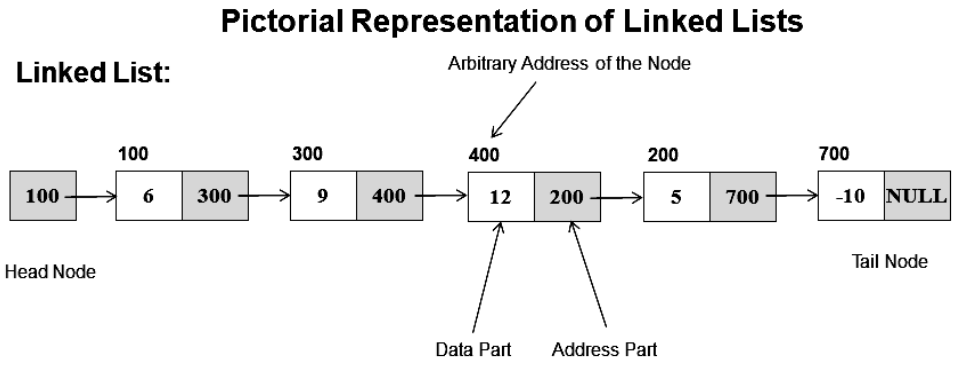


Fig. 8.3

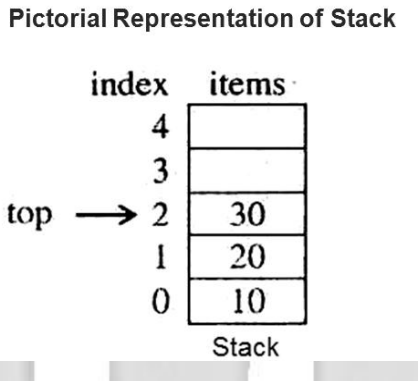


Fig. 8.4

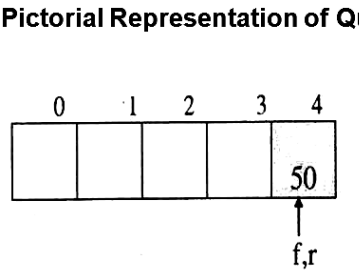


Fig. 8.5

Manipal

b) Nonlinear Data Structure

A data structure is said to be *nonlinear if its elements cannot form a* sequence or a linear list.

Examples:

- Trees
- Graphs

Pictorial Representation of Tree

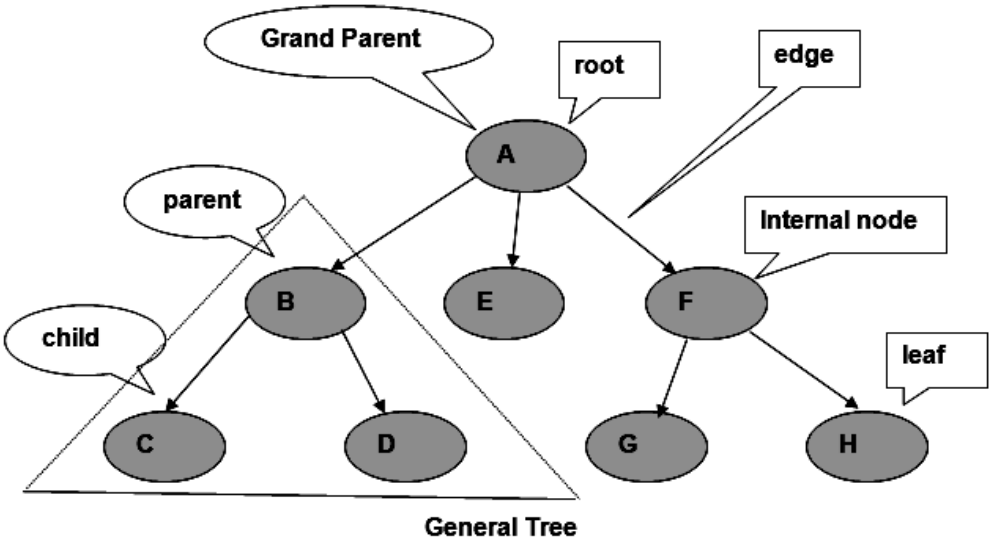


Fig. 8.6

Pictorial Representation of Graph

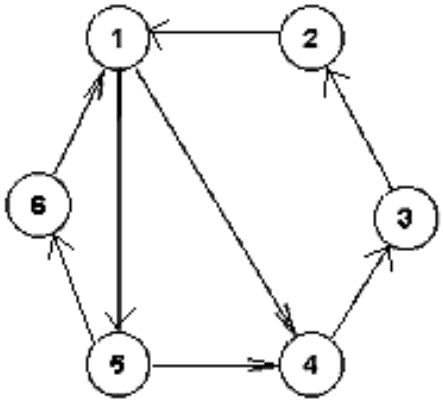


Fig. 8.7

8.6 Applications of Data Structure

- **Traversal:** Travel through the data structure, travelling salesman problem, shortest path problem.
- **Search:** Traversal through the data structure for a given element, appropriate searching methodology, searching a number from thousands of random numbers.
- **Insertion:** Adding new elements to the data structure, constructing index tree, B-tree, insertion of elements into the sorted database.
- **Deletion:** Removing an element from the data structure, garbage collection, memory utilization and optimization.
- **Sorting:** Arranging the elements in some order, appropriate sorting method for appropriate architecture.
- **Merging:** Combining two similar data structures into one

Examples:

1. Adding of two large numbers.
2. Finding shortest path between two cities among number of paths.
3. Designing of Integrated Circuit (IC).
4. Preparation of game planning / meeting conduction with a number common members in minimum durations.
5. Searching of records from large database.
6. Data mining application.
7. Execution of large file in relatively very small memory.
8. Managing of Dead Lock situation of Operating System.
9. Selection of Routing path in Network & Mobile Communication.
10. Resource utilization of Operating System.

8.7 Summary

A data structure is a representation of data and the operations allowed on that data. The proper choice of a data structure can lead to more efficient programs. In computer science, a data structure is a way of storing data in a computer so that it can be used efficiently. It is an organization of mathematical and logical concepts of data. Often a carefully chosen data structure will allow the most efficient algorithm to be used. A well-designed data structure allows a variety of critical operations to be performed, using as few resources, both execution time and memory space, as possible. Abstract Data Type, This unit covers the concept of data structure.

Self Assessment Questions

1. A data structure is a representation of data and the operations allowed on that data. (True / False)
2. Array is an example of data structure. (True / False)
3. ADT means Advance Data Type. (True / False)
4. Tree is the linear data structure. (True / False)

8.8 Terminal Questions

1. What is a Data Structure?
2. Define Abstract Data Type.
3. Differentiate between linear and nonlinear data structure.
4. How many levels are there in Data Structure? Describe.

8.9 Answers to Self Assessment and Terminal Questions

Self Assessment Questions

1. True
2. True
3. False
4. False

Terminal Questions

1. Section 8.1
2. Section 8.3
3. Section 8.5
4. Section 8.4

Manipal