Introduction to Data Structures

A data structure is a model where data is organized, managed and stored in a format that enables efficient access and modification of data. There are various types of data structures commonly available. It is up to the programmer to choose which data structure to use depending on the data.

The choice of a particular one can be considered based on the following points:

It must be able to process the data efficiently when necessary.
It must be able to represent the inherent relationship of the data in the real world.

Why study Data Structures?

"Bad programmers worry about the code. Good programmers worry about data structures and their relationships."

Linus Torvalds

Time and energy are both required to process any instruction. Every CPU cycle that is saved will have an effect on both the

time and energy consumed and can be put to better use in processing other instructions.

A program built using improper data structures will be therefore inefficient or unnecessarily complex. It is necessary to have a good knowledge of data structures and understand where to use the best one. The study includes the description, implementation and quantitative performance analysis of the data structure.

Concept of a Data Type

Primitive Data Types

A primitive data type is one that is inbuilt into the programming language for defining the most basic types of data. These may be different for the various programming languages available. For example, the C programming language has inbuilt support for characters (char), integers (int, long) and real numbers (float, double).

User-Defined Data Type

User-defined data type, as the name suggests is the one that

the user defines as per the requirements of the data to be stored. Most programming languages provide support for creating user-defined data types. For example, C provides support through structures (struct), unions (union) and enumerations (enum).

Abstract Data Type (ADT)

Abstract Data Types are defined by its behaviour from the point of view of the user of the data. It defines it in terms of possible values, operations on data, and the behaviour of these operations.

The definition of ADT only mentions what operations are to be performed but not how these operations will be implemented. That is, it does not specify how the data is being handled under the hood. This concept is known as abstraction.

For eg. The user of the stack data structure only knows about the push and pop operations in a stack. They do not care how the push operation interacts with the memory to store the data. They only expect it to store it in the way specified.

Common Operations in a Data Structure

A data structure is only useful when you can perform operations on it, right? These are the basic operations that should be able to be performed on every data structure.

Access

This operation handles how the elements currently stored in the structure can be accessed.

Search

This operation handles finding the location of a given element of a given structure.

Insertion

This operation specifies how new elements are to be added to the structure.

Deletion

This operation specifies how existing elements can be removed from the structure.

Classification of data structure

A data structure can be broadly classified into 2 types:

Linear Data Structures

A linear data structure's elements form a sequence. Every element in the structure has some element before and after it. Examples of linear structures are:

Arrays

An array holds a fixed number of similar elements that are stored under one name. These elements are stored in contagious memory locations The elements of an array can be accessed using one identifier.

Linked Lists

A linked list is a linear data structure where each element is a separate object, known as a node Each node contains some data and points to the next node in the structure, forming a sequence .

Stacks

Stacks are a type of linear data structures that store data in an order known as the Last In First Out (LIFO) order. This property is helpful in certain programming cases where the data needs to be ordered.

Queues

Queues are a type of linear data structures that store data in an order known as the First In First Out (FIFO) order. This property is helpful in certain programming cases where the data needs to be ordered.

Non-Linear Data Structures

A non-linear data structure's elements do not form a sequence. Every element may not have a unique element before and after it.

Trees

A tree is a data structure that simulates a hierarchical tree, with a root value and the children as the subtrees, represented

by a set of linked nodes.

Heaps

A heap is a specialized tree-based data structure that satisfies the heap property.

Graphs

A graph data structure is used to represent relations between pairs of objects. It consists of nodes (known as vertices) that are connected through links (known as edges). The relationship between the nodes can be used to model the relation between the objects in the graph.

Hash Tables

A Hash Table is a data structure where data is stored in an associative manner. The data is mapped to array positions by a hash function that generates a unique value from each key. The value stored in a hash table can then be searched in O(1) time using the same hash function which generates an address from the key.