IJARSCT

ISSN (Online) 2581-9429

International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)

International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal

Impact Factor: 7.301

Volume 3, Issue 8, January 2023

Data Structures

Janhavi Padhya and Ms. Anjali Yadav

Shri G. P. M. Degree College, Vile Parle (E), Mumbai, Maharashtra, India

Abstract: *A data structure is a specialized format for organizing, processing, retrieving and storing data. There are several basic and advanced types of data structures, all designed to arrange data to suit a specific purpose. Data structures make it easy for users to access and work with the data they need in appropriate ways*

Keywords: data structure

I. INTRODUCTION

A data structure is a specialized format for organizing, processing, retrieving and storing data. There are several basic and advanced types of data structures, all designed to arrange data to suit a specific purpose. Data structures make it easy for users to access and work with the data they need in appropriate ways

OVERVIEW:

A data structure is a particular way of organizing data in a computer so that it can be used effectively. The idea is to reduce the space and time complexities of different tasks.

Classification/Types of Data Structures:

Linear Data Structure

Non-Linear Data Structure.

Linear Data Structure:

Elements are arranged in one dimension, also known as linear dimension.

Example: lists, stack, queue, etc.

Non-Linear Data Structure

Elements are arranged in one-many, many-one and many-many dimensions.

Example: tree, graph, table, etc.

DETAIL INFORMATION:

A data structure is a specialized format for organizing, processing, retrieving and storing data. There are several basic and advanced types of data structures, all designed to arrange data to suit a specific purpose. Data structures make it easy for users to access and work with the data they need in appropriate ways. Most importantly, data structures frame the organization of information so that machines and humans can better understand it.

In computer science and computer programming, a data structure may be selected or designed to store data for the purpose of using it with various algorithms. In some cases, the algorithm's basic operations are tightly coupled to the data structure's design. Each data structure contains information about the data values, relationships between the data and -- in some cases -- functions that can be applied to the data.

For instance, in an object-oriented programming language, the data structure and its associated methods are bound together as part of a class definition. In non-object-oriented languages, there may be functions defined to work with the data structure, but they are not technically part of the data structure.

Why are data structures important?

Typical base data types, such as integers or floating-point values, that are available in most computer programming languages are generally insufficient to capture the logical intent for data processing and use. Yet applications that

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
Impact Factor: 7.301
Volume 3, Issue 8, January 2023

ingest, manipulate and produce information must understand how data should be organized to simplify processing. Data structures bring together the data elements in a logical way and facilitate the effective use, persistence and sharing of data. They provide a formal model that describes the way the data elements are organized.

Data structures are the building blocks for more sophisticated applications. They are designed by composing data elements into a logical unit representing an abstract data type that has relevance to the algorithm or application. An example of an abstract data type is a "customer name" that is composed of the character strings for "first name," "middle name" and "last name."

It is not only important to use data structures, but it is also important to choose the proper data structure for each task. Choosing an ill-suited data structure could result in slow runtimes or unresponsive code. Five factors to consider when picking a data structure include the following:

What kind of information will be stored?

How will that information be used?

Where should data persist, or be kept, after it is created?

What is the best way to organize the data?

What aspects of memory and storage reservation management should be considered?

How are data structures used?

In general, data structures are used to implement the physical forms of abstract data types. Data structures are a crucial part of designing efficient software. They also play a critical role in algorithm design and how those algorithms are used within computer programs.

Early programming languages -- such as Fortran, C and C++ -- enabled programmers to define their own data structures. Today, many programming languages include an extensive collection of built-in data structures to organize code and information. For example, Python lists and dictionaries, and JavaScript arrays and objects are common coding structures used for storing and retrieving information.

Software engineers use algorithms that are tightly coupled with the data structures -- such as lists, queues and mappings from one set of values to another. This approach can be fused in a variety of applications, including managing collections of records in a relational database and creating an index of those records using a data structure called a binary tree.

Some examples of how data structures are used include the following:

Storing data. Data structures are used for efficient data persistence, such as specifying the collection of attributes and corresponding structures used to store records in a database management system.

Managing resources and services. Core operating system (OS) resources and services are enabled through the use of data structures such as linked lists for memory allocation, file directory management and file structure trees, as well as process scheduling queues.

Data exchange. Data structures define the organization of information shared between applications, such as TCP/IP packets.

Ordering and sorting. Data structures such as binary search trees -- also known as an ordered or sorted binary tree -- provide efficient methods of sorting objects, such as character strings used as tags. With data structures such as priority queues, programmers can manage items organized according to a specific priority.

Indexing. Even more sophisticated data structures such as B-trees are used to index objects, such as those stored in a database.

Searching. Indexes created using binary search trees, B-trees or hash tables speed the ability to find a specific sought-after item.

Scalability. Big data applications use data structures for allocating and managing data storage across distributed storage locations, ensuring scalability and performance. Certain big data programming environments -- such as Apache Spark -- provide data structures that mirror the underlying structure of database records to simplify querying.

Characteristics of data structures

Data structures are often classified by their characteristics. The following three characteristics are examples:

Linear or non-linear. This characteristic describes whether the data items are arranged in sequential order, such as with an array, or in an unordered sequence, such as with a graph.

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
Impact Factor: 7.301
Volume 3, Issue 8, January 2023

Homogeneous or heterogeneous. This characteristic describes whether all data items in a given repository are of the same type. One example is a collection of elements in an array, or of various types, such as an abstract data type defined as a structure in C or a class specification in Java.

Static or dynamic. This characteristic describes how the data structures are compiled. Static data structures have fixed sizes, structures and memory locations at compile time. Dynamic data structures have sizes, structures and memory locations that can shrink or expand, depending on the use.

Data types

If data structures are the building blocks of algorithms and computer programs, the primitive -- or base -- data types are the building blocks of data structures. The typical base data types include the following:

Boolean, which stores logical values that are either true or false.

integer, which stores a range on mathematical integers -- or counting numbers. Different sized integers hold a different range of values -- e.g., a signed 8-bit integer holds values from -128 to 127, and an unsigned long 32-bit integer holds values from 0 to 4,294,967,295.

Floating-point numbers, which store a formulaic representation of real numbers.

Fixed-point numbers, which are used in some programming languages and hold real values but are managed as digits to the left and the right of the decimal point.

Character, which uses symbols from a defined mapping of integer values to symbols.

Pointers, which are reference values that point to other values.

String, which is an array of characters followed by a stop code -- usually a "0" value -- or is managed using a length field that is an integer value.

The data structure hierarchy shows how data types and data structures are related.

Types of data structures

The data structure type used in a particular situation is determined by the type of operations that will be required or the kinds of algorithms that will be applied. The various data structure types include the following:

Array. An array stores a collection of items at adjoining memory locations. Items that are the same type are stored together so the position of each element can be calculated or retrieved easily by an index. Arrays can be fixed or flexible in length.

An array can hold a collection of integers, floating-point numbers, stings or even other arrays.

Stack. A stack stores a collection of items in the linear order that operations are applied. This order could be last in, first out (LIFO) or first in, first out (FIFO).

Queue. A queue stores a collection of items like a stack; however, the operation order can only be first in, first out.

Linked list. A linked list stores a collection of items in a linear order. Each element, or node, in a linked list contains a data item, as well as a reference, or link, to the next item in the list.

Linked list data structures are a set of nodes that contain data and the address or a pointer to the next node.

Tree. A tree stores a collection of items in an abstract, hierarchical way. Each node is associated with a key value, with parent nodes linked to child nodes -- or sub nodes. There is one root node that is the ancestor of all the nodes in the tree.

A binary search tree is a set of nodes where each has a value and can point to two child nodes.

Heap. A heap is a tree-based structure in which each parent node's associated key value is greater than or equal to the key values of any of its children's key values.

Graph. A graph stores a collection of items in a nonlinear fashion. Graphs are made up of a finite set of nodes, also known as vertices, and lines that connect them, also known as edges. These are useful for representing real-world systems such as computer networks.

Trie. A trie, also known as a keyword tree, is a data structure that stores strings as data items that can be organized in a visual graph.

Hash table. A hash table -- also known as a hash map -- stores a collection of items in an associative array that plots keys to values. A hash table uses a hash function to convert an index into an array of buckets that contain the desired data item.

Hashing is a data structure technique where key values are converted into indexes of an array where the data is stored.

These are considered complex data structures as they can store large amounts of interconnected data.

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
Impact Factor: 7.301
Volume 3, Issue 8, January 2023

How to choose a data structure?
When choosing a data structure for a program or application, developers should consider the answers to the following three questions:
Supported operations. What functions and operations does the program need?
Computational complexity. What level of computational performance is tolerable? For speed, a data structure whose operations execute in time linear to the number of items managed -- using Big O Notation: O(n) -- will be faster than a data structure whose operations execute in time proportional to the square of the number of items managed -- O(n^2).
Programming elegance. Are the organization of the data structure and its functional interface easy to use?
Some real-world examples include:
Linked lists are best if a program is managing a collection of items that don't need to be ordered, constant time is required for adding or removing an item from the collection and increased search time is OK.
Stacks are best if the program is managing a collection that needs to support a LIFO order.
Queues should be used if the program is managing a collection that needs to support a FIFO order.
Binary trees are good for managing a collection of items with a parent-child relationship, such as a family tree.
Binary search trees are appropriate for managing a sorted collection where the goal is to optimize the time it takes to find specific items in the collection.
Graphs work best if the application will analyze connectivity and relationships among a collection of individuals in a social media network.
This was last updated in March 2021
Continue Reading About data structures
Redis aims for an infinite variety of data structures
The rise of multi-model databases to support data variety
An enterprise architects guide to the data modeling process
How the SHA-3 competition declared a winning has function
7 best courses to learn data structure and algorithms
Related Terms
C++
C++ is an object-oriented programming (OOP) language that is viewed by many as the best language for creating large-scale ... See complete definition
data lakehouse
A data lakehouse is a data management architecture that combines the key features and the benefits of a data lake and a data ... See complete definition
tuple
A tuple, pronounced TUH-pull, is an ordered and finite list of elements in various fields of interest, including computing. See complete definition

II. CONCLUSION

This course covered the basics of data structures. With this we have only scratched the surface. Although we have built a good foundation to move ahead.
Data Structures is not just limited to Stack, Queues, and Linked Lists but is quite a vast area. There are many more data structures which include Maps, Hash Tables, Graphs, Trees, etc. Each data structure has its own advantages and disadvantages and must be used according to the needs of the application. A computer science student at least knows the basic data structures along with the operations associated with them.
Many high level and object oriented programming languages like C#, Java, Python come built in with many of these data structures. Therefore, it is important to know how things work under the hood.

REFERENCES

[1]. Author:Klaus Samelson and Friedrich L. Bauer Year: 1957 Uses: Algorithms

IJARSCT ISSN (Online) 2581-9429

International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)

International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal

Impact Factor: 7.301

Volume 3, Issue 8, January 2023

[2]. https://www.techtarget.com/searchdatamanagement/definition/data-structure

[3]. https://www.geeksforgeeks.org/introduction-to-data-structures/

[4]. :https://www.techtarget.com/searchdatamanagement/definition/data-structure

[5]. https://www.codecademy.com/resources/blog/why-data-structures/