

Data Structure: Theoretical Approach

Durgesh Raghuvanshi

B-Tech Department of Computer Science,
IILM Academy of Higher Learning, Greater Noida, Uttar Pradesh, India

19 May 2022

ABSTRACT

Run with accordance with significance. The first if these this paper explains about the basic terminologies used in this paper in data structure. Better running times will be other constraints, such as memory use which will be paramount. The most appropriate data structures and algorithms rather than through hacking removing a few statements by some clever coding. Data structures serve as the basis for abstract data types (ADT). "The ADT defines the logical form of the data type. The data structure implements the physical form of the data type." Different types of data structures are suited to different kinds of applications, and some are highly specialized to specific tasks. For example, relational databases commonly use B-tree indexes for data retrieval, while compiler implementations usually use hash tables to look up identifiers.

INTRODUCTION

Data structures serve as the basis for abstract data types (ADT). "The ADT defines the logical form of the data type. The data structure implements the physical form of the data type." Different types of data structures are suited to different kinds of applications, and some are highly specialized to specific tasks. For example, relational databases commonly use B-tree indexes for data retrieval, while compiler implementations usually use hash tables to look up identifiers. Data structures provide a means to manage large amounts of data efficiently for uses such as large databases and internet indexing services. Usually, efficient data structures are key to designing efficient algorithms. Some formal design methods and programming languages emphasize data structures, rather than algorithms, as the key organizing factor in software design. Data structures can be used to organize the storage and retrieval of information stored in both main memory and secondary memory. Data structures are generally based on the ability of a computer to fetch and store data at any place in its memory, specified by a pointer—a bit string, representing a memory address, that can be itself stored in memory and manipulated by the program. Thus,

the array and record data structures are based on computing the addresses of data items with arithmetic operations, while the linked data structures are based on storing addresses of data items within the structure itself. Many data structures use both principles, sometimes combined in non-trivial ways (as in XOR linking).[citation needed] The implementation of a data structure usually requires writing a set of procedures that create and manipulate instances of that structure. The efficiency of a data structure cannot be analyzed separately from those operations. This observation motivates the theoretical concept of an abstract data type, a data structure that is defined indirectly by the operations that may be performed on it, and the mathematical properties of those operations (including their space and time cost).[citation needed] An array is a number of elements in a specific order, typically all of the same type (depending on the language, individual elements may either all be forced to be the same type, or may be of almost any type). Elements are accessed using an integer index to specify which element is required. Typical implementations allocate contiguous memory words for the elements of arrays (but this is not necessity). Arrays may be fixed-length or resizable. A linked list (also just called list) is a linear collection of data elements of any type, called nodes, where each node has itself a value, and points to the next node in the linked list. The principal advantage of a linked list over an array, is that values can always be efficiently inserted and removed without relocating the rest of the list. Certain other operations, such as random access to a certain element, are however slower on lists than on arrays. Most assembly languages and some low-level languages, such as BCPL (Basic Combined Programming Language), lack built-in support for data structures. On the other hand, many high-level programming languages and some higher-level assembly languages, such as MASM, have special syntax or other built-in support for certain data structures, such as records and arrays.

SEQUENTIAL SEARCH

When data items are stored in a collection such as a list, we say that they have a linear or sequential relationship. Each data item is stored in a position relative to the others. In Python lists, these relative positions are the index values of the individual items. Since these index values are ordered, it is possible for us to visit them in sequence. This process gives rise to our first searching technique, the sequential search. Starting at the first item in the list, we simply move from item to item, following the underlying sequential ordering until we either find what we are looking for or run out of items. If we run out of items, we have discovered that the item we were searching for was not present.

1 Depth of node

The depth of node is the length of the path from the root to the node. A rooted tree with only one node has a depth of zero.

ALGORITHM COMPLEXITY

Algorithm	Best case	Expected
Selection sort	$O(N^2)$	$O(N^2)$
Merge sort	$O(N \log N)$	$O(N \log N)$
Linear search	$O(1)$	$O(1)$
Binary search	$O(1)$	$O(1)$

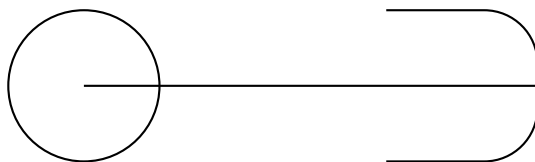
Mathematical Expression

This is a simple math expression $\sqrt{x^2 + 1}$ inside text.

And this is also the same: $\sqrt{x^2 + 1}$

This is an inline matrix here: $\begin{pmatrix} a & b \\ c & d \end{pmatrix}$,

Figure



Reference

- 1.1 Book of Data structures through C G. S Baluja.
- 1.2 Pieren Garry Department of computer science New York University.
- 1.3 Paul Xavier department of algorithms in c Amsterdam.
- 1.4 Surendrakumar Ahuja IITdelhi department of computer science delhi
- 1.5 Nick jones department of data mining Australia.
- 1.6 Wikipedia sequential search.