Data Structure: Theoretical Approach Durgesh Raghuvanshi B-Tech Department of Computer Science, IILM Academy of Higher Learning, Greater Noida, Uttar Pradesh, India

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.

Keywords: Include up to five keywords that capture the main topics or themes of the paper. Separate each keyword with a comma and space.

1. 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.

2. 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.

3. Algorithm complexity

Algorithm	Best Case	Expected
Selection sort	O(N2)	O(N2)
Merge sort	O(NlogN)	O(NlogN)
Linear search	O(1)	O(N)
Binary search	O(1)	O(logN)

4. 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.

5. Threaded binary tree

In a threaded binary tree all the null pinters which wasted the space in linked representation is converted into useful links called threads thus representation of a binary tree using these threads is called threaded binary tree.

6. Analysis of sequential search

To analyze searching algorithms, we need to decide on a basic unit of computation. Recall that this is typically the common step that must be repeated in order to solve the problem. For searching, it makes sense to count the number of comparisons performed. Each comparison may or may not discover the item we are looking for. In addition, we make another assumption here. The list of items is not ordered in any way. The items have been placed randomly into the list. In other words, the probability that the item we are looking for is in any particular position is exactly the same for each position of the list. If the item is not in the list, the only way to know it is to compare it against every item present. If there are n items, then the sequential search requires n comparisons to discover that the item is not there. In the case where the item is in the list, the analysis is not so straightforward. There are actually three different scenarios that can occur. In the best case we will find the item in the first place we look, at the beginning of the list. We will need only one comparison. In the worst case, we will not discover the item until the very last comparison, the nth comparison.

7. Binary Search

Binary search is a fast search algorithm with run-time complexity of (log n). This search algorithm works on the principle of divide and conquer. For this algorithm to work properly, the data collection should be in the sorted form. Binary search looks for a particular item by comparing the middle most item of the collection. If a match occurs, then the index of item is returned. If the middle item is greater than the item, then the item is searched in the sub-array to the left of the middle item. Otherwise, the item is searched for in the subarray to the right of the middle item. This process continues on the sub-array as well until the size of the sub array reduces to zero. B-trees are generalizations of binary search trees in that they can have a variable number of sub trees at each node. While child-nodes have a pre-defined range, they will not necessarily be filled with data, meaning B-trees can potentially waste some space. The advantage is that B-trees do not need to be re-balanced as frequently as other self-balancing trees. Due to the variable range of their node length, B-trees are optimized for systems that read large blocks of data. They are also commonly used in databases. A ternary search tree is a type of tree that can have 3 nodes: a lo kid, an equal kid, and a hi kid. Each node stores a single character and the tree itself is ordered the same way a binary search tree is, with the exception

of a possible third node. Searching a ternary search tree involves passing in a string to test whether any path contains it. The time complexity for searching a balanced ternary search tree is O(log n).

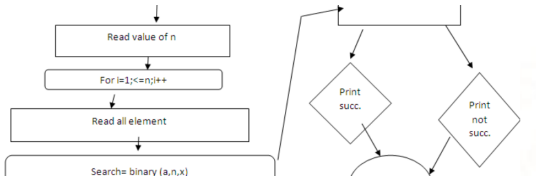

Figure 1. Binary Search flow-chart

8. Second and following pages

The second and following pages should begin 1.0 inch (2.54 cm) from the top edge. On all pages, the bottom margin should be 1-1/8 inches (2.86 cm) from the bottom edge of the page for 8.5 x 11-inch paper. (Letter-size paper)

9. Type-style and fonts

Please note that *Times New Roman* is the preferred font for the text of your paper. **If you must use another font**, the following are considered base fonts. You are encouraged to limit your font selections to Helvetica, Arial, and Symbol as needed. These fonts are automatically installed with the viewing software.

10. Page Numbers

Please DO NOT include page numbers in your manuscript.

11. Equations

$$e = mc^2 \tag{1}$$

$$(a+b)^2 = a^2 + 2ab + b^2 \tag{2}$$

$$log(ab) = log(a) + log(b) \tag{3}$$

12. Graphics/Images

All images must be embedded in your document or included with your submission as individual source files. The type of graphics you include will affect the quality and size of your paper on the electronic document disc. In general, the use of vector graphics such as those produced by most presentation and drawing packages can be used without concern and is encouraged.

- Resolution: 600 dpi
- Color Images: Bicubic Downsampling at 300dpi
- Compression for Color Images: JPEG/Medium Quality
- Grayscale Images: Bicubic Downsampling at 300dpi
- Compression for Grayscale Images: JPEG/Medium Quality
- Monochrome Images: Bicubic Downsampling at 600dpi
- Compression for Monochrome Images: CCITT Group 4

If your paper contains many large images they will be down-sampled to reduce their size during the conversion process. However the automated process used will not always produce the best image, and you are encouraged to perform this yourself on an image by image basis. The use of bitmapped images such as those produced when a photograph is scanned requires significant storage space and must be used with care.

13. Main text

Type your main text in 10-point Times, single-spaced. Do not use double-spacing. All paragraphs should be indented 1/4 inch (approximately 0.5 cm). Be sure your text is fully justified—that is, flush left and flush right. Please do not place any additional blank lines between paragraphs.
Figure and table captions should be 9-point boldface Helvetica (or a similar sans-serif font). Callouts should be 9-point non-boldface Helvetica. Initially capitalize only the first word of each figure caption and table title. Figures and tables must be numbered separately. For example: “Figure 1. Database contexts”, “Table 1. Input data”. Figure captions are to be centered below the figures. Table titles are to be centered above the tables.

14. First-order headings

For example, “1. Introduction”, should be Times 12-point boldface, initially capitalized, flush left, with one 12-point blank line before, and one blank line after. Use a period (“.”) after the heading number, not a colon.

14.1. Second-order headings

As in this heading, they should be Times 11-point boldface, initially capitalized, flush left, with one blank line before, and one after.

14.1.1. Third-order headings. Third-order headings, as in this paragraph, are discouraged. However, if you must use them, use 10-point Times, boldface, initially capitalized, flush left, followed by a period and your text on the same line.

15. Footnotes

Use footnotes sparingly (or not at all) and place them at the bottom of the column on the page on which they are referenced. Use Times 8-point type, single-spaced. To help your readers, avoid using footnotes altogether and include necessary peripheral observations in the text (within parentheses, if you prefer, as in this sentence).

16. References

References and in-text citation should be in line with the format recommended by the Publication Manual of the American Psychological Association (7th edition). The style and grammar guidelines are freely available and can be found at: https://apastyle.apa.org/style-grammar-guidelines.

List and number all bibliographical references in 9-point Times, single-spaced, and in an alphabetical order at the end of your paper. For example, Allen and Santomero, 1997; Castells, 2010 and Bloomberg, 2018 and Allen and Santomero, 1997.

References

Allen, F., & Santomero, A. (1997). The theory of financial intermediation. *Journal of Banking & Finance*, (21(11-12)), 1461–1485.

Bloomberg. (2018). *"wolf of wall street"jordan belfort isn't paying his debts, u.s. says.* https://www.bloomberg.com/news/articles/2018-05-16/-wolf-of-wall-street-belfort-isn-t-paying-his-debts-u-s-says (accessed: 04.01.2022)

Castells, M. (2010). *The rise of the network society (2nd ed.)* Wiley-Blackwell.