INTERNATIONAL JOURNAL FOR INNOVATIVE RESEARCH IN MULTIDISCIPLINARY FIELD ISSN: 2455-0620 Volume - 6, Issue - 7, July – 2020
Monthly, Peer-Reviewed, Refereed, Indexed Journal with IC Value: 86.87 Impact Factor: 6.719
Received Date: 07/07/2020 Acceptance Date: 25/07/2020 Publication Date: 31/07/2020

A Survey Paper on Performance Analysis of Data Structure Types

[1]Himani Bhatt, [2]Hetal Chokshi
[1, 2]Lecturer , Computer Engineering Department,
[1, 2] Parul Institute of Engineering and Technology – Diploma Studies, Vadodara, India.
Email – himanibhatt297@gmail.com, [2] hetalshah108@gmail.com

Abstract: *The Need for Data Structures is to sort out information all the more practically for complex applications. Numerous information structures exist yet we have to choose the seized information structure to meet the arrangement. A review has been done on various kinds of information structures to recognize their characteristics and divisions. This paper depicts unmistakable information structures in a steady way to give a brief examination on execution of information structures. This paper presents a concise report on execution, time unpredictability and uses of information structures. This paper arranges information structures into seven classes that bunch them as per their time multifaceted nature.*

Keywords: *Data Structure, Time Complexity, Performance.*

1. INTRODUCTION:

Data structures are utilized in the circumstances where consistent relationship is required between the data components so as to store the data. The logical or mathematical model of a specific association of data is called as data structure [1]. Data structures are intended to sort out information to suit a particular reason so it very well may be gotten to and worked with in proper manners. In computer programming, an data structure might be chosen or intended to store data to take a shot at it with different algorithms [2]. Data structures give a way to oversee enormous measures of data effectively. Some proper plan techniques and programming dialects underline data structures, instead of algorithms. While Selecting a Data Structure first we have to investigate the issue to decide the asset limitations an answer must meet, and afterward decide the fundamental activities that must be upheld. We have to ascertain the asset constrains for every operations and finally select the data structure that best meets these necessities. In general we can change the data structures dynamically to prepare the data for a given algorithm. The execution of a data structure for the most part requires composing a lot of systems that make and control examples of that structure. This paper gives clear description about the Data Structures, time complexity analysis and their applications. Section II presents related work that is carried out for analyzing the time complexity of data structures and also the classification based on their time complexity. Section III presents the experimental results and discusses its performance over each scenario (different values of N). Section IV presents real time applications and finally section V presents the conclusions of the paper.

2. DISCUSSION:

Premeditation and analysis of each and every data structure are done and based on their run time for each operation with different range of input data (N). Data structures are classified in to seven different class that group them according to their time complexity. These data structures works efficiently according to the user's problem definition with unique performance. The following Table:1 shows the time complexity of each data structure which is further used for classification.

Table 1 : Time Complexity of every Data Structure for Insertion, Deletion and Search Operation

	Name of Data Structure	Time Complexity		
		Insertion	Deletion	Search
Class 1	Stack	O(1)	O(1)	O(1)
	Queue			
Class 2	List	O(1)	O(N)	O(N)
	Linked List			
Class 3	Heap	O(Log N)	O(Log N)	O(N)
	Binary Heap			
Class 4	B-Tree	O(Log N)	O(Log N)	O(Log N)
	2-3-4 Tree			
	B+ Tree			
	Red Black Tree			
	Splay Tree			

Class 5	**Priority Queue** **Fibonacci Heap**	O(1)	O(Log N)	O(1)
Class 6	**Dequeue**	O(1)	O(1)	O(N)
Class 7	**Binary Tree**	O(N)	O(N)	O(Log N)

Table 2 : Run Time for different input data size(N)

Notation	**Complexity**	**Description**	**Run time(sec) for different input data size N**		
			N = 10000	**N = 1000000**	**N = 10000000**
O(1)	Constant	Constant number of operations, not depending on the input data size	Constant time	Constant time	Constant time
O(log N)	Logarithmic	Number of operations proportional to log2(N)	10^{-5} secs (0.00001 secs)	$1.7*10^{-5}$ secs (0.000017 secs)	$2*10^{-5}$ secs (0.00002 secs)
O(N)	Linear	Number of operations proportional to the input data.	10^{-3} secs (0.001 secs)	0.1 secs	1 sec

4. ANALYSIS: Table 3 show the analysis, an input explain how the execution time will change for different data sizes for performing each operation. The table explains how the execution will change when the size of the input data raises and it also tells which data structure is best suited for performing specific operation.

- Stack and queue take constant time for performing all the operations irrespective of size of input data.
- As the input data size increases the execution time rapidly increases for searching an element in a linked list.
- The execution time for inserting and deleting an element from the binary heap is very less, but as the input data size raises execution time for searching an element rapidly increases. If the algorithm involves appending a lot of data then heaps can be used and is best suited if a large number of insertions and deletions are needed.
- Irrespective of the size of the input data the execution time for the data structures which belongs to category 4 for performing insertion, deletion and search operations is very less. To keep data sorted; despite arbitrary inserts and deletes then a red black tree can be preferred. The run time for performing insertion and search operations is constant for priority queue and Fibonacci heap, but for deleting an element run time is very less. Priority queue can be used to order a list by some kind of importance.
- The run time for dequeue is constant for insertion and deletion operations, but as the size of the input data increases run time increases rapidly.
- In contrast to other data structures, for a binary tree the run time for inserting and deleting an element rapidly increases as the input data size increases. But once after inserting all the elements run time for searching an element is very less irrespective of size of input data. A binary tree is a good data structure to use for searching sorted data.

Table 3 : Execution time Analysis for different input data size(N)

	Type of Data Structure	**Execution Time analysis for input data size(N)**					
		Insertion		**Deletion**		**Search**	
		Small Data Size	**As data size increases**	**Small Data Size**	**As data size increases**	**Small Data Size**	**As data size increases**
Class 1	**Stack** **Queue**	constant time	constant time	constant time	constant time	constant time	constant time
Class 2	**List** **Linked List**	constant time	constant time	less	Increases rapidly	Less	Increases rapidly
Class 3	**Heap** **Binary Heap**	Very Less	Very Less	Very Less	Very Less	Less	Increases rapidly
Class 4	**B-Tree** **2-3-4 Tree** **B+ Tree**	Very Less	Very Less	Very Less	Very Less	Very Less	Very Less

	Red Black Tree						
	Splay Tree						
Class 5	**Priority Queue**	constant time	constant time	Very Less	Very Less	constant time	constant time
	Fibonacci Heap						
Class 6	**Dequeue**	constant time	constant time	constant time	constant time	Less	Increse rapidly
Class 7	**Binary Tree**	Less	Increse rapidly	Less	Increse rapidly	Very less	Very less

5. APPLICATION:

- Stacks can be used for converting a decimal number into a binary number, Towers of Hanoi Problem, parsing, and in runtime memory management.
- Queue can be used for Simulation, Ordered requests and Searches.
- Priority Queue can be used for Bandwidth management, Discrete event simulation, Dijkstra's algorithm, Huffman coding, A* and SMA* search algorithms and ROAM triangulation algorithm.
- Dequeue is used for the A-Steal job scheduling algorithm.
- Linked Lists are used to implement several other common abstract data types, including stacks, queues, associative arrays, and symbolic expressions.
- List can be used to store a list of records. The items in a list can be sorted for the purpose of fast search (binary search).
- Heap data structure is used in Heap sort, Selection algorithms, and Graph algorithms.
- B-Trees have wide range of application in Data base, Dictionaries, 1-D range search.
- Binomial Heaps are used in discrete event simulation and Priority queues.
- Red-Black Trees are used in time-sensitive applications such as applications and in functional programming and to construct associative sets.
- 2-3-4 Trees are used as in-memory data structures so user could memory program steps rather than disc accesses when evaluating and optimizing an implementation.
- Binary Tree are Used in many search applications where data is constantly entering/leaving, such as the map and set objects in many languages' libraries.

6. SUMMARY:

This survey paper analyses the run time of information structures for acting totally different operations by considering totally different vary of computer file size. The information structures represented during this paper square measure distinguished and economical. The degree of speed-up in apply can depend on the machines on that they're enforced. During this survey, found some points which will be any explored within the future, like to style algorithms and information structures so as to attenuate the run time even for larger computer file sizes and take a look at to explore deeper during this analysis space.

REFERENCES:

Journal Papers:

1. Sleator, Daniel D.; Tarjan, Robert E. (1985), "Self-Adjusting Binary Search Trees", *Journal of the ACM*.

Books:

1. Dr. N. Kashivishwanath *Data Structure Using C++* Laxmi publications
2. Sartaj Sahni, *Data structures, Algorithms and Applications in C++*.
3. Gopal, Arpita. *Magnifying Data Structures* PHI.
4. Donald Knuth. *The Art of Computer Programming* Volume1: Fundamental Algorithms, Third Edition. Addison-Wesley, 1997
5. *Definition of a linked list*. National Institute of Standards and Technology. 2004-08-16. Retrieved 2004-12-14.
6. Parlante, Nick (2001). *Linked list basics*.Stanford University. Retrieved 2009-09-21.
7. Goodrich, Michael T.; Tamassia, Roberto (2004). 7.3.6. *Data Structures and Algorithms in Java* (3rd ed.). pp. 338–341
8. Atkinson, M.D., J.-R. Sack, N. Santoro, and T. Strothotte. *Programming techniques and Data structures*.
9. Comer, Douglas (June 1979), The *Ubiquitous B-Tree* Computing Surveys 11 (2): 12137.

10. Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms* MIT Press and McGraw-Hill
11. Grama, Ananth (2004). (2,4) Trees. CS251: *Data Structures Lecture Notes*. Department of Computer Science, Purdue University.
12. Ramakrishnan, R. and Gehrke, J. *Database Management Systems* McGraw-Hill Higher Education (2002), 3rd edition.
13. San Diego State University: CS 660: Red–Black tree notes, by Roger Whitne

Web References:

- http://searchsqlserver.techtarget.com/definition/data-structure.
- http://www.cprogramming.com/tutorial/computersciencetheory/stack.html