**LINKED DATA STRUCTURES \***

Chekanin, Vladislav A; Chekanin, Alexander V.

INTRODUCTION

Arrays and Lists are one of the many ways data can be stored, accessed, and manipulated using a programming interface on the memory of your computer. They are quite similar with one another, such that in essence, they are storing memory addresses for the data they hold. The array is the fundamental sequentially-allocated data structure. It is fixed-sized, can be dynamically reallocated, and is efficiently indexed. Each item in an array is able to be referenced, by its index, in constant time. The list is the fundamental linked data structure. Therefore, a list is stored as distinct blocks of memory bound together by pointers (Jerome, 2017).

Linked data structures use a combination of very effective data structures and is used in many algorithms. The implementation of a linked data structure includes an organize set of elements having links to each other. One of the greatest advantages of a linked data structure is that its elements can be located on different memory allocations contrary to a linear array. Since stack, queue and array are linear data structures, we shall establish how these data structures can be implemented, and that includes the operations as well (Chekanin, Chekanin 2017). For this journal review, the data structure to be discussed is about the Stack Data Structure.

STATEMENT OF THE PROBLEM

In general, the process of indexing an array takes more time to execute than does the process of accessing the contents of a pointer. In fact, this is one of the main reasons why pointers are used to access the elements of an array (Kochan 2014). The main issue to be discussed is the optimization of a program, one aspect is through the use of arrays in accessing data or its elements, by looping over every element, in another perspective, the speed of insertion and deletion of elements are also concerned. For small-scale arrays, this sort of method will not affect the overall time consumed, but when this array becomes the size of a thousand or more, this will be a bottleneck for the processing time.

LINK

https://search.proquest.com/pqrl/docview/2190324803/fulltextPDF/4ABB9BBA9C4E472BPQ/1?accountid=34542  
August 31, 2019  
GMT+8 - 8:00 AM

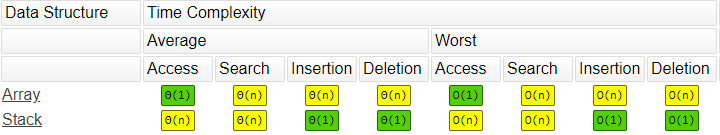
CONCEPTS AND THEORIES

Data in a program, whether stored in an array or a list, is stored in addressable block of memory so that it can be accessed later. Items in either arrays or lists may be referred to as nodes, items, elements, records, or entities. We’ve been using the term “node” to describe the items linked together in lists. In the case of arrays, we’ll use the term “item”, which is conventional. Keep in mind, however, that all of these terms can be used interchangeably (Jerome, 2017).

A stack is a container of objects that are inserted and removed according to the last-in first-out (LIFO) principle. In the pushdown stacks only two operations are allowed: push the item into the stack, and pop the item out of the stack. A stack is a limited access data structure - elements can be added and removed from the stack only at the top. push adds an item to the top of the stack, pop removes the item from the top (Nitesh, 2014). A stack is an example of a linked data structure where the elements of the stack have a memory address to other stack elements, usually having the terms, next and/or prev, meaning the next item and the previous item respectively. To implement a stack, at least two pointers must be available, one for the bottom of the stack, and one for the top of the stack.

SOLUTION

Stack, as it is implemented as a linked data structure, effectively reduces the time constraints of insertion and deletion of elements. Time complexity must also be taken into consideration, through which, we can consider the data structure to be used for a certain program, with optimization in mind.

Green - > Excellent Yellow -> Good

We can see that for the Stack, there are two green areas and two yellow areas (average and worst case). As for the Array, there are three yellow areas. The main advantage of the stack data structure is in the insertion and deletion process.

CONCLUSION

REFERENCES

Adam Drozdek. Data Structures and Algorithms in C++, Third Edition. Thomson Course Technology, 2005. ISBN 0 - 534-49182-0. Chapter 4: Stacks and Queues, pp. 137–169.

Chekanin, V. A., & Chekanin, A. V. (2017). LINKED DATA STRUCTURES \*. *International Journal of Computer Research, 24*(2), 237-267. Retrieved August 31, 2019, from https://search.proquest.com/docview/2190324803?accountid=34542

Chekanin, V. A., Chekanin, A. V. (2016). Основы алгоритмизации и программирования на языках C и C++; ФГБОУ ВО МГТУ “СТАНКИН”: Москва, 2016, pp. 265-278. [Basics of algorithmization and programming on languages C and C++; MSTU “STANKIN”: Moscow, 2016, pp. 265-278].

Donald Knuth. The Art of Computer Programming, Volume 1: Fundamental Algorithms, Third Edition. Addison- Wesley, 1997. ISBN 0-201-89683-4. Section 2.2.1: Stacks, Queues, and Deques, pp. 238–243.

Jerome, D. L. (June 30, 2017). Lists Part III: Lists vs Arrays. Retrieved September 1, 2019 from https://medium.com/@dljerome/lists-part-iii-lists-vs-arrays-60bdced64eed.

Kochan, S. (August 18, 2014). Programming in C (4th Edition). Developer’s Library.

Nitesh, Manbir S., Rahul Y. (2014). RESEARCH PAPER ON STACK AND QUEUE. Volume 1 Issue 7. JIRT.

Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. Introduction to Algorithms, Second Edition. MIT Press and McGraw-Hill, 2001. ISBN 0-262-03293-7. Section 10.1: Stacks and queues, pp. 200–204.

Weiss, M. A. Data Structures and Algorithm Analysis in C++; Pearson Education: Boston, MA, 2014, pp. 77-116.

N.A. Know Thy Complexities. Retrieved September 2, 2019, from https://www.bigocheatsheet.com/

William Ford, William Topp. Data Structures with C++ and STL, Second Edition. Prentice Hall, 2002. ISBN 0-13- 085850-1. Chapter 8: Queues and Priority Queues, pp. 386–390.