Linked List vs. AVL Tree in Modulo Ten Search

Soheir Noori
Department of Computer Science
University of Karbala
Karbala, Iraq
soheirnoori80@gmail.com

***Abstract*—There are various kinds of searching algorithms such as Modulo Ten Search, which has been used to find a certain key. This searching algorithm searches a key first by checking the values of modulo ten, and then it follows a linear search. We have used an AVL tree instead of a linked list. The result shows that using an AVL tree in Modulo Ten search improves the searching process.**

Keywords- Modulo Ten; AVL Tree; Linked List.

I. Introduction

The most fundamental problems in computing are searching problems; many algorithms have been developed to tackle this problem. Searching is the process used for finding a specific item in a group of items. The most basic search algorithm that can be found is the linear search. This algorithm checks every input sequentially until finding the element. It is suitable for minimum inputs [1]. A binary search algorithm, on the other hand, searches for items inside a sorted array. The item is compared with the middle element in the array. If the item is greater than the middle element, it should be in the upper half of the array ; otherwise, it lies in the lower half [1]. A binary search deals with half an array in the worst case, which is medium for a linear search[2]. However, a binary search tree is a binary search that is modified to deal with information rather than values [3]. It is suitable to organize large files because it is “efficient with both random and sequential access of records, and for modification of a file” [4]. Each node in a binary tree keeps the smaller elements on the right and the greater ones on the left. This leads to an easy search since the comparison with the root will decide the direction (right/left) that should be followed to find the element[5]. It is faster than linear search; however, searching a binary tree can be as slow as linear search if all nodes lie on one side, which depends on the sequences of the input [6]. A balanced binary search tree overcomes the previous problem with binary search trees by having a parameter that can be varied to maintain the balance of the tree[7].

There are many data structures that have been used to store information; one of the basic structures is a link list[8], which is a sequence of elements each of which has a pointer to the next sequential element in the list with the ability to add and remove elements from the list [9]. The searching in a linked list is sequential and time consuming[8]. AVL (a highly balanced binary search tree) is another technique of data structure which is first introduced by Adel'son-Vel'skii and Landis [10] and it is named after them. The binary tree is called an AVL tree if the heights of the right subtree differ from those of the left subtree by almost one for each node [11]. AVL is completely balanced, which leads to fast searching.

Modulo Ten Search is actually analogous to binary search in the case of small lists [12]. In our paper, we try to use the same technique that has been used by Francis [12] with another data structure, which is AVL tree and compare it with the previous one , which is a linked list in terms of time complexity. We found that using this technique with an AVL tree reduces the complexity of searching especially when we have a huge data base.

II. Related Work

Most algorithms usually used for searching in databases include simple “linear search”, and a lot of other algorithms use a set of search data structures, for example, hash tables, heaps and binary search trees to accelerate various queries for the same database [3]. Besides, there are a lot of algorithms which are prepared practically for recall in very big databases like fingerprinting databases, electronic documents, and bank account records.

Modulo Ten Search is a searching algorithm used to find certain key inside the list using modulus ten of the number. Practically, automatically, the number is to be referred to one of the lists depending on the factor of modulo ten number. There are ten linked lists, each of which stores numbers depending on the values of modulo ten. Linked list one includes numbers whose values of modulo ten are one. Linked list two, on the other hand, includes numbers whose values of modulo ten are two and so on. Therefore, number 95 is to be stored in linked list five because the value of modulo ten of 95 is 5. At last, a consecutive search is to be utilized to establish whether the item is explicit in the linked list or not.

When Modulo Ten Search is used, at first the value of modulo ten of the number is calculated. Based on the modulo ten value, the corresponding linked list uses a linear search method to check the existence of this item in the linked list [12].In this paper we replaced the linked list with an AVL tree. The same Modulo Ten has been used by computing the modulo ten for the number. After that we used this number to determine the tree that we are going to search in. Each AVL tree contains all the numbers with the same modulo. Fig.1 shows the pictorial implementation of the AVLs.

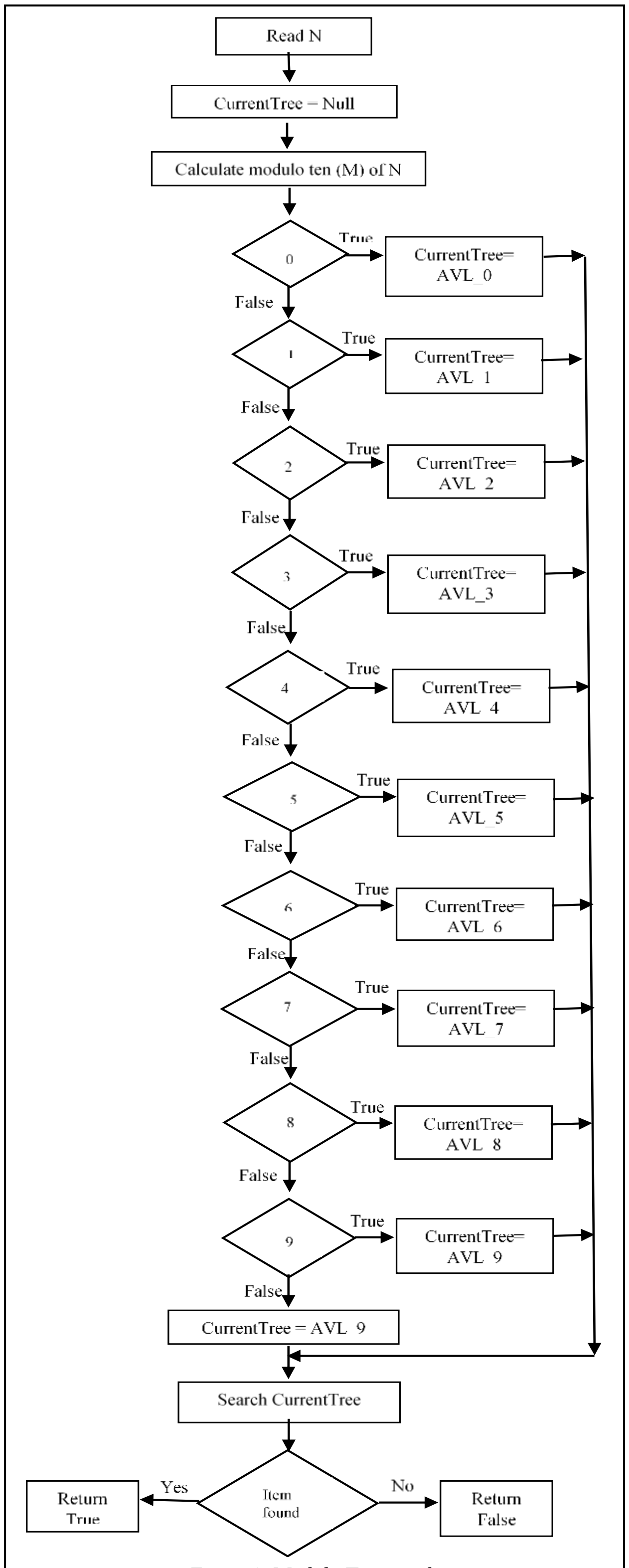

Figure 1. Modulo Ten search

III. METHODOLOGY

In Modulo Ten Search we will have ten AVL trees; each tree will store numbers depending on their values of modulo ten. For instance, tree nine will include numbers whose values of modulo ten are nine and tree eight will include numbers whose values of modulo ten are eight and so on.

A. AVL Modulo Ten Search

Our experiment is based on the Modulo Ten Search algorithm {AVL_0 to AVL_9 includes the items}
Firstly: Starts All AVL equal to Null
Secondly: Read the key entered (X) // that X is the element to be found.
Thirdly: Check M which represents the modulo ten value of X.
Fourthly: Check (M) parameter with switch/ case statement,
case 0: AVL_0
case 1: AVL_1
case 2: AVL_2
case 3: AVL_3
case 4: AVL_4
case 5: AVL_5
case 6: AVL_6
case 7: AVL_7
case 8: AVL_8
case 9: AVL_9
Fifthly: Perform search for AVL tree.
Sixthly: End of Modulo Ten AVL Search.

An AVL-tree uses the binary tree as its backbone. This is a powerful structure that supports a large number of operations on an ordered set efficiently. AVL is a binary search tree with the difference in height between the left and right children of any node is at most 1. If at any time they differ by more than one, rebalancing is done to restore this property [13].

B. Implementation

In our experiment we assume to have two-digit numbers, and it will contain a hundred numbers as maximum: from zero to ninety-nine. By applying modulo ten, all AVL trees will have at most ten elements (numbers).
Let us assume that we have these numbers, which are thirty eight numbers as follows: 71,23,66,17,98,18,28,68,58,49,84,44,14,64,55,25,81,52,3,13,93,43,65,95,6,86,46,10,50,80,61,1,51,36,56,19,69,79

At the beginning the AVL trees do not have any items. Later on, they are inserted inside the AVL trees as in Fig. 2

Assume that a search for the item 86 is applied. Firstly the modulo ten of 86 is to be found, which is actually 6. After that we search the AVL_6. The result of this example will be obtained in three comparisons instead of six comparisons in a linked list.

Assuming that we need to add a new item to the AVL tree, firstly, modulo ten of the number is to be found and then to be added to the identical tree. Assume now that we need to add 87, the modulo ten of 87 is to be calculated, which is 7. Therefore, the number 87 is added to AVL_7.

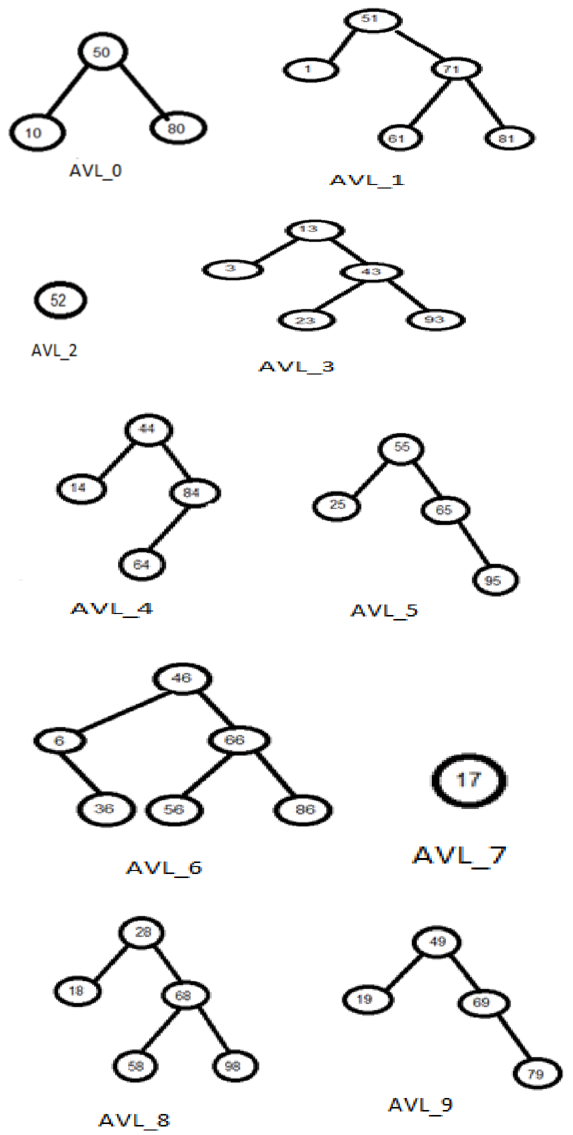

Figure 2. AVL Trees

IV. Result and Discussion

Starting comparisons in Modulo Ten Search is going to reduce the whole linked list to 10 % of Linear Search. Therefore, the complexity of time for Modulo Ten Search is much better than Linear Search [12]. The complexity of time for Modulo Ten Search by using linked list is not better than when we use the AVL tree.

Operations to access an item by position (add ,retrieve, remove) in linked Lists are O(n) in the worst case and O(1) in best case [14] when the item that we are looking for is at the beginning of the Linked List. An AVL tree, however, is a binary search tree with excellent performance guarantees. AVL trees guarantee fast lookup (O(log n)) on each operation [14]. In multithreading, AVL trees are more useful with lots of lookups because of parallel lookups in AVL tree [15].

V. Conclusion

The Modulo Ten Search is usually used as another method for the linear search. Further, it can be well-performed if the list appears to be as an AVL tree. The comparison number can be reduced by putting the item which is indicated frequently at the beginning of the list; however; implementing Modulo Ten search with an AVL tree can give faster results especially with bushy applications.

References

[1] R. Bremananth, V. Radhika, and S. Thenmozhi, "Visualizing Sequence of Algorithms for Searching and Sorting," in Advances in Recent Technologies in Communication and Computing, 2009. ARTCom'09. International Conference on, 2009, pp. 647-649.

[2] Y. D. Liang, Introduction to Java programming: brief version: Pearson, 2013.

[3] J. Mosher, "Search Algorithms," ACM Journal Name, vol. 1, p. 11, 2010.

[4] J. Nievergelt, "Binary search trees and file organization," ACM Computing Surveys (CSUR), vol. 6, pp. 195-207, 1974.

[5] Stoimen. (June 22, 2012). Computer Algorithms: Binary Search Tree.

[6] Stoimen. (July 3, 2012). Computer Algorithms: Balancing a Binary Search Tree.

[7] J. Nievergelt and E. M. Reingold, "Binary search trees of bounded balance," SIAM journal on Computing, vol. 2, pp. 33-43, 1973.

[8] T. Niemann, "Sorting and searching algorithms," ed, 2010.

[9] K. L. Kluge, "Method and apparatus for managing a linked-list data structure," ed: Google Patents, 1999.

[10] M. AdelsonVelskii and E. M. Landis, "An algorithm for the organization of information," DTIC Document1963.

[11] A. K. Tsakalidis, "AVL-trees for localized search," Information and Control, vol. 67, pp. 173-194, 1985.

[12] A. Francis and R. Ramachandran, "Modulo Ten Search-An Alternative to Linear Search," in Process Automation, Control and Computing (PACC), 2011 International Conference on, 2011, pp. 1-4.

[13] C. A. Shaffer, "Data Structures and Algorithm Analysis," Update, vol. 3, p. 0.3, 2012.

[14] katzider. (2017). Data structures time and space complexity. Available: https://quizlet.com/121785926/data-structures-time-and-space-complexity-flash-cards/

[15] AbhiAgarwal. (2017). AVL trees vs Splay trees. Available: https://github.com/AbhiAgarwal/prep