ISSN No. 0976-5697

Volume 4, No. 10, September-October 2013

International Journal of Advanced Research in Computer Science

RESEARCH PAPER

Available Online at www.ijarcs.info

Presenting a tree structure for storing and searching large lists of order O(1)

ShirinAbbasloo*
Department of computer,
Islamic Azad University, Baft, Iran,
abbaslooshirin@gmail.com

FarokhKoroupi
Department of computer,
Islamic Azad University, Baft, Iran,
farokh.koroupi@iaubaft.ac.ir

Reza Noormanid Pour
Department of computer,
Islamic Azad University, Sirjan, Iran,
noormanir@iausirjan.ac.ir

Abstract:Sort the list of greatest concern is that mathematicians are working on optimizing the algorithms. Sort the list so far using linear arrays was performed. Due to the limited size of the linear array and traverse difficult time sorting this type of data structure and it was not desired. Sorting in linear lists must be scanned once for each element and other elements to be compared. Thus, when about twice the size of the list to sort the list (O (n^2)) to spend.Scrolling up to search for the elements as we move toward the desired element. The best way to split the original list into two smaller lists. With this action against O (Logn) to be spent.In this paperhaspresenteda tree structureforstoringandsearching that can order largelistsinO (n) order time.Italsohasasearchfeatureinthelistofpossiblelistelementsdo not dependon the sizeandtypeofthelististhesametime(O (1)).

Keywords:NLR traversal, Searching algorithm, Sorting algorithm, Tree structure.

I. INTRODUCTION

Sorting algorithm, unordered list and take it to the Sort Ascending or Descending to sort out [1, 2]. Lists can contain letters or numbers [3, 4, 5]. Obviously, when we want to sort a list of elements that should be compared to the list and each placed in the right place [6, 7, 8, 9].After the list of accounts that n is at least n times the element because each element of work to do compared to the other elements of the list is a linear list [10, 11].There arealgorithmsforsortinglistsin order ofO (nlogn)andhigher [12].In generalsortingalgorithmscan be dividedintothreecategories.

a. InsertionSortAlgorithm
b. selectivesortingalgorithm
c. divide and conquersorting algorithms

Insertions of the approach taken in the top of the list are inserted in the right place [13, 14].The option element is selected and compared with the rest of the ingredients and put in the right place [15, 16].The divide and conquer approach to general list into smaller lists and we'll sort it lists [17]. The small list and merge the sorted list to get sorted first [18].Algorithm is presented in this paper is a method of Insertion. In this algorithm, arrivals elements are in the right place at the tree and do not spend extra time to do the sorting. Also search in the list that is true, only time spent traversing a vertex of the tree to search for the desired element. The paper concludes in Section sorting and searching algorithms are presented and compared with other algorithms, the simulation results are listed in the chart.

A. *Provided a tree structure:*

As mentioned sorting algorithms on a linear lists work.After a long time of these algorithms is less than O (N) not be. Because these lists are linear and N is exactly the height of the list. The list can never be less than its height. In our algorithm, we use a different data structure.data structure whose height is less than linear lists. Building a tree data structure that is most appropriate for times when you can bring down an algorithm. We consider this algorithm, each node in the tree, the tree, including 36 children. Of these 36 children 26 children 10 children to the letters A..Z and numbers 0..9 are, however, much has changed. Obviously, each node represents a letter or number in which the home is located. Also for the optimal algorithm memory consumption for every knot that tied it in the list of existing allocated memory will be and if it is not tied up in the list, memory occupation.In other words, the tree and the tree is the perfect tree algorithm. Tree in Figure 1, in which the characters are stored in the list, is displayed.

List = john, marry, tomas135, zero

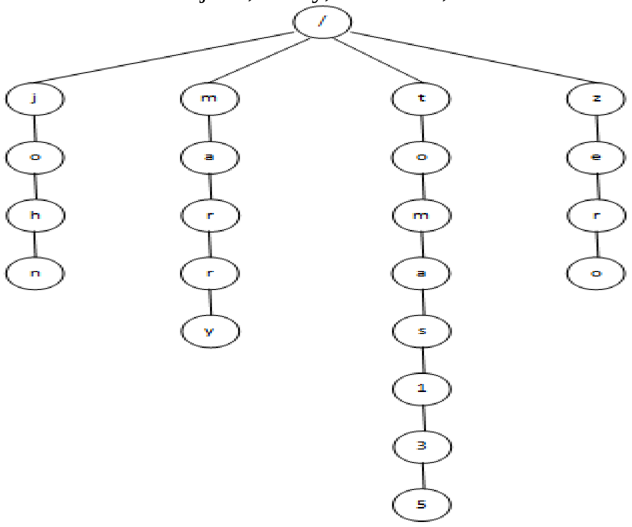
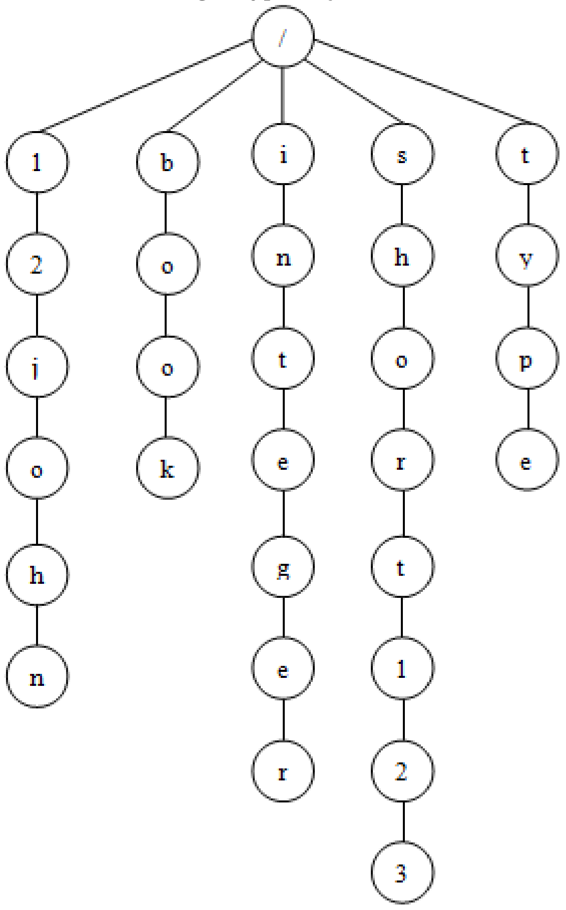


Figure 1: The tree which stores have it in List 1. In this tree with a height of 9, there are 21 nodes.

With regard to the structure of a tree is presented one can understand that algorithm has regular letters and numbers should be entered as character to character padding and in the proper place to phrase should be according to tree form 1 character words of the entrance in the proper place. since peak height times the length of the largest password, then b (tree) = MAX (Len) can be understood by the algorithm in the worst case, for every word of order O (h), but also O (h)Children should also be arranged in the following order of the numbers 0 ... 9 in the higher priority and the characters A .. Z, respectively, should be a lower priority. So in the left navigation tree NLR will start scrolling the numbers and letters are read [19, 20]. The numbers are higher priority because of letters, numbers and letters are the first node after the last node" 0" and" Z" to navigation Skip to assume the following list as input to the resulting tree-like algorithm is given in Figure 2.

List = book, integer, type, 12john, short123, short



Sorted list = 12john, book, integer, short123, type, short

Figure 2 - a tree is built, and then traversal the list NLR, List Sorted list is created.

The fundamental problem is that the tree does not specify a node, the node determines whether the final word. This means that if we traverse the tree String 'type', 'typ', 'ty', ’t’, and all the trees are measurable. To resolve this problem, we consider each node of a B-type Boolean that these bits are initially false for all nodes are equal. If the build tree, char last word was a bit ended to put it to true. Then when scrolling the navigation tree, if you were true to character bits ended up here was a character created and false if it should continue scrolling. For example, the tree in Figure 2, if the words 'short123', 'short', I have to scroll after scroll 'short' work to continue tree traversal word 'short123' is also measurable. As shown in this tree cannot be repeated two or more words. If the List 2 words 'short' can only be navigated after the word 'short' to find. For the sorting algorithm is a great fault for removing the defect of a variable name to another count that at first for all the nodes equal to zero and we only tied for the final if the prophet ended to be true to a unit of the count is added. With this interpretation, each node can determine the end of a word is a count of zero indicates that the number is equal to the number of words in the list have been saved. For example, consider Figure 3 lists:

List = 12, 123, 23, 12

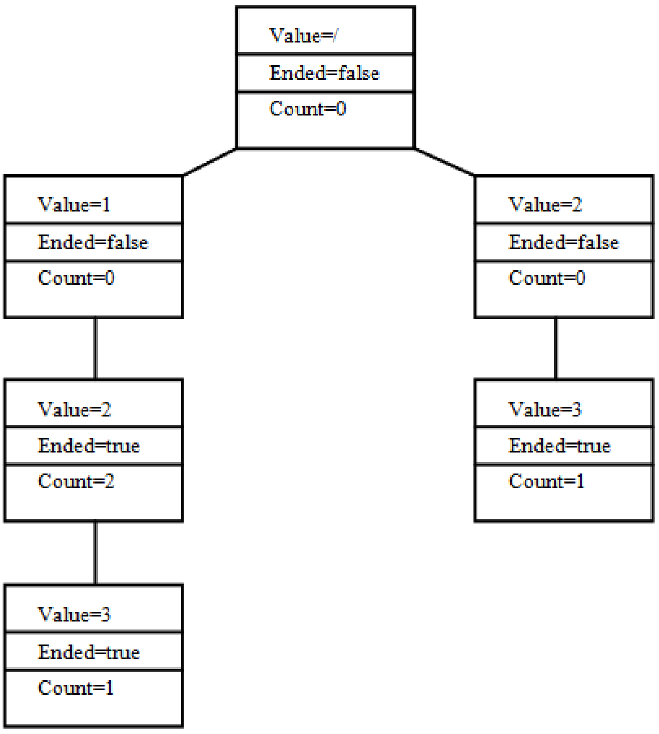


Figure 3: The status of each node in the tree is stored in the list.

We also created variables to test the appropriate tree node count to count the number of elements in the list must be equal. In other words, if SUM (count) =N resulting tree was so true.

Due to the above structures, the tree data structure must be defined as follows:

```
Type    tree_struct {
Char value;
Boolean ended=false;
Int count=0;
Tree child [0..9, 'A..Z'];
} Tree;
```

Tree data structure is represented by a node of type character specifies the value of each node.A byte that specifies the terminal nodes of the Boolean type is an integer that specifies the number of occurrences of each word. Each node in the tree for 36 pointers to child nodes available. The numbers refer to the first 10 pointers and 26 pointers point to the next word A..Z. Each node has 3 pointers and 36 bytes for the data it needs. However, the number of nodes, each node can vary depending on the input data. If the tree is supposed to sort the numbers, the number of pointers that each node should be 10 pointer.Or if this character is the big and small of importance so should 26 pointing to the character is small and 26 pointing to the character was great in pointing out the need for total 62 pointer.

B. Sort in the proposed structure:

The sorting algorithm is applied on the tree in the words entered by the user to put them on the tree. The father of the child nodes can be made at the relevant pointers. You should also check if a node as the last character of a word after it ended variable nodes have to be true. The last character must be also count +1.

```
Void sort () {
Node=get ();
For (i=1; i<=len (node);i++){
If (stree.child[node[i]]==null) {
Tree  child =new tree;
Child. Value =node[i];
If(i==Len(node)){
Child.Ended =true;
Child.Count++;
}}
Stree.child[node[i]] = child;
Stree = child;
}}
```

Since the arrival of the tree are the extra time not require for the sorting operation the proposed sorting algorithm of order O (1) is each point represents a tree in a sorted list.

C. Search on the proposed structure:

Now that the tree was very helpful in searching. Search algorithms to traverse the list to find a specific element [21]. Some of these algorithm are list to list divided into two and then each from the list are based on being a downward spiral or seek upward and finally element on the issue. Some other strategy based on its own investigations to list their tree like Heap that depends on, the Heap or changeover-Heap. Algorithm search for manuscripts list from beginning to end.Measuring to the element to find.Because in tree that presented in this article, can search an element with traversal an edge of tree. Therefore, it's the NLR traversal of this tree, that traversal each node and then children of those node until end of edges with difference that if finished node of this edges equal to finished character of special element. Here is a look at the tree, the worst and best of times, O (h) where h is the length of the word. Since h is a constant of the order of the search tree in O (1) is. Below is the algorithm for the search function in the tree.

```
Void search (){
node= get ();
for(i=1; i<=len(node);i++) {
if (stree.child[node[i]) !=null) {
stree = stree.child[node[i]];
}else{break;}
} if ((i==len(node)&(stree.ende==true))
Return true;
Else
Return false;
}
```

The variables used in the data structure can be used to do other.For example, $\sum \text{count} - \sum \text{ended}$is the number of words that have been repeated in the list or height of the tree with these nodesisthe highest word in list.
The two simple conditions that guarantee the right to the tree of:

a) $\sum count = n$

b) $\sum ended \le \sum count$

If this condition was true then is true tree.

II. CONCLUSION

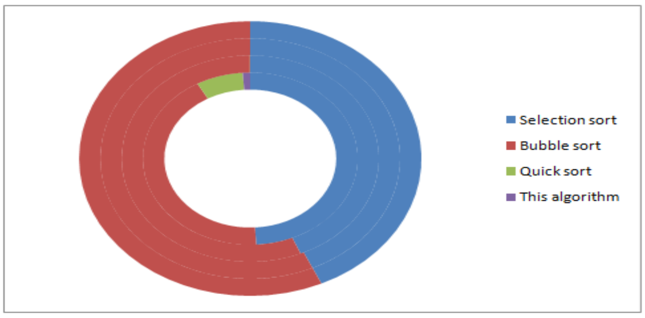
In this section, the results of tests performed of presenter sorting and searching algorithm with other algorithm, then performed advantage and disadvantage of this algorithms. Should be noted CPU Intel 2400MHZ core i7 system with the test results obtained.

Table 1. Spent time of sorting algorithms

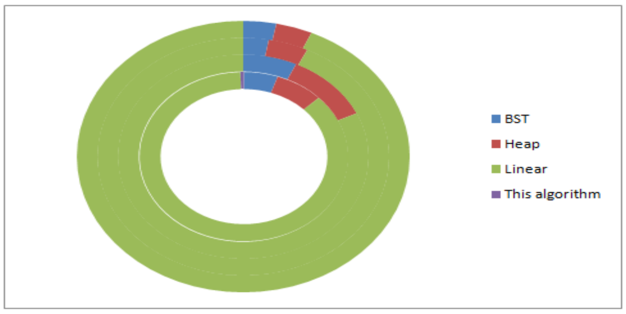
Algorithm	N=10	N=1000	N=100000	N=10000000
Selection sort	0.049	195.743	1900950.119	19000224200.661
Bubble sort	0.042	253.009	2500450.020	25000070000.049
Quick sort	0.007	0.212	20.033	2000.092
This algorithm	0.001	0.025	0.745	10.802

Table 2. Spent time of searching algorithms

Algorithm	N=10	N=1000	N=100000	N=10000000
BST	0.019	1.759	163.760	21496.250
Heap	0.027	2.812	259.305	23698.654
Linear	0.312	20.794	5427.631	627410.853
This algorithm	0.002	0.003	0.002	0.002



(a)



(b)

Figure 4 - Simulation results of sorting algorithms (Section A) and search algorithms (Section B)

Top of the charts and graphs, sorting and searching algorithms are harvested, which was presented in this paper is suitable for small N and large N is very efficient. For a list of small N, the number of elements it lists the maximum length is less than the max (height) <= N. because in other algorithm if there are a list with 1 element, for sorting time does not spent and for searching only 1 comparison needed. While the, presented sorting algorithms need O (h) and presented search algorithms need O (h) time. In general, the algorithms of order O (l) are not useful for small data. In other words, only for storage of linear lists the word as bytes of memory is required. In this structure for store a node, 1 byte needed to store character and 1 byte for store Ended of this node and 2 Byte need to store number of iterations. So 4 byte for information of this character needed and each pointer need 4 byte to store. If there are 36 pointer for a node, so much for a node occupies 166 bytes of memory.

That’s why these algorithms for sorting and storage of information on the numbers and not affordable and in some cases, such as libraries and documentation centers is recommended. However, this algorithm is useful if the memory is not a problem. Quick sort algorithms with a comparison of this algorithm are compared. Suppose that the number of characters that a list of 1000000 elements, 10000000 list of characters. Quick sort algorithm, this list is but a memory to 10MB of memory, our algorithm is equal to 1/5GB. Furthermore, the time consumed in the Quick sort algorithm in worst case O (n2) algorithm and O (n) that if these algorithms are implemented on a cpu Intel 2400MHZ core i7 processors fast sorting algorithm when the 2000000 milliseconds, and the algorithm presented in this paper runs in front of the 2000 MS runs. Compare All you need to sort the list by a 1000000 memory elements, 2000000 is equal to the time of 2000MS arrives. It is a way to search for a particular element in the list of search algorithms have been very little time to spend on the search algorithm of order O (l), and does not depend on the size of the problem.

III. REFERENCE

[1]. Erik Sintorn, Ulf Assarsson, Fast parallel GPU-sorting using a hybrid algorithm, Journal of Parallel and Distributed Computing, Volume 68, Issue10, October 2008, Pages 1381-1388.

[2]. AtifRahman, SwakkharShatabda, Masud Hasan, An approximation algorithm for sorting by reversals and transpositions, Journal of Discrete Algorithms, Volume 6, Issue 3, September 2008, Pages 449-457.

[3]. Wolfgang Rönsch, Henry Strauss, Timing results of some internal sorting algorithms on vector Computers, Parallel Computing, Volume 4, Issue 1, February 1987, Pages 49-61.

[4]. Christos Levcopoulos, Ola Petersson, Splitsort—an adaptive sorting algorithm, Information Processing Letters, Volume 39, Issue 4, 30 August 1991,Pages 205-211.

[5]. VamsiKundeti, SanguthevarRajasekaran, Efficient out-of-core sorting algorithms for the Parallel DisksModel, Journal of Parallel and Distributed Computing, Volume 71, Issue11, November 2011, Pages 1427-1433.

[6]. SanguthevarRajasekaran, An optimal parallel algorithm for sorting multisets, Information Processing Letters, Volume 67, Issue 3, 17 August1998, Pages 141-143.

[7]. Mingming Li, Shuming Liu, Ling Zhang, Huanhuan Wang, FanlinMeng, Lu Bai, Non-dominated Sorting Genetic Algorithms-IIBased on Multi-objectiveOptimization Model in the Water Distribution System, Procedia Engineering, Volume 37, 2012, Pages 309-313.

[8]. José Luis Soncco-Álvarez, Mauricio Ayala-Rincón, Sorting Permutations by Reversals through a Hybrid Genetic Algorithm based onBreakpoint Elimination and Exact Solutions for Signed Permutations, Electronic Notes in Theoretical Computer Science, Volume 292, 5 March 2013, Pages119-133.

[9]. SèverineBérard, Cedric Chauve, Christophe Paul, A more efficient algorithm for perfect sorting by reversals, Information Processing Letters, Volume 106, Issue 3, 30 April2008, Pages 90-95.

[10]. M. Basu, Dynamic economic emission dispatch using nondominated sorting geneticalgorithm-II, International Journal of Electrical Power & Energy Systems, Volume 30, Issue 2,February 2008, Pages 140-149.

[11]. RenataFurtuna, Silvia Curteanu, Florin Leon, An elitist non-dominated sorting genetic algorithm enhancedwith a neural network applied to the multi-objectiveoptimization of a polysiloxane synthesis process, Engineering Applications of Artificial Intelligence, Volume 24,Issue 5, August 2011, Pages 772-785.

[12]. T. Miyadokoro, N. Nishimura, S. Yamamoto, subalpine old-growth coniferous forest, central Japan, Forest Ecology and Management, Volume 182, Issues 1–3, 3September 2003, Pages 259-272.

[13]. Susanne Hambrusch, Chuan-Ming Liu, Walid G. Aref, SunilPrabhakar, Efficient query execution on broadcasted

index tree structures, Data & Knowledge Engineering, Volume 60, Issue 3, March 2007,Pages 511-529.

[14]. Said M. Megahed, Efficient computation algorithm for dynamic modelling of treestructure robot arms, Robotics and Autonomous Systems, Volume 10, Issue 4, 1992,Pages 225-242.

[15]. SéverineFratani, Regular sets over extended tree structures, Theoretical Computer Science, Volume 418, 10 February 2012, Pages 48-70.

[16]. Athar Ali Moinuddin, Ekram Khan, Mohammed Ghanbari, The impact of tree structures on the performance of zerotreebased wavelet video codecs, Signal Processing: Image Communication, Volume 25, Issue 3, March 2010, Pages 179-195.

[17]. Stefan Edelkamp, Stefan Schrödl, Chapter 2 - Basic SearchAlgorithms, Heuristic Search, 2012, Pages47-87.

[18]. Bin Wu, CunhuaQian, Weihong Ni, Shuhai Fan, Hybrid harmony search and artificial bee colony algorithm forglobal optimization problems, Computers &Mathematics with Applications, Volume 64, Issue 8,October 2012, Pages 2621-2634.

[19]. N. Poursalehi, A. Zolfaghari, A. Minuchehr, Differential harmony search algorithm to optimize PWRsloading pattern, Nuclear Engineering and Design, Volume 257, April 2013, Pages161-174.

[20]. N. Poursalehi, A. Zolfaghari, A. Minuchehr, PWR loading pattern optimization using Harmony Searchalgorithm, Annals of Nuclear Energy, Volume 53, March 2013, Pages 288-298.

[21]. R. Krueger, G. Simonet, A. Berry, A General Label Search to investigate classical graph searchalgorithms, Discrete Applied Mathematics, Volume 159, Issues 2–3, 28 January2011, Pages 128-142.