Development a Sequential Search Algorithm by using Triple Structure

Ragheed D. Salim*

Received on: 1/12/2010
Accepted on: 5/5/2011

Abstract

This paper develops sequential search algorithm by using new structure called Triple structure. In this structure data are represented as triple. It consists of three locations (1-Top, 2- Left, and 3- Right)

Sequential search algorithm is a simplest form. This search is applicable to a table organized either as an array or as a linked list, this process makes the maximum number of comparisons (Average case complexity of Search) is ***O***(***n***) (pronounce this "big-***O***h-***n***"or "the order of magnitude"), if we search in a list consists of **N** elements. In this research the number of comparison is reduced to a third by using triple structure, is achieved this process makes the maximum number of comparisons is ***O***(2.5(***n***/3)) if search in a list consist of **N** elements.

تطوير خوارزمية البحث التسلسلي باستخدام الهيكل الثلاثي

الخلاصة

في هذا البحث تم تطوير خوارزمية البحث التسلسلي(Sequential Search) وذلك من خلال استحداث هيكل بياني جديد يسمى الهيكل الثلاثي (Triple Structure) تتمثل فيه البيانات على شكل ثلاثي ويتكون من ثلاث مواقع :

(1- القمة 2- جهة اليسار3- جهة اليمين)

ان خوارزمية البحث التسلسلي (Sequential Search)تحتاج الى مسح(استعراض) جميع عناصر القائمة من بدايتهاوبالتسلسل لحين الوصول الى العنصر المطلوب في حالة وجوده او الوصول الى نهاية القائمة عندما يكون غير موجود ولهذا فان اكبر عدد للمقارنات(معدل التعقيد) سيبلغ تقريبا***O***(***n***)عند البحث في قائمة عدد عناصرها **N**.

في هذا البحث تم تقليص عدد المقارنات الى الثلث وذلك من خلال استخدام الهيكل الثلاثي (Triple Structure) ولهذا فان اكبر عدد للمقارنات سيبلغ تقريبا(2.5(***n***/3))***O***

1- Introduction

A table or a file is a group of elements, each of which is called a record. Associated with each record is a key, which is used to index records or differential records [1].

Each file has at least one set of keys (possible more) that is unique (that is, no two records have the same key). Such a key is called primary key. For example, if the file is stored as an array, the index within the array of an element is a unique external key for that element [1, 2].

A search algorithm is an algorithm that accepts an argument **a** and tries to find a record whose key is **a**. The algorithm may return entire record or, more commonly; it may return a pointer to that record. It is possible that the search for a particular

***Computer Sciences Department, University of Technology/ Baghdad**

https://doi.org/10.30684/etj.29.10.2
2412-0758/University of Technology-Iraq, Baghdad, Iraq

argument in **a** table is unsuccessful; that is, there is no record **i** the table with that argument as its key [3].

Binary search algorithm is faster than sequential and other commonly used search algorithms [8]. The disadvantage of binary search over proposed algorithm is the array must be sorted first, where the proposed algorithm not need sorted array first. This research develops sequential search algorithm by using new structure called Triple structure in this structure data are represented as triple. It consists of three locations (1- Top, 2-Left, and 3- Right).

The advantage of the proposed algorithm over a sequential search is interesting for large numbers. for an array of a two million elements, proposal algorithm, $O(2.5(n/3))$, will find the target element with a worst case of approximately (833000) comparison. Sequential search $O(n)$, on average will take million of comparisons to find the element.

2- Search Algorithms Types

There are many types of search algorithms, searching large amount of data to find one particular piece of information. This research shows in compare the algorithms with the proposed algorithm.

2-1 Sequential Search

Sequential search is the simplest form of search algorithm. This search is applicable to a table organized either as an array or as a linked list. Let us assume that k is an array of n keys, k(0) through k(n-1), and r an array or records, r(0) through r(n-1),such that k(i) is the key of r(i). Let assume that key is a search argument. If to return the smallest integer i such that k (i) equals key if such an i exists and -1 otherwise [1, 8].

As an example: Figure1 shows an array [A], has seven numbers elements, numeric values. To search the array sequentially, sequential search algorithm may be used. The maximum number of comparisons is 7, and occurs when the key we are searching for is in A[6]. The Average time complexity, is $O(n)$, where n is the size of array [8].

The function for doing sequential search for figure 1 is as follows:

```
intfunction SequentialSearch (Array A, int Lb, int Ub, int Key);
    Begin
      For i = Lb to Ub do
        If A[i] = Key then
          Return i;
      Return -1;
    End;
```

2-2 Binary Search

A fast way to search a sorted array is to use the binary search. The idea is to find the element in the middle. If the key is equal to that, the search is finished. If the key is less than the middle element, a binary search on the first half is done. If it's greater, a binary search of the second half will be done [3,4,7].

The advantage of a binary search over a linear search is interesting for large numbers. For an array of a million elements, binary search, $O(\log 2\ n)$, will find the target element with a worst case of only 20 comparisons. Sequential search, $O(n)$, on average will take 500,000 comparisons to find the element [5,7,9].

The function for doing binary search in figure 1 is as follows [4,8]:

```
int function BinarySearch (Array A, int Lb, int Ub, int Key);
    Begin          Do forever
      M = (Lb + Ub)/2;
      If (Key < A [M]) then
        Ub = M – 1
      Else if (Key > A [M]) then
```

```
Lb = M + 1;
            Else
              Return M;
        If (Lb > Ub) then
          Return -1;
  End;
```

2-3 Binary Search Tree (BST)

In the binary tree the nodes are labeled with elements of an ordered dynamic set and the following BST property is satisfied: all elements stored in the left sub tree of any node x are less than the element stored at x and all elements stored in the right sub tree of x are greater than the element at x. The average time complexity of search in BST, is $O(\log 2\ n)+O(n)$, where n is the number of nodes in the tree[7,8].

An Example: Figure 2 shows a binary search tree. Notice that this tree is obtained by inserting the values 13, 3, 4, 12, 14, 10, 5, 1, 8, 2, 7, 9, 11, 6, 18 consequently, starting from an empty tree. in order traverse of a binary search tree always gives a sorted sequence of the values. This is a direct consequence of the BST property. This provides a way of sorting a given sequence of keys: first, create a BST with these keys and then do an in order to traversal of the BST so created [8]. the largest element value in a BST can be found by traversing from the root in the right direction all along until a node with no right link is found (called that the rightmost element in the BST).

While the lowest value element in a BST can be found by traversing from the root in the left direction all along until a node with no left link is found (call that the leftmost element in the BST).

Search is straightforward in a BST. Start with the root and keep moving left or right using the BST property. If the searched key is included in the tree, search procedure will find it other wise the search algorithm will return a null link [8].

3- Triple Structure

A triple structure is a data structure consists of three locations first location is called top, second location is called left and third location is called right. (See figure 3).

The maximum number is stored in top, the number less than top is stored in left, and the number less than left is stored in right.

As example: figure 4 shows nine elements represented in triple structure, the key elements are: (60 40 50 70 90 80 30 20 10)

4- The Proposed Algorithm

The idea is to look at the represented in triple structure and sorted elements in descending order in all triple structure, and start search in sequence. If the key is equal to TOP, the search is finished. If the key is grater than TOP the sequential search on the next triple structure is done. If the key is less than the TOP and the key is less than the Right, then compare the key with Left, if equal the search is finished, if not equal the sequential search on the next triple structure is done.

The algorithm for doing proposal algorithm search is as follows:

Input: A List of elements, and Key (the search key)

Output: Position (such that array [position] =Key)

Step1: Begin

Step2: Representing the elements in triple structure

Step3: Sorting the elements in descending order

Step4: While not end of triple
Structure and not found DO
Begin
If (Key = array [Top]) then Key

Found
Else
If (Key > array [Top]) then
go to next triple structure and go to step4
Else
If (Key < array [Top] then
If (Key = array [Right]) then
Key found
Else
If (Key = array [Left]) then
Key found
Else
Go to next triple structure and go to step4
If (Key < array [Top]) and (Key > array [Left]) then
Go to next triple structure and go to step4
End /end while/
Step5: End

5- Discussion of the Proposed Algorithm

To clarify the proposed algorithm, consider the following keys:
30 20 50 15 10 60 65 35 55 70 33 12 9 7 8

Examlpe1: represented best case time to find the key = 70
1- First step: representing the elements in a triple structure
2- Second step: sorting the elements descending
3-Third step: 70>top[50] go to next triple
4-Fourth step: 70>top[60] go to next triple

5- Fifth step: 70>top[65] go to next triple
6- Sixth step: top[70] = 70 so found key
The equation of best case is (***n***/3)

Example 2: represented worst case time to find key =12
1- First step: representing the elements in a triple structure
2- Second step: sorting the elements descending
3-Third step: 12<top[50] check 12<Left[30] check 12<Right[20] go to next triple
4-Fourth step: 12<top [60] check 12<Left [15] check 12<Right [10] go to next triple
5- Fifth step: 12<top [65] check 12<Left[55] check 12<Right[35] go to next triple
6-Sixth step: 12<top [70] check 12<Left [33] check 12=Right [12] so found key
The equation of worst case is (N/3 + N/2) /2

6- Performance

The advantage of a binary search over a sequential search is astounding for large numbers. For an array of a million elements, binary search, ***O***(log ***n***), will find the target element with a worst case of only 20 comparisons. Sequential search, ***O***(***n***), on average will take 500000 comparisons to find the element. The disadvantage of binary search over sequential search is the array must be sorted first and some of data can not be sorted. The advantage of proposed algorithm over sequential search is better for large numbers. For an array of a million elements, proposed algorithm, ***O***(2.5(***n***/3)), will find the target element with a worst case of only 416666 comparisons. Sequential search, ***O*** (***n***), on average will take 500000 comparisons to find the element. The worst case in binary search tree in the height ***h*** is ***O***(***n***) and ***O***(log ***n***) in the best case.

The average complexity time of binary search (log ***n***) +***O***(N(log N)) ,where ***O***(N(log n)) is a big-***O***-notation of Quick sort algorithm ,the average complexity time of sequential search

(***n*/2**),The average complexity time of the binary search tree (***n*/2**) + (log ***n***), The average complexity time of, proposed algorithm, (2.5(***n***/3) / 2).

7-Results

Table (1): illustrates growth rates for proposed algorithm and other search algorithm, if we search in list consist of N elements.

8- Conclusions

This research, introduces a new data structure called Triple Structure used to implement the sequential search algorithm, this structure makes the search processing faster than sequential search algorithm by reducing the number of compression to a third , insertion and deleting operation is easer than binary search algorithm.

The triple structure is active and efficient in data structure such as Tree Structure, and can use it in sorting algorithms such as Bubble Sort.

We can develop triple structure by making it object (Object Oriented Structure) has name and properties not related with other triple structure.

References

[1] Yedidyal Langsam and Aaron M. Tenenbaum, "Data structures using C and C++, Second Edition",1998.

[2] Sartaj Sahni, "Data Structures, Algorithms, and Applications in C++", 1998.

[3] Rebert Lafor, " Data Structures and Algorithms in Java", 2001.

[4] Algorithms http://planetmath.org/encyclopedia/Binary-search.html ,2008.

[5] Binary Search Algorithms http://www.scriptol.com/programming/binary-search.php ,2008.

[6] Searching Algoritms http://www.fredosaurus.com/notes-cpp/algorithms/searching/binarysearch.html ,2009.

[7] Thomas Niemann, "Sorting and Searching Algorithms", http://epaperpress.com/sortsearch/download/sortsearch.pdf ,2009.

[8] Binary Search Tree http://lcm.csa.iisc.ernet.in/dsa/node91.html,2009.

[9] Algorithms and Data Structures http://www.algolist.net/algorithms/Binary-Search, 2009.

Table (1): results

Size of list ***n***	Average time of Proposal algorithm (2.5(***n***/3) / 2)	Average time of Sequential search (***n***/2)	Average time of binary search tree (***n*/2**) + (log ***n***),	Average time of binary search (log ***n***) + (***n*/2**(log ***n***))
500	**208**	**250**	**259**	**2259**
1000	**416**	**500**	**510**	**5010**
5000	**2083**	**2500**	**2513**	**32513**
10000	**4166**	**5000**	**5014**	**70014**
1000000	**416666**	**500000**	**500020**	**10000020**
20000000	**8333333**	**10000000**	**10000025**	**25000025**

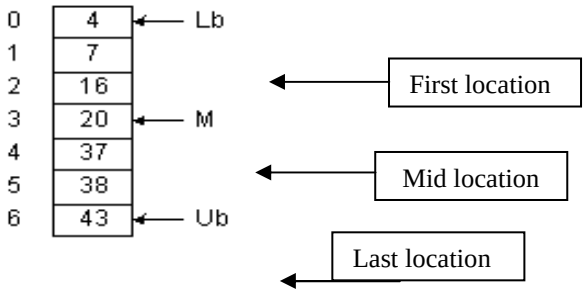


Figure 1: An Array

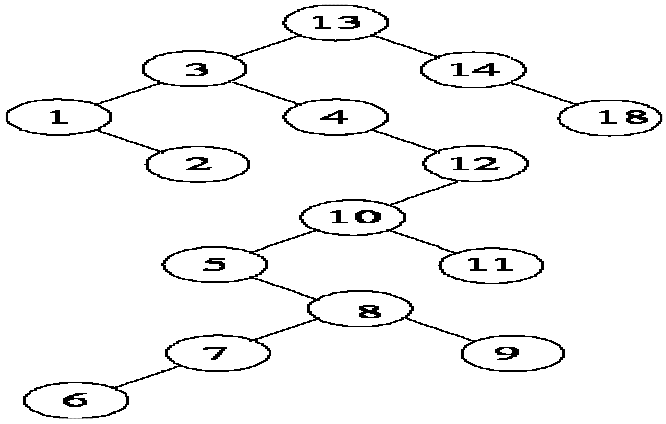


Figure 2: An example of a binary search tree

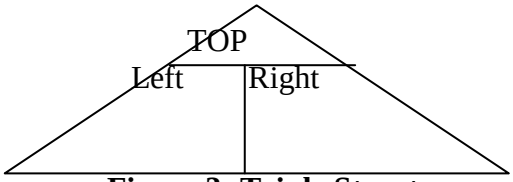


Figure 3: Triple Structure

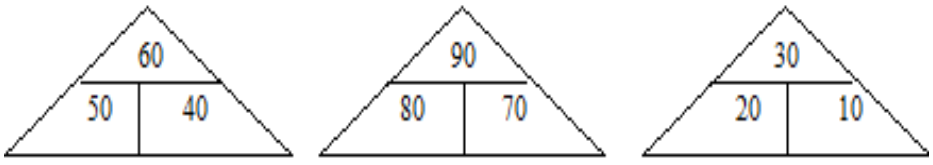
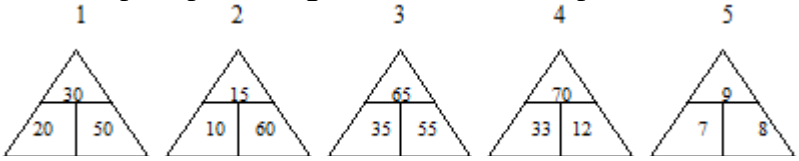


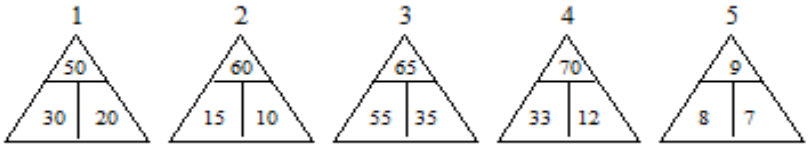
Figure 4: An example of triple structure

Examlpe1 : represented best case time to find the key = 70

1- First step: representing the elements in a triple structure

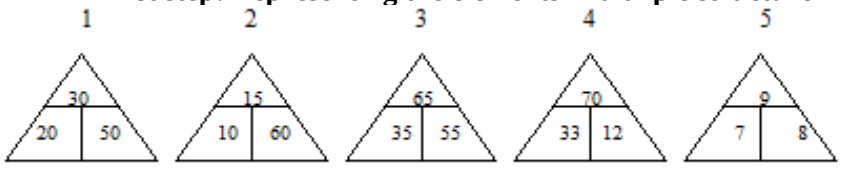


2- Second step: sorting the elements descending



Example 2: represented worst case time to find key =12

1- First step: representing the elements in a triple structure



2- Second step: sorting the elements descending