Development Binary Search Algorithm

Ragheed D. Salim*

Received 3, January, 2011
Accepted 20, May, 2011

Abstract:

There are many methods of searching large amount of data to find one particular piece of information. Such as find name of person in record of mobile. Certain methods of organizing data make the search process more efficient the objective of these methods is to find the element with least cost (least time).
Binary search algorithm is faster than sequential and other commonly used search algorithms.

This research develops binary 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)

Binary search algorithm divide the search interval in half, this process makes the maximum number of comparisons (Average case complexity of Search) is $\boldsymbol{O}(\log 2\ \boldsymbol{n})$ (pronounce this "big-Oh-n" or "the order of magnitude"), if we search in a list consists of ($\boldsymbol{N}$) elements.

In this research the number of comparison is reduced to triple by using Triple structure, this process makes the maximum number of comparisons is $\boldsymbol{O}(\log 2\ (\boldsymbol{n})/3+1)$ if we search key in list consist of ($\boldsymbol{N}$) elements.

Key words: Big Oh (A notation formally describing the set of all functions which are bounded above by a nominated function).

Introduction:

A table or a file is group of elements, each of which is called a record. Associated with each record is a key, which is used to differential among different records [1].For every file there is 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 searching 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 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. This research develops binary search algorithm by using new structure called Triple structure in this structure data are represented as triple. It consists of three locations (Top, Left, and Right).

The advantage of a proposal algorithm over a binary search is astounding for large numbers, for an array of a million elements, proposal algorithm, $\boldsymbol{O}(\log 2\ (\boldsymbol{n})/3)+1$, will find

*University of Technology

the target element with a worst case of only 8 comparison. Binary search ***O***(log2 ***n***), on average will take 20 comparisons to find the element.

Types of Search Algorithms :

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

1 Sequential Search:

This simplest form of a search is the sequential search. This search is applicable to a table organized either as an array. Let 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[1,4,5,6] .

An example: Figure1 shows an array (name of array is A), seven elements long, containing numeric values. To search the array sequentially, may use the algorithm of sequential search. The maximum number of comparisons is 7, and occurs when the key we are searching for is in A [7]. Average case complexity of Search, is O(n), where n is the size of array [4].

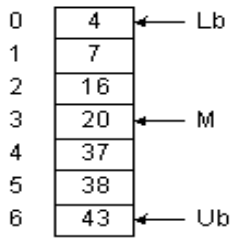


Fig. 1: An Array

The function for doing sequential search for above example is as follows:

```
int function Sequential Search (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- Binary Search:

A fast way to search a sorted array is to use a 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, 5,6].

The advantage of a binary search over a linear search is astounding for large numbers. for an array of a million elements, binary search, ***O***(log2 ***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].

The function for doing binary search in figure 1 is as follows [7]:

```
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;
```

Binary Search Tree:

Search is straightforward in a BST. Start with the root and keep moving left or right using the BST property. If the key we are seeking is present, this search procedure will lead us to the

key. If the key is not present, we end up in a null link [9,10,11].

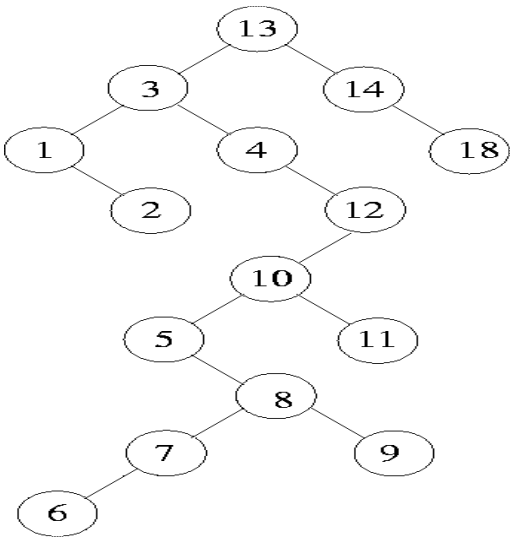


Fig. 2: An example of a binary search tree

Triple Structure:

A triple structure is a structure consists of three locations first location is called top, second location called left and third location called right.

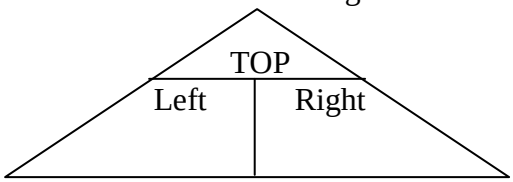


Fig. 3: Triple Structure

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.

An example: figure 4 shows nine elements represented in triple structure, the key elements are:

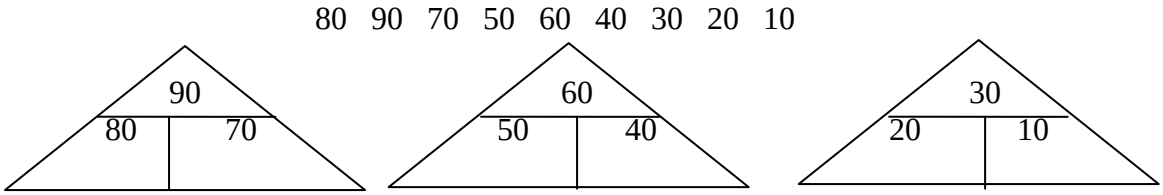


Fig. 4: An example of triple structure

Proposed Algorithm:

The idea is to look at the sorted elements in descending order and represented in triple structure, and find the middle structure. If the key is equal to TOP, the search is finished. If the key is less than the TOP and the key is greater than the Left, then compare the key with Right, if equal the search is finished. If the key is less than the TOP and the key is less than the Left, a binary search on the second half is done. If it's greater, a binary search of the first half will be done.

The algorithm for doing the proposed algorithm search is as follows:

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

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

Step1: Begin

Step2: Sort the elements in descending order

Step3: Represent the elements in triple structure

Step4: While (Lower<=Upper) and (not found) Do

Begin

Find middle structure (mid ← (Lower + Upper) Div 2)

If (Key = array[Top]) then Key found /found is true/

Else If (Key < array [Top]) And(Key > array [Left]) then

if (Key = array [Right]) then Key found /found is true/

Else If (Key < array [Top]) And (Key < array [Left]) then

Upper ← mid + 1

Else If (Key > array [Top]) then

Lower ← mid - 1

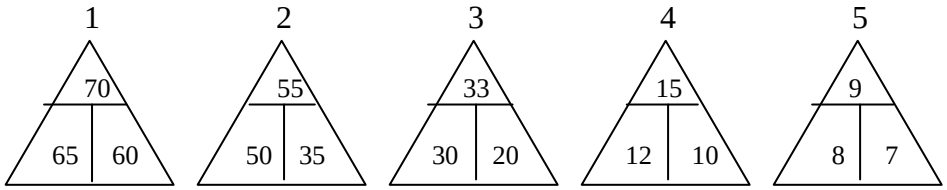
Else
Return step4
End
Step5: If Lower>Upper then Key not found
Step6: End

Explanation of the Proposal Algorithm:

To clarify of the proposed algorithm we will take the following example that regarding the following keys:
30 20 50 15 10 60 65 35 55 70 33 12 9 7 8
to find the key = 70

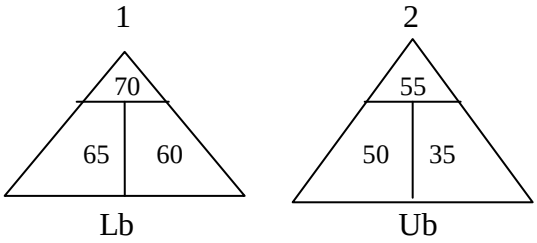
1- First step: sort the elements in descending order
70 65 60 55 50 35 33 30 20 15 12 10 9 8 7

2- Second step: representing the elements in a triple structure

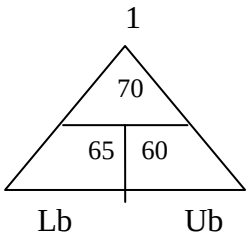


3-Third step: find middle structure
Top[33] < 70 so move Ub to (mid -1)

4-Fourth step: Lb<= Ub and not found
Mid =(1+5)/2 =3



5- Fifth step: Lb<= Ub and not found
Mid =(1+2)/2 =2
Top[55] < 70 so move Ub to (mid -1)



6- Sixth step: Lb<= Ub and not found
Mid =(1+1)/2 =1
Top[70] = 70 so found key

Results and Conclusions:

Figure 5 illustrates growth rates for proposed algorithm and binary search algorithm, if we search in list consist of (***n***) elements.

Fig. 5: results

Size of list (***n***)	Proposed algorithm ***O***(log2 (***n***)/3)+1	binary search algorithm ***O***(log2 ***n***)
1000	4	10
5000	6	12
1000000	8	20
5000000	9	22
10000000	10	23
20000000	11	24

This research, introduces a new structure called (Triple Structure) used to implement the binary search algorithm, this structure makes the search processing faster than binary search algorithm by reducing the number of comparison to triple.
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).

References:

[1] Yedidyal, L. and Aaron, M. T. 1998 "Data structures using C and C++, Prentice-Hall, Second Edition, India, :661.

[2] Sartaj, S., 2002, "Data Structures, Algorithms, and Applications in C++",. MCGraw-Hill, Second Edition, U.S.A, pp814.

[3] Rebert, L. 2003 " Data Structures and Algorithms in Java", McGraw-Hill,First Edition, U.S.A pp 416.

[4]- الصفار، عصام.2001 ."هياكل البيانات" الطبعة الثانية, دارالسفيرللطباعة العراق 301.

[5] Manber, U. and Baeza, R. 2001," An Algorithm for String Matching with a Sequence of Don't Cares".*Information Processing Letters 37*(3):133-136.

[6] Mehlhorn, K. 2005. Dynamic Binary Search. *SIAM J. Computing 8*(2): 175-198.

[7] Thomas, N. 2009. "Sorting and Searching Algorithms", second Edition.Edition, U.S.A,PP305.

[8] Robert, N. 2006. "Generalized Binary Search",IEEE Trans. Inform. Theory, 52(2): 489–509 .

[9] Cormen, T. and Charles E. 2001,"Introduction to Algorithms", McGraw-Hill ,Second Edition,New York, pp 820.

[10] Tenenbaum,A.M and Augenstion, M.J.,2000. "Data Structures Using C++", Prentice-Hall International, Second Edition, India, pp543.

[11] Aho Alfred,V. and Jeffrey D.U.,2003." Data Structures and Algorithms ",Addison-Wesley, Third Edition.U.S.A, PP682.

تطوير خوارزمية البحث الثنائي

رغيد داود سالم*

*الجامعة التكنولوجية.

الخلاصة:

هنالك عدة خوارزميات تستخدم في البحث عن عنصر معين في مجموعة من البيانات اذا كان العنصر موجود. مثلا ايجاد اسم شخص في سجل الموبايل.

ان الهدف من هذه الخوارزميات هو ايجاد العنصر المطلوب باقل كلفة ممكنة(اقل وقت) و تكون هذه الخوارزميات اكثر كفاءة عندما تكون البيانات مرتبة وفق نسق معين.

تعتبر خوارزمية البحث الثنائي (Binary Search) من الخوارزميات التي تحتاج الى وقت قليل في ايجاد العنصر المطلوب لانها سوف تقلص عدد المقارنات الى النصف.

في هذا البحث تم تطوير خوارزمية البحث الثنائي (Binary 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 (Top)

2- جهة اليسار (Left)

3- جهة اليمين (Right)

ان خوارزمية البحث الثنائي (Binary Search) في كل مقارنة تقلص عدد المقارنات اللاحقة الى النصف ولهذا فان اكبر عدد للمقارنات(معدل التعقيد) سيبلغ تقريبا($\boldsymbol{O}(\log 2\ \boldsymbol{n}$)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 تقريب $\boldsymbol{O}(\log 2(\ \boldsymbol{n})/3)+1)$.