Comparing Linear Search and Binary Search Algorithms to Search an Element from a Linear List Implemented through Static Array, Dynamic Array and Linked List

Vimal P.Parmar
Research Scholar,
Department of Computer Science
Saurashtra University, Rajkot
Gujarat, INDIA

CK Kumbharana, PhD
Head, Guide
Department of Computer Science
Saurashtra University Rajkot
Gujarat, INDIA

ABSTRACT
Information retrieval is the most fundamental requirement for any kind of computing application and which requires search operation to be performed from massive databases implemented by various data structures. Searching an element from the list is the fundamental aspects in computing world. Numbers of algorithms are developed for searching an element among which linear search and binary search are the most popular algorithms. In this paper researcher has made efforts to compare these both algorithms to implement on various data structures and to find out the solution to implement binary search on linked linear list. This paper also analyzes both the algorithms at some extent for the applicability and execution efficiency. This paper also analyzes the few data structures to implement these algorithms. At last based on the linear search and binary search algorithms, one algorithm is designed to function on linked linear list.

General Terms
Information retrieval, Data Structures, Searching, Insertion, Deletion, Algorithms, Array, List, Memory Allocation

Keywords
Linear Search, Binary Search, Static array, Dynamic array, Linked List, Binary Search Tree, Time Complexity, Algorithm Efficiency, Algorithm Analysis

1. INTRODUCTION
We always use searching techniques one way or another way to search information. We search specific topic using keywords from a specific book by use of the index given at the end of the book, we search number from telephone directory and we search the English spelling from dictionary. Search operation is involved in day to day life.

Computers are used to process bulk amount of data and searching information from these large database is crucial operation. Almost all computing applications involve searching.

Two widely used and the most fundamental searching techniques are linear search and binary search. Before thinking about any searching techniques it is necessary to determine the organization of data which is termed as data structure. How the data structure is organized is also the responsible for what kind of operations can be performed on data structure. .

Linear search can be applied to both types of arrays static and dynamic as well as linked list, but binary search can not be directly applied to linked list. Here an effort is made to implement binary search for linked list.

Also comparison of array and linked list, comparison of static and dynamic array and comparison of linear search and binary search is described.

Various search algorithms are developed to deal with various data structures. It is not possible to directly implement binary search on linked list. So the solution is developed to apply binary search on linked list.

2. COMPARING ARRAY AND LINKED LIST
Array is the fundamental data structure which is homogeneous, fixed size and static. That means compilers must aware about the information needed for memory allocation. Whereas linked linear list is dynamic in nature.

Following is the detailed comparison between array and linked list.

2.1 Array is static while linked list is dynamic. This implies the size of array must be given prior to compilation whereas size of linked list is changed during the life-span of it. For this reason array can be used only when the number of elements are known otherwise linked list is the best choice.

2.2 Insertion and deletion operation of an element requires shifting of elements which is the time consuming process whereas the same operations can be efficiently implemented using linked list.

2.3 Insertion and deletion operation of an element requires shifting of elements which is the time consuming process whereas the same operations can be efficiently implemented using linked list.

2.4 Searching and element with known index is efficient and directly can be located using an array but it requires searching process for a linked list.

2.5 Array is easy to implement and understand but linked list requires little programming expertise and knowledge of pointer and memory allocation is mandatory.

2.6 Memory allocation of an array is at compile time so efficiently can be sequentially allocated. Memory allocation of an element in linked list is at run time so

runtime overload is imposed and memory allocation of elements does not require sequentially. Elements are allocated in linked list from the free available memory pool.

2.7 Both the linear search and binary search can be applied to array. Whereas linear search can be applied to linked list but binary search can not be directly implemented to linked list.

So it is necessary to design a solution to apply binary search on linked linear list.

3. LINEAR OR SEQUENTIAL SEARCH ALGORITHM

Linear search or sequential search algorithm searches an element from array or linked list by examining each of the elements and comparing it with the search element starting with the first element to the last element in the list. If an element is found then index, flag signal or value can be returned or processed, otherwise special index as -1 or flag signal can be returned.

It is called linear or sequential because this algorithm scans the elements in a list in linear manner or in sequence by scanning one by one element in a list..

Time required to search an element if exist or to determine that element is not found depends on the total number of elements in the list. So the time complexity of linear search is O(N)[1][9].

Worst case requires N comparison if an element is found at the last position N or an element does not exist in the list.

Average time required by linear search on an average we can say an element may be at center of the list, so average time for this algorithm is O(N/2) which is again linear in nature.

The best case time is O(1) which is constant and element is found at the first position.

In general time required by linear search is O(N) as Big-O notation is simply determines the highest order terms and ignores the coefficients and lower order terms.

Due to its simplest implementation it can be applied to array list as well as all types of linked lists. So it is easy to implement but it is not useful when the size of the list is too large. Because the time require is proportional to total number of elements N. So it is useful when we have small size of an array or a linked list but require more time when an array becomes large.

The algorithm of linear search technique is listed below which finds the location of the search term if found otherwise the message of element not found is displayed.[1]

Algorithm LINEAR_SEARCH (V, N, X) : Given a vector V containing N elements, this algorithm searches V for a given value of X. FOUND is a Boolean variable. I and LOCATION are integer variables.

1. [Search for the location of value X in vector V]

```
FOUND ← false
I ← 1
Repeat while I ≤ N and not FOUND
        If      V[I] = X
        then    FOUND ← true
                LOCATION ← I
                Exit
        else
                I ← I + 1
Write ('Value of', X , 'NOT FOUND')
```

2. [Finished]

Exit

Above algorithm searches vector starting from the first element upto the last element in linear manner for the search element X.

Each time an index variable I is increment to search next element. If an element does not exist in the vector then entire vector is scanned and then result in element not found.

The same algorithm can be applied to linked list whose typical node structure contains information field(s) and one pointer field that points to the next node. The last node's pointer field contains NULL means there are no more elements and which is the last node in the list.

Following algorithm is again the linear search algorithm which search the linked list..

Algorithm LINEAR_SEARCH_LINK_LIST (FIRST, X) : Given a linked list whose typical node structure contains INFO and LINK fields to store information and address for the next node respectively. First node address of the list is given by the pointer variable FIRST. This algorithm searches the linked list for a given value of X. If X is found in linked list then it displays the message that element is found otherwise it displays the message that element is not found.

1. [Initiate search]

```
SAVE ← FIRST
```

2. [Search for the X in linked list]

```
Repeat while LINK (SAVE) ≠ NULL
    If      INFO (SAVE) = X
    then Write ('Value of ', X,'FOUND')
            Exit
    else
            SAVE    ← LINK (SAVE)
```

3. [Finished]

```
Write ('Value of', X , 'NOT FOUND')
```

Exit

Above algorithm searches the linked list for the value given by a variable X The linked list whose first node address is given by pointer variable FIRST. If FIRST is NULL means linked list is empty and value of X is not found is displayed otherwise FIRST contains the address which is stored in pointer variable SAVE.

Algorithms starts comparing for the value of X with the INFO filed of first node and continues until the last node whose LINK field is NULL. The pointer SAVE is updated each time to point to the next node in the linked list.

The time required by both of the above algorithms is O(N) that is linearly proportional to total number of elements in the list.

4. BINARY SEARCH ALGORITHM

The performance of linear search algorithm can be realized when the number of element is too large. In such situation binary search algorithm is quite useful.

The binary search algorithm can be applied to an array whose elements are to be required in sorted form [1][2]. Each iteration of binary search narrows the search interval by half of the search interval of its previous iteration.

The time required by binary search is O ($\log_2$ N)[1][5]. Variation of binary search can be possible by selecting a random element between lower and upper bound.[4]

Function BINARY_SEARCH(K, N, X) : Given a vector K, consisting of N elements in ascending order, this algorithm searches the structure for a given element whose value is given by X. The variables LOW, MIDDLE and HIGH denote the lower, middle and upper limits of the search interval. The function returns the index of the vector element if the search is successful and returns 0 otherwise.

1. [Initialize]
   LOW ← 1
   HIGH ← N
2. [Perform search]
   Repeat thru step 4 while LOW ≤ HIGH
3. [Obtain index of midpoint of interval]
   MIDDLE ← (LOW + HIGH) / 2
4. [Compare]
   If X < K [MIDDLE]
   then HIGH ← MIDDLE – 1
   else If X > K [MIDDLE]
   then LOW ← MIDDLE + 1

   elseWrite('SUCCESSFULSEARCH')
   Return (MIDDLE)
5. [Unsuccessful search)
   Write('UNSUCCESSFULSEARCH')
   Return (0)

Above algorithm searches an element X by first examining the center element of the vector K. If it is found then search process is completed otherwise it will check for the element whether exist in first half or the second half by testing the condition. If X < K[MIDDLE] then element might be in first half so the upper limit of the next interval is changed by MIDDLE – 1. If X > K[MIDDLE] then element might be found in second half of the current interval so the lower limit is changed by MIDDLE + 1. This process continues until LOW <= HIGH.

5. COMPARISON LINEAR SEARCH VS BINARY SEARCH

Following table 1 compares the search algorithms linear search and binary search.

Table : 1 Comparison of Linear search and Binary search

Sr. No.	Comparison	Linear Search	Binary Search
1	Time Complexity	O(N)	O ($\log_2$ N)
2	Best case time	O(1) First Element	O(1) Center Element
3	Prerequisite for an array	No prerequisite	Array must be in sorted order
4	Worst case for N = 1000 elements	1000 comparisons required	Can conclude after only 10 comparisons.
5	Can be implemented on	Array and Linked list	Cannot be directly implemented on linked list
6	Insert operation when we use either search	Easily inserted at the end of list	Require processing to insert at its proper place to maintain sorted list.
7	Algorithm type	Iterative in nature	Divide and conquer in nature
8	Usefulness	Easy to use and no need for any ordered elements	Somehow tricky algorithm and elements must be arranged in order
9	Lines of Code	Less	More[6]

Both the algorithms are quite useful depending on the application.

If array is the data structure and elements are organized in sorted order then binary search is preferred for fast searching. If linked list is the data structure no matter how the elements are arranged, linear search is preferred due to unavailability of direct implementation of binary search algorithm.

We require other data structure to obtain the effect of binary search or we require to design the variation of binary search algorithm that can work on linked list too because binary search is faster in execution than a linear search[6]. The original Binary search algorithm can not be applied to linked list because linked list by nature is dynamic and it is not known where the middle element is actually allocated.

So, some of the efforts are required to apply binary search over a linked list to obtain the advantages.

6. STATIC VS DYNAMIC ARRAYS WITH SEARCHING ALGORITHM

Static array is fixed and cannot be altered its size during program execution. If the size is only a matter then the solution is to implement dynamic array. The size of dynamic array can be determined during execution time.

The both types of arrays are discussed with different view point.

Static arrays are compile-time array declaration and size is known at compilation time while dynamic arrays are determined during run time and its size is determined when the program is actually run.

6.1 Static array is fixed and can not be modified its size during execution of a pprogram. Dynamic array is varying and size is determined during execution of a program. Further, the size of dynamic array can be altered during execution time with some more efforts and processing is taken.

6.2 Static array can be accessed by array name as well as pointer where as pointer must be necessary to use for dynamic array.

6.3 Both static and dynamic arrays occupy memory contiguously and both of the linear search as well as binary search algorithms can be implemented on both types of these arrays

6.4 Static array memory management is implicitely being made by language itself while memory management for dynamic array is the explicitelycore responsibility of the programmer itself.

6.5 We can have full access to the static array through out the execution of the program. If we loss the pointer pointing to the dynamically allocated array resulting in to garbage array to which we have no access at all.

6.6 Static arrays are easy to use and manipulate compared to dynamic array in which little expertised is required.

Processing of both the static and dynamic arrays are different but both the searching algorithms that is linear and binary search can be implemented by using of either type of array.

For dynamic array, total number of elements must be kept up to date which is not necessary in static array because the size of the array is already known. But in dynamic array it must be updated when an element is inserted or deleted.

7. IMPLEMETATION OF BINARY SEARCH ALGORITHM FOR OTHER DATA STRUCTURE

Binary search algorithm can only be applied to sorted array whether it is static or dynamic. This algorithm can not be directly implemented to linked list[2].

Although we can obtain the benefits of binary search by organizing the same array elements in non linear data structure tree[8]. Binary search tree is a non-linear data structure which searches an element in equal amount of time as binary search requires $O(\log_2 N)$[1][5]. But it is difficult to maintain and manipulate binary search tree.

The second option to implement binary search on linked list is to copy all the elements of linked list into either sorted array or a binary search tree. This option is again not practical for maintenance of the data as each time searching will be faster but require more processing of creating and copy elements. Extra overload will be faced by processor in obtaining the benefits of binary search.

The third alternative is to derive new algorithm that can equally perform binary search on linked list by making alternative changes in original binary search algorithm.

Directly it is not possible to implement binary search over linked list due to dynamic memory allocation and searching center element through address calculation. So we keep track of all the addresses of each node in a separate array of pointer.

8. IMPLEMENTATION OF BINARY SEARCH FOR SEARCHING AN ELEMENT FROM LINKED LIST

Function BINARY_SEARCH_LINKED_LIST (FIRST, N, X, PTR) : Given a linked list whose typical node structure contains INFO and LINK fields to store information and pointer to the next node of the linked list. FIRST is the pointer to the first node of the linked list. Linked list is organized in order of INFO field in increasing order of value. PTR is an array of pointer whose first element pointing to first node of linked list, second element pointing to second node of the linked list and so on. N is the total number of element in the list. This algorithm searches the linked list structure for a given element whose value is given by X.

The variables LOW, MIDDLE and HIGH denote the lower, middle and upper limits of the search interval. The function returns the index of the linked list element if the search is successful and returns 0 otherwise.

1. [Initialize]
   LOW ← 1
   HIGH ← N
2. [Perform search]
   Repeat thru step 4 while LOW ≤ HIGH

3. [Obtain index of midpoint of interval]
   MIDDLE ← (LOW + HIGH) / 2
   SAVE ← PTR[MIDDLE]
4. [Compare]
   If X < INFO(SAVE)
   then HIGH ← MIDDLE – 1
   else If X > INFO(SAVE)
   then LOW ← MIDDLE + 1
   else
   Write('SUCCESSFULSEARCH')
   Return (MIDDLE)

5. [Unsuccessful search)
   Write('UNSUCCESSFULSEARCH')
   Return (0)

9. CONCLUSION

From the above three algortihms it can be concluded that the binary search is more efficient searching technique than linear search but insertion of an element is not efficient as it requires arranged elements in specific order. Further it is also possible to apply binary search on linked list by making necessary modifications to original binary search algorithm. The selectiom of searching algorithm can be based on the data structure on to which it is applied and which operations are required more. Balance is required between search and maintanance of a data structure.

It is also possible to combine two searching algorithms and obtaining benefits of both. For example binary search algorithm applied for linked list say primary linked list in which each node may represent another linked list say secondary linked list which can be searched using linear search. Hashing technique is another alternate for direct searching an element from a table without concerning where the element is actually allocated. Different applications require different searching techniques base on the insertion, deletion, searching and maintenance operations.

10. REFERENCES

[1] An Introduction to data structures with applications - (Mcgraw Hill Computer Science Series) [Jean-Paul Tremblay, Paul G. Sorenson, P. G. Sorenson]

[2] International Journal Of Innovative Research In Technology © 2014 IJIRT | Volume 1 Issue 5 | ISSN: 2349-6002 IJIRT 100392 800 BINARY SEARCH ALGORITHM Ancy Oommen , Chanchal Pal

[3] Binary Search on linked List – a research project in experimental computer science - Marcin Paprzycki, Firooz Khosraviyani, bark Wagaman Department of mathematics and computer science The university of Texas of the Permian basin Odessa Tx 79762

[4] A Randomized Searching Algorithm and its Performance analysis with Binary Search and Linear Search Algorithms - The International Journal of Computer Science & Applications (TIJCSA) - Volume 1, No. 11, January 2013 ISSN – 2278-1080

[5] Modified Binary Search Algorithm for Duplicate Elements- P PhyuPhyu Thwe, Lai Lai Win Kyi International Journal of Computer & Communication Engineering Research (IJCCER) Volume 2 - Issue 2 March 2014

[6] Analysis of Linear and Binary Search Algorithm – Sapinder, Ritu, Arvinder Singh. International Journal of Computers & Distributed Systems www.cirworld.com Volume 1, No.1, Jun 2012

[7] Linear Search versus Binary Search: A statistical comparison for binomial inputs- Anchala Kumari1, Rama Tripathi2, Mita Pal3 and Soubhik Chakraborty, International Journal of Computer Science, Engineering and Applications (IJCSEA) Vol.2, No.2, April 2012

[8] Binary Search Tree Balancing Methods: A Critical Study Suri Pushpa1, Prasad Vinod IJCSNS International Journal of Computer Science and Network Security, VOL.7 No.8, August 2007

[9] BINARY SEARCH ALGORITHM, Ancy Oommen , Chanchal Pal, 2014 IJIRT | Volume 1 Issue 5 | ISSN: 2349-6002

[10] Searching Large Indexes on Tiny Devices Optimizing Binary Search with Character Pinning