International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 05 Issue: 04 | Apr-2018 www.irjet.net p-ISSN: 2395-0072

Design and Analysis of Bisecting Linear Search for Sorted Array

Rahul Sharma[1], Rohit Kumar[2]

[1]*Dept. of Computer science and Engineering, IIMT Engineering College, Meerut, Uttar Pradesh, India*
[2]*Assistant Professor, Dept. of Computer science and Engineering, IIMT Engineering College, Meerut, Uttar Pradesh, India*

---***---

Abstract – *Searching algorithms are helpful in performing search operations on the array. There are many search operations that use different approach and techniques to search the desired data. Linear search performs search operation linearly. For large array, linear search takes large time to search the value if the value is present at the last. Bisecting linear search is designed for sorted array and works faster as compared to linear search.*

Key Words: array, data structure, linear search, searching, bisecting linear search.

1. INTRODUCTION

In computer science, a search algorithm is an algorithm that solves a search problem, namely, to retrieve information stored within the array or some data structure. Search algorithms are classified based on the search mechanism that they use for searching the value. Digital search algorithms work based on the properties of digits in data structure that use numerical keys. One of the fundamental operations that are performed on computer is searching operation. There are many known searching operations that can perform searching in different conditions. Web development, software development, and databases use searching operations extensively. Search algorithms are analyzed and evaluated on the basis of their complexity. A search algorithm should be fast and less complex. Linear search[1] and Binary search[1,2] are two basic searching algorithms that are used to perform search operations. Computer systems often store huge amount of data from which individual record of data is to be retrieved. Thus the efficient search algorithm will search the data in less time. There are searching algorithms which work faster as compared to other algorithms, but to use those algorithms there are certain conditions. Binary search works on sorted array only whereas linear search can work on both sorted as well as unsorted array. But if the search is to be performed on the sorted array, linear search performs similar operation to search the value. Bisecting linear search is a modification over linear search. Bisecting linear search will work only on array those are sorted in ascending or descending order. It concentrates on only one portion of the array, where the search value may be present, this mechanism makes Bisecting linear search faster.

1.1 Related Work

Linear search is a sequential search which begins at the beginning of the list and searches for the value till the end of the list. If the search value is found it returns the index position of the search value, if the search value is not found in the array then it returns -1. The general concept of searching a value is used by linear search. Linear search works both on the sorted and unsorted array. The pseudocode of linear search is given below:

```
int searchLinear (int[] array, int value)
    for (i = 0; i < array.length; i++)
        if (array[i] == value)
            return i
    return -1
```

The linear search pseudocode has a *searchLinear* method which takes the array and the search value as a parameter. The loop iterate from *0* to *array.length.* Each value of array is compared with the search value. If any value of array matches with the search value then it returns the index position, but if the search value is not found then the method return *-1.*

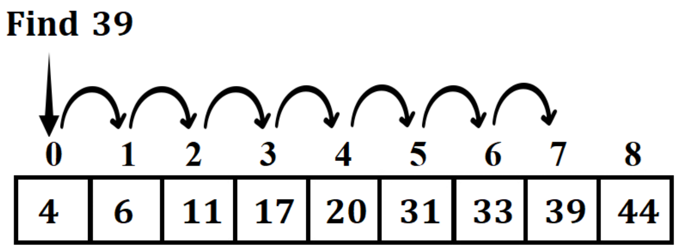

Fig -1: Linear search

Linear search is shown in Fig-1 which is used to search 39 from the given sorted array. Although the array is sorted, it uses the same technique to search the value, and hence the time taken by linear search in both sorted and unsorted array is similar.

1.2 Analysis of Linear search

A linear searching algorithm that searches the specific item in an array. Search cases of a searching algorithm can be categorized as best case, average case and worst case. In some algorithms, all the three cases may be same. In some algorithms, the cases might have a large difference. In most of the cases, the average behavior of the algorithm helps in determining the algorithms usefulness. Linear search operates a simple loop to *length-1* of the array that goes to each element until the search value is found. In the worst case, the loop will iterate from *0* to *length -1.* Consequently, worst case time complexity of linear search would be:

$$f(n) = O(n)$$

In the best case, the desired search value will be present in the first position of the array, and hence only one comparison is made. So the complexity of linear search in the best case would be:

$$f(n) = O(1)$$

2. PROBLEM DEFINITION

While working with the huge data set and array, sometimes searching is necessary to perform the further operations. The value which is to be searched from the array may be present anywhere in the array although the array is sorted, the sorted sequence cannot be used to increase the search speed if linear search is used.

3. THE PROPOSED ALGORITHM

To perform the search operation faster on the sorted array, a new Bisecting linear search algorithm is proposed. In bisecting linear search algorithm, the middle position of the array is found. In case of ascending order, if the search value is less than the middle value then the algorithm performs linear search from both ends of the left sub array. But if the search value is greater than the middle value then the algorithm perform linear search from both ends in the right sub array. Similarly, in descending order, if the search value if less than the middle value then the search is performed on right sub array and so on. The linear search performed by the algorithm is a special type of linear search, which search from both the ends. The pseudocode of Bisecting linear search is given:

```
int search(int a[],int n, int val, int mode)

        if (val==a[n/2])

                return n/2

        else if ((val>a[n/2]) && (mode==1))
//descending order

                i←0
                j← (n/2)-1
                while(i<=j)

                        if (a[i]==val)
                                return i
                        else if (a[j]==val)
                                return j
                        i++
                        j--

        else if ((val<a[n/2]) && (mode==1))
//descending order

                i←n/2+1
                j←n-1
                while(i<=j)

                        if (a[i]==val)
                                return i
                        else if (a[j]==val)
                                return j
                        i++
                        j--

        else if ((val<a[n/2]) && (mode==0))
//ascending order

                i←0
                j← (n/2)-1
                while(i<=j)

                        if (a[i]==val)
                                return i
                        else if (a[j]==val)
                                return j
                        i++
                        j--

        else if ((val>a[n/2]) && (mode==0))
//ascending order

                i←n/2+1
                j←n-1
                while(i<=j)

                        if (a[i]==val)
                                return i
                        else if (a[j]==val)
                                return j
                        i++
                        j--

        return -1
```

The above pseudocode has a *search* method which performs the search operation. It takes an array, size of the array, the search value, and *mode* as a parameter. The variable *mode* may have either *0* or *1*. If the array is ascending then the value of mode should be *0*, but if the array is in descending order then the value of *mode* should be *1*.

If the search value is equal to the middle value, the algorithm return *n*/2. Otherwise, the further operation is performed. If the search value is greater than the middle value and if the mode is *1*, *i* iterates from *0* and *j* iterates from (*n*/2)-1. Search value is compared with values of *i* and *j*. It returns *i* or *j*, when the condition is fulfilled. The pseudocode has another *else if* condition, which executes when the search value is less than the middle value. Similar operation is performed for further conditions. The pseudocode *return -1* when the search value is not present in the array.

Fig-2 and Fig-3 shows the working of Bisecting linear search algorithm for both the conditions.

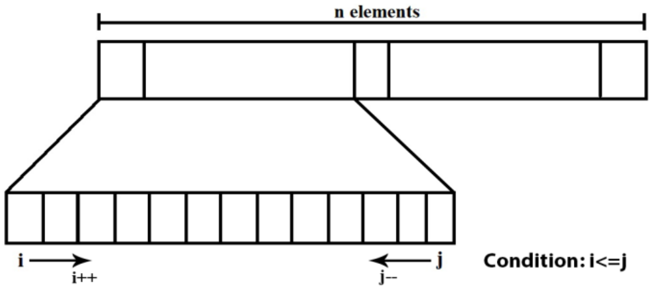

Fig -2: Search value is less than middle value

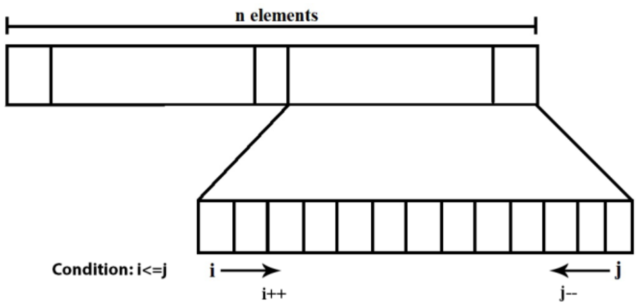

Fig -3: Search value is greater than middle value

3.1 Analysis of Bisecting linear search

For an array with size *n*, the best case is when the search value is the first value of the array. Only one comparison is performed in the best case. The best case of Bisecting linear search will be:

$$f(n) = O(1)$$

In worst case, the search value is not present in the array. In bisecting linear search although the size of the array is *n*, but the loop will run less than *n* times. The time complexity of the algorithm is O(*n*) because the loop variable *i* and *j* increment by a constant value and the most significant term of *n* is taken. The worst case complexity of bisecting linear search is:

$$f(n) = O(n)$$

3.2 Example

Fig – 4 show an example of bisecting linear search. Random values are sorted and search operation is performed using bisecting linear search algorithm. The search value is *33* and array is in ascending order, and hence mode is *0*.

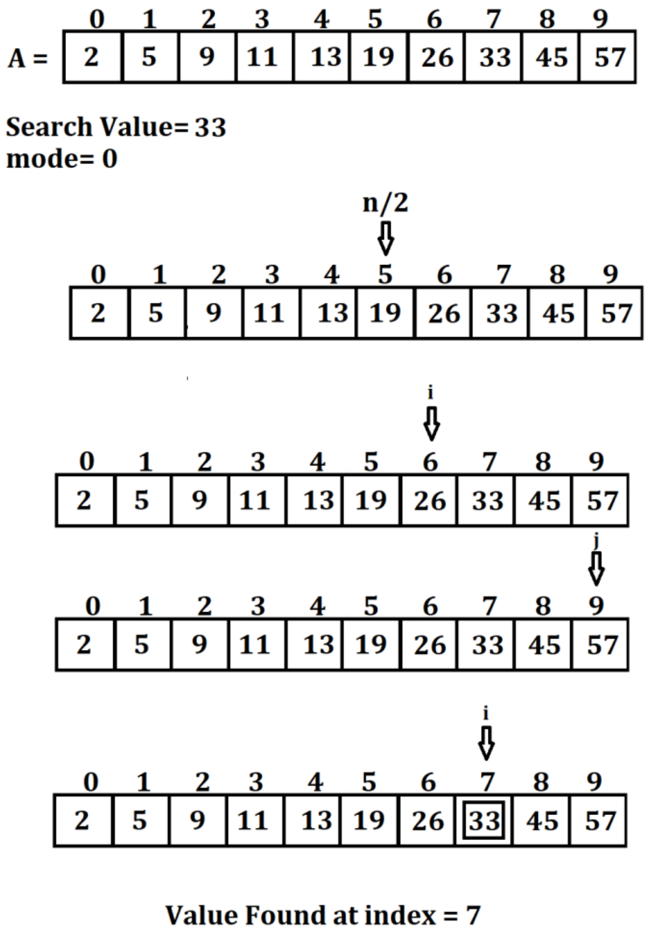

Fig -4: Bisecting linear search example

4. RESULT

The time taken by linear search and bisecting linear search is calculated on a machine of 64-bit Operating system having Intel(R) Core(TM) i5 1.60GHz and 8 GB RAM.

Table -1: Time taken by linear search and Bisecting linear search in worst case

N	Linear search (seconds)	Bisecting Linear search (seconds)
10000	0.000286	0.00022
20000	0.000576	0.00033
30000	0.000849	0.00046
40000	0.001036	0.00065
50000	0.001309	0.00082

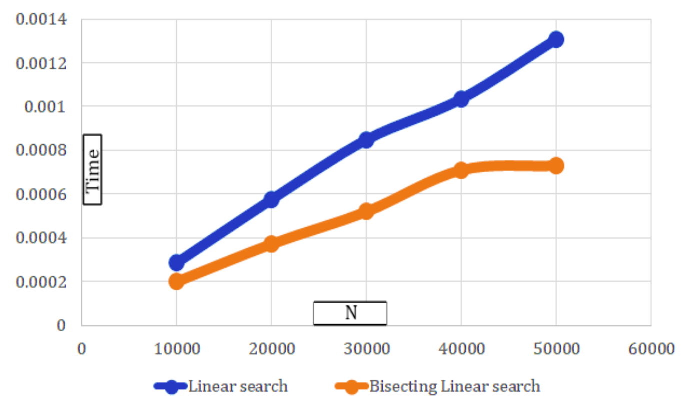

Chart -1: Performance of Linear search and Bisecting linear search in the worst case.

4. CONCLUSIONS

Linear search and Bisecting linear search is performed on the array with sorted data. The data were sorted in ascending order. Linear search can perform the search operation on unsorted as well as on sorted array, but Bisecting linear search is designed for searching the desired value when the array is sorted in ascending or descending order. The Table-1 shows the time taken by both the algorithms to perform the search operation in worst case. The Chart-1 show the graph plotted by the calculated time. The loop in Bisecting linear search iterates for very less number of times as compared to linear search. With the results, we conclude that Bisecting linear search algorithm works faster as compared to linear search.

ACKNOWLEDGEMENT

Writing this research paper has been a fun and rewarding experience for me. My parents encouraged me throughout the research work. I would thank Mr. Pradip Sharma for encouraging me so that I can achieve my target. Mr. Rohit Kumar helped me with his immense knowledge and showed me the direction that is good and adequate for me.

REFERENCES

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

[2] K. J. Overholt, "Optimal binary search methods," BIT, vol. 13, no.1, pp. 84-91, 1973.

[3] D. Coppersmith, "Fast evaluation of logarithms in finite fields of characteristic two," IEEE Trans. Inform. Theory, vol. IT-30, pp. 587-594, 1984.

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

[5] Aho A., Hopcroft J., Ullman J., "The Design and Analysis of Computer Algorithms," Addison Wesley, 1974.

BIOGRAPHIES

Rahul Sharma was born in Jamshedpur, Jharkhand, India in 1996. He currently pursuing the B.Tech degree in computer science and engineering from IIMT Engineering College, Meerut, India.

Rohit Kumar was born in Bihar Sharif, Bihar, India in 1988. He received the B.Tech in Information Technology from Dr. MGR University in 2009 and M.Tech. degrees in Computer Science from NIT Kurukshetra, Haryana, in 2010.