Volume 4, Issue 3, March 2014 ISSN: 2277 128X

International Journal of Advanced Research in Computer Science and Software Engineering

Research Paper

Available online at: www.ijarcsse.com

Bi Linear Search a New Session of Searching

Smita Paira*
B.Tech Student
Department of CSE
Calcutta Institute of Technology, India

Sourabh Chandra
Asst Professor
Department of CSE
Calcutta Institute of Technology, India

Sk Safikul Alam
Asst Professor
Department of CSE
Calcutta Institute of Technology, India

Subhendu Sekhar Patra
Asst Professor
Department of CSE
Calcutta Institute of Technology, India

Abstract— ***It is the age of Technology. Today, the research is putting focus on innovative development and comparing and analysing the existing technologies. The field of Software Technology is always associated with some basic operations. These operations operate, based on some internal codes. One of these basic operations is searching. It has both practical as well as real life applications. Searching is basically of two types-sequential (Linear Search) and random search, which include the Binary, Jump and Interpolation Search. Studies have proved Binary Search to be the best, though it requires the sorted order of the array. This paper deals with the introduction of a new searching technique (Bi-linear Search), that can operate on both sorted and unsorted array sequentially, and yet more efficient than many other searching algorithms.***

Keywords— ***Bi Linear Search, Linear Search, Binary Search, Interpolation Search, Jump Search, Time Complexity.***

I. INTRODUCTION

In our day-to-day life, sometimes a situation arises when we need to find out some item from a collection of things. Searching is the process that works behind this. It is the process of finding/extracting a particular element from a given list of elements. For example, if we want to access a particular disk from a drive, we require searching operation. The access can be sequential or random. There are many searching algorithms developed namely-Linear Search, Binary Search and Interpolation Search. Both Interpolation and Binary Search works randomly and executes faster than the Linear Search, the algorithm that runs sequentially. Hence, they are highly efficient. But, for both the algorithms, the list of items should be in sorted order.

So, our main objective is to overcome these drawbacks and thus the Bi-linear Search exists. This technique executes sequentially, provided the array can be sorted or unsorted. The Bi-linear Search works from both end of the array. In first iteration, both the first and last elements are compared simultaneously with the search key. If the data is found, the searching method stops.

Otherwise, the second element and the second to last element are compared simultaneously and so on. Thus, the number of steps/comparisons required is n/2 i.e. half of what is required by Linear Search. Let us consider an array of 8 elements. The steps for searching an element 7 is shown below:-

Given array

1	2	3	4	5	6	7	8
0	1	2	3	4	5	6	7

Step 1: The element is searched at the locations shown by the arrows.

1	2	3	4	5	6	7	8
0	1	2	3	4	5	6	7
↑							↑

Step 2: Since, 7 is not found, so the search continues and the next locations are searched as shown by the arrows.

1	2	3	4	5	6	7	8
0	1	2	3	4	5	6	7
	↑						↑

Now, the element 7 is found and the Bi-linear Search stops. Thus, only two executions are required. But if the same thing is done by Linear Search then the latter would have taken seven steps. The Bi-linear Search is the best among all others searching techniques, having a worst case time complexity of O (1).

II. PSEUDO CODE

Let us consider an array, a of size n. Let i and j be the loop variables and item be the element to be searched. The step-by-step procedure for implementing the Bi-linear Search is as follows:-

```
Step 1: Enter the value of item.
Step 2: Initialise i to 0 and j to n-1.
Step 3: Continue step 4 - step 5 until i is incremented to n/2 and j is decremented to n/2.
Step 4: Check whether a[i] or a[j] is equal to item.
Step 5: If equal then break the loop and go to step 6 otherwise Go to step 3.
Step 6: If j is equal to ((m/2)-1) then print “Item Not Found” and go to step 9 otherwise follow step 7.
Step 7: If a[i] is equal to item then:
            x=i+1 otherwise make x=j+1

Step 8: Print that the item is found in position x
Step 9: End
```

III. IMPLEMENTATION OF BI-LINEAR SEARCH USING JAVA AND C

Let’s see how the pseudo code can be realized using object oriented programming approach and a function oriented programming language as well.

The Bi-linear Search in Java is as follows:-

```
import java.io.*;
class search
{
public static void main(String args[])throws IOException
{
BufferedReader buf=new BufferedReader(new InputStreamReader(System.in));
 int a[]=new int[100];
System.out.println("Enter the number of elements :-");
 int n=Integer.parseInt(buf.readLine());
int i,j;
 System.out.println("\n Enter the array elements :-");
for(i=0;i<n;i++)
a[i]=Integer.parseInt(buf.readLine());
System.out.println("\n The given array is :- \n");
for(i=0;i<n;i++)  //printing the array
        System.out.print(a[i]+" ");
Bilinear(a);  //calling the Bilinear function

static void Bilinear(int b[])
{
 int item,x;
System.out.println("Enter the item to be searched for:-");
int item=Integer.parseInt(buf.readLine());
for(i=0,j=n-1;i<=(n/2),j>=(n/2);i++,j--)
{
        if((b[i]==item)||(b[j]==item))
        break;
}
if(j==((n/2)-1))
        System.out.println("\n Item not found!!!!");
else
{
        if(b[i]==item)
        x=i+1;
        else
        x=j+1;
        System.out.println("\n Item found in %d position.",x);
}
}
}
}
```

The Bi-linear search in C is as follows:-

```
#include<stdio.h>
#include<conio.h>
void bilinear(int *,int);
void main()
{
        int a[100],i,n;
        clrscr();
        printf("\n Enter the number of        elements...");
        scanf("%d",&n);
        printf("\n Enter the elements...");
        for(i=0;i<n;i++)
                scanf("%d",&a[i]);
        printf("\n The given array is...\n");
        for(i=0;i<n;i++)
                printf("%d ",a[i]);
        bilinear(a,n);
        getch();
}
void bilinear(int *b,int m)
{
int i,j,x,item;
printf("\n Enter the item to be searched....");
scanf("%d",&item);
for(i=0,j=m-1;i<=(m/2),j>=(m/2);i++,j--)
{
        if((b[i]==item)||(b[j]==item))
        break;
}
if(j==((m/2)-1))
        printf("\n Item not found!!!!");
else
{
        if(b[i]==item)
                x=i+1;
        else
                x=j+1;
        printf("\n Item found in %d position.",x);
}
}
```

IV. TIME COMPLEXITY

The efficiency of the searching algorithm lies in its time complexity or the number of comparisons made. Let the number of elements of an array be n [1, 2, 3, 4]. For a single iteration, as the value of i increases, the value of j decreases simultaneously. The algorithm makes only one comparison for the first and last positions of the array. It makes two comparisons for the second and second to last positions of the array. Thus, for two positions of the array only one comparison is made and for four positions of the array, two comparisons are made and so on. So, if there is n number of elements, the loop iterates for n/2 number of times. Hence, the average case time complexity of Bi-linear Search is O (n).

If the item is present in the first and last position of the array, then only one comparison is made. The first execution of the loop finds the item and the search process stops. Hence, the best case and worst case time complexities of Bi-linear Search are both O (1).

V. COMPARISON ANALYSIS

The working efficiency of Bi-linear Search is very high in compared to other searching algorithms. If we make a study of the pseudo codes of the basic searching algorithms, we detect that the average case analysis of both Binary and Interpolation search is the best [7,9]. Jump Search is better than Binary Search when the array is very large and the search key lies close to the starting element. Binary Search directly checks the middle of the array and then access the key from the middle which takes a lot of time, whereas Jump Search does not do the same [10]. But binary search, interpolation search and jump search are applicable only when the array is sorted [11]. If the list is not sorted, then it has to be sorted first and then the searching operation is performed. Thus, the above three algorithms consume more time [1,5].Whereas, Linear and Bi-linear Search algorithms run for both sorted and unsorted arrays We have made a brief comparison study of Bi-linear Search with other searching algorithms like Linear Search and Binary Search, in the form of a table as shown below:-

Table 1: Comparison Study of Bi-linear Search with different Searching algorithms

Conventional Searching Algorithm		**Linear Search**	**Bi Linear Search**	**Binary Search (Considering Sorted Array)**
Time Complexity	**Best**	O(1)	O(1)	O(1)
	Average	O(n)	O(n)	O(log n)
	Worst	O(n)	O(1)*	O(log n)
Best Case Analysis (Unsorted Array Size of 20,50 and 100 respectively)	**No. of comparisons**	1 1 1	1 1 1	1 1 1
	Execution Time (ms)	0.47252747 0.47252747 0.47252747	0.47252352 0.47252352 0.47252352	0.4527472527 0.4527472527 0.4527472527
Average Case Analysis (Unsorted Array Size of 20,50 and 100 respectively)[Considering target data in 14th,30th and 70th position respectively]	**No. of comparisons**	14 30 70	7 21 30	4 5 6
	Execution time (ms)	0.52747252 0.81318681 0.87912087	0.49120879 0.56043956 0.57142857	0.46901098 0.46702967 0.49450540
Worst case Analysis (Unsorted Array Size of 20,50 and 100 respectively)	**No of comparisons**	20 50 100	1 1 1	4 5 6
	Execution time (ms)	0.61538461 0.89010982 0.95054945	0.47252352 0.47252352 0.47252352	0.46813186 0.52197802 0.59890109
Can search target data in		Sorted and Unsorted array	Sorted and Unsorted array	Sorted array only

*Since only one comparison step is required to search the last element for both sorted and unsorted array.

Linear search performs in sequential order [5, 6].Thus, if the number of elements is very large then the execution time increases and to search the last element, n number of comparisons are made [2, 8]. As a result, its worst case time complexity is O(n). Same thing happens in case of Interpolation search [12].The Jump search also uses the linear search mechanism in its sub lists. Thus, if an element is present second to the last element, then the Jump Search makes an initial comparison of n. In this case, the Binary search is advantageous [4].It requires the random access of the data, having time complexity O(log n) [5].But, Binary search requires more space in stack due to its recursive nature[13].Linear search is better than binary search in case of small list of data[3,6].

The Bi-linear search beats Binary search with a time complexity of O(1),when the searched item is present in the last position of the array. Therefore, the Bi-linear search's execution time is very fast and is highly efficient for searching items from a large number of records.

VI. APPLICATIONS

Data are stored in various servers like mail servers, file servers, etc for security. To retrieve these data, Bi-linear Search can be used. It allows insertion, deletion and any type of modifications at the searched positions within a short time. It can be used to find out a bug present in a big program. Bi-linear Search can be used to search for a particular disk from the drives, even in case of tight memory space. It can find the last record by making only one comparison. It does not even require the ordered storage of records. The Bi-linear Search can be used to find someone's name in a telephone directory, to search a student's record from a Student's Database, to track any type of files from a directory and many more within a short span of time. Having a less execution time, the Bi-linear Search is highly efficient and faster in locating a data from a large collection of data

VII. CONCLUSION

Developments are going on in different genres of Software Engineering. In this paper, we have established a new searching technique (i.e. the Bi-linear Search) and made a brief comparison analysis with different searching algorithms. The paper has proved Bi-linear Search to be highly efficient for unsorted list of data. It takes comparatively less time to search an item from a large collection of records than a normal Linear Search. Even in a sorted list, it sometimes beats the Binary Search, especially when the element is present in the last position of the array. The Bi-linear Search has a worst case time complexity of O(1). A new establishment is always followed by a new invention. Ideas should be based on the needful demand. It is basically a step towards the increasing development of Technology. We ensure similar interesting types of paper in the near future.

REFERENCES

[1] Debasis Samanta,"Searching", Classic Data Structures.PHI Learning Private Limited, Second Edition, PP 714-738, ISBN: 978-81-203-3731-2

[2] S.K. Shrivastava, Deepali Shrivastava,"Searching, Hashing and Storage Management", "Data Structures Through C In Depth", BPB Publications, Eighth Edition, PP 424-428,ISBN:81-7656- 741-8

[3] Seymour Lipschutz,"Sorting and Searching", "Data Structures with C", Tata McGraw Hill Education Private Limited, Fourth Edition, PP 9.38-9.41, ISBN: 978-0-07- 070198-4

[4] Amitava Nag,Jyoti Prakash Singh,"Searching","Data Structures and Algorithms Using C", Vikas Publishing House Pvt. Ltd., First Edition, PP 271-281, ISBN: 978-81-259-3132-4

[5] Arpita Gopal,"Sorting Searching and Algorithm Complexity", "Magnifying Data Structures", PHI Learning Private Limited, First Edition, PP 423-430, ISBN: 978-81-203-4019-0

[6] Dr. Amiya Kumar Rath,Alok Kumar Jagadev, "Searching & Sorting", "Data Structures using C", Scitech Publications(India) Pvt. Ltd.,Second Edition, PP 11.1-11.10,ISBN:978-81-8371-232-3

[7] http://airccse.org/journal/ijcsea/papers/2212ijcsea03.pdf

[8] www.cirworld.com/index.php/IJCDS/**article**/download/11-IJCDS.../pdf

[9] www.princeton.edu/~achaney/tmve/wiki100k/docs/Interpolation_search.html

[10] http://www.stoimen.com/blog/2011/12/12/computer-algorithms-jump-search/

[11] kartikkukreja.wordpress.com/2013/08/17/beating-binary-search-the-interpolation-search/

[12] www.ics.uci.edu/~dan/class/161/notes/2/Interpolation.html

[13] http://scientia africana.uniportjournal.info/v9n2/pdfvol9no2/12.pdf

.