International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 07 Issue: 01 | Jan 2020 www.irjet.net p-ISSN: 2395-0072

A Survey on Different Searching Algorithms

Ahmad Shoaib Zia[1]

[1]*M.Tech scholar at the Department of Computer Science, Sharda University Greater Noida, India*

Abstract - This paper presents the review of certain important and well discussed traditional search algorithms with respect to their time complexity, space Complexity , with the help of their realize applications. This paper also highlights their working principles. There are different types of algorithms and techniques for performing different tasks and as well as same tasks, with each having its own advantages and disadvantages depending on the type of data structure. An analysis is being carried out on different searching techniques on parameters like space and time complexity. Dependent upon the analysis, a comparative study is being made so that the user can choose the type of technique used based on the requirement.

***Key Words*: Searching algorithms, binary search, linear search, hybrid search, interpolation search, and jump search.**

1. INTRODUCTION

A searching algorithm [1] [2] [3] is that type of algorithm that allows the efficient retrieval of a particular item from a set of many items. Searching is the algorithm process of finding a specific item in a collection of item. A search typically answers the user whether the item he searched for is present or not. Computer systems are often used to store large amounts of data from which individual records can be retrieved according to some search criterion so, it is our need to search and fetch the data in that manner so that it will take lesser time and will be efficient. [2] For this purpose some approaches are needed that not only saves our time but also fetches the required data efficiently. In this study we will discuss linear search, binary search, Interpolation search, hybrid search, algorithms on the basis of their efficiency and time complexity.

Searching falls into two categories:

a) **External searching**: External searching [3] means searching the records using keys where there are many records, which resides in the files stored on disks. This is the type of searching in which the data on which searching is done resides in the secondary memory storage like hard disk or any other external storage peripheral device.

b) **Internal searching:** [3] Internal searching is that type of searching technique in which there is fewer amounts of data which entirely resides within the computer's main memory. In this technique data resides within the main memory on.

2. Exiting Search Algorithms

2.1 Binary Search

It is a fast search algorithm [9] as the run-time complexity is O (log n). Using Divide and conquer Principle for it search algorithm. This algorithm performs better for sorted data collection. In binary search, we first compare the key with the item in the middle position of the data collection. If there is a match, we can return immediately. If the key is less than middle key, then the item must lie in the lower half of the data collection; if it is greater, then the item must lie in the upper half of the data collection [8].

Algorithm

1. Input an array A of n elements I sorted form.
2. LB=0,UB=n; mid=int((LB+UB))/2)
3. Repeat step 4 and 5 while(LB<=UB and (A[mid]!=item)
4. If (item<A[mid]) UB=mid-1
   Else
   LB=mid+1
5. mid=int((LB+UB)/2)
6. If (A[mid]==item)
   Print" Item is found"
   Else
   Print "Item is not found"
7. End.

Illustration

An array with seven elements, search for "5":

12	4	95	32	7	24	5
12	4	95	32	7	24	5
12	4	95	32	7	24	5
12	4	95	32	7	24	5
12	4	95	32	7	24	5
12	4	95	32	7	24	5
12	4	95	32	7	24	5

2.2 Linear Search

Linear search is a simple search algorithm [8]. It is a sequential search which performed on sequences of numbers that are ascending or descending or unordered. And it checks each and every element of the entire list to search a particular data from the list. If the comparison is equal, then the search is stopped and declared successful. For a list with n items, the best case is when the value of item to be searched is equal to the first element of the list, in this case only one comparison is needed. Worst case is when the value is not in the list or occurs only once at the end of the list, in this case n comparisons are needed [9].

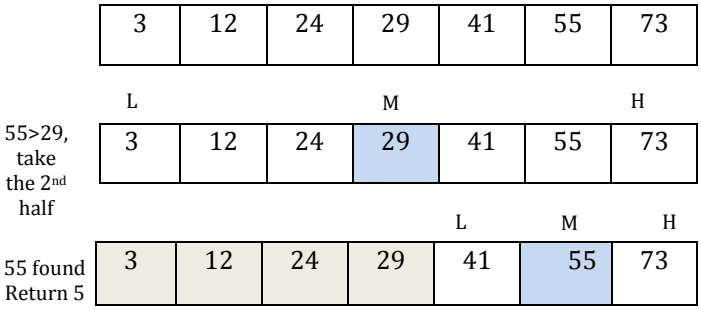
Algorithm

Here A is a linear array with N elements, and ITEM is a given item of information. This algorithm finds the location LOC of ITEM in A [3].

1. Set ctr=L
2. Repeat steps 3 through 4 until ctr>Upper bound.
3. If A[ctr]==ITEM then { print "Search successful"
   Print ctr, "is the location of", ITEM
   Go out of loop }
4. 4.ctr=ctr+1
5. If ctr>Upper bound then
   Print "Search unsuccessful"
6. End.

Illustration

Searching for 55 in 7-element array:



2.3 Hybrid Search

Hybrid Search algorithm [6] combines properties of both linear search and binary search and provides a better and efficient algorithm. This algorithm can be used to search in an unsorted array while taking less time as compared to the linear search algorithm. As mentioned this algorithm is combines two searching algorithms, viz. Linear Search and Binary Search. As with Hybrid Search algorithm, the array is divided into two sections and then searched in each of the sections. The algorithm starts with comparing the key element to be searched with the two extreme elements of the array, the first and the last, as well as the middle element. If a match is found, the index value is returned. However, if it is not, the array is divided into two sections, from the middle index. Now the search is carried out in the section on the left in a similar way. The extreme elements and the middle element of the left division are compared with the key value for a match, which if found, returns the index value. If not, the left section is again divided into two parts and this process goes on till a match is found in the left section [5]. If no match is found in the left division, then the algorithm moves on to the right division, and the same procedure is carried out to find a match for the key value. Now, if no value is found that matches the key value even after searching through all sections, then it is further divided and the process repeats iteratively until it reaches the atomic state. If the value is not present in the array, as a result of which the algorithm returns -1.

Algorithm

1. mid = (low + high)/2
2. if a[low] = key then return low
3. else if a[high] = key then return high
4. else if a[mid] = key then return mid
5. else if low >= high – 2 then return -1
6. else
7. p = recLinearBinary(a, low + 1, mid – 1, key)
8. if p = -1
9. p = recLinearBinary(mid + 1, high – 1, key)
10. return p

International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 07 Issue: 01 | Jan 2020 www.irjet.net p-ISSN: 2395-0072

Illustration

Searching for 2 in 8-elemetnts array:

14	5	71	37	56	2	98	11

14	5	71	37	56	2	98	11

14	5	71	37	56	2	98	11

14	5	71	37	56	2	98	11

2.4 Interpolation Search

Interpolation search algorithm [7] is improvement over Binary search. The binary search checks the element at middle index. But interpolation search may search at different locations based on value of the search key. The elements must be in sorted order in order to implement interpolation search. As mentioned the Interpolation Search is an improvement over Binary Search for instances, where the values in a sorted array are uniformly distributed. [3] Binary Search always goes to the middle element to check. On the other hand, interpolation search may go to different locations according to the value of the key being searched. For example, if the value of the key is closer to the last element, interpolation search is likely to start search toward the end side.

Algorithm

1. Initialize the values of start to 0 and end to n-1.
2. Calculate the value of
   $$x = start + \frac{(end - start)}{(K[end] - K[start])} * (p - K(start))$$
   where K is an array and p is the search key.
3. If K[x]==p, then stop and return.
4. If K[x]!=p, then
   If p>K[x] then make start=x+1
   If p<K[x] then make end=x-1.
5. Repeat step 2 till the search element is found.
6. End

Illustration

Searching for 4 in 9-elments array:

Start End

1	2	4	7	9	12	13	14	17
Array [0]	Array [1]	Array [2]	Array [3]	Array [4]	Array [5]	Array [6]	Array [7]	Array [8]

1	2	4	7	9	12	13	14	17
Array [0]	Array [1]	Array [2]	Array [3]	Array [4]	Array [5]	Array [6]	Array [7]	Array [8]

1	2	4	7	9	12	13	14	17
Array [0]	Array [1]	Array [2]	Array [3]	Array [4]	Array [5]	Array [6]	Array [7]	Array [8]

2.5 Jump Search

Jump search algorithm, [4] also called as block search algorithm. Only sorted list of array or table can use the Jump search algorithm. In jump search algorithm, it is not at all necessary to scan every element in the list as we do in linear search algorithm. We just check the m element and if it is less than the key element, then we move to the m + m element, where all the elements between m element and m + m element are skipped. [7] This process is continued until m element becomes equal to or greater than key element called boundary value. The value of m is given by m = √n, where n is the total number of elements in an array. Once the m elements attain the boundary value, a linear search is done to find the key value and its position in the array. And the numbers of comparisons are equal to (n/m + m -1). [3] It must be noted that in Jump search algorithm, a linear search is done in reverse manner that is from boundary value to previous value of m.

Algorithm

1. Set i=0 and m= √n
2. Compare A[i] with item. If A[i] != item and A[i] < item, then jump to the next block. Also, do the following:
   1. Set i = m
   2. Increment m by √n
3. Repeat the step 2 till m < n-1
4. If A[i] > item, then move to the beginning of the current block and perform a linear search.
   1. Set x = i

2. Compare A[x] with item. If A[x]== item, then print x as the valid location else set x++
3. Repeat Step 4.1 and 4.2 till x < m

5. End

Illustration

Searching for 24 in 9-elments array:

n = 9
m = √9 = 3

2	5	7	11	24	30	45	78	99

7 < 24

2	5	7	11	24	30	45	78	99

30 > 24

2	5	7	11	24	30	45	78	99

Do linear search backward

2	5	7	11	24	30	45	78	99

3. Comparison table of search algorithms

Algorithms	Binary Search	Linear Search	Hybrid Search	Interpolation Search	Jump Search
Features	As well-known as half interval search or logarithmic search which follows divides and reduces method.	Sequentially Checks the target element in the list until and unless it is found or all the element are checked	Combines the advantages of Binary and Linear algorithms and provides an effective way to search for a given key element in an unsorted array, in limited time.	Improved variant of binary search. Works on the probing position of required value to search a particular data from a list. Works only on sorted elements.	Also known as block search, where step number is calculated from the list length and searching is done in some interval of blocks.
Time Complexity Analysis	Best case =O(1) Average Case =O(log n) Worst case =O(log n)	Best case =O(1) Average case =O(n) Worst Case =O(n)	Best case =O(1) Average case =O($\log_2$n) Worst Case =O(n)	Best case =O(1) Average case =O(log(log N)) Worst case =O(n)	
Advantage	Average case and worst case order are better than that of linear search. Worst case order is also better than interpolation search.	Simple to understand, Works on both sorted and unsorted elements. Easy to implement.	Takes lesser time compared to linear search and array need not be sorted.	Execution time for average case is much lower than linear and binary search.	It is very useful when jumping back is significantly lower than jumping forward. More efficient than linear search.
Disadvantage	Works only on sorted elements.	Is not very efficient than binary and interpolation search as it requires lot of comparisons to find a particular data.	The worst case is n iteration.	Works only on sorted elements and worst case is n iterations	Works only on sorted elements. Implementation of this approach is considered to be more difficult than binary search.

4. Motivation

During the research I carried out upon various searching algorithms and throughout reading various papers from different publishers. I was motivated to perform more and more research in this field which caused me to come up and publish a survey paper to find difference between different types of search algorithms and there best way to be suitable for data set.

5. Conclusion

The paper discusses about various searching techniques. It shows the methodology for various searching techniques. Searching is one of the important operation of data structure. Different searching algorithms enable us to look for a particular data from the entire list. The analysis shows the advantages and disadvantages of various searching algorithms along with examples. We analyzed based on time complexity and space complexity. On analysis, we found that binary search is suitable for mid-sized data items and is applicable in arrays and in linked list, whereas jump search is best for large data items. Also we found that Hybrid search used for unsorted list with more elements.

ACKNOWLEDGEMENT

First of all, I would like to thank the most merciful and the most gracious Allah who teaches and shows us the way of happiness of this and next word by the agency of his messenger the owner of honor Mohammad (peace be open him) and give command for learning the knowledge. I, Ahmad Shoaib Zia, would like to express my sincere appreciation to all those who provided me the possibility to complete this paper. I give a gratitude to my guide Prof. (Dr.) Nitin Rakesh and a special thanks from Dr. Mandeep Kaur whose contribution in stimulating suggestions and encouragement helped me to write this paper at the end a special thanks from my classmate Zubair Salarzai, he help me to write this paper.

References

[1] Survey and Analysis of Searching Algorithms, Tahira Mahboob, Fatima Akhtar, Moquaddus Asif, Nitasha Siddique, Bushra Sikandar, International Journal of Computer Science Issues (IJCSI), 2015

[2] Comparative Analysis on Sorting and Searching Algorithms, B Subbarayudu, L Lalitha Gayatri, P Sai Nidhi, P. Ramesh, R Gangadhar Reddy, Kishor Kumar Reddy C, International Journal of Civil Engineering and Technology (IJCIET), 2017

[3] A Comparative Study of Sorting and Searching Algorithms, Ms Roopa K, Ms Reshma J. International Research Journal of Engineering and Technology (IRJET), 2018

[4] Jump Searching: A Fast Sequential Search Technique, S.L. Graham, R.L. Rivest, Ben Shneiderman University of Maryland, 1978

[5] Analysis And Comparative Study Of Searching Techniques, Ayush Pathak, International Journal Of Engineering Sciences and Research Technology (IJESRT), 2015

[6] Hybrid Search Algorithm, Asha Elza Jacob, Nikhil Ashodariya, Aakash Dhongade, International Conference on Energy, Communication, Data Analytics and Soft Computing (ICECDS), 2017

[7] A Brief Study and Analysis Of Different Searching Algorithms, Najma Sultana, Smita Paira, Sourabh Chandra. Sk Safikul Alam, IEEE, 2017

[8] A Randomized Searching Algorithm and its Performance analysis with Binary Search and Linear Search Algorithms, Pranesh Das, Prof. Pabitra Mohan Khilar, The International Journal of Computer Science and Applications (TIJCSA), 2012

[9] Comparison Searching Process of Linear, Binary and Interpolation Algorithm, Robbi Rahim, Saiful Nurarif, Mukhlis Ramadhan, Siti Aisyah, Windania Purba, International Conference on Information and Communication Technology (IconICT) , 2017