Performance Analysis of Search Algorithms on Workstation system

Mubashir Ali^{1,*}, Muhammad Affan Abbas², Maaz Asghar³

^{1,2}Department of Computer Science, Bahauddin Zakariya University, Multan, Pakistan.

³FSKTM Universiti Tun Hussein Onn Malaysia

¹dr.mubashirali1@gmail.com

ABSTRACT

Searching is a common issue in computer science. It is defined as a process in which elements are to be found from a given list. Search algorithms used in daily life for finding values in array or list, data retrieving, finding passwords and combinations, etc. In the past, many search algorithms have been introduced and existing ones have been improved keeping a view of performance in terms of time and space complexity. Search algorithms are defined based on their framework. Well-known types of search algorithms are binary, linear, jump, and interpolation search. Binary search (interval, logarithmic search) is defined as a search algorithm, which finds the position of an element or target value within a sorted array; linear search (sequential search) is a search over the list of items in a sequential way (step-by-step process). Jump search focuses on fewer elements by jumping (skipping) to the next elements in fixed steps while Interpolation search is an improvement over binary search may go to different locations according to the value of the key being searched. In this paper, we have performed a comparative analysis of these search algorithms on the workstation system. Lenovo S-20 workstation with windows as an operating system is used for experimentation and analysis. Finding out which search algorithm is best in an associated scenario with a comparison of a single-core processor to a workstation of a six-core processor.

Keywords: Search Algorithms, Performance Analysis, Comparative Analysis, Sequential Search, Linear Search, Binary Search, Jump Search, Interpolation Search, Workstation System (Lenovo S20)

1. INTRODUCTION

A search algorithm is defined as a specific set of instructions given to find an item from a list or group of items [1]. Search algorithms are one of the most important parts of programming. In the

age of technology, these serve many purposes like finding value in array or list, data retrieving, finding password and combinations, etc. A good example of an application of a search algorithm is the browser engine which let us

search all type of things i.e. google. Another example of a search algorithm is to find personal details of a person from a record of millions of people i.e. a police station database. An interpolation search algorithm is an advanced form of the binary search algorithm. The efficiency of the searching algorithm lies in its time complexity or the number of comparisons made [2]. Searching is the most fundamental operation in a data structure which is used to find specific data from a list of items (array). Searching is operable on both sorted and unsorted type of data structure. The search algorithm is chosen by the data type provided [3]. Many types of Search algorithms are used to search for a different or similar type of data. Some examples of traditional search algorithms are linear search, binary search, jump search, and interpolation search.

Jump Search is a searching algorithm meant for sorted arrays that skips some elements based on pre-requisites and then jump on to the next ones [4]. The basic theme is to skip elements that are not meant to be searched and to jump ahead to check if the element is present there. The time complexity of the jump search lies between linear search $O(n)$ and binary search $O(\log n)$. Jump search focuses on fewer elements by jumping (skipping) to the next elements and thus by doing this jump case avoids a sequential flow and attains a much more efficient way of searching an item from a given list. It is slower than binary search but can prove efficient in some cases. Almost for every scenario, a jump search algorithm is a convenient way to perform the task of search.

The binary search performed well in terms of complexity by $O(\log(n))$ but in the case where, there is a very large group of items, it is a bad idea to jump directly at the interval of the list, as the element to be found can be located near the first elements or the last ones. For this purpose, we should make a large step back if the searched value is located at either end of the list. In a comparison of jumping, back Jump search is very useful and is significant while in terms of jumping forward it is significantly slower.

Interpolation search is an improved version of binary search [5]. This algorithm was modified into a search algorithm that works on the defined position for the list of items [6]. The original idea to develop an Interpolation search was for uniformly distributed data that points to a linear relationship between values of sorted data as well as the positions of the elements within the dataset. Thus, the only drawback for the interpolation search algorithm is that the data should present in a uniform sorted form and equally divided to work on. The time complexity for interpolation search depends upon if the array is sorted and uniformly distributed or not. Results depend on the cases, in the best case, it's $O(\log(n))$ and in the worst case, it can be $O(n)$. Interpolation search algorithm proves to be costly and time taking in worst-case scenario. Interpolation search has been found to underperform other searching techniques, which are in practice due to the reason that it uses computationally expensive calculations.

Binary and linear search are well-known types of search algorithms. The

past work and research papers have signified that binary search has always proved to be a more efficient way of searching in terms of cost and time complexity [7]. A better example of linear searching is that suppose your driver parked your car at the parking lot. The only thing you know is the face of the driver and license plate of the car so how are you going to find your car. The obvious thing would be to just go to the parking lot where the car is parked, and look for the car one by one until you find your car. This technique of searching elements from a list of elements is called linear search, where we check all elements and do a comparison with the desired value (brute force search). Other examples include the searching of books, medicines, movies, grocery items in stores, and the market. The algorithm of binary search is comparatively more efficient as it deletes unnecessary comparison at initial stages. The binary search focuses on a divide and conquers approach with time complexity of $O(\log(n))$ while linear search focuses on a sequential flow with time complexity of $O(n)$. For example, in a list of hundred items to be searched, binary search will divide a hundred items into fifty using the given requirement, and then in the next step, the further fifty elements would be narrowed down to half of them and so on. In linear search a sequential flow, the searching would start from the first element and would end upon the last hundredth item. Thus, this concludes how efficient binary search is but there are other search algorithms (jump search and interpolation search algorithm) that are more efficient in different scenarios. Binary search technique has now become a major part of daily life computer science

application and is now used in 99% of 3D games and applications. One of its first greatest showcases was Doom.

Figure 1: Search Process

This research work is divided into six sections. The literature section discusses the previous work with theories and logic on which searching would base, existing knowledge like substantive findings, as well as methodological and theoretical contributions. Section 3 precisely elaborates on the underlying searching algorithms along with their complexity. Section 4 shows the methodology and experimental criteria. Section 5 shows the experimental results and discussion by comparative analysis. Finally, section 6 concludes the work.

2. LITERATURE REVIEW

The previous research suggests that data searching is a practical field and a major sub-problem in computer science applications. Chang et. al [7] Provided basic information about different types of search algorithms of data structure like linear, binary, and hash search. Technical aspects of these searching algorithms have been covered with detailed analysis and their working

process. They have concluded that which type of search algorithm is best for an associated scenario. The result deduced from analysis and experimental work is that hashing is faster for large lists in comparison with binary search. Binary search is efficient concerning mid-size lists or small lists in comparison with linear search [7].

Kumari et. al focus on the analysis and comparison between existing searching algorithms like linear and binary search. The comparison is made based on time complexity for a given amount of data. This paper concludes the results for different searching techniques by the use of time complexity. The result deduced from the experimental work and analysis is that a linear search is suitable for a small amount of data. While binary is efficient for large amounts of data that is in the sorted form [8].

Subbarayudu et al analyze search algorithms based on performance factors i.e. computational, complexity, memory, and interrelated factors. This paper analyzed the running time of binary and linear search on different processors with some java concurrency tools. The result shows binary search is 1.98 times faster than a linear search on a dual-core processor and a quad-core processor. The running time is 3.83 times faster in comparison to a linear search. Thus results conclude that binary search is a better searching algorithm than a linear search as on dual-core processors [9].

Jacob et al analyze three different types of searching techniques (binary, linear, and interpolation search) to find elements from a list of data by the

process of checking. A comparison between binary, linear, or sequential and interpolation search has been done with the experimental results. The importance and application for searching algorithms and techniques have been discussed and summarized for the database management system. The conclusion states the advantages of binary search in comparison to other searching techniques, [10].

Dubey et al analyze the performance of different searching algorithms such as linear, brute force, and binary search algorithms. Performance analysis of searching algorithms is done on different machines and execution time is measured with the help of time complexity. The result deduced from the experiments show that binary search is better in time complexity and brute force search is best in finding searching patterns [11].

Parmar et al proposed a new searching algorithm with hash function, which is applicable for both sorted and unsorted array. This searching algorithm is different from traditional search algorithms (binary and linear search algorithm). This research concludes that hash functions complexity is very less as compared to Binary and Linear searching techniques thus it performs better [12].

Sultana et. al analyze the problem of growing data and how to manage it by studying different searching algorithms and techniques. Several algorithms have been developed for searching an element from a list of items but choosing the right technique and right algorithm for an associated scenario is challenging. This research analyzes

different searching techniques by making a comparison between them. They also studied the applicability and execution efficiency of algorithms. The results deduced from the paper concludes that binary search is better than linear search. But in the case of insertion, it requires elements to be in an arranged manner. Apart from binary and linear search, this paper provides analysis on hashing technique which is an alternate solution for direct searching [13].

Pathak et al highlight the working principles of traditional techniques for both sorted and unsorted list of elements. Some well-known proposed searching algorithms like binary, linear search, vector approach have been discussed and a comparison between them has been drawn in the form of a table [14].

Nishihara et al explained the use and importance of jump search an alternative searching technique to binary and linear search. The analysis of the jump search is done by exploring multiple levels of jump strategies and are discussed by evaluating the performance. This paper depicts the usefulness of jump search when the binary search is unfeasible. Further, the types of jump searches are discussed and analyzed [5].

3. SEARCH ALGORITHMS

Linear search or sequential search is a technique that checks for a specific value, one at a time in a list of elements to be searched until the desired value is found. It is the simplest searching technique. The time complexity for linear search is $O(n)$. The best case for

the n items would be when on the first iteration the element to be found. The worst-case would be when the element to be found does not exist in the list or is present at the last point. This means for n iteration the loop will start from the starting point and will end at the last point checking the entire collection.

Binary search is also known as half interval search or logarithmic search. It searches for the desired value by pinning down its location, it's based on the divide and conquer approach [15]. It compares the target value with the desired value by comparing it with the middle element of the array. If found unequal; then both halves are compared and then the half in which the element cannot lie is eliminated and then the other half is further divided and the same technique is applied until the element to be found. The time complexity of Binary Search is $\log(n)$. The best case for binary search would be when on the first attempt the element to be found is located on mid-value. The worst-case would be when the element to be searched is not located in the array or is found at the last iteration. Binary search requires elements to be searched in a sorted manner [16].

Jump Search is a searching algorithm meant for sorted arrays that skips some element based on pre-requisites and then jumps on to the next ones. The basic theme is to skip irrelevant elements and to jump ahead to check if the element is present there. The time complexity of the jump search lies between linear Search $O(n)$ and binary search $O(\log n)$. Jump search focuses on fewer elements by jumping (skipping) to the next elements and thus by doing this jump case avoids a sequential flow

and attains a much more efficient way of searching an item from a given list. It is slower than binary search but can prove efficient in some cases. The best case for the jump search would be to find the desired value on the first attempt or iteration in n items to be searched. The worst-case would be when the element to be found is not in the list or is skipped repeatedly due to particular conditions set in the algorithm.

Interpolation search is an improved version of binary search. This algorithm was modified into a search algorithm that works on the defined position for the list of items. The original idea to develop an interpolation search was for uniformly distributed data that points to a linear relationship between values of sorted data as well as the positions of the elements within the dataset. Thus, the only drawback for the interpolation search algorithm is that, for it to work, the data should present in a uniform sorted form and equally divided. The time complexity for interpolation search depends if the array is sorted and uniformly distributed or not. Results depend on the cases in the best case its $O(\log(n))$ and in the worst case. It can be $O(n)$.

4. METHODOLOGY

An experimental analysis is conducted to check the performance of sorting algorithms in different scenarios. Lenovo S20 workstation system with windows operating systems was used experimentation.

Lenovo s20 was designed for professionals, users of mid-range and finance, and medical specialists. The core focus of this workstation system

was to provide performance, durability, and reliability. A standard PC has enough power to do simple tasks such as web surfing, e-mail, and word processing, etc. Comparatively a workstation system offers more power and performance. It can perform multiple tasks at the same time handling CAD, data analysis, photorealistic renderings (video and audio edit), and animations. Workstation PC is equipped with ECC memory capability which decreases Blue screen issues as well as memory resolve issues [17]. Workstation systems are designed for heavy work and multi-tasking. They are also designed to match your application requirements like complex graphics are rendered fast, more processing power for intensive apps, greater and fast memory for large tasks, hard drives of massive size for large storage, reliability, and less downtime. The use of this workstation system will provide the programmers with a significant advantage by reducing the time of a given algorithm. Xeon x5670 is a multi-core processor with 6 physical and 6 logical cores and 12 megabytes cache memory. The previous research paper concludes that search algorithms work faster on multi-core systems than on single-core systems.

Choosing an operating system for the workstation system is an essential part. An operating system is a set of instructions that provides knowledge of common services for computer programs as well as manages hardware resources for computers. Dev C++ is the compiler used to compile codes and measure time based on C++ implementation for involved searching algorithms.

5. PERFORMANCE ANALYSIS OF SEARCH ALGORITHM

Integer type data is used for the search process. The data range is 300 integer values to 2100 integer values. The distribution of data is random in the array. All search algorithms are executed under the same data set with the same queries. Running time is measured in milliseconds. The data range is an increase from 300 to 2100 with a static ratio.

Figure 3: Linear Search Profile

Figure 3 shows the execution profile of a linear search. The data range is drawn on the x-axis and running time is drawn on the y-axis. Running time is measured in a millisecond that the search algorithm takes to return the required result. Graphs show the linear curve of execution time in case of linear search. It takes near 0.01 milliseconds for search in 300-integer size array and almost 0.06 seconds for 2100 integer size array.

Figure 4: Binary Search Profile

Figure 4 illustrates the running time profile of binary search against a given data set and search query. The data set is populated on the x-axis and execution time is drawn on the y-axis. Execution time is measured in a millisecond that the binary search algorithm takes to return the required result. The binary search takes less time to search a query against a linear search.

Figure 5: Jump Search Profile

Figure 5 shows the execution profile of the jump search. The data range is drawn on the x-axis and running time is drawn on the y-axis. Running time is measured in milliseconds that the search algorithm takes to return the required result. It takes near 0.01 milliseconds for search in 300-integer size array and almost 0.045 seconds for 2100 integer size array.

Figure 6: Interpolation Search Profile

Figure 6 illustrates the running time

profile of the interpolation search against the given data set and search query. The data set is populated on the x-axis and execution time is drawn on the y-axis. Execution time is measured in a millisecond that the interpolation search algorithm takes to return the required result.

Figure 7: Performance Analysis with Comparative Profile

Figure 7 shows the comparative profile of linear, binary, jump, and interpolation search. The data range is drawn on the x-axis and execution time is taken on the y-axis. Searching profiles of different search algorithms are differentiated with colors. The comparison depicts that Binary search is best among all search algorithms. It also shows that binary search is efficient for small as well as large sizes of elements in an array to be searched. Thus, Binary search provides maximum efficiency with the minimum runtime execution of 0.33 milliseconds for an array of size 2100 as compared to all other searching algorithms. Between linear and jump search algorithms,

Jump search provides better results with 0.45 milliseconds for an array of size 2100. Jump search and interpolation search perform almost the same. Comparatively, the linear search takes extra time for searching.

6. CONCLUSION

Searching is one of the common problems that arise in computer science. Searching is used in various applications as a sub problem. Multiple sorting algorithms are devised in literature. A better sorting algorithm improves the overall performance of an application. This paper analyzes the performance of four different searching techniques on a workstation system. Experimental analysis is conducted to analyze the comparative performance of linear, binary, jump, and interpolation search. Binary search shows maximum efficiency with an execution time of 0.033 ms in an array of size 2100 while linear search shows maximum runtime execution of 0.056 ms in an array of size 2100. Interpolation search and jump search shows average results. Thus, it is concluded that Binary search is the best searching technique for integer type random drawn data.

REFERENCES

- [1] M. T. McClellan, J. Minker, and D. E. Knuth, "The Art of Computer Programming, Vol. 3: Sorting and Searching," *Mathematics of Computation*, vol. 28, no. 128. pp. 1175, 1974.
- [2] M. Ali, Z. Nazim, W. Ali, A. Hussain, N. Kanwal, and M. K. Paracha, "Experimental Analysis of On

($\log n$) Class Parallel Sorting Algorithms," Int. J. Comput. Sci. Netw. Secur., vol. 20, no. 1, pp. 139-148, 2020.

[3] J. Selander, S. U. Marnetoft, B. Åkerström, and R. Asplund, Locus of control and regional differences in sickness absence in Sweden, vol. 27, no. 16. 2005.

[4] B. Shneiderman, "Jump Searching: A Fast Sequential Search Technique," Commun. ACM, vol. 21, no. 10, pp. 831-834, 1978.

[5] S. Nishihara and H. Nishino, "Binary Search Revisited: Another Advantage of Fibonacci Search," IEEE Trans. Comput., vol. C-36, no. 9, pp. 1132-1135, 1987.

[6] A. R. Chadha, R. Misal, and T. Mokashi, "Foundation of Computer Science," 2014.

[7] S. S. Iyengar and H. Chang, "Efficient algorithms to create and maintain balanced and threaded binary search trees," Softw. Pract. Exp., vol. 15, no. 10, pp. 925-941, 1985.

[8] A. Kumari, "Linear Search Versus Binary Search: A Statistical Comparison For Binomial Inputs," Int. J. Comput. Sci. Eng. Appl., vol. 2, no. 2, pp. 29-39, 2012.

[9] B. Subbarayudu, L. Lalitha Gayatri, P. Sai Nidhi, P. Ramesh, R. Gangadhar Reddy, and C. Kishor Kumar Reddy, "Comparative analysis on sorting and searching algorithms," Int. J. Civ. Eng. Technol., vol. 8, no. 8, pp. 955-978, 2017.

[10] A. E. Jacob, N. Ashodariya, and

A. Dhongade, "Hybrid search algorithm: Combined linear and binary search algorithm," in 2017 International Conference on Energy, Communication, Data Analytics and Soft Computing, ICECDS 2017, 2018, pp. 1543-1547.

[11] S. Dubey, "Comparative Performance Analysis of Binary Search in Sequential and Parallel Processing," Int. J. Res. Appl. Sci. Eng. Technol., vol. V, no. X, pp. 1339-1347, Oct. 2017.

[12] V. P. Parmar and C. Kumbharana, "Comparing Linear Search and Binary Search Algorithms to Search an Element from a Linear List Implemented through Static Array, Dynamic Array and Linked List," Int. J. Comput. Appl., vol. 121, no. 3, pp. 13-17, Jul. 2015.

[13] N. Sultana, S. Paira, S. Chandra, and S. K. S. Alam, "A brief study and analysis of different searching algorithms," Proc. 2017 2nd IEEE Int. Conf. Electr. Comput. Commun. Technol. ICECCT 2017, no. February, 2017.

[14] A. Pathak, "Analysis and comparative study of searching & researching techniques," Int. J. Eng. Sci. Res. Technol., vol. 4, no. 3, pp. 235-237, 2015.

[15] A. Immirzi, L. Erra, and C. Tedesco, "Finding crystal structures from few diffraction data by a combination of a random search with genetic algorithms," J. Appl. Crystallogr., vol. 41, no. 4, pp. 784-790, 2008.

[16] S. Dubey, "Comparative Performance Analysis of Binary Search

in Sequential and Parallel Processing,"
Int. J. Res. Appl. Sci. Eng. Technol., vol.
V, no. X, pp. 1339-1347, 2017.