A Brief Study and Analysis of Sequential and Interval Searching Algorithms

Arcena, Jodie Lance A. — BSIT — Batangas State University

Aclo, Maricar B. — BSIT — Batangas State University

Bation, John Michael L. — BSIT — Batangas State University

ABSTRACT

The two strategies for looking for particular data in a collection are sequential search and interval search. Sequential search is a simple and straightforward method that checks each element in a collection one at a time until it discovers the target value or reaches the collection's end. It has a worst-case time complexity of O(n) and is memory-efficient, although it may not be the most efficient technique for searching for patterns or approximate matches in data. Interval search, on the other hand, is a specialized algorithm that searches for overlapping intervals in a set of data. These methods have a temporal complexity of O(log n + k) and may be used on any form of interval data. Interval search algorithms are fast and scalable, and they may be used for a wide range of tasks such as scheduling, resource allocation, and database searches. Overall, the decision between sequential and interval search methods is influenced by the unique requirements and features of the data being searched. Sequential search is appropriate for small collections or unsorted data, but interval search methods are required for complicated searches in bigger or interval-based datasets. Researchers and data analysts may pick the optimal technique for their unique needs by studying the differences and benefits of each algorithm.

Keywords: Interval search, Interval data, Sequential search, Search, Data, Methods, Time, Technique, Collection

I. INTRODUCTION

[1] A searching algorithm is a computer technique meant to discover a certain value or group of values in a set of data. It's a fundamental notion in computer science that's employed in a variety of applications including databases, search engines, and artificial intelligence. When the amount of data is too enormous to search manually, search algorithms are utilized. These techniques can be used to find specific values in an array, linked list, tree structure, or other data structure. Depending on the type of data and the specific requirements of the search, numerous searching algorithms can be utilized. The time complexity of a searching algorithm is measured by the amount of time it takes to finish the search. The smaller the temporal complexity, the quicker the algorithm. Overall, searching algorithms are important tools for tackling a wide range of issues in computer science, and they are widely employed in many disciplines.

In this case study, we will look at and analyze Sequential Search and Interval Search. We shall investigate their parallels and differences, benefits and drawbacks, and application scenarios. We'll also look at their time and space complexity for various operations including searching, insertion, and deletion. I intend to give a thorough understanding of these two data structures so that readers may use this information to select the best data structure for their applications.

II. SEQUENTIAL SEARCH

[2] Sequential search, also known as linear search, is a basic searching technique that evaluates each member of a collection consecutively until a match is discovered or the collection is exhausted. It operates by comparing each collection element to the search key until a match is discovered. The sequential search method begins at the beginning of the collection and iteratively checks each piece. [3] If the search key is discovered, the process is terminated, and the index of the matching element is returned. If the algorithm reaches the end of the collection without finding a match, it

produces a special number, such as -1, to indicate that the search key was not found. Any sort of collection, including arrays, linked lists, and other data structures, can benefit from sequential search. It is particularly beneficial for tiny collections or unsorted data since it does not require the data to be sorted first. Sequential search, on the other hand, has a temporal complexity of O(n), where n is the number of items in the collection. This means that the search time grows linearly with collection size, making it less efficient than alternative searching methods like binary search or hash tables.

III. INTERVAL SEARCH

[3] Interval search is a search technique that determines whether or not a given value occurs inside a set of intervals. It is sometimes referred to as an interval query, an interval tree, or a segment tree. Interval search is widely used in computer science and data analysis, particularly in applications involving scheduling, resource allocation, and time series analysis. Interval search, for example, can be used in scheduling applications to assess resource availability during certain time periods. The search method walks the tree and compares the target value with the ranges of each node to find a value inside the intervals. If the target value is inside a node's range, the algorithm descends farther into the tree to look for it among the child nodes. If the target value is not inside a node's range, the algorithm searches in the relevant child node. A collection of intervals is grouped into a data structure such as a binary search tree or a segment tree in interval search. Each node in the data structure represents an interval range, and the tree is built so that each node's range overlaps the ranges of its sibling nodes. For average and worst-case circumstances, interval search is a moderately efficient searching algorithm with a time complexity of O(log n). The development of the interval search data structure, on the other hand, might be computationally costly, particularly for large collections of intervals.

IV. COMPARISON OF BOTH SEARCH ALGORITHMS

[9] This section compares and contrasts two searching algorithms: Sequential Search and Interval Search. The search algorithms are compared based on their characteristics, time complexity, space complexity, benefits, and downsides.

Table 1.1 Comparison Table of Sequential Search and Interval Search in terms of features

Searching Algorithm	**Features**
Sequential Search	Sequential search is a basic algorithm that is straightforward to implement, making it an excellent choice for novices or situations where simplicity is desired. It is most effective when dealing with limited amounts of data but can be sluggish and inefficient for bigger collections. It works well with unsorted data, but the temporal complexity of sequential search is O(n), which increases linearly with data size.
Interval Search	[6] Interval search is a simple data structure that can be dynamically changed, allowing intervals to be added or deleted without rebuilding the entire data structure. It can be used to a variety of data formats, such as numeric, dates, and texts, and has

	several applications, such as scheduling, resource allocation, time series analysis, and database searches.

Table 1.2 Comparison Table of Sequential Search and Interval Search in terms of Time Complexity

Searching Algorithm	Time Complexity
Sequential Search	[3][8]Sequential search, also known as linear search, has a temporal complexity of O(n), where n is the number of items in the collection being searched. [4] Sequential search may require verifying every element in the collection before finding a match or reaching the end of the collection in the worst-case scenario. This can occur when the search key is near the end of the collection or when the search key is not there at all. As a result, the time required for sequential search rises linearly with collection size. For example, if you have a collection of 1000 pieces and use sequential search to discover a certain value, it may take up to 1000 comparisons in the worst-case scenario to find the match or conclude that the value is not present in the collection. It is crucial to note that the layout or order of the components in the collection has no effect on the temporal complexity of sequential search. [4] Regardless of whether the items are sorted or unsorted, sequential search must examine each element one by one until a match is discovered or the collection is exhausted, resulting in a temporal complexity of O(n).
Interval Search	[11]The temporal complexity of an interval search algorithm is determined by the implementation and data structures employed. [7] An interval tree, on the other hand, can give an efficient interval search algorithm with a time complexity of O(log n + k), where n is the number of intervals in the tree and k is the number of intervals in the result set. Because an interval tree is a balanced binary search tree, each node represents an interval and has information about the maximum endpoint of all intervals in its subtree, this time complexity is attained. When looking for intervals that overlap with a particular interval, the method can rapidly predict which nodes in the tree may contain overlapping intervals, allowing it to trim the search space and minimize time complexity.

Table 1.3 Comparison Table of Sequential Search and Interval Search in terms of Space Complexity.

Searching Algorithm	**Space Complexity**
Sequential Search	[5] A sequential search algorithm's space complexity is determined by the data structure used to store the search space. The space complexity of sequential search is O(n) if the search space is stored in an array, where n is the number of entries in the array. This is due to the fact that the complete array must be held in memory in order to execute the search. [7] The space complexity of sequential search is also O(n) if the search space is stored in a linked list, where n is the number of entries in the linked list. This is due to the fact that each node in the linked list includes data as well as a reference to the next node, and the full linked list must be held in memory in order to execute the search.
Interval Search	[10] The space complexity of an interval search algorithm is also affected by its implementation and data structures. The space complexity of an interval tree is O(n), where n is the number of intervals in the tree. This is due to the fact that each node in the tree represents an interval and maintains data such as its endpoints and the maximum endpoint of all intervals in its subtree. Other data structures, such as segment trees or range trees, can be utilized to build interval search algorithms in addition to the interval tree. Depending on the implementation specifics, the space complexity of various data structures might vary.

Table 1.4 Comparison Table of Sequential Search and Interval Search in terms of Advantages.

Searching Algorithms	**Advantages**
Sequential Search	[5] The algorithm is simple to comprehend and implement, since it just requires a basic loop structure and a comparison operation to compare each member in the collection to the goal value. Any sort of collection, including arrays, linked lists, and other data structures, may be searched

	using sequential search. Because sequential search does not require any new data structures, it requires less memory and may be employed in memory-constrained contexts. Sequential search code is simple to comprehend and alter, making it simple to maintain and update as required. Sequential search can be a faster and more efficient choice than more sophisticated search algorithms for small datasets, especially if the data is already sorted.
Interval Search	[12] Interval search techniques based on interval trees, for example, can yield search durations of O(log n + k), where n is the number of intervals in the search area and k is the number of overlapping intervals returned. Interval search techniques scale well to big datasets, making them valuable in data-intensive applications. Interval search techniques may be used to any interval data type, including numerical ranges, temporal intervals, and geographic areas. Interval search algorithms have several uses, including database searches, data analysis, and scheduling systems. Interval search algorithms usually consume little memory, making them appropriate for usage in memory-constrained contexts.

Table 1.5 Comparison Table of Sequential Search and Interval Search in terms of Disadvantages.

Search Algorithm	**Disadvantages**
Sequential Search	[14] Sequential search can be wasteful in some circumstances, especially for huge datasets. If the target value is at the end of the collection, the method may have to compare every element in the collection, resulting in a worst-case time complexity of O(n), where n is the collection's size. Sequential search can only locate precise matches and cannot be used to look for patterns or approximate matches in data. Because the full collection must be saved in memory, it may consume a substantial amount

	of RAM if the data being searched is vast. Binary search or other search methods may be more efficient than sequential search if the collection is sorted.
Interval Search	[13] Implementing an interval search technique, such as an interval tree, may be difficult and necessitates a thorough grasp of data structures and algorithms. When compared to simpler search algorithms, the interval search technique may have more overhead, especially for tiny datasets or when just a few intervals are predicted to overlap with the search interval. Maintaining an interval search data structure, such as an interval tree, can be difficult and time-consuming, particularly if the information is often updated or altered. Interval search methods are designed to find intervals that overlap with another interval and may not be appropriate for other sorts of search queries or processes.

V. CONCLUSION

In conclusion, both sequential search and interval search algorithms are suitable for searching for specific data in a collection. Sequential search is a basic and easy-to-implement technique that may be used to search tiny collections or unsorted data. It is also memory-efficient and has a low overhead. However, it may not be the most efficient technique for looking for patterns or approximate matches in huge datasets.

Interval search algorithms, on the other hand, are specialized algorithms created for looking for overlapping intervals in a collection. They are practical for many applications since they are effective, scalable, and work with any kind of interval data. The implementation of an interval search method, however, can be difficult and time-consuming, particularly if the dataset is often updated or modified. For limited datasets or when just a few numbers of intervals are anticipated to coincide with the search interval, interval search may also incur more overhead than simpler search techniques.

Finally, the decision between sequential and interval search algorithms is dictated by the unique requirements and features of the data being searched. Sequential search may be sufficient for simple searches in small datasets. Interval search techniques may be required to obtain efficient and accurate results in more complicated searches in bigger or interval-based datasets.

REFERENCES:

[1] Margaret Rouse (2017) https://www.techopedia.com/definition/21975/search-algorithm

[2] Igor Deshko and Victor Tsvetkov (2022) https://iopscience.iop.org/article/10.1088/1742-6596/2373/5/052021/pdf

[3] Sudhiksha Malla (2023) https://www.programming9.com/tutorials/competitive-programming/433-search-algorithms-sequential-search-vs-interval-search

[4] Rohit Sharma (2022) https://www.upgrad.com/blog/linear-search-vs-binary-search/#:~:text=The%20time%20complexity%20of%20the,search%2C%20jump%20search%2C%20etc.

[5] (2019)

https://ozaner.github.io/sequential-search/

[6] Kale Casey (2020)

https://codersera.com/blog/let-us-understand-searching-algorithms/#:~:text=Interval%20Search%3A%20These%20algorithms%20are,For%20Example%3A%20Binary%20Search.

[7] Ashutosh Krishna (2022)

https://www.freecodecamp.org/news/search-algorithms-linear-and-binary-search-explained/

[8] Sohail Khan

https://www.quora.com/What-is-the-time-complexity-of-a-sequential-search

[9] Kamlesh Kumar Pandey, Narendra Pradhan (2014)

https://www.researchgate.net/publication/308119139_A_Comparison_and_Selection_on_Basic_Type_of_Searching_Algorithm_in_Data_Structure

[10] Admin AfterAcademy (2019)

https://afteracademy.com/blog/time-and-space-complexity-analysis-of-algorithm/

[11] Prashant Sharma (2021)

https://www.analyticsvidhya.com/blog/2021/09/searching-in-data-structure-different-search-methods-explained/#:~:text=Since%20linear%20search%20uses%20no,element%20in%20the%20search%20array.

[12] Arnold S. Korkhin (2017)

https://www.researchgate.net/post/What-is-the-advantages-of-Interval-Analysis#:~:text=The%20interval%20analysis%20allows%20to%20take%20into%20account,statistical%20methods%20are%20applicable%20not%20in%20all%20cases.

[13] Pritha Bhandari (2020)

https://www.scribbr.com/statistics/interval-data/

[14] Kamlesh Kumar Pandey (2014)

https://www.researchgate.net/figure/ADVANTAGE-DISADVANTAGE-OF-SEARCHING-TECHNIQUES_tbl2_308119139