Searching Algorithms

Samar Ahmed, Abdallah Ibrahim, and Bushra Alqhatani

Department of Computer Science and Engineering, The American University in Cairo

CSCE 1101, Section 2

Dr. Howaida Ismail

4/29/2023

Table of Contents:-

Searching Algorithms

Introduction

A search algorithm is a methodical process used to locate particular data within a collection of data. It is regarded as a basic computing operation. In computer science, choosing the right search algorithm can make the difference between a fast application and a slow one while looking for data. Without them, you would need to examine each piece of information, such as a phone number or business address, separately to see whether it is what you are searching for. It will take a lot of time to perform this for a sizable quantity of data. Instead, you can use a search algorithm to find the piece of data you're looking for. Because search algorithms are a fundamental concept in computer science, developers must comprehend them. A collection of guidelines called an algorithm is used to solve a problem in a certain number of steps. Consequently, search algorithms consider or investigate a problem. Search algorithms' primary function is to check or retrieve an element from any data structure in which it is stored (Krishna, 2022). The Linear Search Algorithm and the Binary Search Algorithm are the two most significant categories of search algorithms. Sudoku fans could solve a problem using a search algorithm in a matter of minutes as an example of the type of problem that these algorithms can tackle. Or they can use it to search through a sizable database for information like fingerprints or pictures (CSUN, n.d.). The speed of numerous algorithms will be compared in this study, along with suggestions for how to speed up some of them. Additionally, it will present a number of published works that suggest surveys of search methods. Enabling the organization of the search algorithm field in a manner that makes the issues and solutions clear.

2. Motivation

There has been work done in recent years to find quick, effective search algorithms. These strategies and algorithms play a crucial part in many applications or challenges found in the real world. Therefore, it is crucial to work tirelessly all the time to improve these algorithms. The goal of this research article is to examine several search algorithms in order to identify their differences as well as the benefits and drawbacks of each approach. The fact that the research will be useful and valuable regardless of its findings is one of its main incentives.

3. Problem Definition

This study will compare the linear search algorithm, the binary search algorithm, and the interpolation search algorithm. The Linear Search Algorithm is a straightforward search technique that looks up the target element in an array one by one. It takes O(n) time to complete. In contrast, an array or list can also be searched for elements using the Binary Search Algorithm. It operates by frequently splitting the list in half until all of the entries are located. It takes O(log n) amount of time to complete. The binary search version is then enhanced using the interpolation search algorithm. To estimate the position of the target element, it takes into account the value of the target element as well as the maximum and lowest values. It is quicker than binary search and has an O(log log n) time complexity (tutorialspoint, 2016). The explanation and analysis of these search methods will be included in the paper that follows. Finally, it will evaluate their performance while comparing their time and spatial complexity.

4. Literature Survey of Algorithms

4.1 Introductory to Surveys

In a survey by Nowak (2011), he looks into the issue of figuring out a binary-values function through a series of carefully chosen queries. The Generalised Binary Search algorithm was the major focus. This paper will provide more details on a well-known algorithm. In order for GBS to accomplish the information-theoretically ideal query complexity, he was able to create new conditions. The novel geometric relationship between the query and hypothesis spaces serves as the foundation for the new condition (Nowak, 2011). Furthermore, the following survey presented by Muja & Lowe, (2014), proposed new algorithms and compared them with previous algorithms. They demonstrate how the properties of the data set affect the parameters of the optimal closest neighbor algorithm and propose an automated setup process for selecting the most effective algorithm to search a specific data set. They provided a distributed nearest-neighbor matching framework that may be used with any of the algorithms mentioned in the study in order to scale to very huge data sets that would otherwise not fit in the memory of a single computer (Muja & Lowe, 2014). The new method outperformed the linear search algorithm in several ways. Interpolation Search was the subject of the final survey. In a study published by Yilmaz et al., (2022) they also provide new algorithms that are an improvement on the Interpolation Search method. In the face of Pattern Recognition Search (PRS), which is faster than interpolation search and binary search in terms of operating time, they developed a novel method of searching through a sorted array (Yilmaz et al., 2022). In the sections that follow, the three basic algorithms—binary search, linear search, and interpolation search—will each be described in detail and their benefits and drawbacks will be contrasted. To further understand the foundation upon which these surveys are based on.

4.2 Algorithm Description

A. Linear Search

```
int search(int arr[], int N, int x)
{
  int i;
  for (i = 0; i < N; i++)
    if (arr[i] == x)
      return i;
  return -1;
}
```

(GeeksforGeeks, 2019a)

This algorithm will have two methods to accept an array. First, it will either use a searchable value or an integer value as the input. Furthermore, this algorithm uses a dynamic array therefore it will accept a value of N. After searching the array, the function will return an index of the first occurrence. Moreover, the "==" operator is used to check each element to see if it equals a target value. If the target value is identified, the index of the array will subsequently be returned. A -1 is returned by the algorithm if no match is found (StarAgile, 2022).

B. Binary Search

```
int binarySearch(int arr[], int l, int r, int x)
{
   while (l <= r) {
      int m = l + (r - l) / 2;

      // Check if x is present at mid
      if (arr[m] == x)
         return m;

      // If x greater, ignore left half
      if (arr[m] < x)
         l = m + 1;

      // If x is smaller, ignore right half
      else
         r = m - 1;
   }

   // if we reach here, then element was
   // not present
   return -1;
}
```

(GeeksforGeeks, 2019b)

This code can be used to find a desired integer in an array. The main concept is that the array will be divided into half. Every time it does so the algorithm will compare between the middle element and the target element. Therefore, until the element is found the interval will be empty. Furthermore, if the interval is not found then the function will return the left and right digits of the search intervals. That said, binary search only works with sorted arrays.

C. Interpolation Search

```
// C++ program to implement interpolation
int interpolationSearch(int arr[], int lo, int hi, int x)
{
  int pos;

  // Since array is sorted, an element present
  // in array must be in range defined by corner
  if (lo <= hi && x >= arr[lo] && x <= arr[hi]) {

    // Probing the position with keeping
    // uniform distribution in mind.
    pos = lo
        + (((double)(hi - lo) / (arr[hi] - arr[lo]))
          * (x - arr[lo]));

    // Condition of target found
```

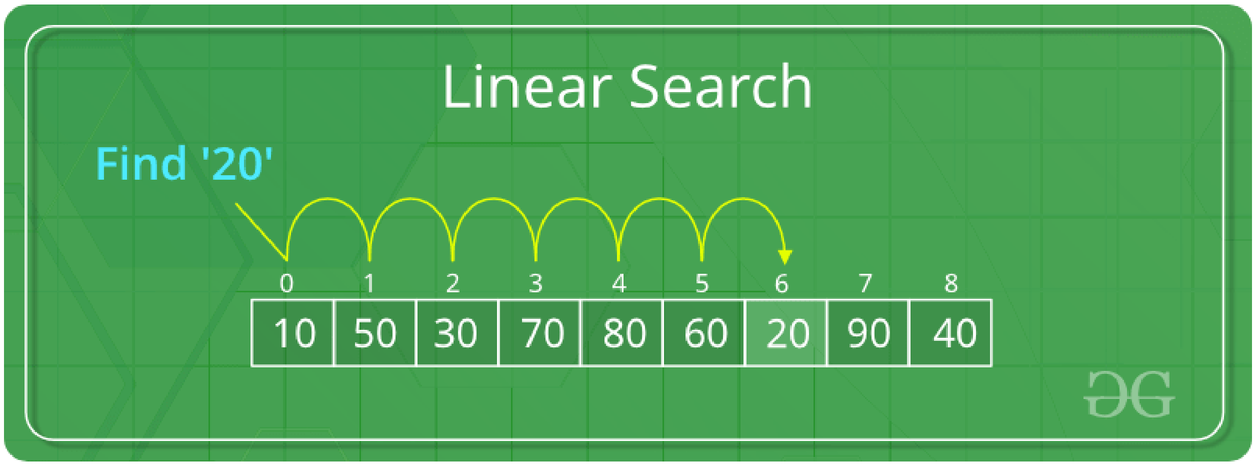
```
        if (arr[pos] == x)
            return pos;

        // If x is larger, x is in right sub array
        if (arr[pos] < x)
            return interpolationSearch(arr, pos + 1, hi, x);

        // If x is smaller, x is in left sub array
        if (arr[pos] > x)
            return interpolationSearch(arr, lo, pos - 1, x);
    }
    return -1;
}
```

(GeeksforGeeks, 2016)

In this C++ program's application of the interpolation search strategy, recursion is used to identify a specific element in an array with sort order. The procedure, which considers that the elements are spread evenly, guesses the element's position based on the array's endpoint values and the value of the element being looked for (tutorialspoint, 2016). The program's "interpolationSearch" function accepts four claims: the array to search, the bottom and upper bounds of the search range, and the element to be searched for. Before utilizing the interpolation procedure to estimate the element's position, the function determines whether it is inside the range. If the value at the approximate location corresponds to the target element, it returns the position. If not, the process is repeated with a new search range based on the expected position and the starting range. When the target element cannot be located or the function determines that the element does not exist in the array, -1 is returned (GeeksforGeeks, 2016).

5. Examples

A. Linear Search Algorithm

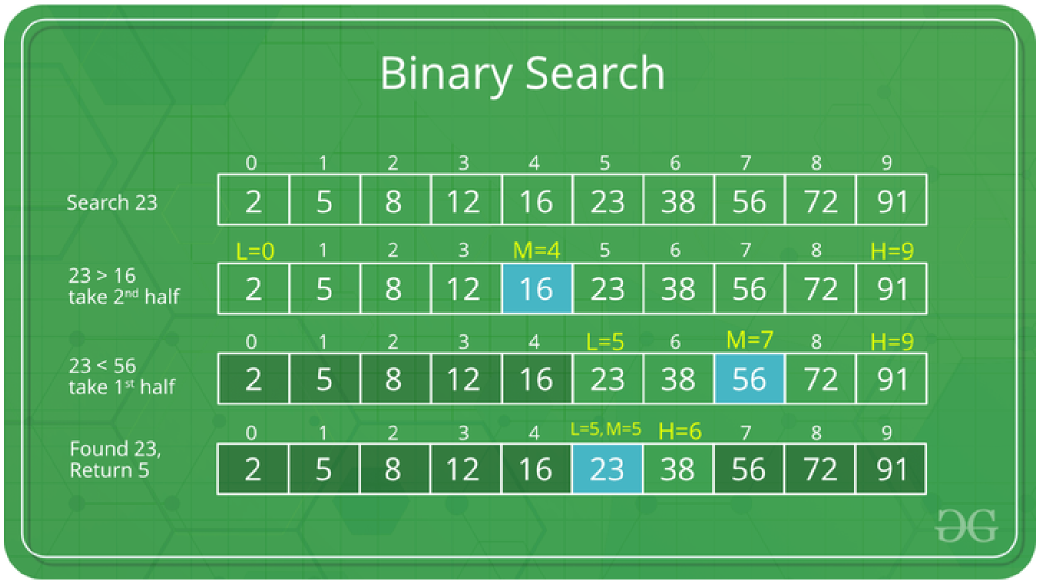


(GeeksforGeeks, 2019a)

This is a simple process in which the algorithm will read the target element, it will check the first index to see if it matches with the target element if not then the index will continue to be increment. If the key matches then the index will be returned.

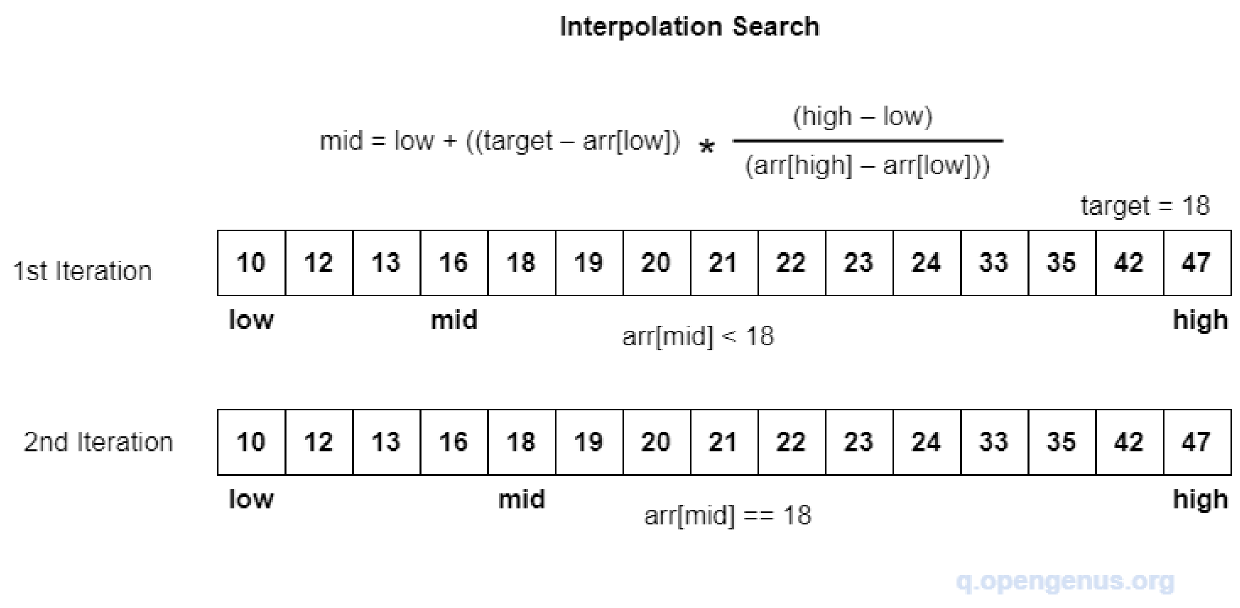
B. Binary Search Algorithm

A brief explanation of the example shown below is that initially, the search space is 9. The algorithm will get the midpoint M which is 4. Furthermore, the target will be compared with the first 4 elements of the array. If the target is not found, switch the search bar to the right in which the start is 5 and the end is 9 giving a midpoint M of 7. This is a continuous process until the target element has been found.



(GeeksforGeeks, 2019b)

C. Interporlation Search Algorithm



There are two main iterations for this process.

1st Iteration:

mid = 0 + 8 * 13/37-> 3

arr[3] = 16 < 18, therefore mid + 1 = 4

After this first step the search space is reduced, we know the target lies between index 3+1 to last index.

2nd Iteration:

mid = 4 + 0 * 9/29-> 4

arr[4] = 18 == target, terminate and return mid.

(OpenGenus, 2021)

6. Analysis and Critiques of Algorithms

A. Linear Search Algorithm

Sequential search can also be known as linear search. This can be seen as a search strategy that is simple. The strategy can help identify whether a particular element can be found within an array or list. The element being searched for is compared by the algorithm alongside the target element. The algorithm works its way through the complete list of elements, doing so in order to search for a match or usually until it reaches the end of the imputed list (StarAgile, 2022).

Because the list contains n elements, a linear search takes O(n) time. This shows how the size of the list influences how long it takes to discover a record linearly. As a result, linear search fails miserably on excessively large lists while excelling on small lists or arrays.

One of the advantages of linear search is that it is simple to use and consumes less memory. It is especially useful for unordered lists because it is unaffected by entry orders. That said, another advantage of linear search is that it is capable of providing quick and efficient solutions to a short list or an array. However, this is ineffective in the use of large lists, unlike a binary search (StarAgile, 2022).

However, linear search has some drawbacks. The approach is inefficient for big lists or arrays due to its O(n) time complexity, which means that it will have to search the entire list at worst. As a result, it is insufficient for applications that require speedy search times on large databases (GeeksforGeeks, 2019a).

B. Binary Search Algorithm

The binary search method works by halving the search frequency on a regular basis. When exploring sorted arrays, this method frequently makes use of the property that components are arranged in increasing order.

The array's core component is the starting comparison point in the method. If the desired value is lower than the array's center member, the array's bottom half is examined. If the target value is greater than the middle element, the search proceeds to the upper half of the array. This approach is performed until either the desired value is attained or all searchable items are removed (Journal, 2015).

The main advantage of this algorithm is that it has a high speed of O(log n), and n is the size of the array. This outperforms a linear search, which has an O(n) time complexity, as the size of the array grows (GeeksforGeeks, 2019b).

Furthermore, There are some limitations to using binary search. A pre-processing step could be needed if the array used is not ordered. This is done in order to sort the array, as an array needs to be sorted to be used. This could result in the process becoming costly if the array is large, this cancels out the purpose of using a binary search (Journal, 2015). Another limitation is that binary search is only useful for static arrays with elements that are not changed on a regular basis. A binary search may not be the best approach if the array is continually changing, with elements being added or removed, because it requires sorting the array after each change (tutorialspoint, 2016).

To summarize, binary search is a powerful search strategy that works best when searching over large arrays of sorted data. It is not always the best solution because it requires the array to be sorted and only works with static arrays.

C. Interpolation Search Algorithm

When looking at the components found in a sorted array, this is a search method that is used to locate said components. Using an element's value and range of the array, the search method guesses the position of the element.

The method first compares the target element to the array's start or end value. If this is the case, the operation returns the element's index. If not, it uses the index to conduct the following calculation before interpolating to find the target element (Yadav, 2021).

If the expected position is outside the array's coverage range, the procedure determines that the target component is not in the array. If the projected position corresponds to the target element, the technique returns the target element's index. Depending on whether the target element is higher or lower than the expected value, the approach repeats the estimating and comparison phases until it finds the target element or determines that it is not in the array (Yadav, 2021).

The worst scenario is O(n), or uniformly distributed if the array's elements are clustered. As a result, it is less efficient than binary search for tiny or concentrated arrays, but faster for large arrays with equally distributed items. Furthermore, the array must be sorted, and if the elements are distributed unevenly or the target element is at the beginning or end of the array, it may not perform well (tutorialspoint, 2016).

Interpolation search is a useful searching approach for big, equally split arrays, however, it does not always function well. It necessitates sorting the array and is time-dependent on the array's element arrangement and the placement of the target element.

7. Conclusion

The field of search algorithms includes a wide variety of techniques. This essay attempts to clarify any confusion, arrange the existing information on the subject, explain how different approaches are likely to work together, and identify any lingering issues that may require additional research. This study comprised four distinct methodologies, each with a detailed explanation, to illustrate the differences between temporal complexity and auxiliary space. Each algorithm has its own examples, which serve to emphasize the distinctions between them as well as any flaws and unrecognized benefits and drawbacks of each method.

References

CSUN. (n.d.). *Search Algorithms*. Www.csun.edu. http://www.csun.edu/~jam79241/jennifermosher.html

GeeksforGeeks. (2016, October 14). *Interpolation Search*. GeeksforGeeks. https://www.geeksforgeeks.org/interpolation-search/

GeeksforGeeks. (2019a, February). *Linear Search - GeeksforGeeks*. GeeksforGeeks. https://www.geeksforgeeks.org/linear-search/

GeeksforGeeks. (2019b, April). *Binary Search - GeeksforGeeks*. GeeksforGeeks. https://www.geeksforgeeks.org/binary-search/

Journal, I. (2015). A REVIEW ON COMPARISION OF BINARY SEARCH AND LINEAR SEARCH. *Www.academia.edu*. https://www.academia.edu/16738886/A_REVIEW_ON_COMPARISION_OF_BINARY_SEARCH_AND_LINEAR_SEARCH

Krishna, A. (2022, January 11). *Search Algorithms – Linear Search and Binary Search Code Implementation and Complexity Analysis*. FreeCodeCamp.org. https://www.freecodecamp.org/news/search-algorithms-linear-and-binary-search-explained/

Muja, M., & Lowe, D. G. (2014). Scalable Nearest Neighbor Algorithms for High Dimensional Data. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *36*(11), 2227–2240. https://doi.org/10.1109/tpami.2014.2321376

Nowak, R. D. (2011). The Geometry of Generalized Binary Search. *IEEE Transactions on Information Theory*, *57*(12), 7893–7906. https://doi.org/10.1109/tit.2011.2169298

OpenGenus. (2021, November 24). *Time and Space Complexity of Interpolation Search*. OpenGenus IQ: Computing Expertise & Legacy. https://iq.opengenus.org/time-complexity-of-interpolation-search/

StarAgile. (2022). *Data Visualization in R*. Staragile.com. https://staragile.com/blog/linear-search

tutorialspoint. (2016). *Data Structure - Interpolation Search - Tutorialspoint*. Www.tutorialspoint.com. https://www.tutorialspoint.com/data_structures_algorithms/interpolation_search_algorithm.htm

Yadav, R. (2021, July 26). *Scaler Topics*. Www.scaler.com. https://www.scaler.com/topics/data-structures/interpolation-search-algorithm/

Yilmaz, S., Ashraf, B., & Mehdi, S. A. (2022, February 1). *Pattern Recognition Search: An Advancement Over Interpolation Search*. IEEE Xplore. https://doi.org/10.1109/DELCON54057.2022.9753537