Searching and Sorting Algorithm in Data Structure

Maimoona Khilji

Data Science,

Institute of management Sciences

Peshawar Pakistan

Maimoon.khilji@gmail.com

***Abstract* – This document is about searching and sorting algorithms. I have formatted this document according to IEEE standard. But also added the Table of content.**

Searching Algorithm

Algorithms that are used to find or locate an element in a set of data are known as searching algorithm.

Linear Search

A **linear search** or **sequential search** is a method for finding an element within a list. It sequentially checks each element of the list until a match is found or the whole list has been searched. [1]

Linear Search performed on sequences of numbers that are ascending or descending or unordered. The linear search was given by comparing the data sought (X) with data in row A [1] ... A [n] starting with the first element data in sequence A. If the comparison is equal, then the search is stopped and declared successful. Whereas if the comparison is not the same value then,

1. If the data not sorted (random data), then the search will proceed to the next data.
2. If the data sequentially sorted, then the search process will proceed to the next data which is to the right of the data, given that the data sought (X) is greater than the data being compared.

If data is descending, then the search will only continue to the next data which is to the right of the data being compared if the data sought (X) is smaller than the data being compared. [2]

Algorithm

```
1. set found to false;
2. set position to –1;
3. set index to 0
    a. while (index < number of elements) and (found is false)
          i. if list[index] is equal to search value
         ii. found = true
        iii. position = index
         iv. end if
          v. add 1 to index
    b. end while
    c. return position
```

Properties

- It is easy to implement.
- It can be applied on random as well as sorted arrays.
- It has more number of comparisons.
- It is better for small inputs not for long inputs.

Binary Search

Binary search is an efficient algorithm for finding an item from a sorted list of items. It works by repeatedly dividing in half the portion of the list that could contain the item, until you've narrowed down the possible locations to just one. [3]

Binary Search can only perform on sequences of numbers that have been sorted either ascending or descending. Binary Search performs a search of X data in row A [1] ... A [n] with starting from intermediate data in sequence A. If X data value is equal to the middle data value of row A, then the search is stopped and declared success. While if not the same then,

1. For ascending data, the search will be continued to the left ½ if the data value of X is smaller than the middle data value in sequence A. whereas if the data value of X is greater than the middle data value in sequence A, Will continue to ½ right.
2. For descending data, the search will be continued to the left ½ if the X data value is greater than the middle data value in sequence A. whereas if the X data value is less than the middle data value in sequence A, Will continue ½ left.

The search will be stopped and declared to fail if ½ left or right half is a single data and the data is not the same as the X data being searched. [2]

Algorithm

1. calculate middle position
2. if (first and last have "crossed") then
    a. "Item not found"
3. Else if (element at middle = to_find) then
    a. "Item Found"
4. Else if to_find < element at middle then
    a. Look to the left
5. else
    a. Look to the right

Comparing Search Algorithms

- Sequential search is O(n) worst-case
- Binary search is O($\log_2$ n) worst-case
- Given n = 1,000,000 items
    - O(n) = O(1,000,000) /* sequential */
    - O($\log_2$ n) = O(19) /* binary */

- Clearly binary search is better in worst-case for large values of n, but there is always trade-offs that must be considered
    - Binary search requires the array to be sorted
    - If the item to be found is near the extremes of the array, sequential may be faster
- The sequential search starts at the first element in the list and continues down the list until either the item is found or the entire list has been searched. If the wanted item is found, its index is returned. So it is slow.
- Sequential search is not efficient because on the average it needs to search half a list to find an item.
- A Binary search is much faster than a sequential search.
- Binary search works only on an ordered list.
- Binary search is efficient as it disregards lower half after a comparison.

Time complexity of searching algorithm

The time complexities of sorting algorithms are given in Table 1 Time Complexity of Searching Algorithm

Table 1 Time Complexity of Searching Algorithm

Algorithm	**Worst-Case**	**Average-Case**	**Best-Case**
Linear Search	O (n)	O (n)	O (1)
Binary Search	O (n log (n))	O (n log (n))	O (1)

Space complexity of searching algorithm

The space complexities of sorting algorithms are given in Table 2 Space Complexity of Searching Algorithm

Table 2 Space Complexity of Searching Algorithm

Algorithm	**Space Complexity**
Linear Search	O (1)
Binary Search	O (1)

Sorting Algorithm

Algorithms that are used to organize the elements of data in ascending or descending order are known as sorting algorithm.

Selection Sort

It is in-place comparison sorting algorithm. It has an O (n^2) time complexity, which makes it inefficient on large lists, and generally performs worse than the similar insertion sort. Selection sort is noted for its simplicity and has performance advantages over more complicated algorithms in certain situations, particularly where auxiliary memory is limited. [4]

Algorithm

1. Set MIN to location 0
2. Search the minimum element in the list
3. Swap with value at location MIN
4. Increment MIN to point to next element
5. Repeat until list is sorted [5]

Bubble Sort

Bubble sort is a simple sorting algorithm. This sorting algorithm is comparison-based algorithm in which each pair of adjacent elements is compared and the elements are swapped if they are not in order. This algorithm is not suitable for large data sets as its average and worst case complexity are of O (n^2) where **n** is the number of items. [6]

Algorithm

1. Initiate with the first element i.e. position=0, and compare it with the next element of that particular array.
2. If the element to be compared is greater than the next element, then we need to swap the elements.
3. If the element to be compared is smaller than the next element, move to the next position to check for another element.
4. Repeat Step 1-3 until all the elements get sorted in an ascending or descending fashion. [7]

Merge-Sort

It is a Divide and Conquer algorithm. It divides the input array into two halves, calls itself for the two halves, and then merges the two sorted halves. **The merge () function** is used for merging two halves. The merge (arr, l, m,

r) is a key process that assumes that arr[l..m] And arr[m+1..r] are sorted and merges the two sorted sub-arrays into one. [8]

Divide-and-Conquer

The divide-and-conquer pattern consists of the following three steps:

1. **Divide:** If the input size is smaller than a certain threshold (say, one or two elements), solve the problem directly using a straightforward method and return the solution so obtained. Otherwise, divide the input data into two or more disjoint subsets.
2. **Conquer:** Recursively solve the sub problems associated with the subsets.
3. **Combine:** Take the solutions to the sub problems and merge them into a solution to the original problem. [9]

Time complexity of sorting algorithm

The time complexities of sorting algorithms are given in Table 3. Time Complexities of Sorting Algorithms

Table 3. Time Complexities of Sorting Algorithms

Algorithm	Worst-Case	Average-Case	Best-Case
Bubble Sort	O(n^2)	O(n^2)	O (n)
Selection Sort	O(n^2)	O(n^2)	O(n^2)
Merge Sort	O(n log(n)	O(n log(n))	O(n log(n))

Space complexity of sorting algorithm

The space complexities of sorting algorithms are given in Table 4. Space Complexity of sorting algorithm

Table 4. Space Complexity of sorting algorithm

Algorithm	Space Complexity
Bubble Sort	O (1)
Selection Sort	O (1)
Merge Sort	O (n)

Big-O Complexity

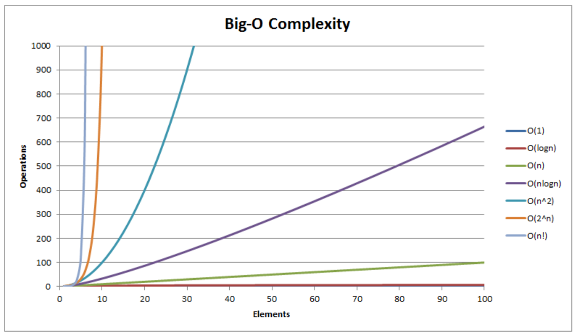

Figure 1 Graphical Representation of Big-O complexity [10]

References

[1]. "Linear search - Wikipedia", *En.wikipedia.org*, 2021. [Online]. Available: https://en.wikipedia.org/wiki/Linear_search#CITEREFKnuth1998. [Accessed: 25- Sep- 2021].

[2]. R. Rahim, S. Nurarif, M. Ramadhan, S. Aisyah and W. Purba, "Comparison Searching Process of Linear, Binary and Interpolation Algorithm", *Journal of Physics: Conference Series*, vol. 930, p. 012007, 2017. Available: 10.1088/1742-6596/930/1/012007 [Accessed 25 September 2021].

[3]. Binary search (article) | Algorithms | Khan Academy", *Khan Academy*, 2021. [Online]. Available: https://www.khanacademy.org/computing/computer-science/algorithms/binary-search/a/binary-search. [Accessed: 25- Sep- 2021].

[4]. En.wikipedia.org. 2021. *Selection sort - Wikipedia*. [online] Available at: <https://en.wikipedia.org/wiki/Selection_sort> [Accessed 24 September 2021].

[5]. Tutorialspoint.com. 2021. *Data Structure and Algorithms Selection Sort*. [online] Available at: <https://www.tutorialspoint.com/data_structures_algorithms/selection_sort_algorithm.htm> [Accessed 24 September 2021].

[6]. Tutorialspoint.com. 2021. *Data Structure - Bubble Sort Algorithm*. [online] Available at: <https://www.tutorialspoint.com/data_structures_algorithms/bubble_sort_algorithm.htm> [Accessed 24 September 2021].

[7]. JournalDev. 2021. *Bubble Sort Algorithm - JournalDev*. [online] Available at: <https://www.journaldev.com/35480/bubble-sort-algorithm> [Accessed 24 September 2021].

[8]. GeeksforGeeks. 2021. *Merge Sort - GeeksforGeeks*. [online] Available at: <https://www.geeksforgeeks.org/merge-sort/> [Accessed 24 September 2021].

[9]. Goodrich, M., Tamassia, R. and Goldwasser, M., n.d. *Data structures and algorithms in Java™*.

[10]. *Linear and Binary Search Comparison*. 2021