**Search Algorithm: Logarithmic with Linear**

Codillo, Ezekiel RobinA. Abeles, Liera Chavez, Thea frances

BSIT – 1204

JPLPC – Malvar Campus

**ABSTRACT**

The study conducted, aims to find the best suitability of two algorithms for different task scenarios through process and time complexity analysis. Will logarithmic be a better useful algorithm for wide varieties of tasks or linear will provide better results with the best time output. Both algorithms will be compared with different sizes of arrays and methods of usage to determine time complexity and therefore the best use for each algorithm.

**INTRODUCTION**

  Algorithms, a set of specific instructions or command followed by the system to solve or implement a specific task or solution, it is mostly used in many large scale programs that perform many mathematical methods and functions at high speed, so it is important to use the proper algorithm on a specific task or function that the system is going to implement.

But in this case we are going to primarily focus on searching algorithms. In terms of searching in a program, we have a very wide variety of algorithms to choose for, but not because they are all searching algorithms, we still need to choose the best algorithm for our searching scenario, that is why in this study we are going to compare two (2) searching algorithms; the “Logarithmic” and “Linear” searching algorithms, we will find both algorithms’ advantages and disadvantages in different scenarios and specific tasks with their time complexity.

**Logarithmic Search**

A searching algorithm that uses a file in which the sort - keys are in ascending order. The middle key in the file is examined and, depending upon whether this is less than or greater than the desired key, the top or bottom part of the file is again examined. Continuing in this way the algorithm either finds the desired record or discovers its absence from the file. Thus the algorithm treats the file as though it were a binary search tree.

**Linear Search**

Sequential search algorithm is another name for linear search. It is the most straightforward search algorithm. In a linear search, we merely go through the list in its whole and match each element with the object whose location needs to be determined. The algorithm returns the item's location if a match is made; otherwise, it returns NULL.

It is frequently used to search for a certain element in an unordered list, or a list where the entries are not sorted. Linear search has a worst-case temporal complexity of O(n).

**Comparison**

In terms of Time Complexity, Best case scenario, average case scenario, and worst case scenario.

The target element being the first element in the array is the Best Case scenario. In this instance, there has just been one comparison. Therefore, O (1) is the time complexity.

The Average Case: Typically, the target element will be near the center of the array. In this scenario, N/2 comparisons will be made. As a result, the time complexity will be O(N) (ignoring the constant).

The Worst Case scenario is when the target element is either the array's last element or not there at all. Since we must traverse the complete array in this situation, N comparisons will be made. So, O(N) will be the temporal

complexity.

‘

The logarithmic search compares the target element to the array's middle value. The search moves to the right half if the target element is bigger than the center element, otherwise, the search moves to the left half if the target element is less than the middle value, repeat this step until the target element is equal to the middle element or the middle element no longer exists in the array. The target element's index is returned if it is located; else, -1 is returned.

While Linear search, we start at one end of the list and continue through each element until the target element is located. It is the most straightforward search algorithm.

The target element being the middle element in the array is the Best Case scenario. In this instance, there has just been one comparison. Therefore, O (1) is the time complexity.

The Average Case: The target element will often be present someplace in the array. Therefore, O(logN) will be the time complexity.

If the target element is not on the list or is not close to the middle element, that is the worst-case scenario. Therefore, O(logN) will be the time complexity.

In short, logarithmic is more useful and faster for large – scale database and arrays, and linear is only at its best when handling elements less than one hundred (100).

**Conclusion**

In conclusion, The logarithmic search is better and more efficient than linear search, not only on data count or scale, but mainly on time complexity, the process of linear takes too long for it has to compare the target variable to ach element of the array which will take much time of implementation, while logarithmic search follows a procedure by looking for the middle value of the sorted array and searches the right part of the array based on the comparison with the middle value whether greater or less than the value, on the other hand Linear search is still the best algorithm used for small – value arrays, but still this makes logarithmic search the best searching algorithm on different platforms and different complicated programming scenarios.

**References**

Logarithmic Search and Linear Search Time complexity - FreeCodeCamp (January 11, 2022):

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

Linear Search meaning - Javatpoint (No Date)

<https://www.javatpoint.com/linear-search>

Logarithmic Search meaning – ENCYCLOpedia (No Date)

<https://www.encyclopedia.com/computing/dictionaries-thesauruses-pictures-and-press-releases/logarithmic-search-algorithm>