O-BIG NOTATION

Castaño Silva Laura Daniela

{est.laura.castano@unimilitar.edu.co}

Teacher: Miguel Monroy

The O Big notation is known as a measure. It is used to examine or study the performance of an algorithm and categorize it depending on the criteria of the input data, for example, to evaluate the execution speed depending on the size of the input data. And, in turn, algorithms are ranked depending on the processing time with respect to the size of the input data, regardless of their hardware. Furthermore, this can be known by means of a function: constant, quadratic, linear or logarithmic. Therefore, two or more algorithms that perform the same task are examined, but, the one with lower levels of complexity is chosen and good performance is determined. In such a way that: [1]*"we need to assess the feasibility of our solutions in certain situations".*

In addition, there are different types of algorithms and/or functions, for example, the time constant (1) where it will be similar in different executions without taking much relevance of the size of the input dataset. The function of this time constant will be maintained at the time of execution, regardless of whether (n) is increasing. Array reads are a key data structure for reading any part of the algorithm. Also, it is said that: [2] *"the access time will always be constant, regardless of the size and number of elements contained in the array".*

Also, there is a logarithmic time O (log n) where there is a division by half of certain initial criteria of the algorithms to find a value by means of a logarithm of the input dimension of the data, or, on the contrary, a binary search can be done where, when comparing an element of the medium to know if it is larger or smaller, an analysis is done again on the subgroup to compare it again. In this case, *"we can see it easier with two algorithms <selectionSort> and <insertionSort>"* [3].

On the other hand, it is possible to visualize a linear time (n), where the functions are used by means of loops that go through the elements to implement the sum of numbers that, growing linearly, depending on an increase in (n). [2] *"The more elements are added to the processing, the execution time increases proportionally."* Where the time grows in a straight line. Implying that, if we have twice as many components in the input, it will take twice as long. However, the increment is still in a linear fashion.

The O-Big notation is often recognized for its efficiency, but not for its time, because it is a constant variable, for example, *"the amount of processed data is better to send it to another person, because if there is little data it is better to send the document digitally, while the larger the document becomes, the better it is to send it physically"* [4]. Determining also the cost comparison of two or more algorithms to solve a specific problem.

It is possible, the implementation of tools to the creators of algorithms to find a more viable solution and, that manages to satisfy specific needs to be supplied. When solving these problems, it is essential to document or comment on the algorithmic complexity of the problem. When the complexity is quantified, there are very few times that an exact result is obtained, so it mostly depends on estimates *"That will help any programmer who comes after, both to use it and to have to optimize it, and will know what is the logic to beat"* [4].

Even, the O-Big notation *"allows us to compare several equivalent algorithms without worrying about doing performance tests that depend on the hardware used"* [5]. So, the validated results establish is that the algorithm with higher performance will be independent of its hardware if the data set is large, *"on small sets faster hardware may give faster results for a less efficient algorithm, but as the set grows this will no longer be the case"* [5]. Subsequently, the large O notation says that in an algorithm [O(f(n))] the number of criteria or single operations to be performed by the computer is less than a constant, for example. the times of execution [f(n)], n will be increasing.

In conclusion, the O-Big notation is a mathematical process that can be useful when evaluating the processing speed and performance of an algorithm that can meet specific needs. By showing the behavior as the size of the component to be executed increases, it is useful to obtain a classification of the efficiency of these algorithms. Essential when examining processing needs and the space required to carry out the compilation of algorithms. Finally, to assess how good an algorithm can be to solve problems that become large as well as its performance remains timely to solve small values or damage by the computer.

Referencies

[1] P. Cianes, «Notacion big O,» 20 10 2020. [En línea]. Available: https://pablocianes.com/notacion-big-o/. [Último acceso: 10 08 2023].

[2] Aprender BIG DATA, «Big-O Para Principiantes,» 03 10 2023. [En línea]. Available: https://aprenderbigdata.com/big-o/. [Último acceso: 10 10 2023].

[3] NetMentor, «Notacion Big-O,» 17 03 2021. [En línea]. Available: https://www.netmentor.es/entrada/notacion-big-o. [Último acceso: 12 10 2023].

[4] G. Cortez, E. Diaz, F. Gómez y I. Sandoval, «Ordenamiento Big-O,» 06 11 2019. [En línea]. Available: https://www.studocu.com/latam/document/universidad-tecnologica-de-panama/estructura-de-datos-ii/ordenamiento-big-o-este-es-un-pequeno-trabajo-de-como-funciona-la-notacion-big-o-con-todo-y-sus/5532099. [Último acceso: 11 10 2023].

[5] J. M. Alarcón, «Rendimiento de algoritmos y notación Big O,» 17 06 2016. [En línea]. Available: https://www.campusmvp.es/recursos/post/Rendimiento-de-algoritmos-y-notacion-Big-O.aspx. [Último acceso: 11 10 2023].