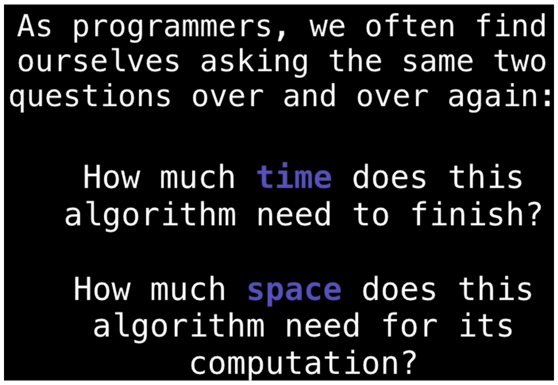
Computational Complexity Analysis

Monday, April 29, 2024   11:52 AM



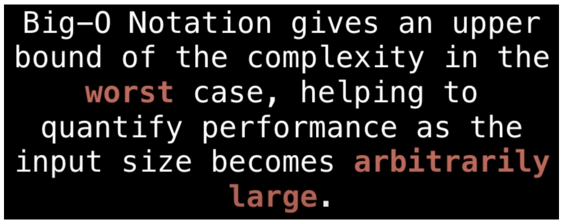
As programmers, we often find ourselves asking the same two questions over and over again:

How much **time** does this algorithm need to finish?

How much **space** does this algorithm need for its computation?

Basically, most of time, we are mostly concerned about are the two things, How much time out program is taking and how much space it is consuming being on run time, if it takes a lot amount of time and space, we have we have done something wrong and that need to be rectified and fixed.

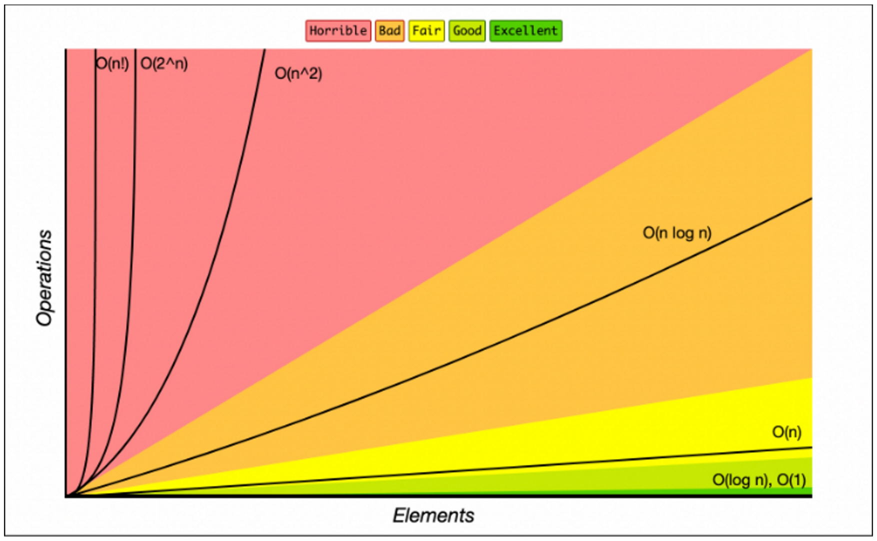
So in order to identify both the parameters of the Programming to ensure, who well our system is designed is called Big O Notation.
Big O Notation gives upper bound of complexity in the worst case scenario.



Big-O Notation gives an upper bound of the complexity in the **worst** case, helping to quantify performance as the input size becomes **arbitrarily large**.

Now lets take an example of finding the #7 from unordered list, now imagine the length of list is a million numbers, not the worst case scenario of 7 appearing in the list is very last of the list So the time complexity is linear to the number of data in the list.

Complexities	Order of n	Example
Constant Time Complexity	O(1)	Finding first element from an array OR Accessing the Array Index Inserting a Node in Linked List Push or Pop action on Stack Enqueue or Dequeue on Queue
Logarithmic	O(log(n))	Finding a number from sorted array Finding a number from Binary Search Tree.
Linear Time	O(n)	Finding a number from un sorted array Or Traversing an array. Traversing a linked list Comparison of two strings. Checking of Palindrome. (Word reads same backward or forward, Eg Madam)
Linear Logarithmic Time	O(n log(n))	Recursive sorting algorithms.

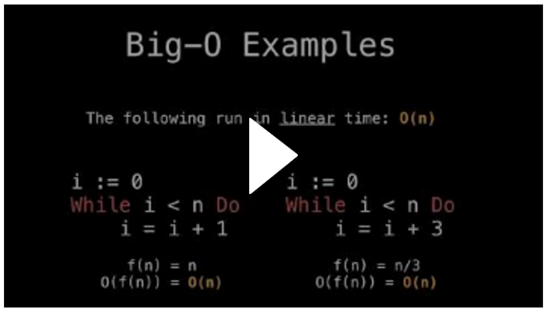


References

https://stackoverflow.com/questions/1592649/examples-of-algorithms-which-has-o1-on-log-n-and-olog-n-complexities

https://www.freecodecamp.org/news/big-o-cheat-sheet-time-complexity-chart/

Introduction to Big-O



Big-O Examples

The following run in linear time: O(n)

```
i := 0                    i := 0
While i < n Do            While i < n Do
    i = i + 1                 i = i + 3

    f(n) = n                  f(n) = n/3
O(f(n)) = O(n)            O(f(n)) = O(n)
```