Big-O Notation

...

January 13, 2022

/in/agrawalshreya10

What is Big-O Notation?

By the definition of Wikipedia,

“**Big O notation** is a mathematical notation that describes the limiting behavior of a function when the argument tends towards a particular value or infinity.”

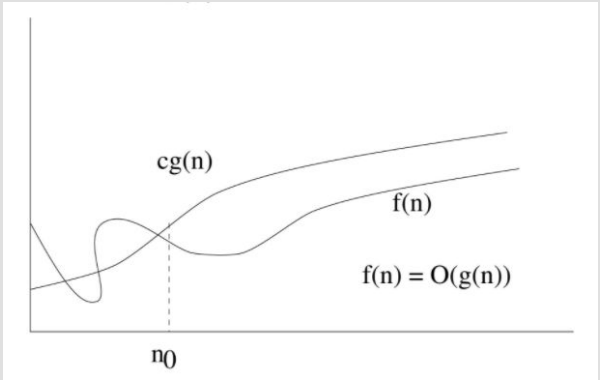
In plain words, Big O notation describes the complexity of your code using algebraic terms.

Big-O as defined by University of Washington:

- Big-O is about finding an asymptotic upper bound.
- **Formal Definition of Big-O:**
  f(N) = O(g(N)) if $\exists$ *positive constants c,* N_0 *such that*
  $f(N) \leq c \cdot g(N) \quad \forall N \geq N_0$

Few points to remember here:

- We are concerned with how *f* grows when *N* is large. (Not concerned with small *N* or constant factors.)
- *"f(N) grows no faster than g(N)."*

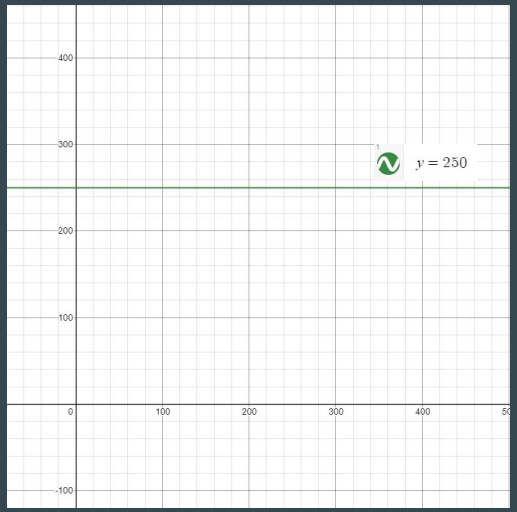


Time Complexity

Constant Time Complexity

By the definition of Wikipedia,

An algorithm is said to be **constant time** (also written as O(1) time) if the value of T(n) is bounded by a value that does not depend on the size of the input.



- **For example,** accessing any single element in an array takes constant time as only one operation has to be performed to locate it.
- Finding the median value in a sorted array of numbers is also the case of **Constant Time Complexity**.

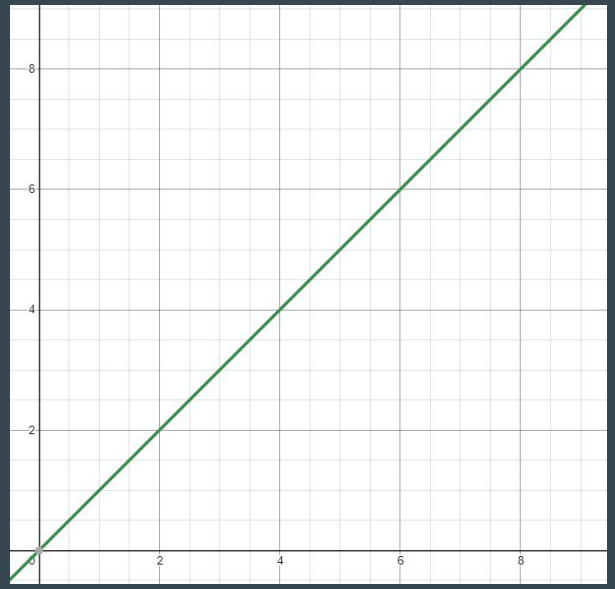
/in/agrawalshreya10

Linear Time Complexity

Linear Time Complexity describes an algorithm whose complexity will grow in direct proportion to the size of the input data.

It is represented as O(n).

- **For example,** printing all the elements of an array.
- Finding the minimal value in an unordered array of numbers is also an example of **Linear Time Complexity**.

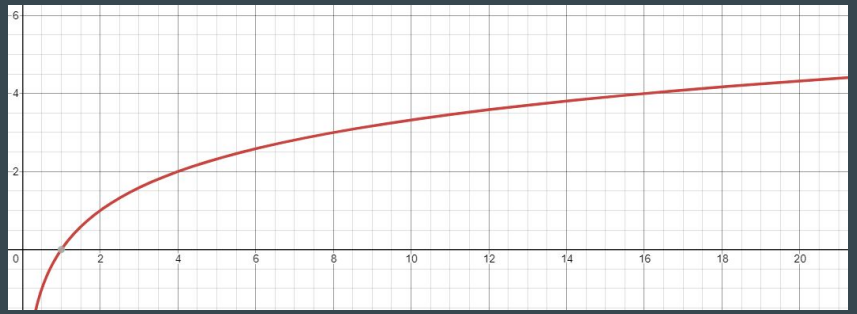


Logarithmic Time Complexity

By the definition of Wikipedia,

An algorithm is said to be taking logarithmic time when T(n) = O(log n). Since $\log_a$ n and $\log_b$ n are related by a constant multiplier, and such a multiplier is irrelevant to big-O classification, the standard usage for logarithmic - time algorithms is O(log n) regardless of the base of the logarithm appearing in the expression of T.

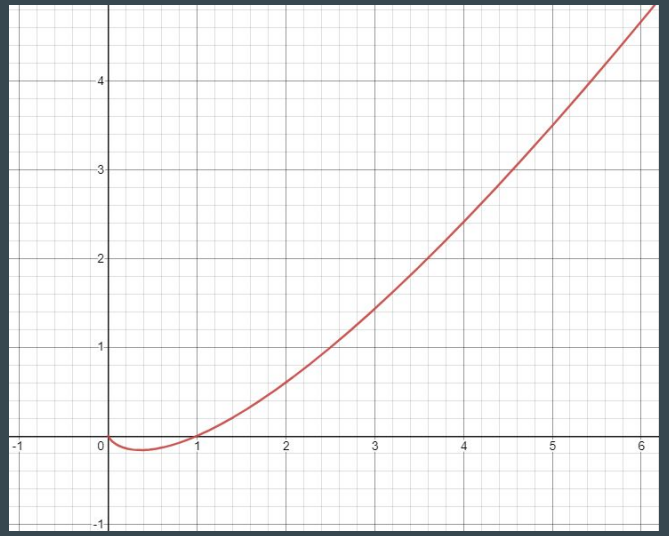
- They are considered highly efficient.
- These algorithms are commonly found in operations on binary trees or when using binary search.



Log-Linear Time Complexity

By the definition of Wikipedia,
An algorithm is said to run in **log-linear time** (also referred to as **quasilinear time**) if $T(n) = O(n \log^k n)$ for some positive constant k; linearithmic time is the case k = 1.

- In many cases, the n log n running time is simply the result of performing a $\Theta(\log n)$ operation n times.
- Merge Sort and Quick Sort are two very good examples of log-linear time complexity.

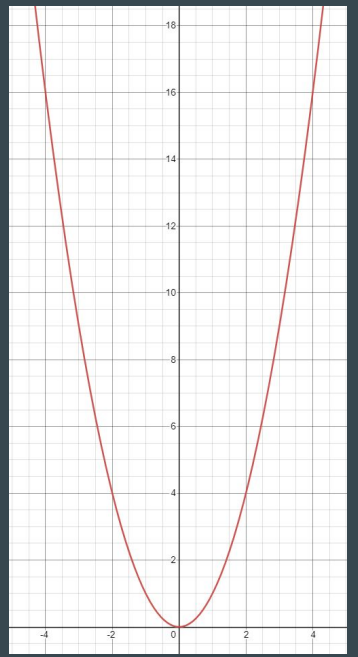


Quadratic Time Complexity

Quadratic Time Complexity - $O(n^2)$ represents an algorithm whose performance is directly proportional to the squared size of the input data set.

Within our programs, this time complexity will occur whenever we nest over multiple iterations within the data sets.

- **For example,**
  To sort an array of size n in ascending order, we need to use the **Insertion Sort Algorithm**, which has $O(n^2)$ time complexity in the worst case.

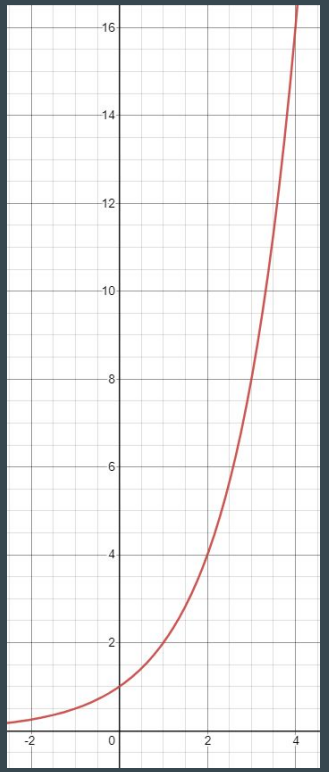


Exponential Time Complexity

Exponential Time Complexity represents an algorithm where T(n) is bounded by $O(2^{n^k})$ for some constant k.

In simple words, the growth of this time complexity doubles with each addition to the input data set.

- *Fibonacci Series* is the best example of **Exponential Time Complexity.**



/in/agrawalshreya10

Basic Differences between Best, Average and Worst Time Complexities

Best Time

The term *best-case performance* is used to describe an algorithm's behavior under optimal conditions. For example, the best case for a simple linear search on a list occurs when the desired element is the first element of the list.

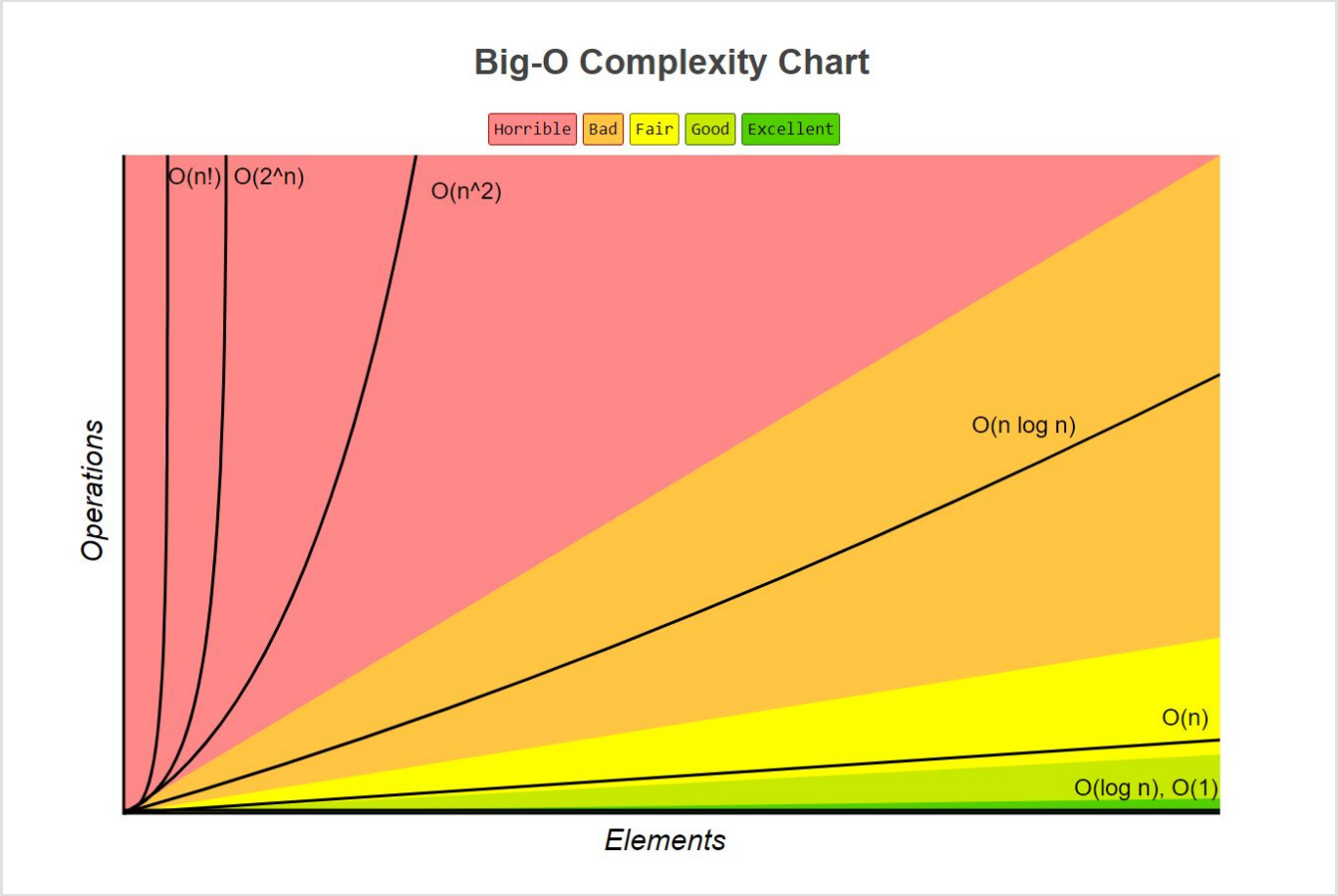
Average Time

The *average-case complexity* of an algorithm is the amount of some computational resource (typically time) used by the algorithm, averaged over all possible inputs.

Worst Time

The *worst-case complexity* measures the resources (e.g. running time, memory) that an algorithm requires given an input of arbitrary size (commonly denoted as n or N). It gives an upper bound on the resources required by the algorithm.

/in/agrawalshreya10



Space Complexity

Space Complexity

By the definition of Wikipedia, the space complexity of an algorithm or a computer program is the amount of memory space required to solve an instance of the computational problem as a function of characteristics of the input.

- In simpler words, it is the memory required by an algorithm until it executes completely.

/in/agrawalshreya10

Thank You!

/in/agrawalshreya10

Q & A

/in/agrawalshreya10