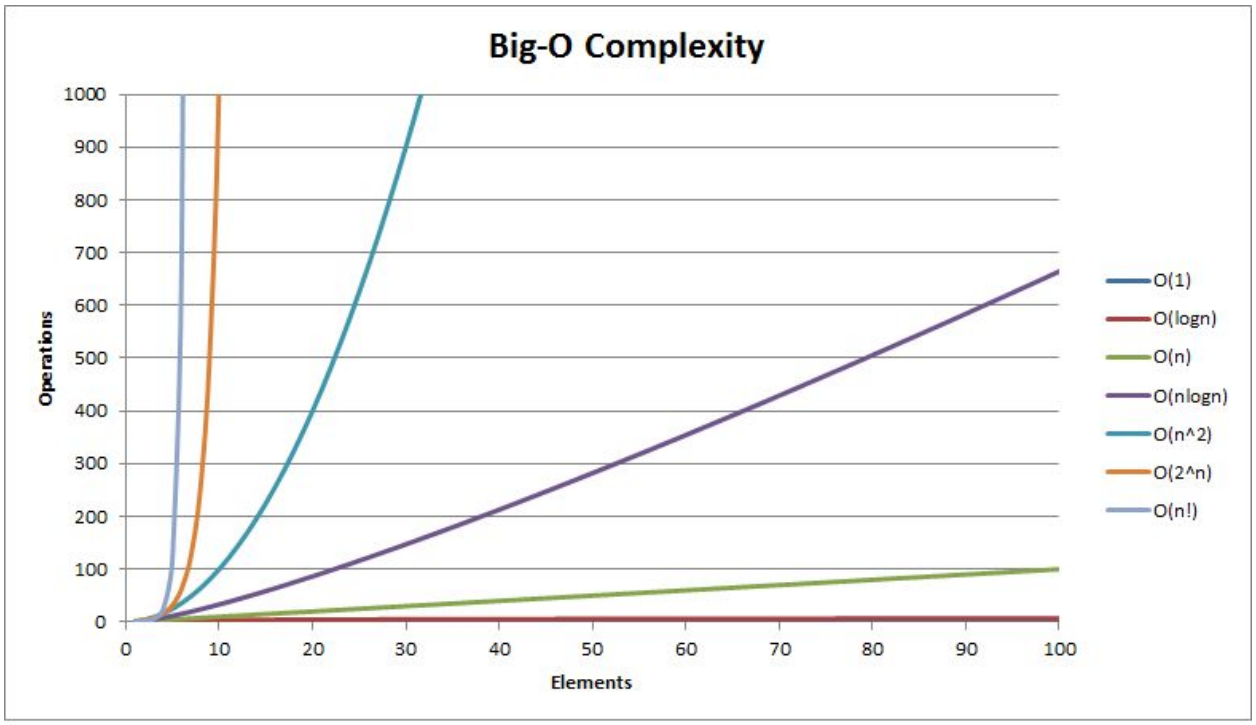
Big-Oh intro

Prof Bill, Jan 2020

"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."

- en.wikipedia.org/wiki/Big_O_notation

What function best describes the performance of your algorithm for N items?
/ my favorite chart of the course */*



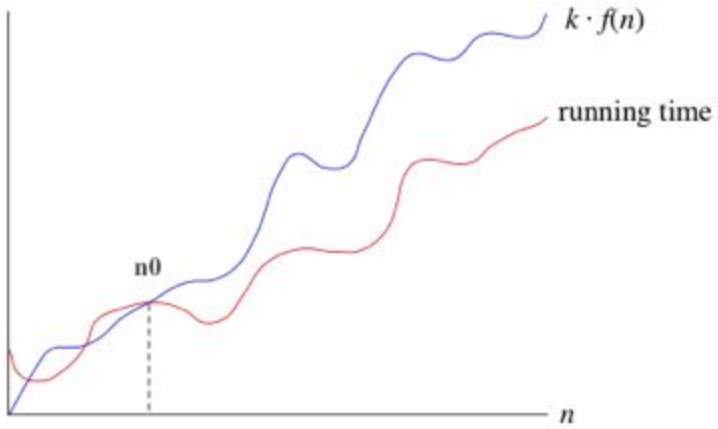
Source:www.hackerearth.com/practice/notes/big-o-cheatsheet-series-data-structures-and-algorithms-with-thier-complexities-1/

Seven performance categories are most common, for a problem of size = n:

- O(1) - constant time
- O(log(n)) - logarithmic time
- O(n) - linear time
- O(n log(n)) - quasi-linear or "n log n" time
- O(n^2) - polynomial time
- O(2^n) - exponential time
- O(n!) - factorial time

Formally, for O(f(n)) defines a function f(n) where:

```
running time <= k * f(n), for n > n0
```



For this Big-Oh, f(n) defines an **asymptotic limit** of our running time.

Constants, multipliers, and lower-order terms are ignored. Why? Because they are insignificant compared to the performance function for large N.

Example: Ignore constants, multipliers, and lower-order terms.

$f(n) = 5n^2 + 7n + 101$ is $O(n^2)$

Try: Each Big-Oh function above for (piddly) N=100...function dominates growth.

Try: What is Big-Oh for the array operations: add, get, remove?

Big-Oh is **not** program timing or running benchmarks. It is a theoretical estimate, independent of specific program or computer.

Links:

- ➔ Another fun Big-OH summary, bigocheatsheet.com
- ➔ Wikipedia summary, en.wikipedia.org/wiki/Big_O_notation