Big-O Notation

Time and Space Complexity

Big-O Notation

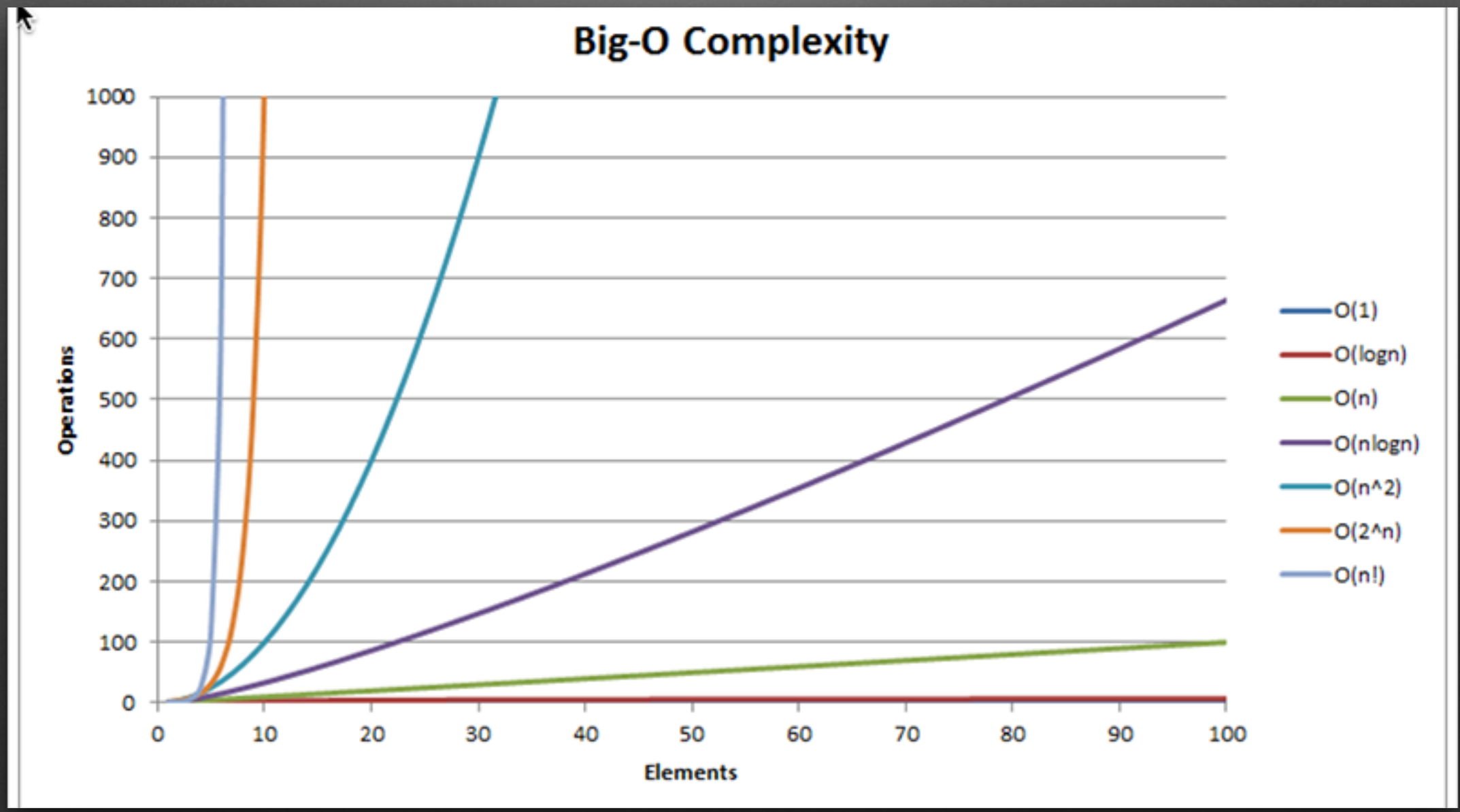
- A measure of how rapidly space or time requirements grow relative to the input, as the input gets arbitrarily large.
- AKA Asymptotic Complexity, or Complexity

An Academic Aside

- Academics differentiate between Big-O, Big-Omega and Big-Theta

- In industry practice, and interviews, Big-O is the tightest description of the asymptotic complexity, a merger of Big-O and Big-theta

- If this doesn't make sense to you, ignore it.



Asymptotic Complexity

cribbed from: http://bigocheatsheet.com

Best, Worst, Expected

- Big-O sometimes differs based particular inputs or scenarios, leading to best, worst and expected case measures.

- We need to consider the worst and the expected case. (Best case is seldom of interest) The worst case may be unlikely, and an algorithm is selected based on its expected case complexity.

Big-O Math

- If complexity is constant (doesn't change with size of input), complexity is O(1)

- Ignore constants, 2n, 5n and n are all O(n)

- As input grows, the result is dominated by the largest term, discard smaller terms. $O(n^2 + n)$ is $O(n^2)$

Multi-part Algorithms

If an algorithm has several steps, how are complexities combined?

- Add complexities if steps are sequential [do this, then do that]

- Multiply complexities if a step executes each time another step executes [do this each time you do that]

Amortized Complexity

"average time taken per operation, if you do many operations".

Consider a dynamic array.

- Insert is O(1), unless the underlying array is full and needs to be resized. Resizing takes O(n) time where n is the current size of the array.

- In the long term, the total time taken for adding m items to the array is O(m), and so the amortized insert time is O(1).

Complexity Examples

- O(1) - constant complexity, index into an array
- O(log n) - Problem divides work into two pieces at each step
- O(n) - linear complexity, loop over each item once
- O(n log n) - for each item, perform some function with complexity O(log n), i.e. merge sort
- $O(n^x)$ - polynomial complexity, X Nested loops over each item, i.e. Bubble Sort
- $O(2^n)$ - Exponential complexity, Constraint satisfaction problems, Backtracking search (branch and bound)
- O(N!) - Factorial complexity, generate all permutations of a list, i.e. traveling salesman

Resources

- Big-O Cheat Sheet: http://bigocheatsheet.com
- Cracking the Coding Interview, 6th Edition