Codesmith

CSPrep

Day 6

Overview

- Algorithms
- Time complexity
- Big O notation
- Real-world examples

What is an algorithm?

- An algorithm is a step by step set of instructions that provide a solution to a problem.
- In programming, most of the code you write is algorithmic.

What is time complexity?

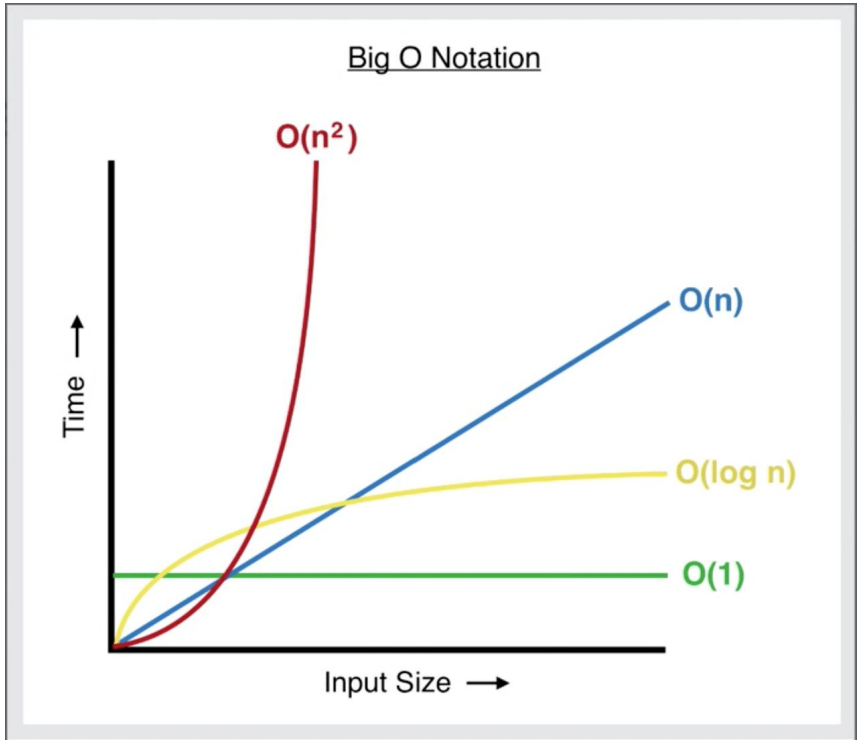
- In computer science, the time complexity is the computational complexity that describes the amount of computer time it takes to run an algorithm 🤔
- It describes the relationship between the size of an algorithm's input and the number of computational steps it takes for the algorithm to complete.
- Programmers use a common vocabulary to talk about time complexity.

Big O Notation

- We communicate about time complexity using Big O Notation.
- Big O is a mathematical notation that describes the rate at which the number of computational steps grows in relation to the input size.
- It refers to the maximum number of steps the algorithm could take under the worst-case scenario.

Common time complexities

Time Complexity	Big O Notation
Constant	O(1)
Logarithmic	O(log(n))
Linear	O(n)
Quasilinear	O(n log(n))
Quadratic	$O(n^{2)}$
Exponential	$O(2^n)$



Interpretation of Big O Notation

- My Algorithm has O(n) (linear) time complexity
  - Meaning: as the input size grows, the maximum number of steps my algorithm might take to complete will grow at the same rate.
- My Algorithm has $O(n^2)$ (quadratic) time complexity
  - Meaning: as the input size grows, the maximum number of steps my algorithm might take to complete will grow at a rate equal to the input size squared.
- My Algorithm has O(1) (constant) time complexity
  - Meaning: regardless of the size of the input, my algorithm will always take the same number of steps to complete.

Common time complexities in practice

Time Complexity	Big O Notation	Scenario	Speed
Constant	O(1)	Key lookup	Fastest
Logarithmic	O(log n)	Binary Tree Search	
Linear	O(n)	1 Level Looping	
Quadratic	$O(n^2)$	2 Nested Loops	
Exponential	$O(2^n)$	Finding subsets	Slowest
Factorial	O(n!)	Generating permutations	

n represents the size of the input. For functions of arrays, n is the length of the array. For functions of integers, n is the number of digits.

Why care?

Why does it matter?

- Computers have limited resources (space and processing power). Writing algorithms with better time complexity saves time!

Time = Performance, efficiency, money

Suppose I have an algorithm with $O(20n^2 + 3n)$ time complexity. Usually, we just call it $O(n^2)$. Why is this okay?

Big O Generalizes

Difference in time complexity matters only at big numbers for n.

Compare $50n$ to $2n^2$ when $n = 1{,}000$

$50 * 1{,}000 \Rightarrow 50{,}000$ | $O(n)$

$2 * (1{,}000^2) \Rightarrow 2{,}000{,}000$ | $O(n^2)$

(40 times bigger)

Remember, we're describing the rate of growth. Constants and coefficients will always add the same number of steps to an algorithm, regardless of the input size - so we drop them to create an approximation.

Time Complexity of Built-In Array Methods

- push: *O(1)*. Adds an element to the end of the array - doesn't need to access any other elements. The length of the array does not affect how many steps it takes to complete.

- pop: O(1) Removes the last element of the array, again without accessing any other elements.

- unshift: O*(n)*. Adds an element to the beginning of the array, at index 0. This means that all existing elements must be moved up one index - we're iterating through the whole array every time this method runs.

Built-In Methods (continued)

- slice: O(n). Makes a copy of a subset of an array, between the indexes passed in - in order to do this, it must iterate through that subset and copy each item individually.

- sort: *O(n log n)* (generally). However, different JavaScript engines implement the sort method using different algorithms, so its time complexity may vary.

Summary

- An algorithm is a set of instructions that provides an answer to a problem.
- Time-complexity describes the rate at which the number of computations grows as the input grows.
- Big O Notation provides a way to represent time-complexity in a meaningful, but approximate way.
- Big O notation describes the worst case scenario.

Further Reading

Big O Notation (Interview Cake)

Big O Notation (Wikipedia)