Runtime Analysis

CSC236 - Fall 2022

October 3rd / 4th

Learning Objectives

01.

Review and understand the big O, big Theta and big Omega complexity classes

02.

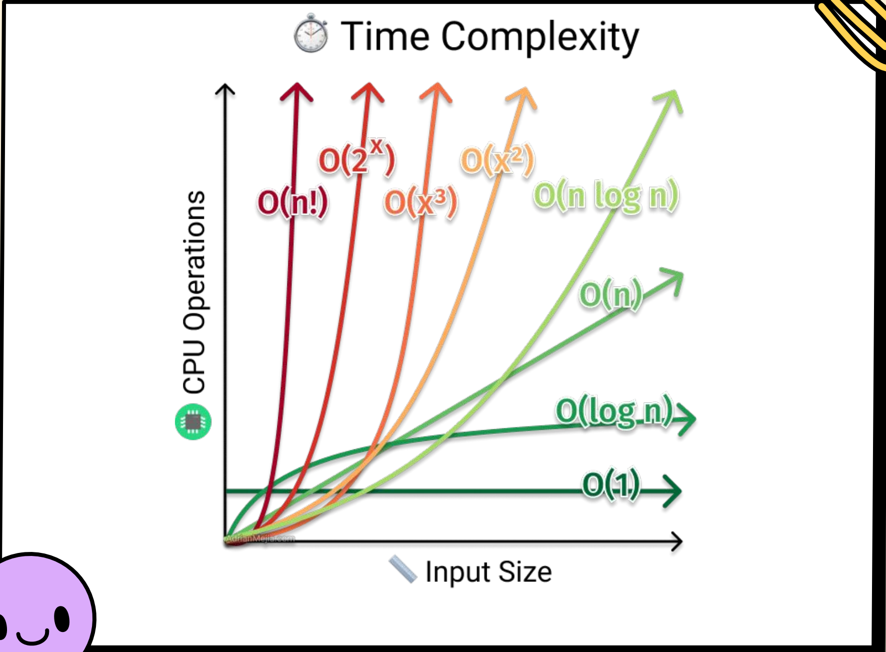
Practice finding the big O complexity for mathematical expressions

03.

Practice writing proofs about complexity

04.

Analyze code to compute exactly how many steps a program takes

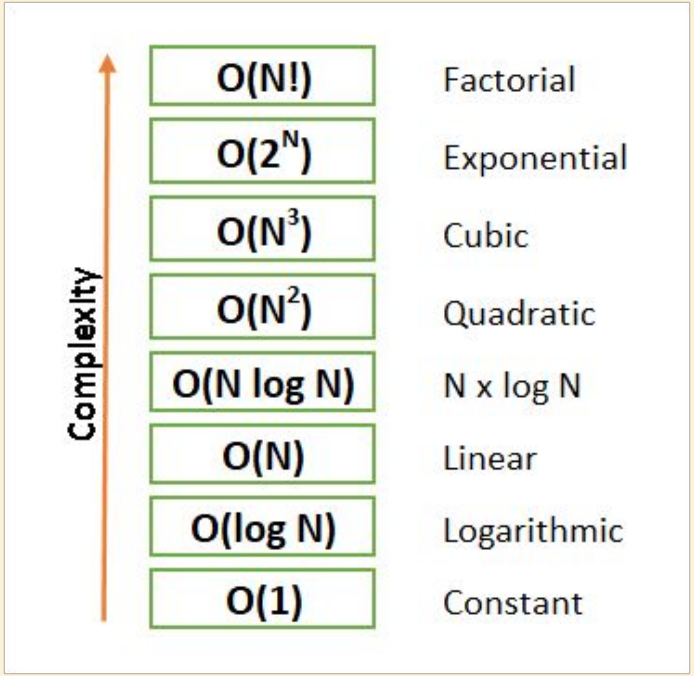


Runtime Analysis

The running time of a piece of code is **proportional to a function of the number of steps** carried out by the computer running the code

Source: https://thecodingbay.com/wp-content/uploads/2022/04/Complexities-Graph1.png

Asymptotic Bounding



How the program behaves as input gets large!!!

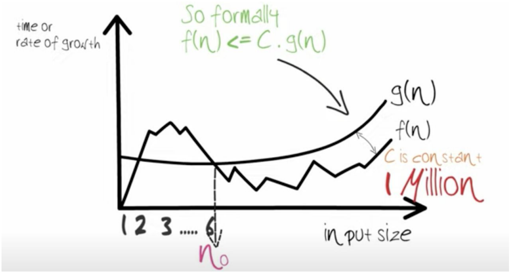
Source: https://miro.medium.com/max/742/1*WBYUz6Lh2Z21DQnEk-MWFQ.png

Asymptotic Bounding

Big-Oh O

Gives **upper bound** to the expression

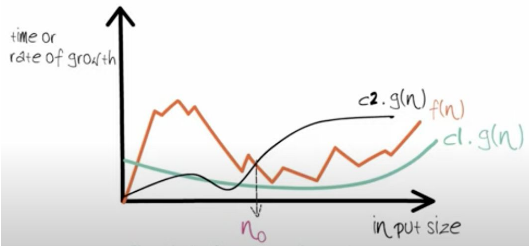
$$f(n) \in O(g(n)) \text{ if: }$$
$$f(n) \leq c \cdot g(n), \quad \forall n \geq n_0$$



Big Theta Θ

upper bound = lower bound

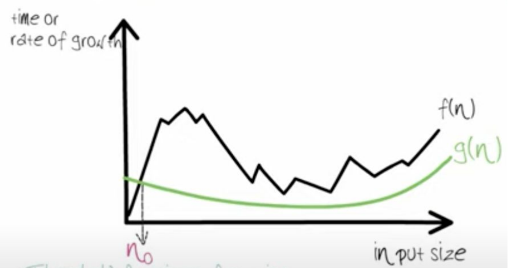
$$f(n) \in \Theta(g(n)) \text{ if: }$$
$$f(n) \in O(g(n)) \text{ and } f(n) \in \Omega(g(n))$$



Big Omega Ω

Gives **lower bound** to the expression

$$f(n) \in \Omega(g(n)) \text{ if: }$$
$$f(n) \geq c \cdot g(n), \quad \forall n \geq n_0$$



The images used in this slide are from this very useful video: https://www.youtube.com/watch?v=bxgTDN9c6rg

Big O for Mathematical Expressions

Expression $f(n)$	$O(f(n))$
$3 \cdot 2^n$	
$\frac{2n^4+1}{n^3+2n-1}$	
$(n^5+7)(n^5-7)$	
$\frac{n^4+n \cdot log_2 n}{n^1+1}$	
$\frac{n \cdot log_2 n}{n-5}$	
$8+\frac{1}{n^2}$	
2^{3n+1}	
$n!$	
$\frac{5 \cdot log_2 n+1}{1+n \cdot log_2 3n}$	

For each of the mathematical expressions in the table, provide a big O upper bound

Worksheet

Questions
2 - 4