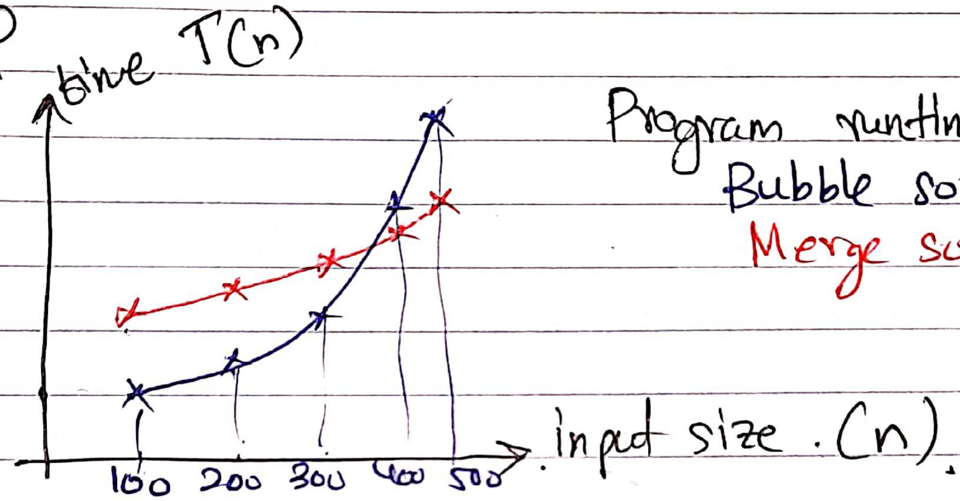
Time complexity

01/Sept/2021
CMSC 351
Review session
— Gihan —

Big-O • Other.

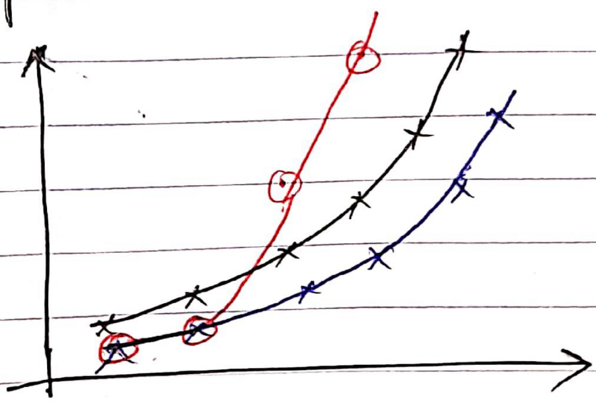
1. why?



Program runtime for
Bubble sort
Merge sort

What is the runtime T(n) for a particular input size n?

1. Experimental measure.



Bubble sort.
· 2GHz processor
3GH processor
Low ram

∴ experiments are not good enough.

2. Theoretical analysis for run $\boxed{\text{time}}$?
We measure time in terms of unit operations done inside an "ideal" computer.

Weak | strong statements

eg:- Everyone in this class is at least 2 feet tall — (A)

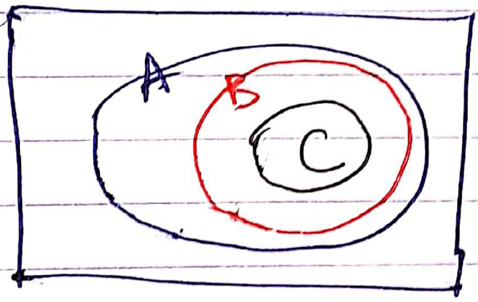
Everyone in this class is at least 3 feet tall — (B)

Everyone in this class is more than 3 feet tall — (C).

$\boxed{A}$ is a weak statement compared to B.

$\boxed{B}$ is a strong statement compared to A.

$\boxed{C}$ is stronger than B and A.



Back to time complexity

$f : \mathbb{N} \longrightarrow \mathbb{N}$ — natural numbers $\{0, 1, 2 \ldots\}$

$g : \mathbb{N} \longrightarrow \mathbb{N}$

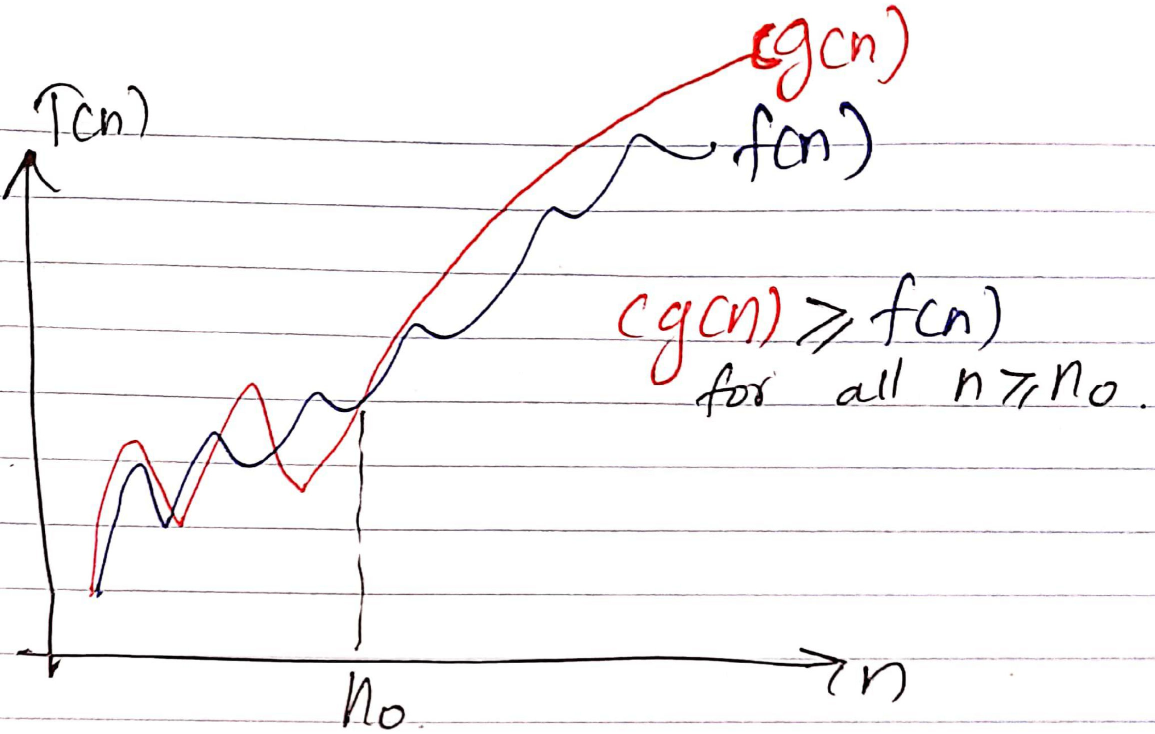
$f(n) = O(g(n))$ if there exists a c, n_0 such that for all $n \geq n_0$

$$f(n) \leq \underline{c\,g(n)}$$

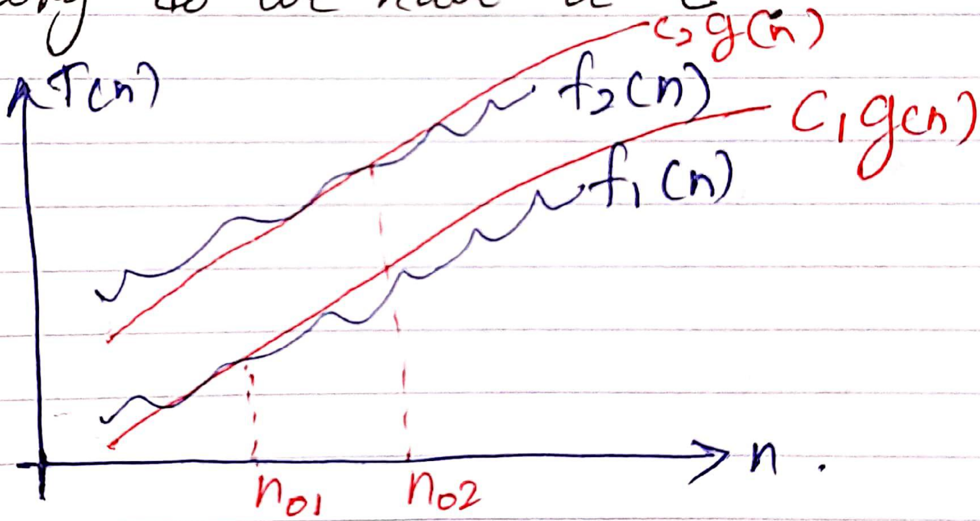
Upper bound of $f(n)$

$c \in \mathbb{R}^+$

$n \in \mathbb{N}$



Q 1: why do we have a c?



$f_1(n)$ and $f_2(n)$ have similar shapes. We would like to give the same Big-O for them.

Q 2: why do we think only about $n \geq n_0$?

We focus on how runtime grows for large n. We are worried about algorithms slowing down for large inputs.

Other definitions

$f, g : \mathbb{N} \to \mathbb{N}$, $c_i \in \mathbb{R}^+$, $n_j \geq n_0$, $n \in \mathbb{N}$

Big O, Ω, Θ

$$f(n) \leq c_1 g(n) \Rightarrow \cancel{\Theta}\ f(n) = O(g(n))$$

$$f(n) \geq c_2 g(n) \Rightarrow f(n) = \Omega(g(n))$$

$$f(n) = O(g(n)) \text{ and } f(n) = \Omega(g) \Rightarrow f(n) = \Theta(g(n))$$

Small o, ω

$$\lim_{n \to \infty} \frac{f(n)}{g(n)} = 0 \Rightarrow f(n) = o(g(n))$$

$$\lim_{n \to \infty} \frac{g(n)}{f(n)} = 0 \Rightarrow f(n) = \omega(g(n))$$

$$\lim_{n \to \infty} f(n) < g(n) \Rightarrow f(n) = o(g(n))$$

$$\lim_{n \to \infty} f(n) > g(n) \Rightarrow f(n) = \omega(g(n))$$

Some important points

→ Small o, ω are stronger statements.

$$f(n) = o(g(n)) \Rightarrow f(n) = O(g(n))$$

$$f(n) = \omega(g(n)) \Rightarrow f(n) = \Omega(g(n))$$

How to rigorously prove Big O / Ω?

Approach 1:

→ Start with arbitrary c and n_0.

→ Show $cg(n) \geq f(n)$ for all $n \geq n_0$.

Approach 2:

→ Assume $cg(n) \geq f(n)$ for all $n \geq n_0$

→ Show the existence of at least one value each for c and n_0

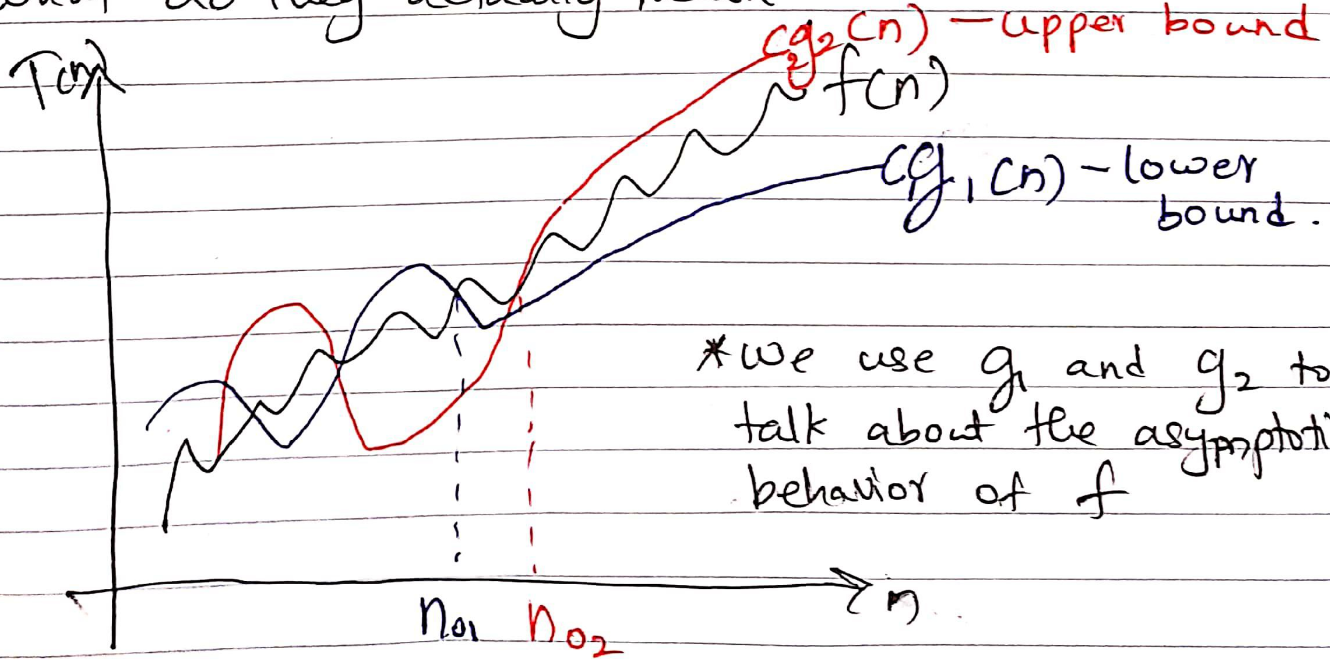
→ You can show there are multiple c, n pairs as well (even infinitely many).

How to rigorously prove Big Θ?

→ Rigorously prove O, Ω

→ That will imply Θ.

What do they actually mean?



*We use g_1 and g_2 to talk about the asymptotic behavior of f