Aula 1 OBI

Três pilares da computação: Autômatos, Teoria da Complexidade e Teoria da Computabilidade.

Ordenação é um problema fácil.
Escalonamento é um problema difícil, com entrada 1000 ordenar leva minutos ou segundos, mas escalonar leva séculos.

O que faz um problema ser fácil ou difícil? Ainda não sabemos dizer, mas existe uma classificação, análoga à tabela periódica.

P = NP? Problema em aberto envolvendo a nossa "tabela periódica".

P é a classe dos problemas resolvidos em tempo polinomial. **[Polinomiais]**
NP é a classe dos problemas resolvidos em tempo polinomial em **Máquina Não-Determinística de Turing**. (E possuem verificador de tempo polinomial) [**Polinomiais em Máquina Não-Determinística de Turing]**
NPC é a classe dos problemas em que um problema pode ser reduzido a outro. É a classe dos nossos problemas difíceis. **[NP-Completo]**

A classe **NPC** possui uma implicação forte. Se um problema for resolvido todos os problemas da classe são resolvidos (pela questão da redutibilidade dos problemas da classe). Mas isso também torna improvável de alguém conseguir a solução ótima de algum problema da classe, visto que, se alguém trabalhou em um problema, então trabalhou em todos.

$B \in$ NPC se $B \in$ NP e todo $A \in$ NPC é redutível em tempo polinomial a B.
A grande questão é: Se $B \in$ NPC e $B \in$ P, então P = NP

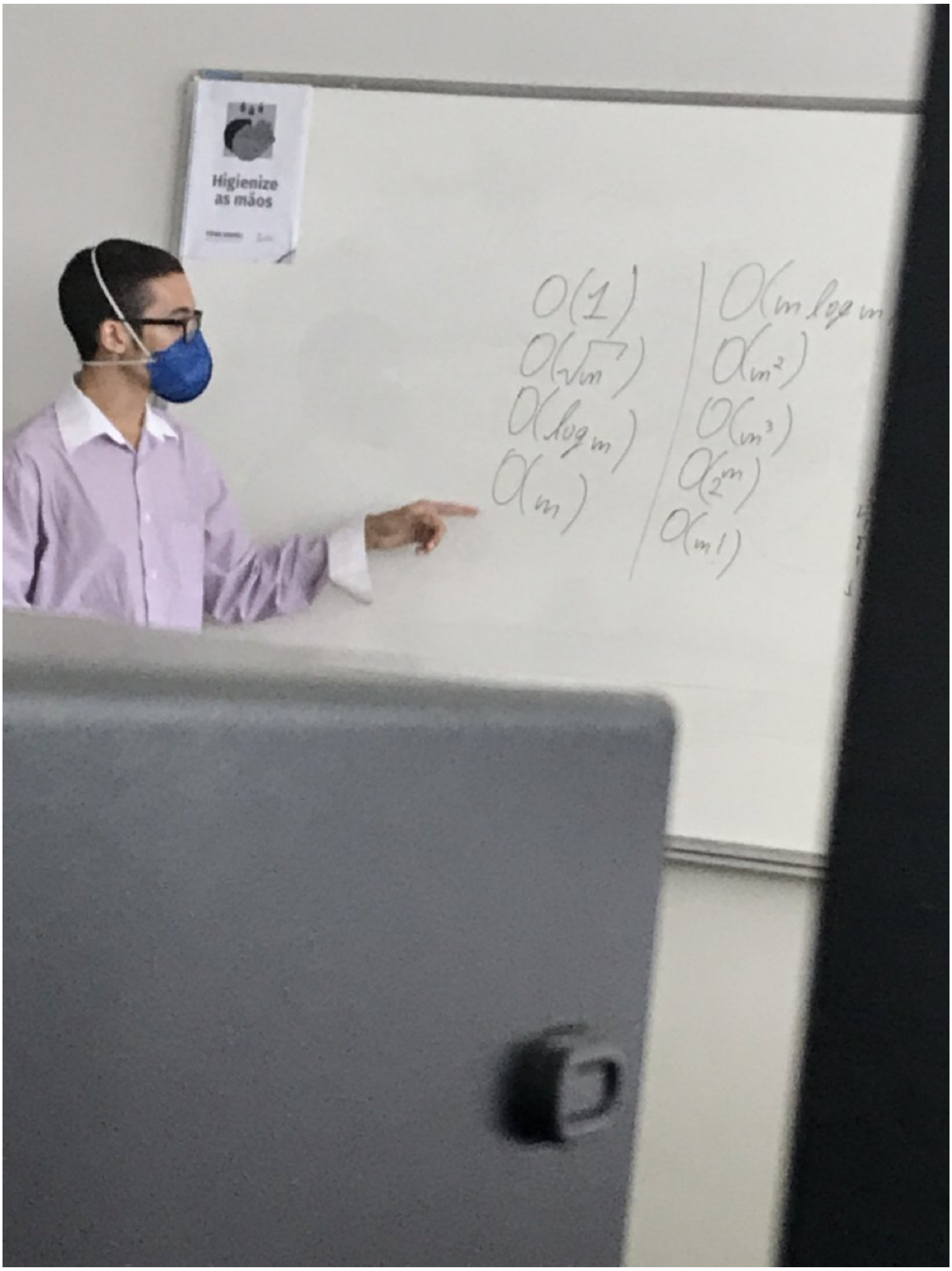
Adiante faremos uso de logaritmos, então uma breve explicação.

$$\log_b n = \frac{\log_2 n}{\log_2 b}$$

A notação assintótica não especifica base, já que desconsidera fatores constantes. Então usaremos a seguinte forma:
$\log_b n = \log n$
Para qualquer base teremos o logaritmo de base 2, então escreveremos só log n.

Tempos:
n^c é polinomial.
2^n é exponencial.

Notação Big-O (O) e Little-O (o):
O(n) especifica que a função cresce no máximo até n.
o(n) especifica que a função cresce menos que n.



1 GHZ

$O(n)$ 10^9

$O(n^2)$ 10^4

$O(n^3)$ 10^3

$O(2^n) \simeq 20$

$O(1)$

$O(\sqrt{n})$

$O(\log n)$

$O(n)$

$O(n \log n)$

$O(n^2)$

$O(n^3)$

$O(2^n)$

$O(n!)$