Aula dia 02/02/2022

Revisão de Complexidade

Classificações de algoritmos/problema quanto à eficiência

$O(g(n))$ dog(n)*

$\Theta(g(n))$

$\Omega(g(n))$ ω(g(n))*

$= \{ f(n)$: existem constantes $c_1$, $c_2$ e $n_0$ tal que: $0 \leq c_1 g(n) \leq f(n) \leq c_2 \cdot g(n)$ para todo $n \leq n_0$

f g Analogia

$a < b$

$a \leq b$

$a = b$

$a > b$

$a \geq b$

$\Leftarrow$

$f(n) = o(g(n))$ cresce mais devagar

$f(n) = O(g(n))$ não cresce mais rápido

$f(n) = \Theta(g(n))$ não cresce mais devagar nem mais rápido

$f(n) = \omega(g(n))$ cresce mais rápido

$f(n) = \Omega(g(n))$ não cresce mais devagar

| | Vetor | Lista encadeada | Pilha Fila | hash | Árvore Binária de busca | especial |
|---|---|---|---|---|---|---|
| Busca | $O(n)$ | $O(n)$ | — | $O(n)$ | $O(\log n)$ * | |
| Busca ordenada | $O(\log n)$ | $O(n)$ | — | — | — | |
| Insuys | $O(1)$ | $O(n)$ | $O(n)$ | $O(1)$ $O(n)$ | $O(\log n)$ * | |
| Remuys | $O(n)$ | $O(n)$ | $O(n)$ | $O(1)$ $O(n)$ | $O(\log(n)$ * | |
| Ordenação | $O(n\log n)$ | $O(n\log n)$ | — | — | — | * $O(n)$ |

A notação $O$ dá um limite superior

$\Omega$ dá um limite inferior

$\Theta$ dá uma igualdade

# Classificação de Problemas (quanto ao tipo saída)

PREFERÍVEL

**Problemas de Decisão**
Problemas cuja saída é SIM ou NÃO

**Problema de Localização**
Problemas cuja saída é uma estrutura qualquer

**Problemas de Otimização**
Problemas cuja saída é o valor ótimo de alguma propriedade ou estrutura

# Certificado:

Um certificado para algum problema é uma prova da resposta para aquele problema.

Ex: 100 é primo? Não

Por quê? $25 \times 4 = 100$

O certificado para a resposta Não: fatores primos

37 é primo? Sim

Por quê? Ele não é divisível por 2, 3, 4, 5, 6 — Certificado

A prova final de um algoritmo correto é um certificado!

# Quanto à tratabilidade

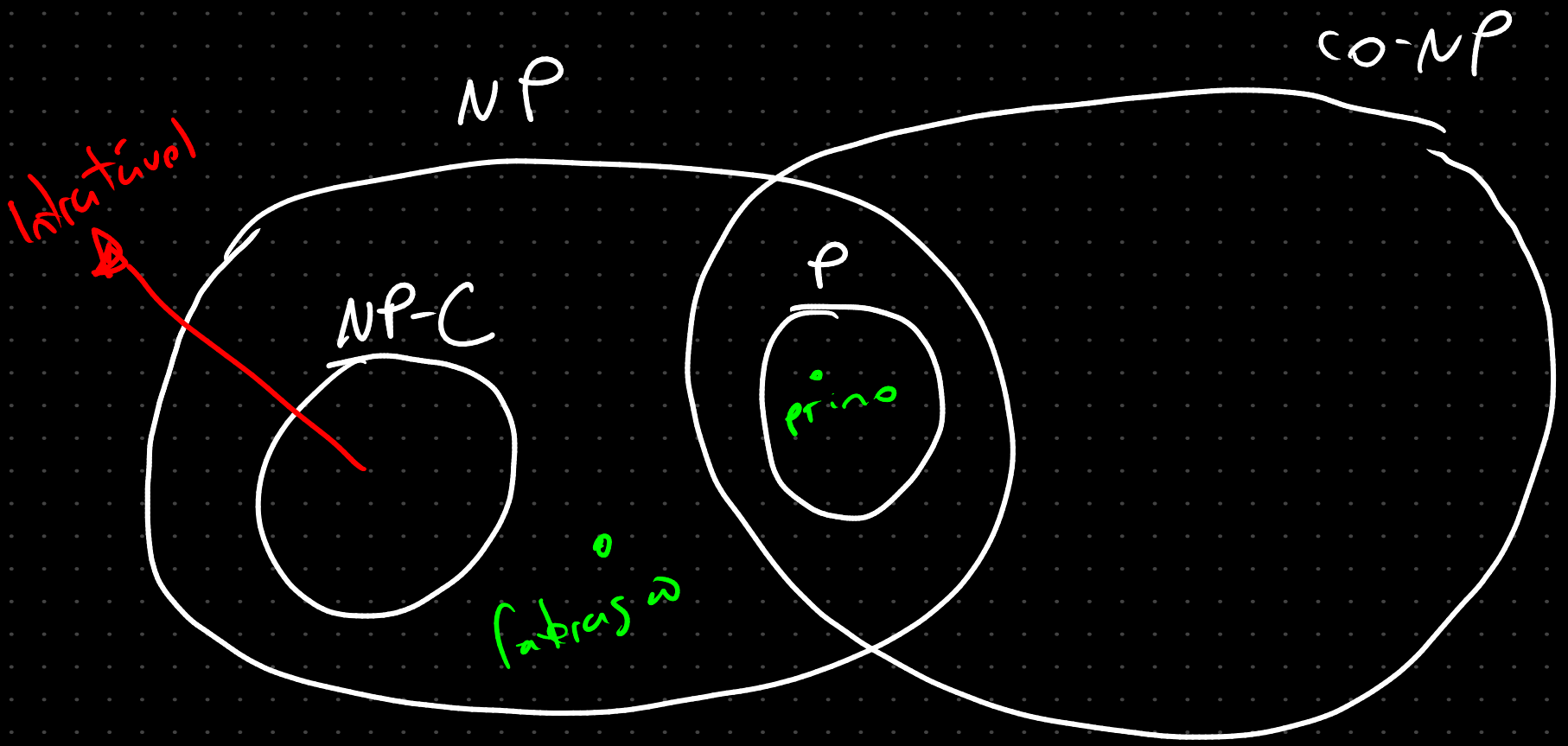
## Classe P (Eficientes)

Um problema pertence à classe P se existem um algoritmo que o resolve com complexidade temporal polinomial $O(n^k)$ para alguma constante $k$

## Classe NP (Classe co-NP)

Um problema de decisão pertence à classe NP se existe um certificado para a resposta SIM/NÃO que pode ser verificado por um algoritmo polinomial

Se existe um algoritmo polinomial, ele é um certificado para o sim e o não logo $P \subset NP$ e $P \subset co\text{-}NP$




Um problema NP-completo é um problema em NP tal que se for encontrado um algoritmo polinomial para ele, as classes P e NP colapsam de forma que P = NP.

Porém não se conhecem nenhum algoritmo desse tipo

$$O(1) \subset O(\log n) \subset O(\log^2 n) \subset O(\sqrt{n}) \subset O(n) \subset$$
$$\subset O(n \log n) \subset O(n^2) \subset O(n^k) \subset O(n^k \log^2 n) \subset$$

subconjunto — permutação — arranjo

$$O(2^n) \subset O(n!) \subset O(n^n) \ldots$$

NP / P

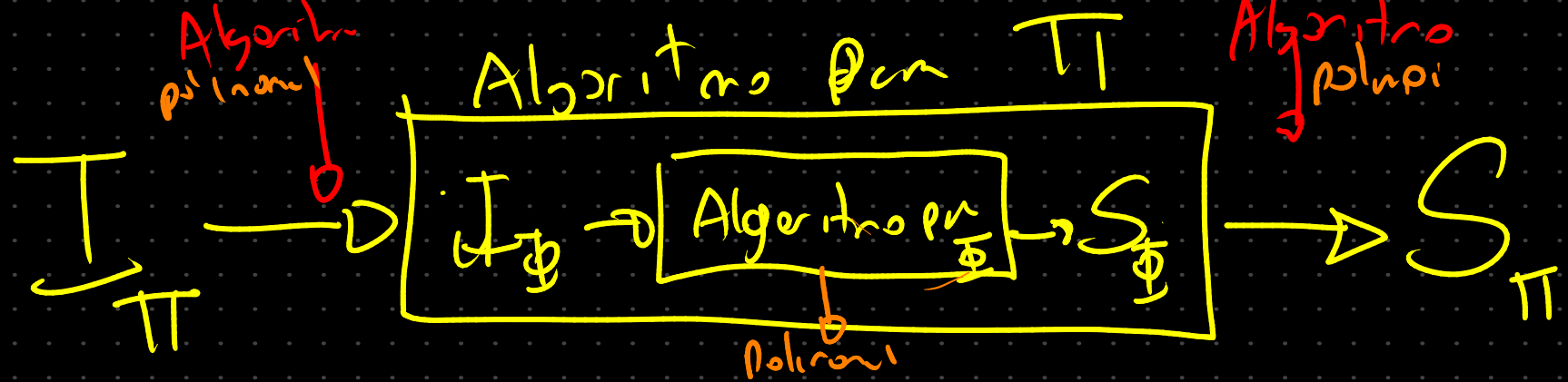
P

Para todos problemas existe uma abordagem de força bruta

# Grande Sacada

Redução: Seja $\Pi$ um problema $I_\Pi$ uma instância para $\Pi$. Da mesma forma $\Phi$ e $I_\Phi$ $S_\Pi$ $S_\Phi$

Uma Redução é uma transformação de $I_\Pi$ em $I_\Phi$ e uma forma de transformar $S_\Phi$ em $S_\Pi$




Analogia Multiplicação via Logaritmo

Transformada de Laplace

Transformando um problema de otimização em um problema de decisão

O: Encontrar o maior elemento numa lista L

D: A lista L possui um elemento maior que k

k é um parâmetro