Técnicas de Programação e Análise de Algoritmos

Prof. Dr. Lucas Rodrigues Costa

Aula 8: Análise de Algoritmos

lucas.costa@idp.edu.br

@lucasrodri

www.linkedin.com/in/lucas-rodri

OBJETIVOS

- → Compreender o conceito de notação assintótica
- → Conhecer os diferentes tipos de análise assintótica
- → Conhecendo algumas classes de problemas

RECORDANDO...

- Vimos na aula passada
  - O conceito de algoritmo eficiente
  - O conceito de complexidade computacional
  - Abordagens empírica, matemática e assintótica

Tipos de Análise Assintótica

Notação Big-O

- A notação big-O é a forma mais conhecida e utilizada de análise assintótica
  - Complexidade do nosso algoritmo no pior caso
    - Seja de tempo ou de espaço
  - É o caso mais fácil de se identificar
    - Limite superior sobre o tempo de execução do algoritmo
    - Para diversos algoritmos o pior caso ocorre com frequência

Diferentes tipos de análise assintótica

- No entanto, existem várias formas de análise assintótica
  - Notação big-Omega, Ω
  - Notação big-O, O
  - Notação big-Theta, Θ
  - Notação little-o, o
  - Notação little-omega, ω
- A seguir, são matematicamente descritas outras formas de análise assintótica.

Diferentes tipos de análise assintótica

- Notação big-Omega, Ω
  - Descreve o **limite assintótico inferior**
  - É utilizada para analisar o **melhor caso** do algoritmo
  - A notação **$\Omega(n^2)$** nos diz que o custo do algoritmo é, assintoticamente, maior ou igual a **n^2**
    - Ou seja, o custo do algoritmo original é no mínimo tão ruim quanto **n^2**

Diferentes tipos de análise assintótica

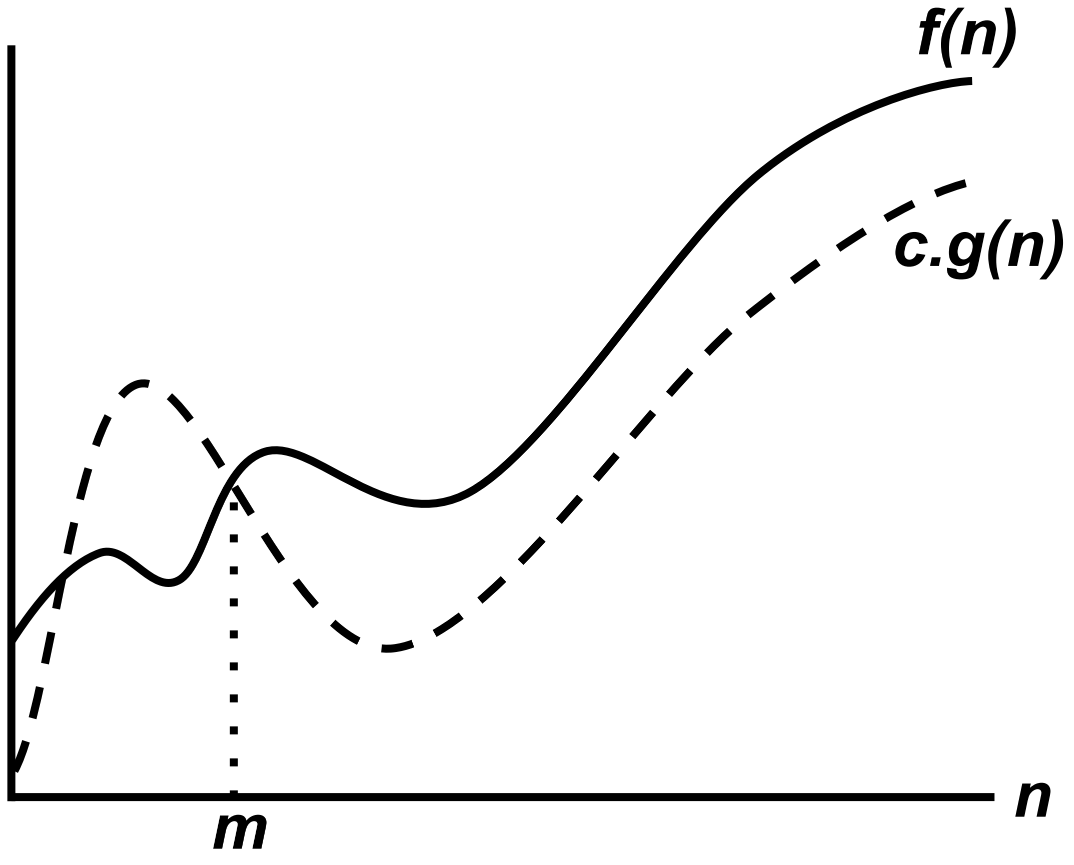
- Notação big-Omega, Ω
  - Matematicamente, a notação Ω é assim definida
    - Uma função custo $f(n)$ é $\Omega(g(n))$ se existem duas constantes positivas c e m tais que
    - Para $n \geq m$, temos $f(n) \geq c.g(n)$
    - Confuso?

Diferentes tipos de análise assintótica

- Notação big-Omega, **Ω**
  - Em outras palavras, para todos os valores de **n** à direita de **m**, o resultado da função custo **f(n)** é sempre **maior ou igual** ao valor da função usada na notação **Ω, g(n)**, multiplicada por uma constante **c**

Diferentes tipos de análise assintótica

- Notação big-Omega, Ω



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

Diferentes tipos de análise assintótica

- Exemplo: mostrar que a função custo é:

$$\mathbf{f(n) = 3n^2 + n \text{ é } \Omega(n)}$$

  - Temos que encontrar constantes c e m tais que:
  - $3n^2 + n \geq cn$
  - Dividindo por $\mathbf{n^2}$, temos:
  - $3 + 1/n \geq c/n$
  - Considerando **c=4** e **n>0**, temos que $\mathbf{f(n) = 3n^2 + n}$ **é** $\mathbf{\Omega(n)}$

Diferentes tipos de análise assintótica

- Notação big-O, **O**
  - Descreve o **limite assintótico superior**
  - É utilizada para analisar o **pior caso** do algoritmo
  - A notação **O(n^2)** nos diz que o custo do algoritmo é, assintoticamente, menor ou igual a **n^2**
    - Ou seja, o custo do algoritmo original é no máximo tão ruim quanto **n^2**

Diferentes tipos de análise assintótica

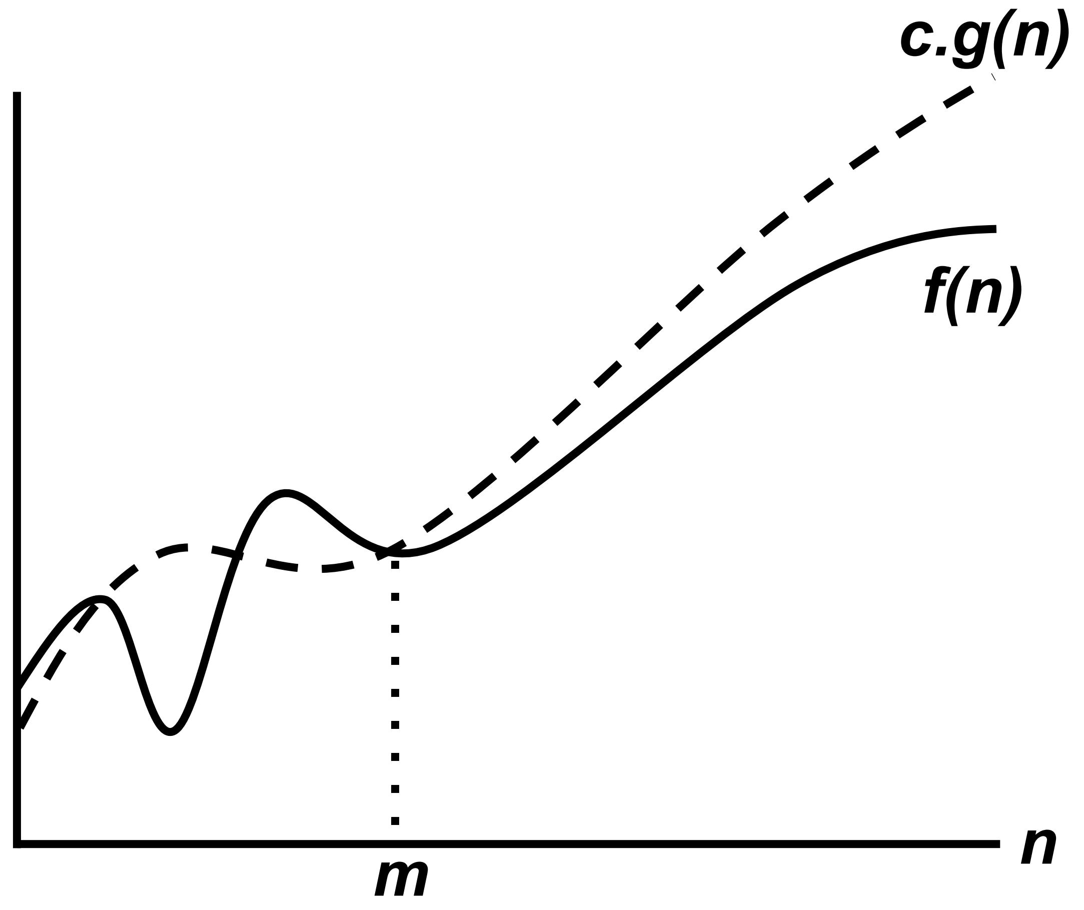
- Notação big-O, **O**
  - Matematicamente, a notação **O** é assim definida
    - Uma função custo **f(n)** é **O(g(n))** se existem duas constantes positivas **c** e **m** tais que
    - Para **n ≥ m**, temos **f(n) ≤ c.g(n)**
    - Confuso?

Diferentes tipos de análise assintótica

- Notação big-O, **O**
  - Em outras palavras, para todos os valores de **n** à direita de **m**, o resultado da função custo **f(n)** é sempre **menor ou igual** ao valor da função usada na notação **O, g(n)**, multiplicada por uma constante **c**.

Diferentes tipos de análise assintótica

- Notação big-O, **O**



f(n) = O(g(n))

Diferentes tipos de análise assintótica

- Exemplo: mostrar que a função custo é:

$$\mathbf{f(n) = 2n^2 + 10 \text{ é } O(n^3)}$$

  - Temos que encontrar constantes c e m tais que:
  - $2n^2 + 10 \leq cn^3$
  - Dividindo por $\mathbf{n^3}$, temos:
  - $2/n + 10/n^3 \leq c$
  - Considerando **c=1** e **n≥4**, temos que $\mathbf{f(n) = 2n^2 + 10}$ **é** $\mathbf{O(n^3)}$
    - Dá para melhorar essa análise!

idp

Diferentes tipos de análise assintótica

- Exemplo: mostrar que a função custo é:

  $$\mathbf{f(n) = 2n^2 + 10 \text{ é } O(n^2)}$$

  - Temos que encontrar constantes c e m tais que:
  - $2n^2 + 10 \leq cn^2$
  - Dividindo por $\mathbf{n^2}$, temos:
  - $2 + 10/n^2 \leq c$
  - Considerando **c=12** e **n>0**, temos que $\mathbf{f(n) = 2n^2 + 10 \text{ é } O(n^2)}$

Diferentes tipos de análise assintótica

- Exemplo: mostrar que a função custo é:

  f(n) = 4n + 7 é O(n)

  - Temos que encontrar constantes **c** e **m** tais que:
  - $4n + 7 \le cn$
  - Dividindo por **n**, temos:
  - $4 + 7/n \le c$
  - Considerando **c=8** e **n>1**, temos que **f(n) = 4n + 7 é O(n)**

Diferentes tipos de análise assintótica

- Exemplo: mostrar que a função custo é:

  $f(n) = n^2$ não é **$O(n)$**

  - Temos que encontrar constantes **c** e **m** tais que:
  - $n^2 \leq cn$
  - Dividindo por **n**, temos:
  - $n \leq c$
  - A desigualdade é inválida!
    - O valor de **n** está limitado pela constante **c**
    - A análise assintótica não é possível (entrada tendendo ao infinito)

Diferentes tipos de análise assintótica

- Notação big-O, **O**
  - Essa notação possui algumas operações
  - A mais importante é a **regra da soma**
    - Permite a análise da complexidade de diferentes algoritmos em sequência
  - Definição
    - Se dois algoritmos são executados em sequência, a complexidade será dada pela complexidade do maior deles
    - **$O(f(n)) + O(g(n)) = O(max(f(n),g(n)))$**

Diferentes tipos de análise assintótica

- Notação big-O, **O**
  - Exemplo da **regra da soma**. Se temos
    - Dois algoritmos cujos tempos de execução são **$O(n)$** e **$O(n^2)$**, a execução deles em sequência será **$O(max(n,n^2))$** que é **$O(n^2)$**

    - Dois algoritmos cujos tempos de execução são **$O(n)$** e **$O(n \log n)$**, a execução deles em sequência será **$O(max(n,n \log n))$** que é **$O(n \log n)$**

idp

Diferentes tipos de análise assintótica

- Notação big-Theta, **Θ**
    - Descreve o **limite assintótico firme**
    - É utilizada para analisar o **limite inferior** e **superior** do algoritmo
    - A notação **$\Theta(n^2)$** nos diz que o custo do algoritmo é, assintoticamente, igual a **n^2**
        - Ou seja, o custo do algoritmo original é **n^2** dentro de um fator constante **acima** e **abaixo**

Diferentes tipos de análise assintótica

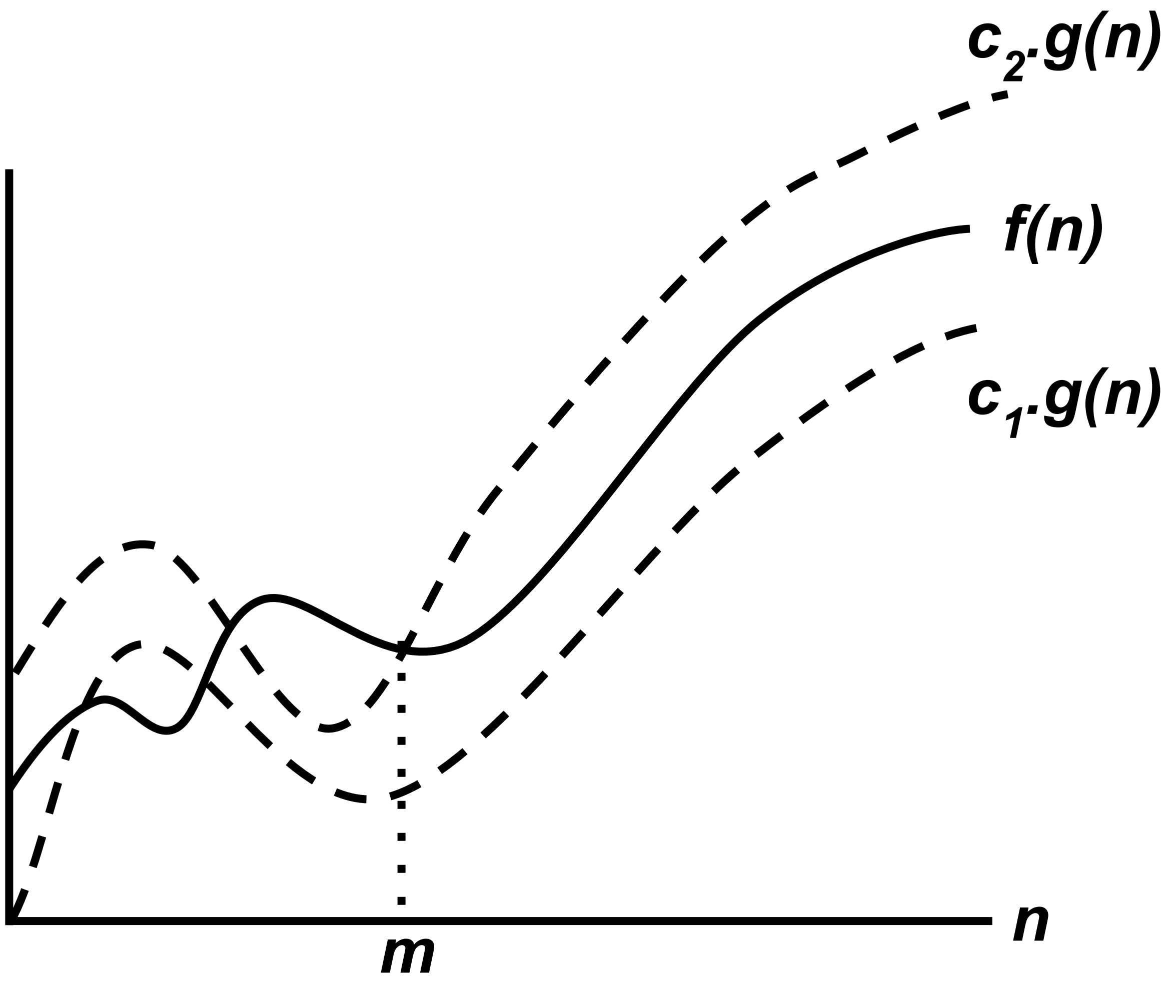
- Notação big-Theta, **Θ**
  - Matematicamente, a notação **Θ** é assim definida
    - Uma função custo **f(n)** é **Θ(g(n))** se existem três constantes positivas $\mathbf{c_1}$, $\mathbf{c_2}$ e **m** tais que
    - Para $\mathbf{n \geq m}$, temos $\mathbf{c_1.g(n) \leq f(n) \leq c_2.g(n)}$
    - Confuso?

Diferentes tipos de análise assintótica

- Notação big-Theta, **Θ**
  - Em outras palavras, para todos os valores de **n** à direita de **m**, o resultado da função custo **f(n)** é sempre **igual** ao valor da função usada na notação **Θ, g(n)**, quando está é multiplicada por constantes c_1 e c_2

Diferentes tipos de análise assintótica

➔ Notação big-Theta, Θ



$f(n) = \Theta(g(n))$

Diferentes tipos de análise assintótica

- Exemplo: mostrar que a função custo

$$f(n) = 1/2\ n^2 - 3n \text{ é } \mathbf{\Theta(n^2)}$$

  - Temos que encontrar constantes c1 e c2 e m tais que:
  - $c_1 n^2 \leq 1/2\ n^2 - 3n \leq c_2 n^2$
  - Dividindo por n^2, temos
  - $c_1 \leq 1/2 - 3/n \leq c_2$

Diferentes tipos de análise assintótica

- Exemplo: mostrar que a função custo

$$f(n) = 1/2\ n^2 - 3n \text{ é } \mathbf{\Theta(n^2)}$$

  - $c_1 \leq 1/2 - 3/n \leq c_2$
  - A desigualdade do lado direito é válida para $\mathbf{n \geq 1}$ escolhendo $\mathbf{c_2 \geq 1/2}$
  - A desigualdade do lado esquerdo é válida para $\mathbf{n \geq 7}$ escolhendo $\mathbf{c_1 \geq 1/14}$
  - Assim, para $\mathbf{c_1 \geq 1/14, c_2 \geq}$ ½ e $\mathbf{n \geq 7}$, $f(n) = 1/2\ n^2 - 3n$ é $\mathbf{\Theta(n^2)}$

Diferentes tipos de análise assintótica

- Exemplo: mostrar que a função custo

  $f(n) = 6n^3$ não é $\mathbf{\Theta(n^2)}$

  - Temos que encontrar constantes c1 e c2 e m tais que:
  - $c_1n^2 \leq 6n^3 \leq c_2n^2$
  - Dividindo por n^2, temos
  - $c_1 \leq 6n \leq c_2$

Diferentes tipos de análise assintótica

- Exemplo: mostrar que a função custo

  $f(n) = 6n^3$ não é $\mathbf{\Theta(n^2)}$

  - $c_1n^2 \leq 6n^3 \leq c_2n^2$
  - A desigualdade do lado direito é inválida!
  - $n \leq c_2/6$
  - O valor de **n** está limitado pela constante c_2
  - A análise assintótica não é possível (entrada tendendo ao infinito)

Diferentes tipos de análise assintótica

- Notação little-o, **o**, e little-omega, **ω**
  - Parecidas com as notações big-O e big-Omega
  - As notações big-O e big-Omega possuem uma relação de menor ou igual e maior ou igual
  - As notações little-o e little-omega possuem uma relação de menor e maior

Diferentes tipos de análise assintótica

- Notação little-o, **o**, e little-omega, **ω**
  - Ou seja, essas notações não representam limites próximos da função
  - Elas representam limites estritamente
  - **superiores**: sempre maior
  - **inferiores**: sempre menor

Classe de Problemas

Classe de Problemas

- A seguir, são apresentadas algumas classes de complexidade de problemas comumente usadas
  - O(1): ordem constante
    - As instruções são executadas um número fixo de vezes. Não depende do tamanho dos dados de entrada
  - O(log n): ordem logarítmica
    - Típica de algoritmos que resolvem um problema transformando-o em problemas menores
  - O(n): ordem linear
    - Em geral, uma certa quantidade de operações é realizada sobre cada um dos elementos de entrada

idp

$O(1) < O(\log n) < O(n) < O(n \log n) < O(n^2) < O(n^3) < O(2^n) < O(n!)$

Classe de Problemas

- Mais classes de problemas
  - $O(n \log n)$: ordem log linear
    - Típica de algoritmos que trabalham com particionamento dos dados. Esses algoritmos resolvem um problema transformando-o em problemas menores, que são resolvidos de forma independente e depois unidos
  - $O(n^2)$: ordem quadrática
    - Normalmente ocorre quando os dados são processados aos pares. Uma característica deste tipo de algoritmos é a presença de um aninhamento de dois comandos de repetição

$O(1) < O(\log n) < O(n) < O(n \log n) < O(n^2) < O(n^3) < O(2^n) < O(n!)$

Classe de Problemas

- Mais classes de problemas
  - $O(n^3)$: ordem cúbica
    - É caracterizado pela presença de três estruturas de repetição aninhadas
  - $O(2^n)$: ordem exponencial
    - Geralmente ocorre quando se usa uma solução de **força bruta**. Não são úteis do ponto de vista prático
  - $O(n!)$: ordem fatorial
    - Geralmente ocorre quando se usa uma solução de **força bruta**. Não são úteis do ponto de vista prático. Possui um comportamento muito pior que o exponencial

$O(1) < O(\log n) < O(n) < O(n \log n) < O(n^2) < O(n^3) < O(2^n) < O(n!)$

Classe de Problemas

- Comparação no tempo de execução
  - Computador executa 1 milhão de operações por segundo

$f(n)$	$n = 10$	$n = 20$	$n = 30$	$n = 50$	$n = 100$
n	1,0E-05 segundos	2,0E-05 segundos	4,0E-05 segundos	5,0E-05 segundos	6,0E-05 segundos
$n \log n$	3,3E-05 segundos	8,6E-05 segundos	2,1E-04 segundos	2,8E-04 segundos	3,5E-04 segundos
n^2	1,0E-04 segundos	4,0E-04 segundos	1,6E-03 segundos	2,5E-03 segundos	3,6E-03 segundos
n^3	1,0E-03 segundos	8,0E-03 segundos	6,4E-02 segundos	0,13 segundos	0,22 segundos
2^n	1,0E-03 segundos	1,0 segundo	2,8 dias	35,7 anos	365,6 séculos
3^n	5,9E-02 segundos	58,1 minutos	3855,2 séculos	2,3E+08 séculos	1,3E+13 séculos

Classe de Problemas

- Cuidado
  - Na análise assintótica as constantes de multiplicação são consideradas irrelevantes e descartadas
    - Porém, elas podem ser relevantes na prática, principalmente se o tamanho da entrada é pequeno
  - Exemplo: qual função tem menor custo?
    - **$f(n) = 10^{100} * n$**
    - **$g(n) = 10n \log n$**

Classe de Problemas

- Cuidado
  - Análise assintótica: o primeiro é mais eficiente
    - f(n) = 10^{100} * n tem complexidade O(n)
    - g(n) = 10n log n tem complexidade O(n log n)
  - No entanto, 10^{100} é um número muito grande
    - Neste caso, 10n log n > 10100 * n apenas para

$$n > 2^{10^{99}}$$

    - Para qualquer valor menor de n o algoritmo de complexidade O(n log n) será melhor

Perguntas?

Exercícios de Fixação

idp

Exercícios

1. O que significa dizer que uma função $g(n)$ é $O(f(n))$?
2. O que significa dizer que uma função $g(n)$ é $\Theta(f(n))$?
3. O que significa dizer que uma função $g(n)$ é $\Omega(f(n))$?
4. Suponha um algoritmo A e um algoritmo B com funções de complexidade de tempo $a(n) = n^2 - n + 549$ e $b(n) = 49n + 49$, respectivamente. Determine quais são os valores de n pertencentes ao conjunto dos números naturais para os quais A leva menos tempo para executar do que B.

Justifique suas respostas!

Referências

- BACKES, A. Ricardo. Algoritmos e estruturas de dados em linguagem C. 1. ed. Rio de Janeiro: LTC, 2023.
- Prof. Dr. André Backes; Estrutura de Dados 2; 2012

idp

INSTITUTO BRASILEIRO DE ENSINO,
DESENVOLVIMENTO E PESQUISA

lucas.costa@idp.edu.br

@lucasrodri

www.linkedin.com/in/lucas-rodri