ECM306 – Tópicos Avançados em Estrutura de Dados

Aula 04

Engenharia da Computação – 3ª série

Função de Complexidade – Resolução (E1, E2)

2024

Atividade 1

- Considere dois algoritmos, A e B, com complexidades $8n^2$ e n^3, respectivamente. Qual o maior valor de n, para o qual o algoritmo B é mais eficiente que o algoritmo A ?

Função de Complexidade – Resolução

Resolução 1

- Função Complexidade $8n^2$:

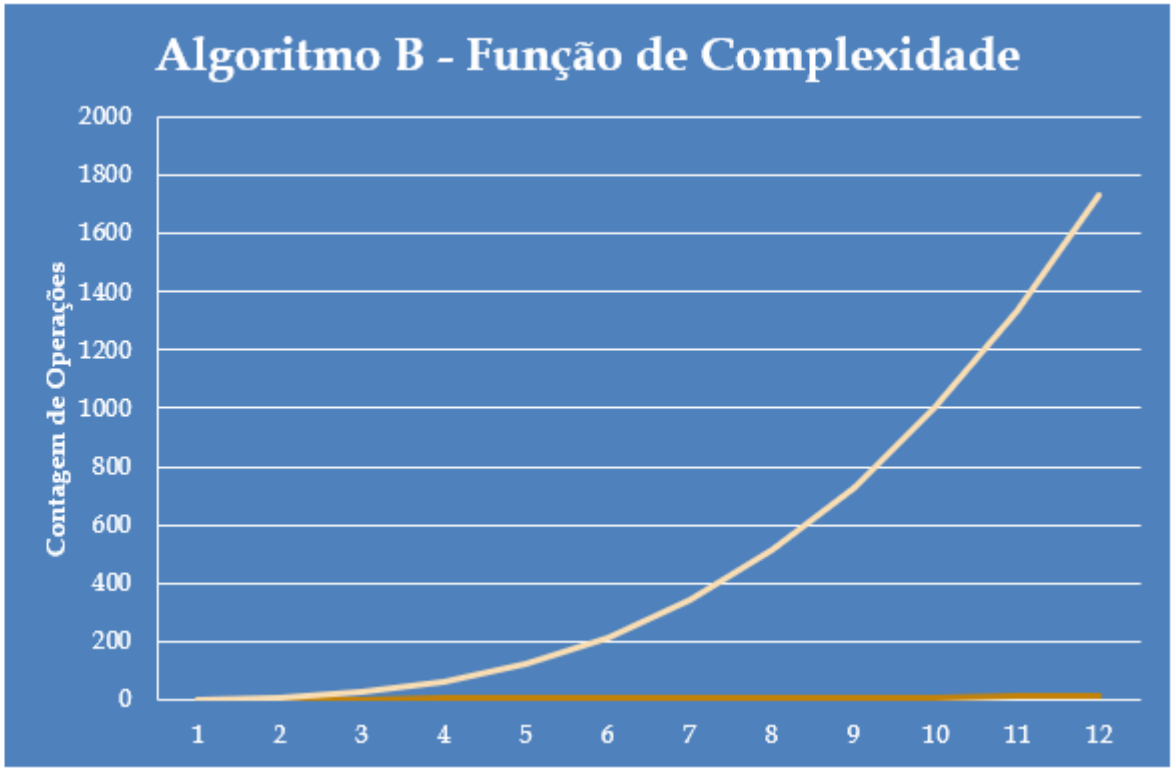
n	$8n^2$
1	8
2	32
3	72
4	128
5	200
6	288
7	392
8	512
9	648
10	800
11	968
12	1152

Função de Complexidade – Resolução

Resolução 1

- Função Complexidade n^3:

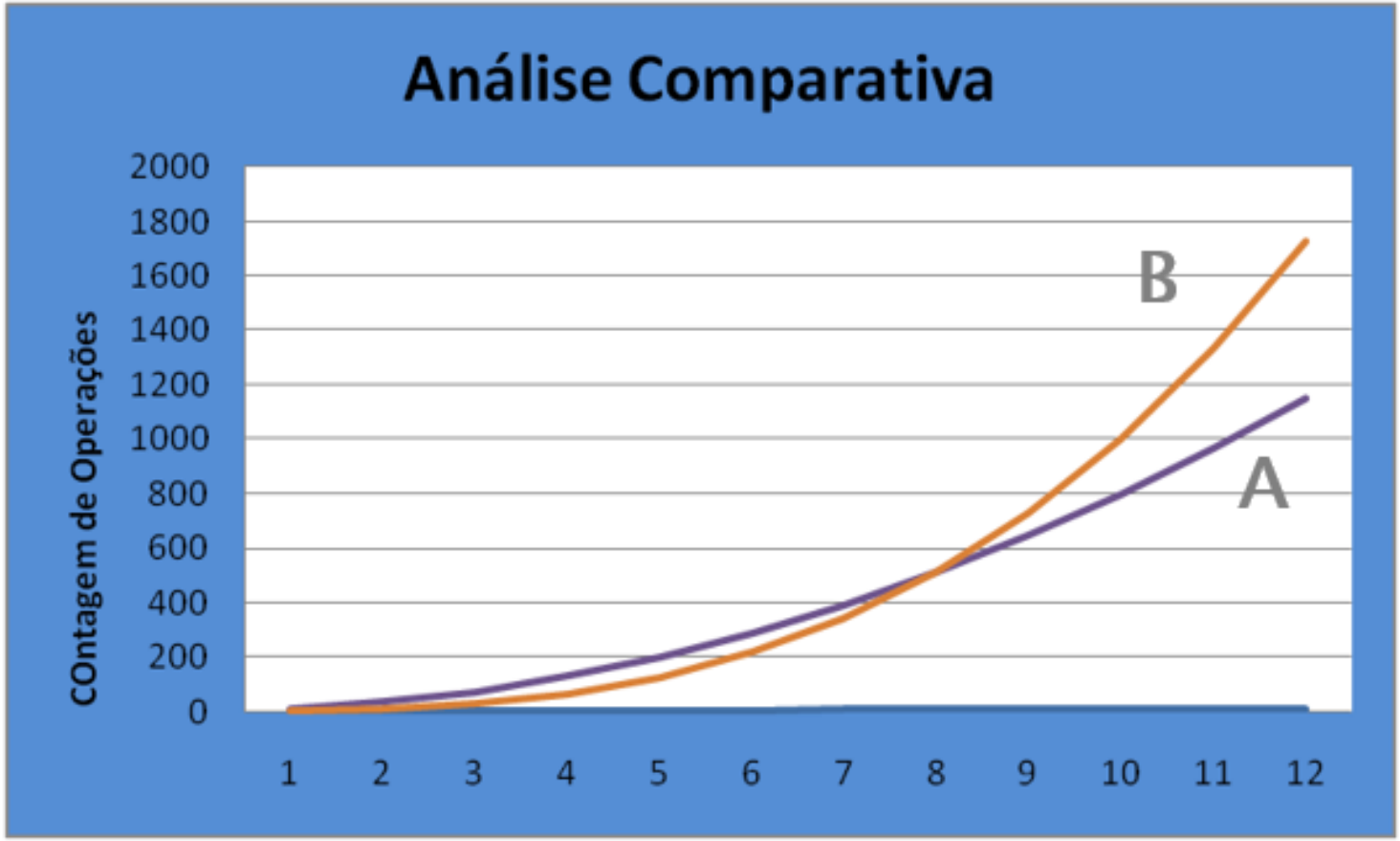
n	n^3
1	1
2	8
3	27
4	64
5	125
6	216
7	343
8	512
9	729
10	1000
11	1331
12	1728



Função de Complexidade – Resolução

Resolução 1

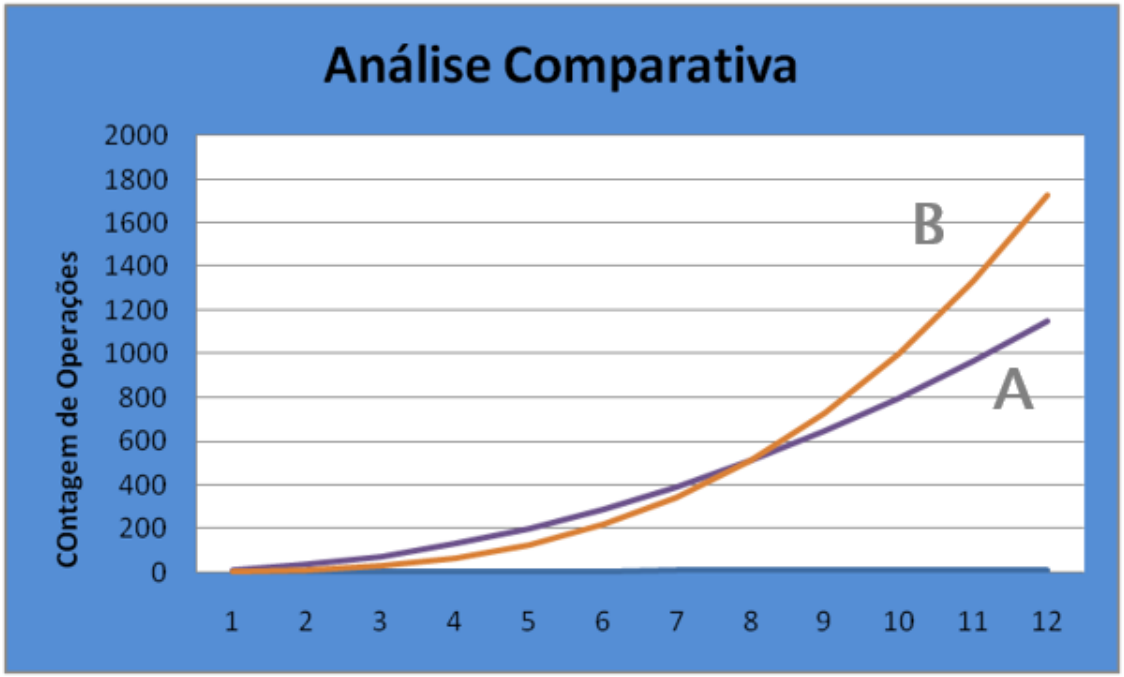
- Análise Gráfica:



Função de Complexidade – Resolução

Resolução 1

- Até um determinado valor de instância de entrada, o algoritmo B é melhor;
- A partir de um certo valor de ***n*** o algoritmo A fica melhor.



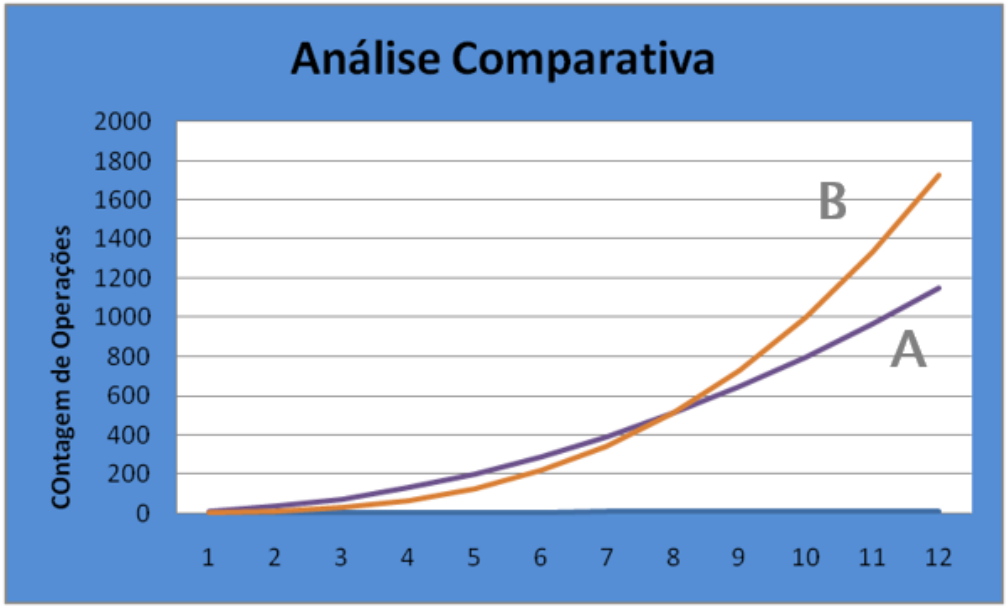
Função de Complexidade – Resolução

Resolução 1

- O ponto de equilíbrio ocorre quando $n^3 = 8n^2$

  $n^3 = 8n^2 \Rightarrow n^3 - 8n^2 = 0 \Rightarrow n^2 (n - 8) = 0 \Rightarrow n = 0$ ou $n - 8 = 0$

- Portanto, o ponto de equilíbrio é $n = 8$, ou seja, o algoritmo B é mais eficiente até $n = 7$.



Atividade 2

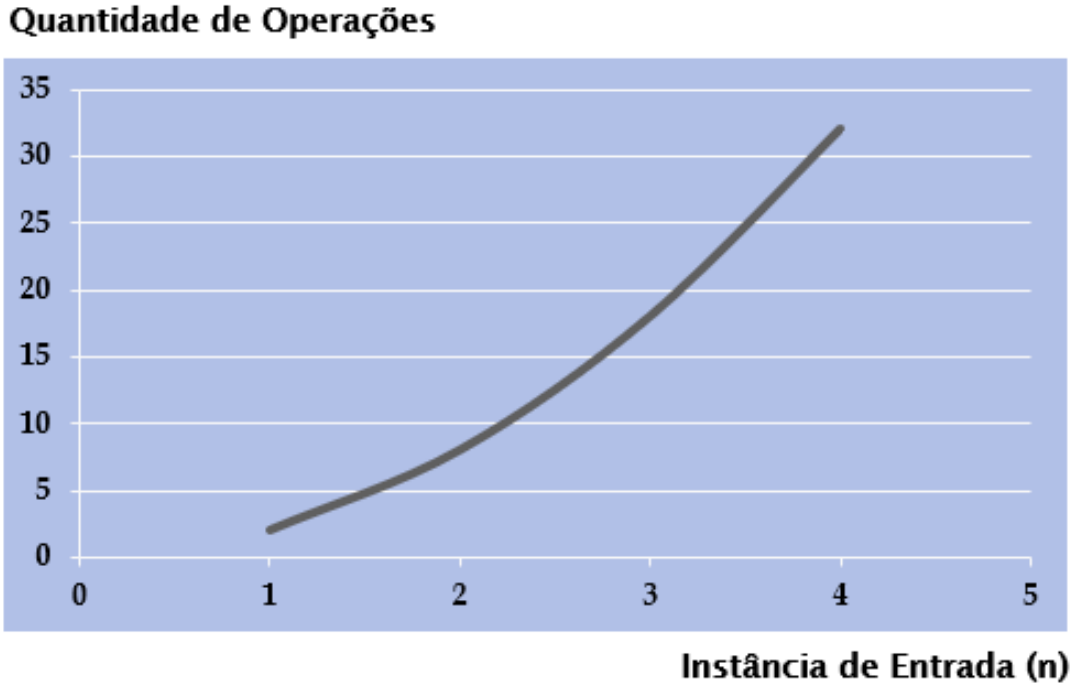
- Um algoritmo tem complexidade $2n^2$. Num certo computador, num tempo ***t***, o algoritmo resolve um problema de tamanho ***25***. Imagine agora que se tenha disponível um computador ***100*** vezes mais rápido. Qual o tamanho máximo de problema que o mesmo algoritmo resolve no mesmo tempo ***t*** no computador mais rápido?
- Considere o mesmo problema para um algoritmo de complexidade 2^n.

Função de Complexidade – Resolução

Resolução 2a

- Função de Complexidade $2n^2$:

n	$2n^2$
1	2
2	8
3	18
4	32
5	50
10	200
25	1250



Função de Complexidade – Resolução

Resolução 2a

- Analisando-se o comportamento da função de complexidade, pode-se afirmar que para ***n=25***, são necessárias ***1250*** operações;
- Estas operações são executadas no computador antigo em um determinado tempo ***t***;
- No computador novo as operações são executadas num total de ***100*** vezes mais rápidas ;
- Assim, no computador novo, no mesmo tempo ***t***, pode-se executar ***1250*100 = 125.000*** operações.

Função de Complexidade – Resolução

Resolução 2a

- Como o algoritmo é o mesmo, tanto no computador novo quando no antigo, a função de complexidade é a mesma:

$$f(n) = 2n^2$$

- Assim, no mesmo tempo ***t***, o computador novo executa ***125.000*** operações, o que representa uma instância maior;
- Tem-se, então:

$$f(n) = 2n^2 \Rightarrow 125000 = 2n^2 \Rightarrow$$

$$\Rightarrow 62500 = n^2 \Rightarrow$$

$$\Rightarrow n = 250$$

Resolução 2a

Resposta:

- O tamanho máximo de problema que o mesmo algoritmo resolve no tempo ***t***, no computador mais rápido, é ***250***.

Observação:

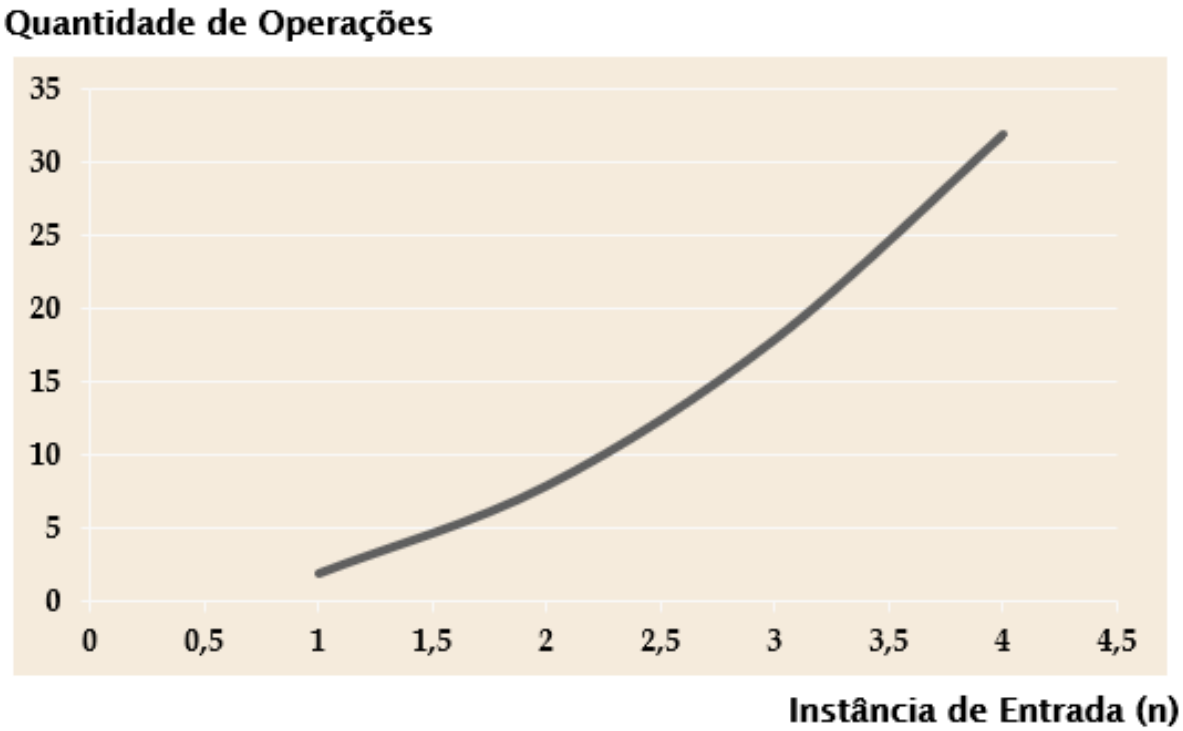
- Embora o computador mais novo seja ***100*** vezes mais rápido, o tamanho do problema aumentou apenas ***10*** vezes, ou seja, de ***25*** para ***250***.

Função de Complexidade – Resolução

Resolução 2b

- Função de Complexidade 2^n:

n	2^n
1	2
2	4
3	8
4	16
5	32
10	1024
25	33554432



Função de Complexidade – Resolução

Atividade 2b

- Analisando-se o comportamento da função de complexidade, pode-se afirmar que para ***n=25***, são necessárias ***33.554.432*** operações;
- Estas operações são executadas no computador antigo em um determinado tempo ***t***;
- No computador novo as operações são executadas num total de ***100*** vezes mais rápidas ;
- Assim, no computador novo, no mesmo tempo ***t***, pode-se executar ***33.554.423*100 = 3.355.443.200*** operações.

Função de Complexidade – Resolução

Atividade 2b

- Como o algoritmo é o mesmo, tanto no computador novo quando no antigo, a função de complexidade é a mesma:

$$f(n) = 2^n$$

- Assim, no mesmo tempo ***t***, o computador novo executa ***3.355.443.200*** operações, o que representa uma instância maior;
- Tem-se, então:

$$f(n) = 2^n \Rightarrow 2^n = 3.355.443.200 \Rightarrow$$

$$\Rightarrow \log_2 2^n = \log_2 3.355.443.200 \Rightarrow$$

$$\Rightarrow n = 31$$

Atividade 2b

Resposta:

- O tamanho máximo de problema que o mesmo algoritmo resolve no tempo ***t***, no computador mais rápido, é ***31***.

Observação:

- Embora o computador mais novo seja ***100*** vezes mais rápido, o tamanho do problema aumentou apenas ***1,24*** vezes, ou seja, de ***25*** para ***31***.

Função de Complexidade – Resolução

Atividade 3

- Suponha que uma empresa utiliza um algoritmo de complexidade n^2 que, em um tempo t, na máquina disponível, resolve um problema de tamanho x. Suponha que o tamanho do problema a ser resolvido aumentou em ***20%***, mas o tempo de resposta deve ser mantido. Para isso, a empresa pretende trocar a máquina por uma mais rápida. Qual percentual de melhoria no tempo de execução das operações básicas é necessário para atingir sua meta, considerando-se a execução do mesmo algoritmo?
  - Suponha que no problema anterior, mantendo-se o mesmo algoritmo, ainda se queira reduzir em ***50%*** o tempo de resposta. Qual a melhoria esperada para a nova máquina?

  - ?

Função de Complexidade – Resolução

Atividade 3

Máquina Velha

Qtde. de Operações	Tempo de cada Operação	Tempo Total
x^2	t_v	$tt_v = x^2 . t_v$

Máquina Nova

Qtde. de Operações	Tempo de cada Operação	Tempo Total
$(1.2x)^2$	t_n	$tt_n = 1.44x^2 . t_n$

Função de Complexidade – Resolução

Atividade 3

- ✓ O problema afirma que o **tempo total da máquina nova** deve ser **igual ao tempo total da máquina velha**;
- ✓ Assim, $\mathbf{tt_n = tt_v}$
- ✓ Portanto, como: $\mathbf{tt_v = x^2 . t_v}$ e $\mathbf{tt_n = 1.44.x^2 . t_n}$
- ✓ Teremos: $\mathbf{x^2 . t_v = 1.44 . x^2 . t_n}$
- ✓ Portanto: $\mathbf{t_v = 1.44 . t_n}$
- ✓ Assim: $\mathbf{t_v = 1.44 . t_n \Rightarrow tv = 1.t_n + 0.44\, t_n}$ => $t_v = t_n +$ **44/100.** t_n

A máquina velha é **44%** mais lenta que a nova, ou
A máquina nova é **44%** mais rápida que a velha

Função de Complexidade – Resolução

Atividade 3

Máquina Velha

Qtde. de Operações	Tempo de cada Operação	Tempo Total
x^2	t_v	$tt_v = x^2 . t_v$

Máquina Nova

Qtde. de Operações	Tempo de cada Operação	Tempo Total
$(1.2x)^2$	t_n	$tt_n = 1.44x^2 . t_n$

Função de Complexidade – Resolução

Atividade 3

- ✓ O problema afirma que o tempo total da máquina nova deve ser **50% inferior** ao tempo total da máquina velha.
- ✓ Assim, $tt_n = (50\%)\ tt_v \Rightarrow tt_n = 0{,}5\ tt_v$
- ✓ Portanto, como: $tt_v = x^2 . t_v$ e $tt_n = 1{,}44.x^2 . t_n$
- ✓ Teremos: $1{,}44 . x^2 . t_n = 0{,}5 . x^2 . t_v$
- ✓ Portanto: $1{,}44 . t_n = 0.5 . t_v$
- ✓ Assim: $2{,}88 . t_n = t_v \Rightarrow t_v = t_n + 1{,}88 . t_n \Rightarrow t_v = t_n + 188/100 . t_n$

A máquina velha é 188% mais lenta que a nova, ou
A máquina nova é 188% mais rápida que a velha

ECM306 – Tópicos Avançados em Estrutura de Dados

Referências bibliográficas

- CORMEN, T.H. et al. Algoritmos: Teoria e Prática (Caps. 13). Campus. 2002.
- ZIVIANI, N. Projeto de algoritmos: com implementações em Pascal e C (Cap. 1). 2.ed. Thomson, 2004.
- FEOFILOFF, P. Minicurso de Análise de Algoritmos, 2010. Disponível em:

  http://www.ime.usp.br/~pf/livrinho-AA/
- DOWNEY, A.B. *Analysis of algorithms* (Cap. 2), Em: *Computational Modeling and Complexity Science.* Disponível em:

  http://www.greenteapress.com/compmod/html/book003.html
- ROSA, J.L. Notas de Aula de Introdução a Ciência de Computação II. Universidade de São Paulo. Disponível em:

  http://coteia.icmc.usp.br/mostra.php?ident=639

Referências bibliográficas

- GOODRICH, Michael T. et al: *Algorithm Design and* Applications. Wiley, 2015.
- LEVITIN, Anany. *Introduction to the Design and Analysis of Algorithms*. Pearson, 2012.
- SKIENA, Steven S. *The Algorithm Design Manual.* Springer, 2008.
- Série de Livros Didáticos. *Complexidade de Algoritmos.* UFRGS.
- BHASIN, Harsh. *Algorithms – Design and Analysis.* Oxford University Press, 2015.
- FREITAS, Aparecido V. de – 2022 – Estruturas de Dados: Notas de Aula.
- CALVETTI, Robson - 2015 – Estruturas de Dados: Notas de Aula.

FIM