Análise de Eficiência – Notação Ordem de

Códigos de Alta Performance

PROFa. PATRÍCIA MAGNA - profpatricia.magna@fiap.com.br

Eficiência de um Algoritmo

Tempo de Execução do Programa

Quantidade de Memória Utilizada

- Existem 2 aspectos de eficiência considerados: tempo requerido e espaço de armazenamento.
- Geralmente, o programador deverá otimizar um aspecto às custas do outro.

Como analisar Eficiência de um Algoritmo

- A medida de tempo seria dada pelo **número de operações críticas (ou essenciais)** que são efetuadas durante a execução do algoritmo, a fim de resolver um específico problema (por exemplo, a busca de um registro em um arquivo)
- O tamanho do arquivo no qual o algoritmo atua sempre é referenciado em relação à quantidade **n** de registros que o compõe.

Quais são as operações críticas?

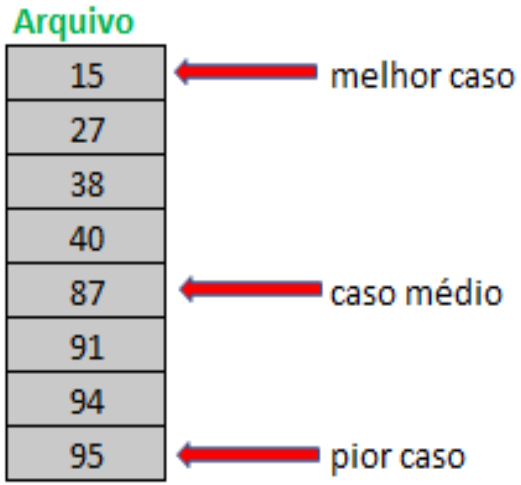
- São as operações essenciais que deve ser realizadas para resolução do aplicação
- Exemplos de operações críticas:
  - Para algoritmos de **busca** seriam apenas as **comparações** sobre chaves
  - Para algoritmos de **soma de elementos** de vetores seriam apenas as **adições** sobre os elementos dos vetores
  - Para algoritmos de **ordenação** seriam as **comparações** sobre chaves e movimentação de registros (ou **trocas**)
  - etc...

Como analisar Eficiência de um Algoritmo?

- A análise pode ser basear na identificação das seguintes situações:
  - Melhor caso
  - Caso médio
  - Pior caso

Exemplo de como analisar Eficiência de um Algoritmo

- Nos métodos de busca sequencial, supondo um arquivo com 8 registros:
  - Melhor caso: chave ser 1º registro (1 comparação)
  - Caso médio: chave estar na 4ª posição do arquivo (5 comparações)
  - Pior caso: chave ser do 8º registro do arquivo (8 comparações)

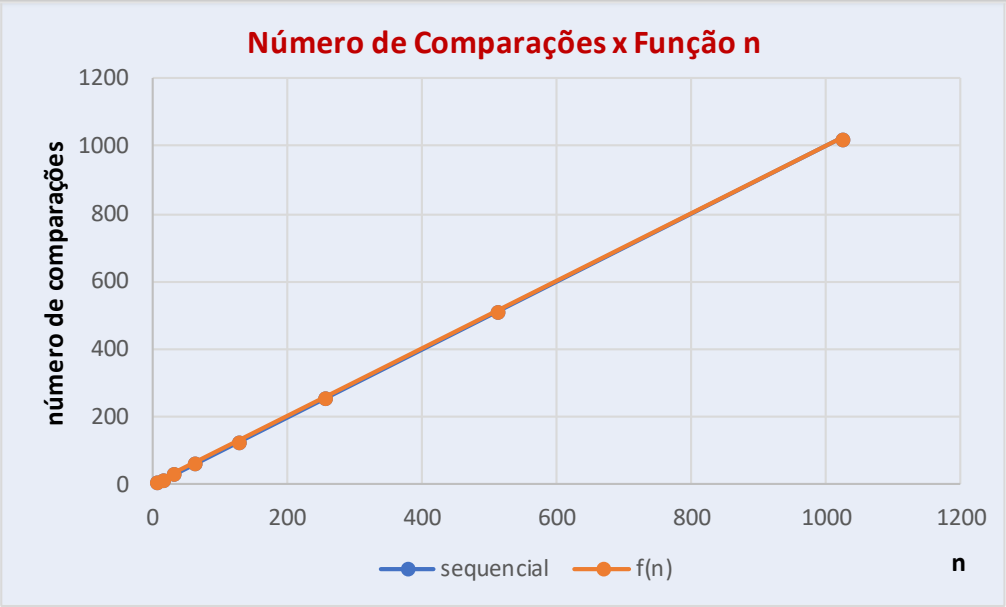


Mas...
Como descrever de “de forma mais geral” o comportamento de um algoritmo em relação a quantidade de dados a serem manipulado?

Descrevendo o comportamento da eficiência dos Métodos de Busca Sequencial e Busca Binária

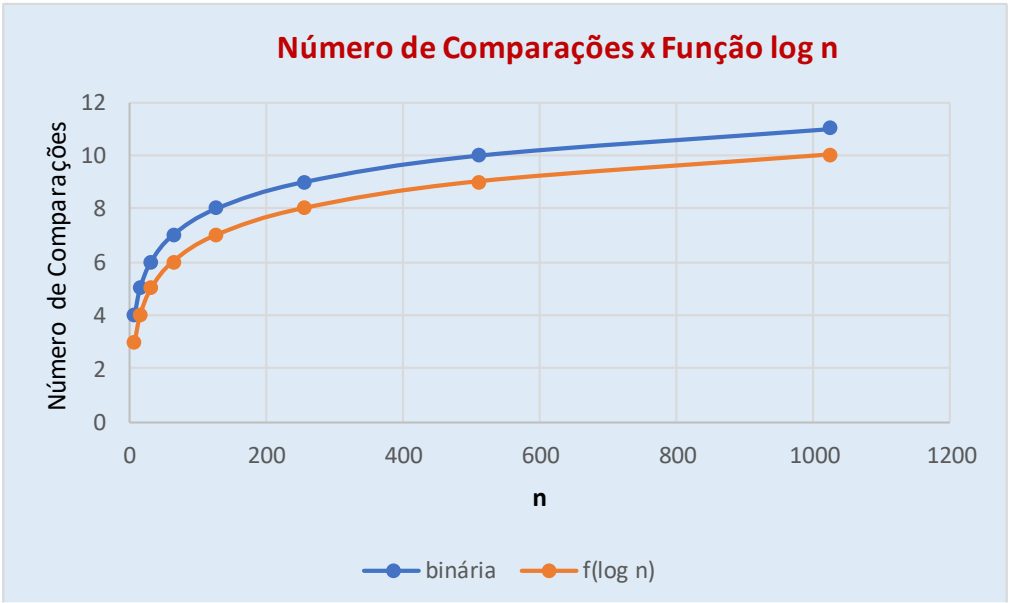
n	sequencial	f(n)
8	7	8
16	15	16
32	31	32
64	63	64
128	127	128
256	255	256
512	511	512
1024	1023	1024

Função que melhor descreve o número de comparações medidos na execução



n	binária	f(log n)
8	4	3
16	5	4
32	6	5
64	7	6
128	8	7
256	9	8
512	10	9
1024	11	10

* Foi usado como log n => $\log_2 n$



Como analisar a Eficiência de um Algoritmo

- No caso dos métodos de busca encontrar uma função que descreva o comportamento do método em relação ao número de registros foi simples.
- Podem ser encontradas funções bem diferentes fazendo as medidas de número de operações críticas em relação ao número de dados.
- Suponha que 2 métodos diferentes que executam a mesma tarefa chega-se às seguintes funções:

$$f_{metodo1}(n) = 0{,}01n^2 + 10n$$

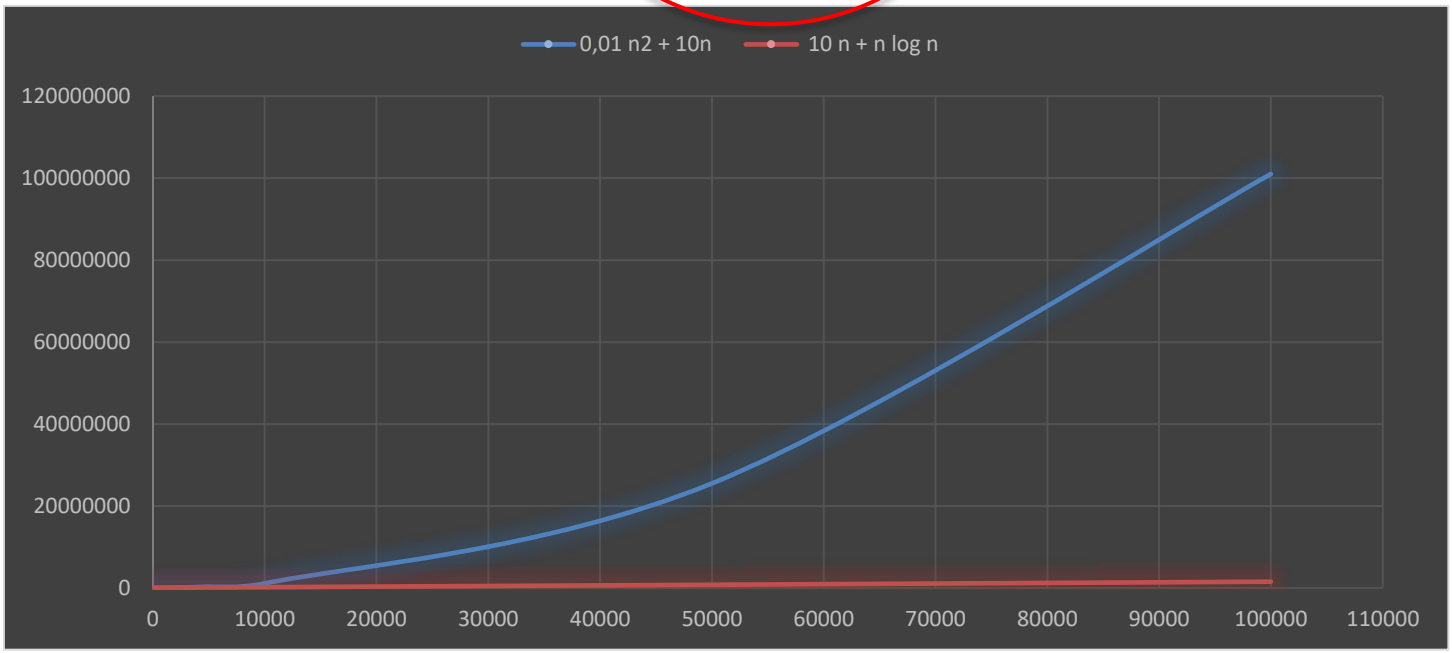
$$f_{metodo2}(n) = 10\,n + n \log n$$

- Qual dos métodos é o mais eficiente?

Comparando de Forma Numérica as Funções

n	0,01 n2 + 10n	10 n + n log n
100	1100	1200,0
500	7500	6349,5
1000	20000	13000,0
5000	300000	68494,9
10000	1100000	140000,0
50000	25500000	734948,5
100000	101000000	1500000,0

É mais eficiente por exigir menos comparações quando a quantidade de dados aumenta



Como analisar a Eficiência de um Algoritmo

- Analisando o comportamento das funções, observamos que para valores de n grandes a função do método 1 supera e muito o número de operações críticas em relação ao 2º método

$f_{metodo1}(n) = 0{,}01n^2 + 10n$ > $f_{metodo2}(n) = 10\,n + n \log n$

- Mas, deve haver uma forma mais fácil de descobrir o comportamento da variação das operações críticas sem ter que construir gráficos. Será?

Sim, existe e para isso existe uma notação conhecida como “da ordem de” ou big O.

O( )

Iniciando com a Notação “da Ordem de”- O ( )

- Suponha a seguinte função $0,01n^2 + 10n$

n	A=0,01 n^2	B= 10 n	A+B	(A+B) / n^2
10	1	100	101	1,01
50	25	500	525	0,21
100	100	1.000	1100	0,11
500	2.500	5.000	7500	0,03
1.000	10.000	10.000	20.000	0,02
5.000	250.000	50.000	300.000	0,01
10.000	1.000.000	100.000	1.100.000	0,01
50.000	25.000.000	500.000	25.500.000	0,01
100.000	100.000.000	1.000.000	101.000.000	0,01

Notação “da Ordem de”- O ( )

- No exemplo anterior, quando n torna-se grande a função fica apenas proporcional a n^2.
- Diz-se que de $0,01n^2 + 10n$ é “da ordem” da função n^2.
- Utiliza-se a seguinte notação para representar essa proporcionalidade:
  - **$O(n^2)$. Diz-se da ordem de n^2.**
- Essa notação é muito utilizada para analisar com que proporção varia o tempo (ou número de operações críticas) são efetuadas, e desta forma, ter de alguma forma ideia da eficiência do algoritmo utilizado.

Propriedade Assintótica

- Se f(n) for O(g(n)) ocasionalmente (n ≥ b) tornar-se-á permanentemente menor ou igual a algum múltiplo de g(n). Isto significa que f(n) está limitada por g(n) de cima para baixo, ou que f(n) é uma função menor que g(n).
- Outro modo, mais formal, é dizer que f(n) é assintoticamente limitada por g(n).

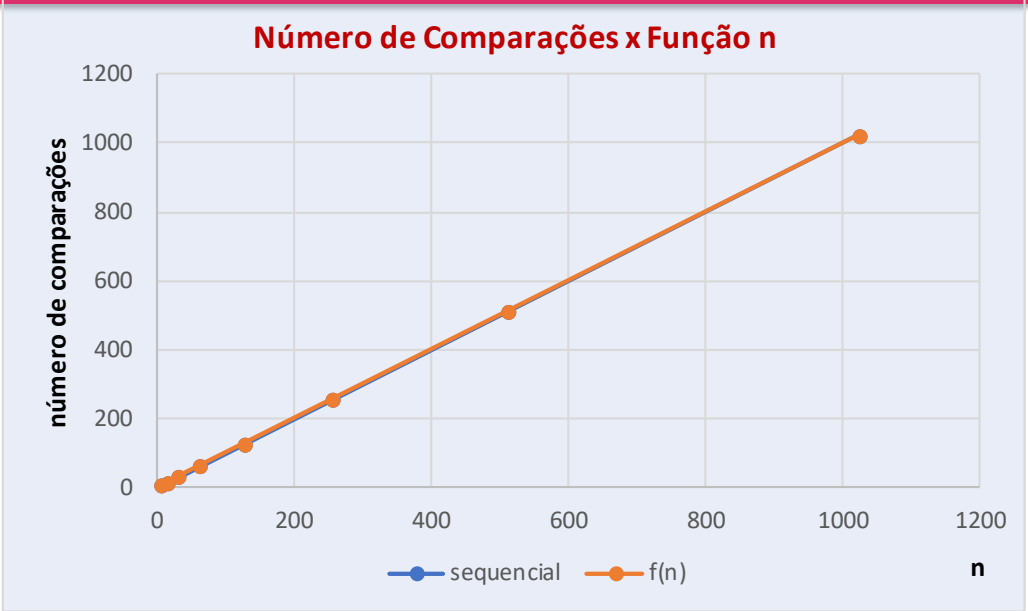
TRADUZINDO:

A função g(n) para valores altos de n sempre vai resultar um valor maior do que f(n).

Desta forma, f(n) sempre será limitada pela função g(n)

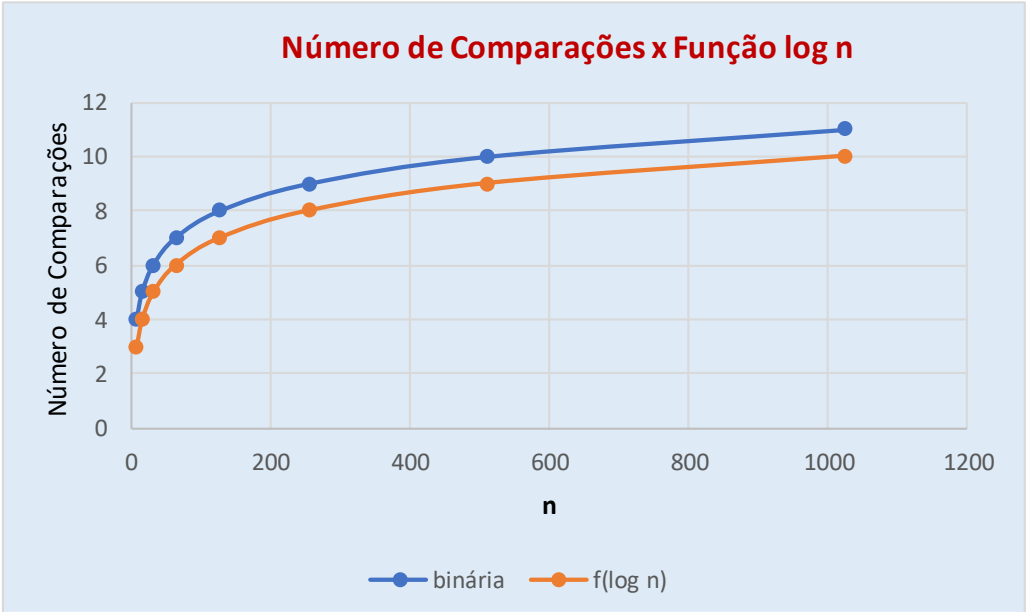
Descrevendo o comportamento dos Métodos de Busca

n	sequencial	f(n)
8	7	8
16	15	16
32	31	32
64	63	64
128	127	128
256	255	256
512	511	512
1024	1023	1024



O(n)

n	binária	f(log n)
8	4	3
16	5	4
32	6	5
64	7	6
128	8	7
256	9	8
512	10	9
1024	11	10

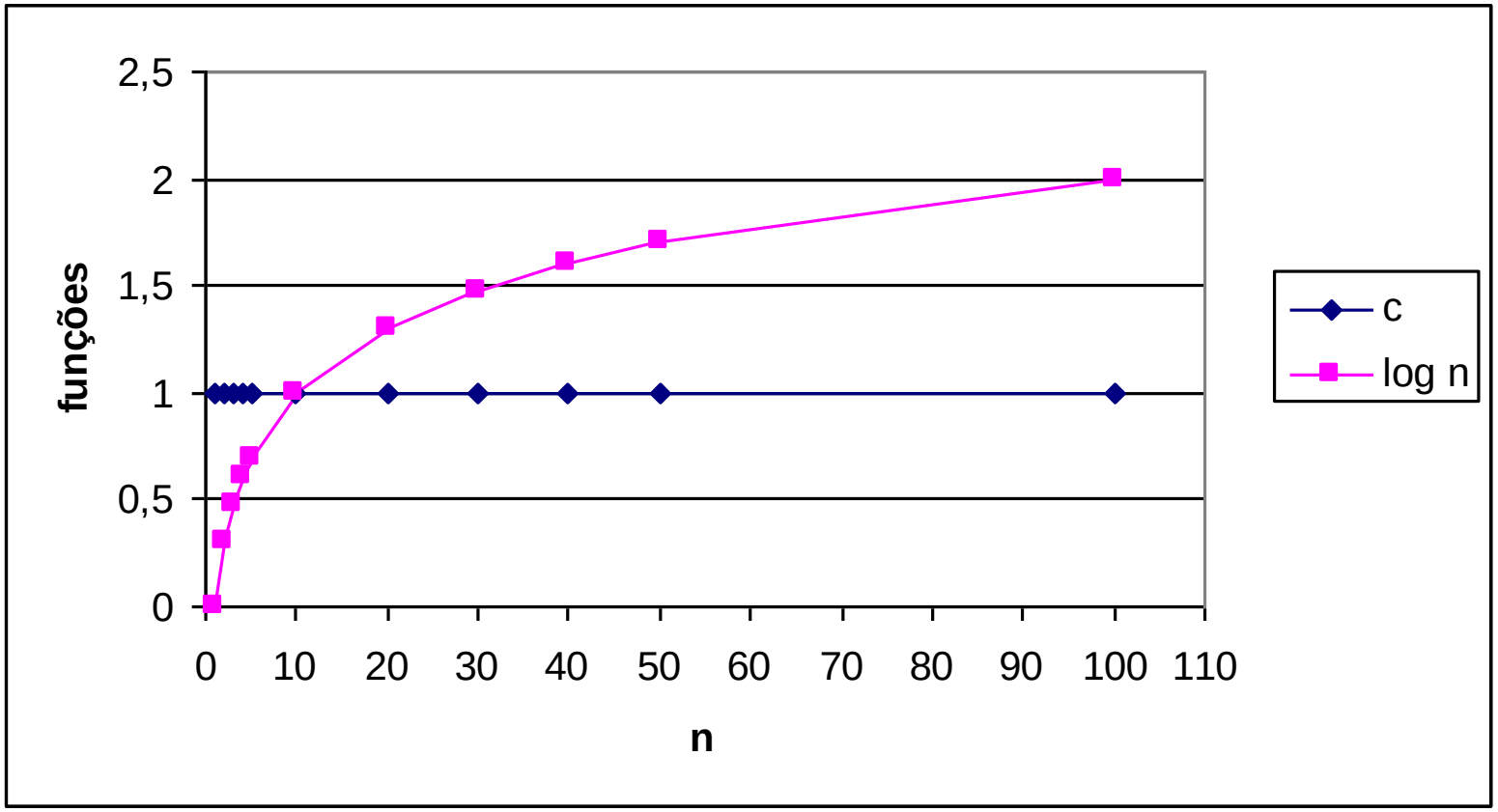


O(log n)

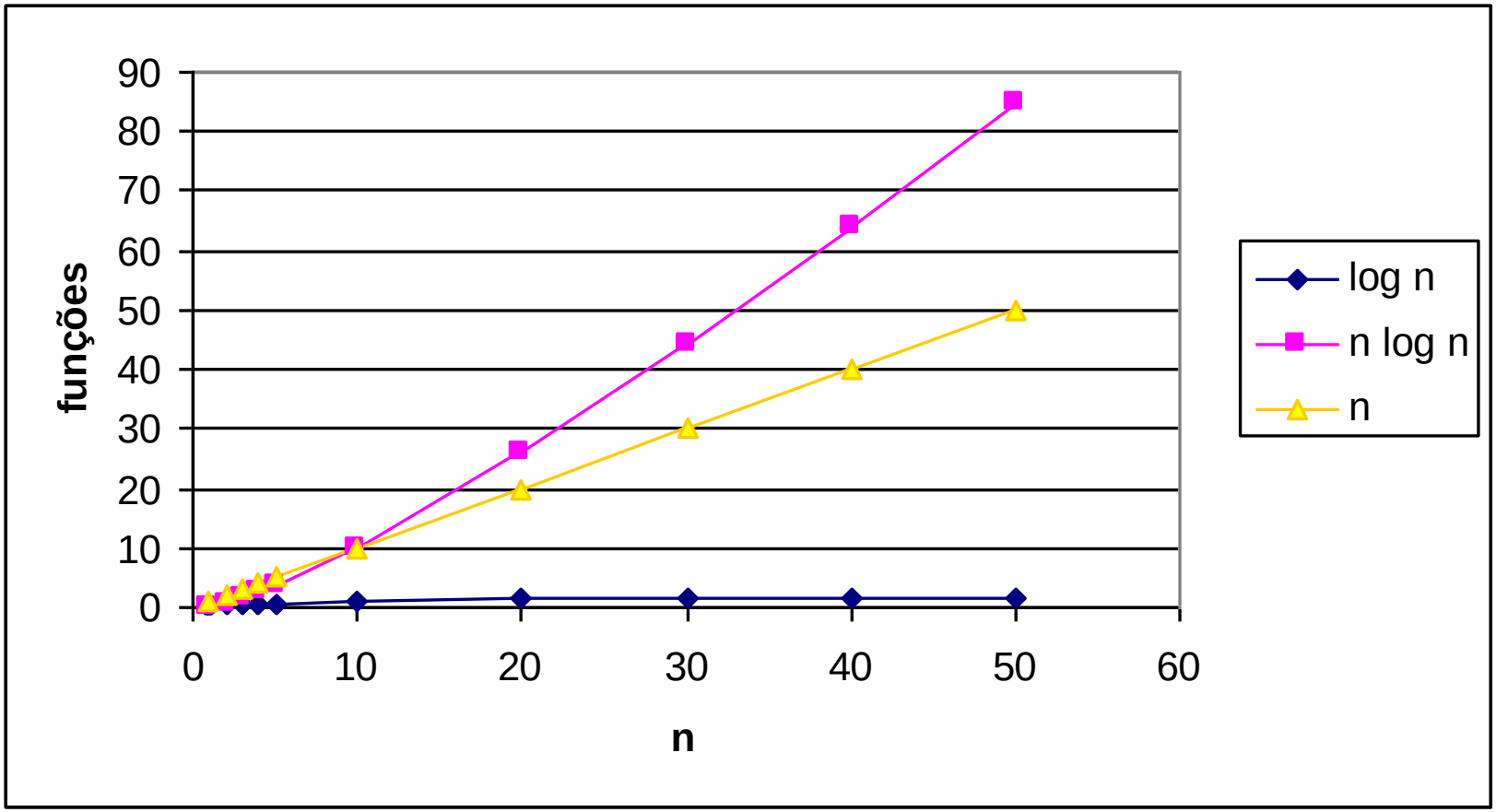
Notação “da Ordem de”- O ( )

- Funções que são usadas para descrever o comportamento dos algoritmos, são :
  - **O(constante)**
  - **O(log n)**
  - **O(n)**
  - **O(n log n)**
  - **O(n^2)**
  - **O(n^k)**
  - **O(2^n)**
- As funções já estão apresentadas de forma hierárquica , conforme será demonstrado nos próximos slides...

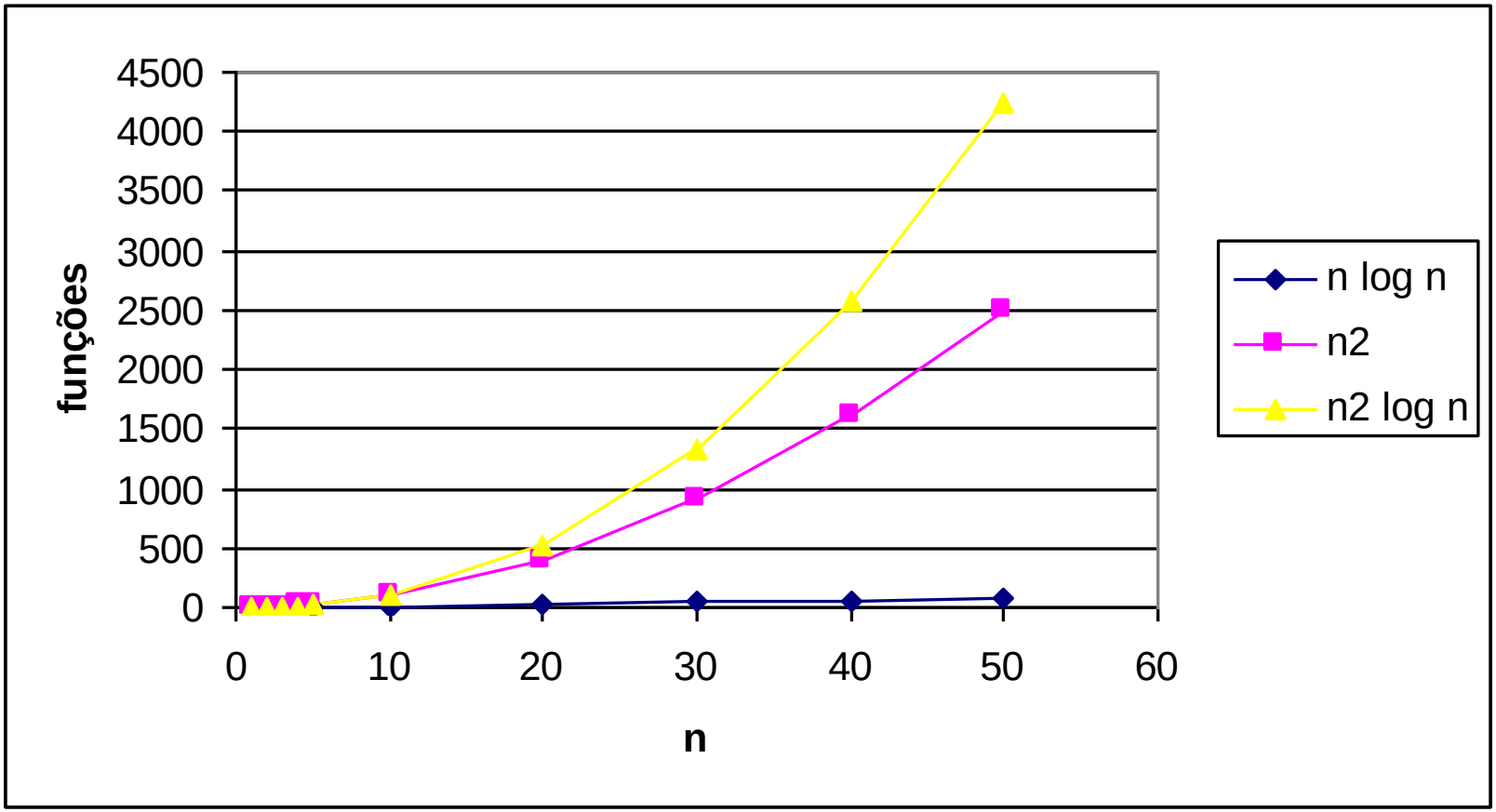
Comparação entre as funções c=1 e log n



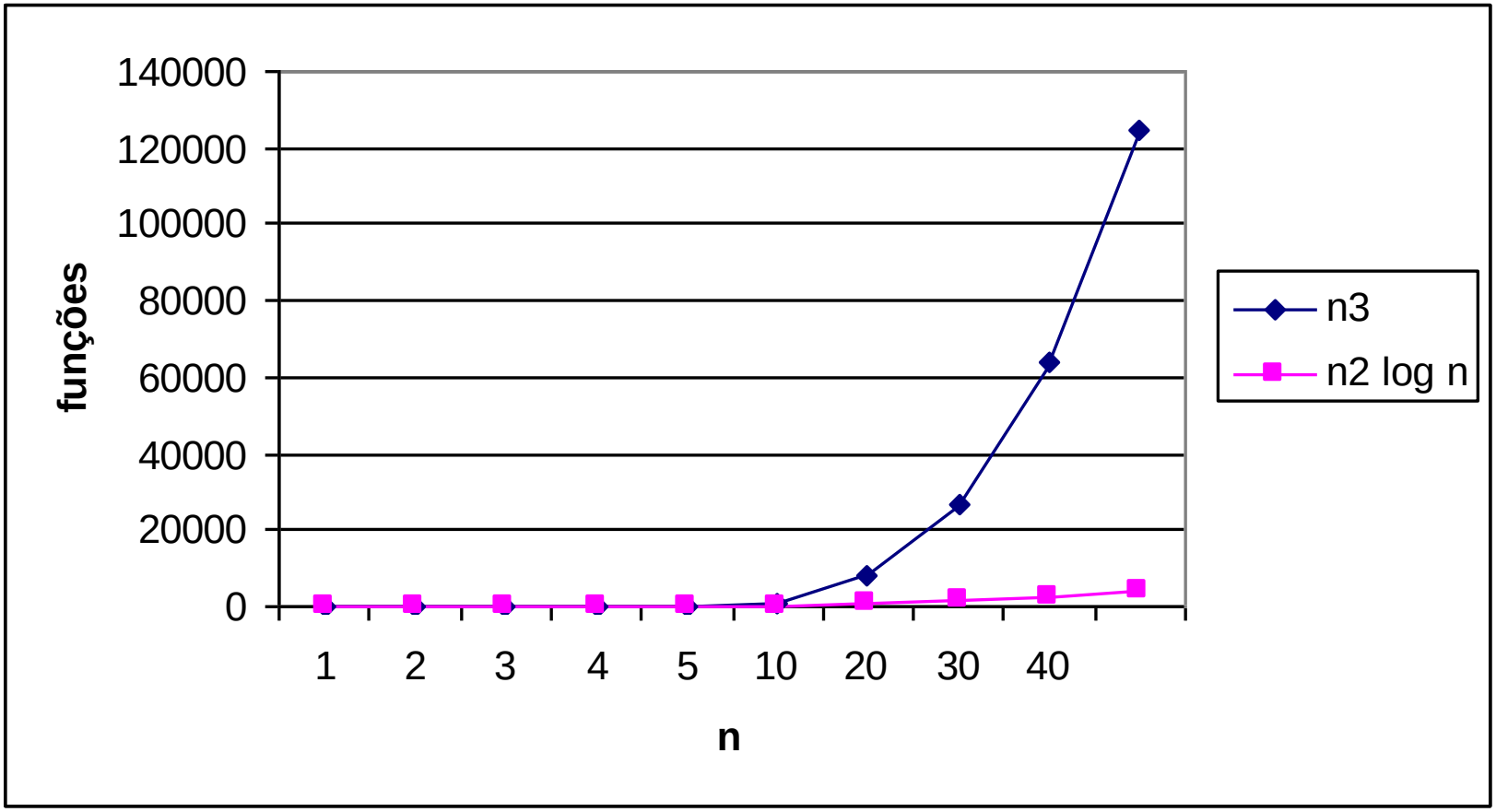
Comparação entre log n, n log n e n



Comparação entre n log n, $n^2 \log n$ e n^2



Comparação entre as funções $n^2 \log n$ e n^3



Notação “da Ordem de”- O ( )

- **Funções que são usadas para descrever o comportamento dos algoritmos:**
  - **O(constante)**
  - **O(log n) : busca binária**
  - **O(n) : busca sequencial**
  - **O(n log n)**
  - **O(n^k): complexidade polinomial (problemas P, NP e NP completo)**
  - **O(2^n): algoritmos conhecidos como intratáveis**

Eficiência de Algoritmos de Ordenação

- Através do conceito de “ordem de” pode-se analisar a ordem de um algoritmo e categorizá-lo como eficiente ou ineficiente.
- Vamos a partir de agora descrever a eficiência de algoritmos e métodos usando essa notação da ordem de.

#partiuAlgoritmoOrdenação

REFERÊNCIAS

- ASCÊNCIO, A.F.G; ARAUJO, G.S. **Estruturas de Dados: Algoritmos, Análise de Complexidade e Implementações em JAVA e C/C++.** São Paulo, Ed.Pearson Prentice Hall, 2010.
- PEREIRA, S.L.; **Estruturas de Dados Fundamentais: Conceitos e Aplicações.** São Paulo, Ed. Érica, 1996.
- TENEMBAUM, A.M et al.; **Estruturas de Dados usando C.** Makron Books Ltda, 1995.
- ZIVIANI, Nivio. Projeto de algoritmos com implementações em Pascal e C. São Paulo: Pioneira, 2000.