# NOTAÇÃO ASSINTÓTICA DE COMPLEXIDADE

## DCE770 - Heurísticas e Metaheurísticas

Atualizado em: 28 de setembro de 2022

Iago Carvalho

Departamento de Ciência da Computação

# MÉDIA DE TEMPO DE EXECUÇÃO DE UM ALGORITMO

O projeto de algoritmos é fortemente influenciado pelo estudo de seuscomportamentos

Depois que um problema é analisado e decisões de projeto sãofinalizadas, é necessário estudar as várias opções de algoritmos aserem utilizados, considerando os aspectos de tempo de execução eespaço ocupado

Muitos desses algoritmos são encontrados em áreas como pesquisa operacional, otimização, teoria dos grafos, estatística, probabilidades, entre outras.

# TIPOS DE PROBLEMAS NA ANÁLISE DE ALGORITMOS

Análise de um algoritmo particular

- Qual é o custo de usar um dado algoritmo para resolver um problema específico?
- Características que devem ser investigadas
  - Número de vezes que cada parte do algoritmo deveser executada
  - Gasto de memória do algoritmo

Análise de uma classe de algoritmos

- Qual é o algoritmo de menor custo possível para resolver um problema particular?
- Coloca-se limites para a complexidade computacional dos algoritmos pertencentes à classe

# CUSTO DE UM ALGORITMO

Determinando o menor custo possível para resolver problemas de uma dada classe, temos a medida da dificuldade inerente para resolver o problema

Podem existir vários algoritmos para resolver o mesmo problema

Quando o custo de um algoritmo é igual ao menor custo possível, o algoritmo é **ótimo** para a medida de custo considerada

Se a mesma medida de custo é aplicada a diferentes algoritmos, então é possível compará-los e escolher o mais adequado

# MEDIDA DO CUSTO PELA EXECUÇÃO DO PROGRAMA

Tempo de relógio

- Tempo de execução em segundos, minutos ou horas
- Influenciável pelo hardare

Modelo matemático

- Modelo teórico que descreve a complexidade de algoritmos
- Complexidade computacional

# FUNÇÃO DE COMPLEXIDADE

Função de complexidade $f$

- Um algoritmo tem complexidade $f(n)$ para uma entrada de tamanho $n$

```
function Max (var A: Vetor): integer;
var i, Temp: integer;
begin
  Temp := A[1];
  for i := 2 to n do if Temp < A[i] then Temp := A[i];
  Max := Temp;
end;
```

O que esse algoritmo faz? Qual é a função de complexidade dele?

# TRÊS CASOS DE EXECUÇÃO DE UM ALGORITMO

Dependendo da entrada, o tempo de execução de um algoritmo pode variar

Pensem num algoritmo de ordenação

- A disposição inicial dos itens a serem ordenados modifica a complexidade computacional do algoritmo

Deste modo, temos três casos

1. Pior caso
2. Caso médio (ou caso esperado)
3. Melhor caso

# FUNÇÃO DE COMPLEXIDADE

Não é prático medirmos a função de complexidade exata de um algoritmo

Quando temos problemas pequenos, *tanto faz* a complexidade do algoritmo

Quando resolvemos problemas grandes, é interessante escolhermos algoritmos eficientes

- Problema grande = valor de $n$ muito alto
- Na maioria das vezes, estamos interessados no comportamento assintótico de sua função de complexidade

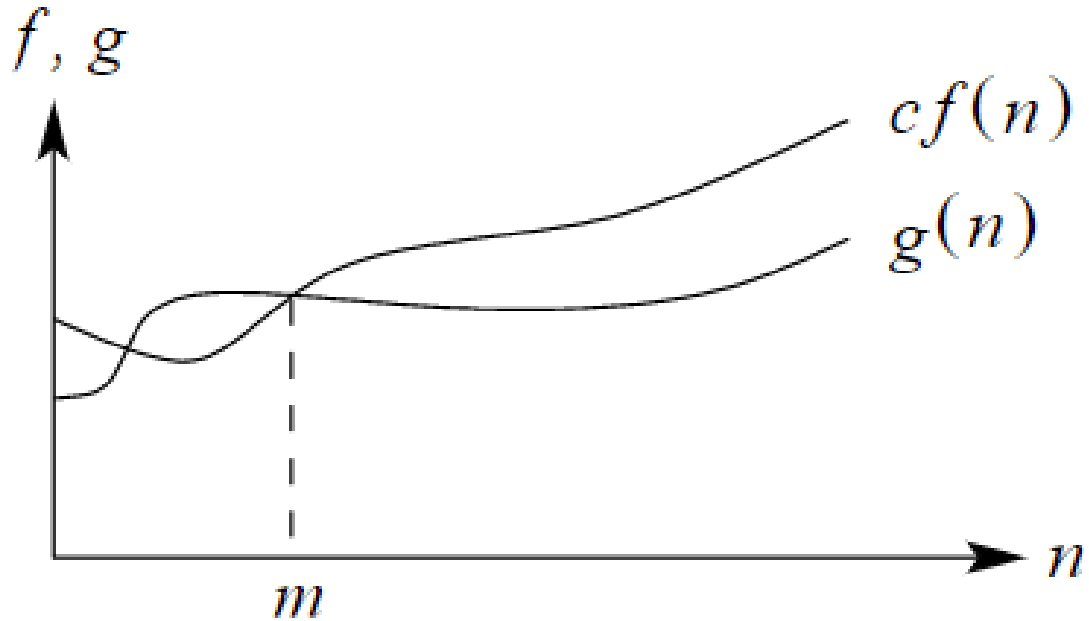
O comportamento assintótico de $f(n)$ define o limite computacional do algoritmo para quando o valor de $n$ cresce

# DOMINAÇÃO ASSINTÓTICA

Uma função $f(n)$ **domina** assintoticamente outra função $g(n)$ se existem duas constantes positivas $c$ e $m$ tais que

$$|g(n)| \leq c\,|f(n)|, \quad \forall\, n \geq m$$

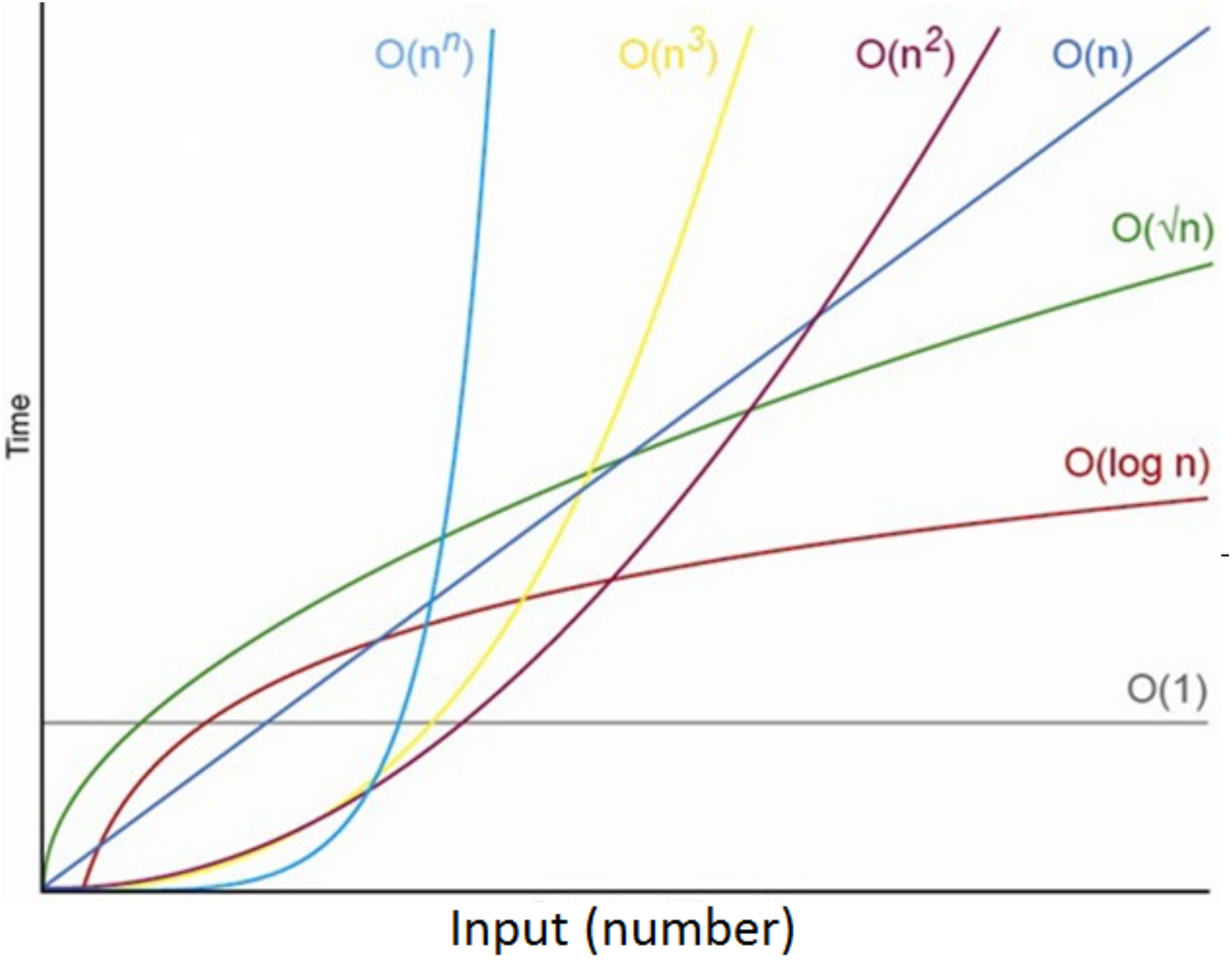
Dizemos que $g(n) = O(f(n))$ para denotar que $g(n)$ domina assintoticamente $f(n)$



# NOTAÇÃO O

$$
\begin{aligned}
f(n) &= O(f(n)) \\
c \times O(f(n)) &= O(f(n)) \quad c = constante \\
O(f(n)) + O(f(n)) &= O(f(n)) \\
O(O(f(n)) &= O(f(n)) \\
O(f(n)) + O(g(n)) &= O(max(f(n), g(n))) \\
O(f(n))O(g(n)) &= O(f(n)g(n)) \\
f(n)O(g(n)) &= O(f(n)g(n))
\end{aligned}
$$

# NOTAÇÃO O



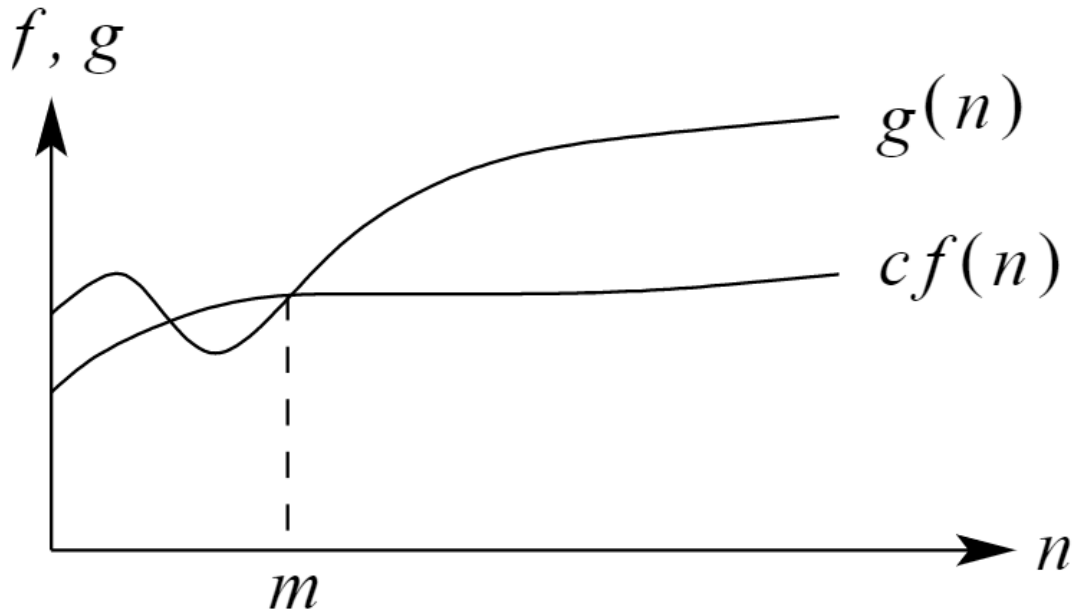
# NOTAÇÃO O

| | Complexidade de tempo | | |
|---|---|---|---|
| | **Melhor** | **Médio** | **Pior** |
| **Insertion Sort** | $O(n)$ | $O(n^2)$ | $O(n^2)$ |
| **Selection Sort** | $O(n^2)$ | $O(n^2)$ | $O(n^2)$ |
| **Bubble Sort** | $O(n)$ | $O(n^2)$ | $O(n^2)$ |
| **Heap Sort** | $O(n \log(n))$ | $O(n \log(n))$ | $O(n \log(n))$ |
| **Merge Sort** | $O(n \log(n))$ | $O(n \log(n))$ | $O(n \log(n))$ |
| **Quick Sort** | $O(n \log(n))$ | $O(n \log(n))$ | $O(n^2)$ |

# NOTAÇÃO Ω

Uma função $g(n)$ **é dominada** assintoticamente outra função $f(n)$ se existem duas constantes positivas $c$ e $m$ tais que

$$|g(n)| \geq c\,|f(n)|, \quad \forall\, n \geq m$$



# NOTAÇÃO Θ

Uma função $g(n)$ é dita ser $\Theta(f(n))$ se existem três constantes positivas $c_1$, $c_2$ e $m$ tais que

$$0 \leq c_1 f(n)| \leq |g(n)| \leq c_2 |f(n)|, \quad \forall n \geq m$$

Neste caso, $f(n)$ é dito ser um limite assintótico firme