

# Análise de algoritmos

# Pontos da aula

- Operações e propriedades da notação  $O$ .
- Análise para um pior caso.
- Exemplos de análise de algoritmos.

# Análise de algoritmos

- Operações e Propriedades da Notação  $O$ .

# Operações e propriedades da notação $O$

- $f(n) = O(f(n))$
- $c \cdot O(f(n)) = O(f(n))$ , onde  $c$  é uma constante
- $O(f(n)) + O(f(n)) = O(f(n))$
- $O(O(f(n))) = O(f(n))$
- $O(f(n)) + O(g(n)) = O(\max(f(n), g(n)))$
- $O(f(n)) \cdot O(g(n)) = O(f(n) \cdot g(n))$
- $f(n) \cdot O(g(n)) = O(f(n) \cdot g(n))$

# Operações e propriedades da notação O

- **Multiplicação por uma função não constante:**

Se  $f(n) = O(n^2)$  e  $g(n) = n$ , então  $f(n) \cdot g(n) = O(n^2 \cdot n) = O(n^3)$ .

- **Divisão por uma função:**

Se  $f(n) = O(n^3)$  e  $g(n) = O(n^2)$ , então  $\frac{O(f(n))}{O(g(n))} = O\left(\frac{n^3}{n^2}\right) = O(n)$ .

- **Potência de uma função:**

Se  $f(n) = O(n)$ , então  $f(n)^k = O(n^k)$ , onde  $k$  é uma constante.

# Operações e propriedades da notação O

- **Exponenciação com uma constante na base:**  
Se  $f(n) = O(n^2)$ , então  $2^{O(f(n))} = 2^{O(n^2)} = O(2^{n^2})$ .
- **Logaritmos de uma função:**  
Se  $f(n) = O(n^k)$ , então  $\log O(f(n)) = O(\log n^k) = O(k \log n) = O(\log n)$ .
- **Soma infinita de funções:**  
Se  $f(n) = O(1) + O(n) + O(n^2)$ , então a complexidade total é a do termo dominante:  $O(n^2)$ .

# Operações e propriedades da notação $O$

- Propriedades da Notação  $O$ .

- **Reflexividade:**

$$f(n) = O(f(n)).$$

- **Transitividade:**

Se  $f(n) = O(g(n))$  e  $g(n) = O(h(n))$ , então  $f(n) = O(h(n))$ .

- **Monotonicidade:**

Se  $f(n) = O(g(n))$  e  $g(n) \geq h(n)$  para  $n$  suficientemente grande, então  $f(n) = O(h(n))$ .

- **Assintoticamente maior:**

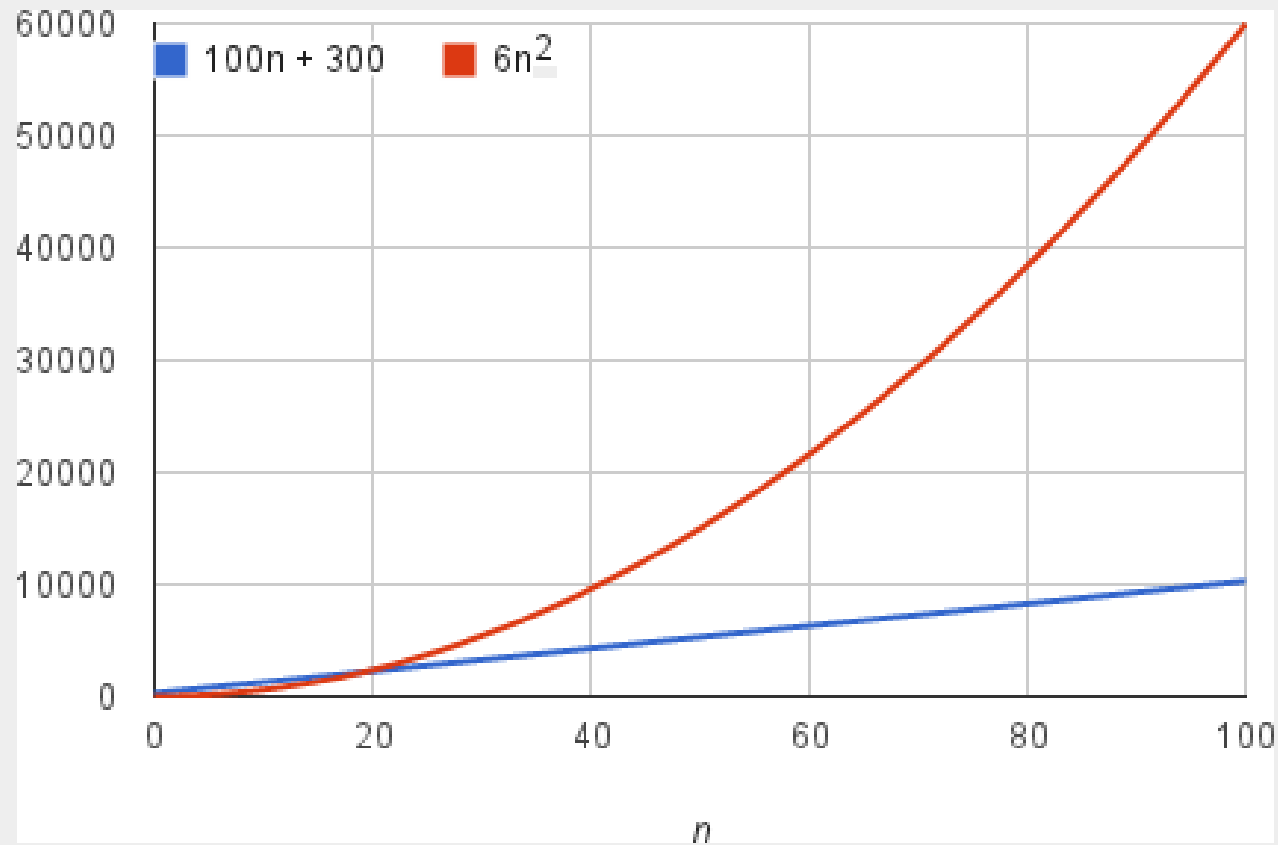
Se  $f(n) = O(n^2)$  e  $g(n) = O(n)$ , então  $O(n^2) \geq O(n)$ .

# Análise de algoritmos

- Análise para um pior caso.



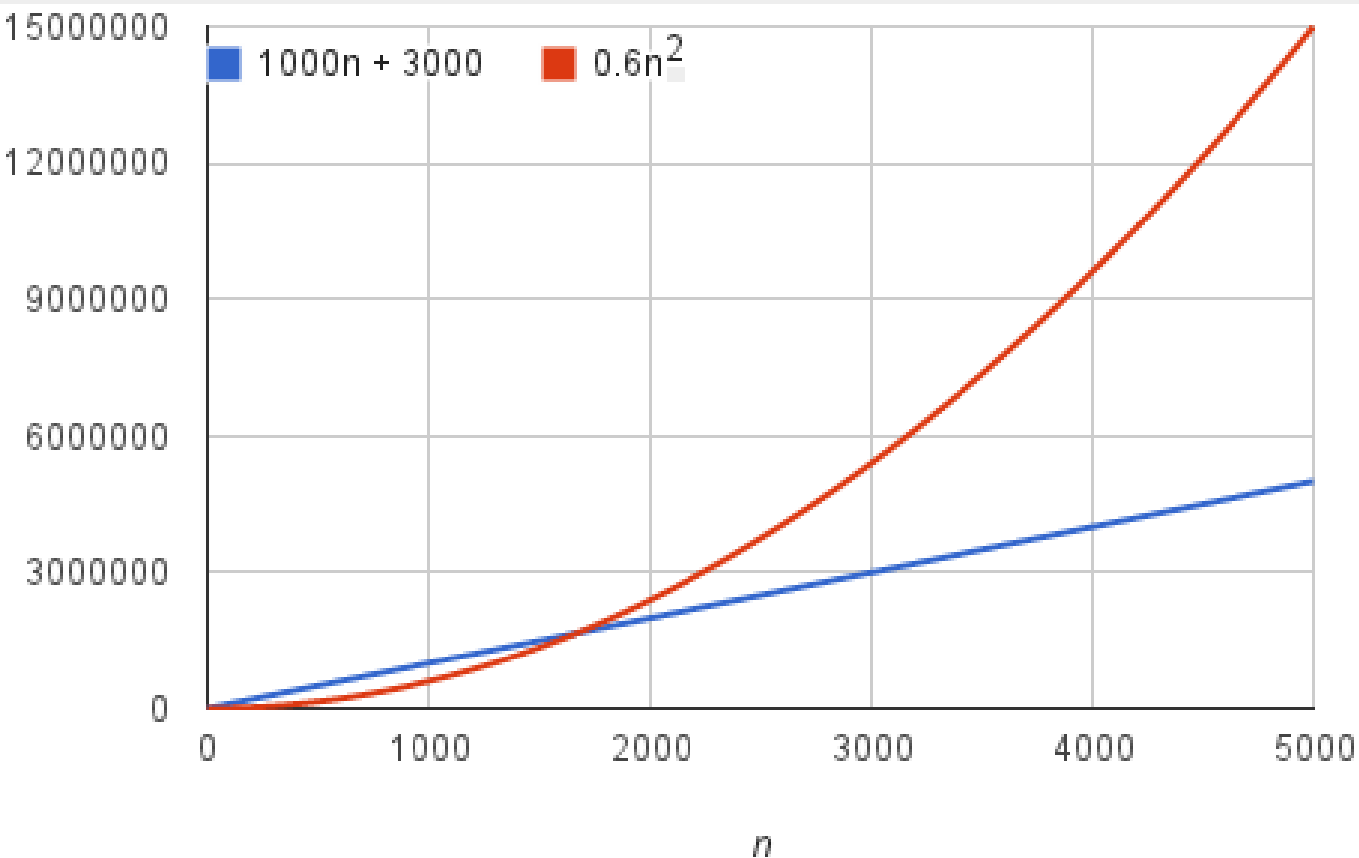
# Análise para um pior caso



$$T(n) = 6n^2 + 100n + 300$$

$$\rightarrow n^2 \rightarrow O(n^2)$$

# Análise para um pior caso



$$T(n) = 0,6n^2 + 1000n + 300$$

$$\rightarrow n^2 \rightarrow O(n^2)$$

# Análise para um pior caso

		Tamanho da entrada		
		10	30	50
Tempo de Execução	$5n$	50 <sub>s</sub>	150 <sub>s</sub>	250 <sub>s</sub>
	$5n^2$	500 <sub>s</sub>	4500 <sub>s</sub>	208 <i>minutos</i>
	$2^n$	1024 <sub>s</sub>	34 <i>anos</i>	357 mil <i>séculos</i>

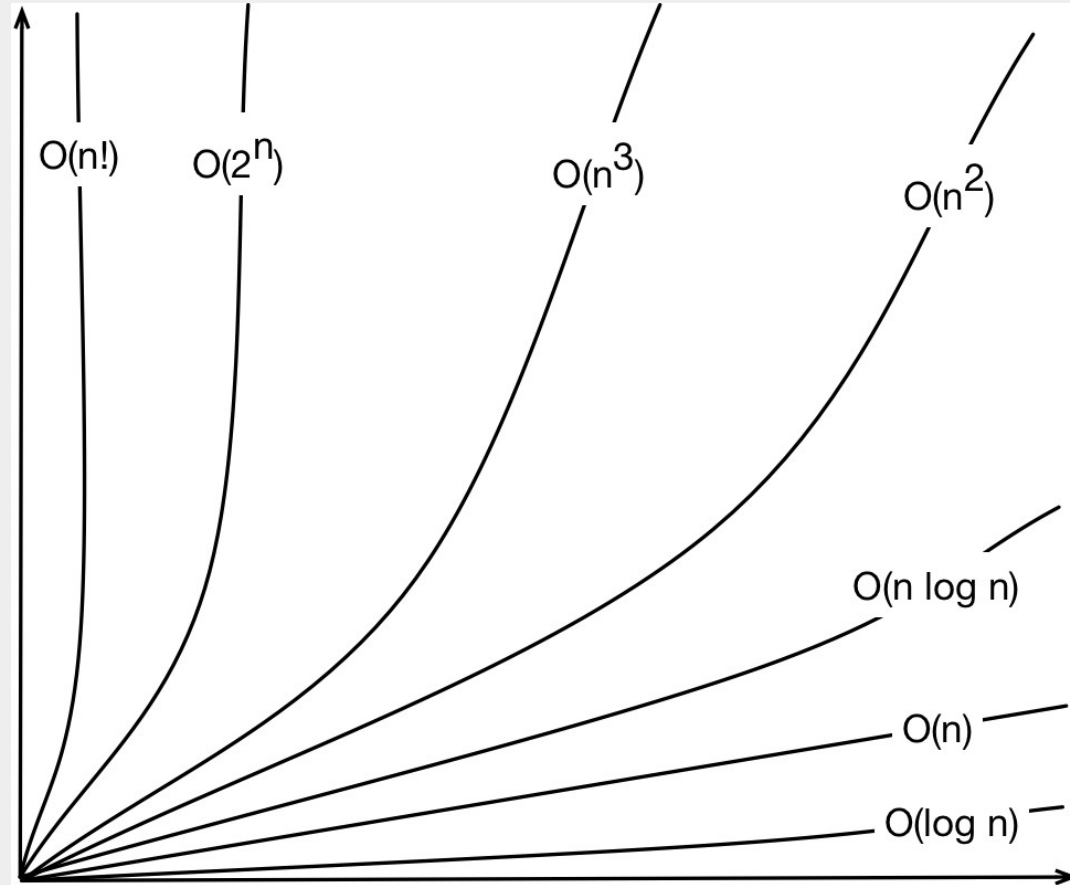
		Tamanho da entrada		
		10	30	50
Tempo de Execução	$n$	10 <sub>s</sub>	30 <sub>s</sub>	50 <sub>s</sub>
	$n^2$	100 <sub>s</sub>	900 <sub>s</sub>	41 <i>minutos</i>
	$2^n$	1024 <sub>s</sub>	34 <i>anos</i>	357 mil <i>séculos</i>

# Análise para um pior caso

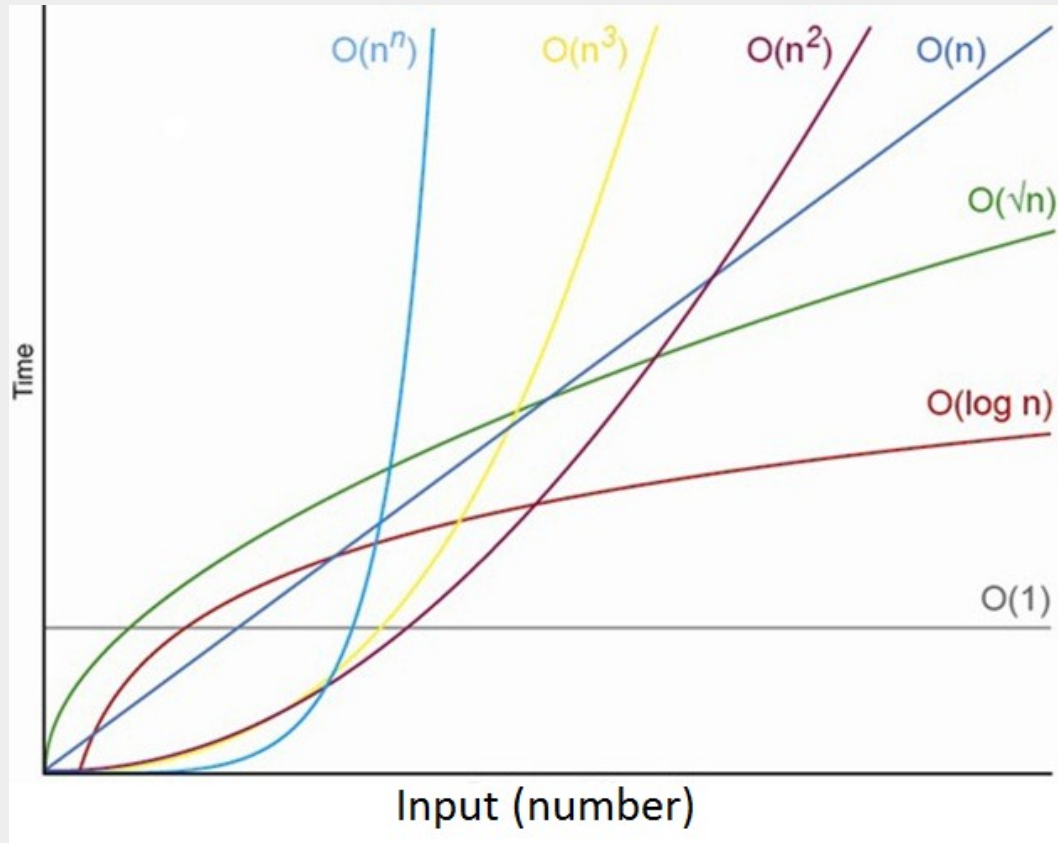
Ordem	Nome	Descrição
$O(1)$	constante	mais rápido, impossível
$O(\log n)$	logarítmico	muito bom
$O(n)$	linear	é o melhor que se pode esperar se algo não pode ser determinado sem examinar toda a entrada
$O(n \log n)$		limite de muitos problemas práticos. Ex: ordenação
$O(n^2)$	quadrático	muitos algoritmos simples de ordenação
$O(n^k)$	polinomial	ok se $n$ for pequeno
$O(k^n)$	exponencial	evite!

Ordem de crescimento

# Análise para um pior caso



# Análise para um pior caso



# Análise para um pior caso

**$O(1)$**

- De ordem constante.
- Independentemente do tamanho dos dados de entrada, a **quantidade** de instruções executadas é **sempre a mesma**.

# Análise para um pior caso

## $O(\log n)$

- De ordem logarítmica.
- Soluções que **dividem** o problema em **problemas menores**, processando a cada iteração, em geral, a metade dos dados da vez anterior.



# Análise para um pior caso

**$O(n)$**

- De ordem linear.
- O **mesmo número** de instruções é executado **para cada um** dos elementos de entrada.

# Análise para um pior caso

**$O(n\log(n))$**

- De log linear.
- Soluções de algoritmos que transformam o problema a ser resolvido em **problemas menores**, que por sua vez possuem **soluções independentes**.

# Análise para um pior caso

$O(n^2)$

- De ordem quadrática.
- Algoritmos que possuem **alinhamentos de laços** para trabalhar com os dados de entrada em pares.

# Análise para um pior caso

$O(n^3)$

- De ordem cúbica.
- Algoritmos que possuem **alinhamentos de três laços** para trabalhar com os dados de entrada.

# Análise para um pior caso

**$O(2^n)$**

- De ordem exponencial.
- Algoritmo de força bruta.

# Análise para um pior caso

**$O(n!)$**

- De ordem fatorial.
- Algoritmo de força bruta, com desempenho pior que o de ordem exponencial.

# Análise de algoritmos

- Exemplos de análise de algoritmos.

# Exemplos de análise de algoritmos

// Exemplo 1:


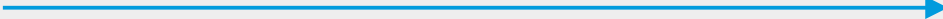

```
int x = 0;
```

```
x = x + 1;
```

```
printf("%d", resultado);
```



# Exemplos de análise de algoritmos

<code>int x = 0;</code>		<b>1</b>
<code>x = x + 1;</code>		<b>1</b>
<code>printf("%d", resultado);</code>		<b>1</b>

$$T(n) = 1 + 1 + 1 \rightarrow 3 \rightarrow O(1)$$

# Exemplos de análise de algoritmos

// Exemplo 2:

```
int encontrar_valor(int v[], int n, int c){  
    int p = -1;  
    int i = 0;  
    while(i < n && p == -1){  
        if(v[i] == c)  
            p = i;  
        i++;  
    }  
    return p;  
}
```

# Exemplos de análise de algoritmos

```
int encontrar_valor(int v[], int n, int c){
```

```
    int p = -1;           → 1
```

```
    int i = 0;           → 1
```

```
    while(i < n && p == -1){ → n + 1
```

```
        if(v[i] == c)     → n
```

```
            p = i;        → 0
```

```
            i++;          → n
```

```
    }
```

```
    return p;            → 1
```

```
}
```

$$T(n) = 3n + 4 \rightarrow n \rightarrow O(n)$$

# Exemplos de análise de algoritmos

## // Exemplo 3:

```
float calcula_ma(int n, float v[]){  
    float soma = 0.0;  
    media = 0.0;  
    for(int i=0; i < n; i++){  
        soma = soma + v[i];  
    }  
    media = soma / n;  
    return media;  
}
```

# Exemplos de análise de algoritmos

```
float calcula_ma(int n, float v[]){
```

```
    float soma = 0.0;            $\longrightarrow$  1
```

```
    media = 0.0;                  $\longrightarrow$  1
```

```
    for(int i=0; i < n; i++){  $\longrightarrow$  1 + (n + 1) + n
```

```
        soma = soma + v[i];}  $\longrightarrow$  n
```

```
    media = soma / n;            $\longrightarrow$  1
```

```
    return media;                 $\longrightarrow$  1
```

```
}
```

$$T(n) = 1 + 1 + 1 + n + 1 + n + n + 1 + 1 = 3n + 6 \rightarrow O(n)$$

# Exemplos de análise de algoritmos

// Exemplo 4:

```
int encontra_maior(int n, int v[]){  
    int maior = 0;  
    for(int i=1; i < n; i++)  
        if(v[i] > v[maior])  
            maior = i;  
    return maior;  
}
```

# Exemplos de análise de algoritmos

```
int encontra_maior(int n, int v[]){
```

```
    int maior = 0;            $\longrightarrow$  1
```

```
    for(int i=1; i < n; i++)  $\longrightarrow$   $1 + (n - 1 + 1) + n - 1$ 
```

```
        if(v[i] > v[maior])  $\longrightarrow$   $n - 1$ 
```

```
            maior = i;        $\longrightarrow$   $n - 1$ 
```

```
    return maior;            $\longrightarrow$  1
```

```
}
```

$$T(n) = 1 + 1 + n + n - 1 + n - 1 + n - 1 + 1 = 4n \rightarrow O(n)$$

**Fim da aula**