

# Teoria da Computação

Guilherme Henrique de Souza Nakahata

Universidade Estadual do Paraná - Unespar

10 de Abril de 2024

# Complexidade de Tempo

- Algoritmos de ordenação;
- Analisar a quantidade de operações;
- Bubblesort;

- Bubblesort:
  - Primitivo;
  - Percorre um vetor de  $n$  posições  $n$  vezes;
  - Compara dois elementos;
  - Troca caso o primeiro seja maior que o segundo.

# Ordenação Bolha (Bubble Sort)

---

**Algorithm 1** Ordenação Bolha

---

```
1: procedure BUBBLESORT( $a[]$ ,  $n$ )
2:   for  $i \leftarrow 1$  to  $n - 1$  do
3:     for  $j \leftarrow 2$  to  $n$  do
4:       if  $a[j - 1] > a[j]$  then
5:          $x \leftarrow a[j - 1]$ 
6:          $a[j - 1] \leftarrow a[j]$ 
7:          $a[j] \leftarrow x$ 
8:       end if
9:     end for
10:  end for
11: end procedure
```

---

# Pior Caso do Bubble Sort

	5	4	3	2	1	TROCAS
i=1, j=2	4	5	3	2	1	4
i=1, j=3	4	3	5	2	1	
i=1, j=4	4	3	2	5	1	
i=1, j=5	4	3	2	1	5	
i=2, j=2	3	4	2	1	5	3
i=2, j=3	3	2	4	1	5	
i=2, j=4	3	2	1	4	5	
i=3, j=2	2	3	1	4	5	2
i=3, j=3	2	1	3	4	5	
i=4, j=2	1	2	3	4	5	1

- Análise:
  - Execução interna ( $a[j-1] > a[j]$ ) vai ser executada  $(n-1).(n-1)$  vezes;
  - No pior caso a quantidade de trocas será:  
 $((n-1)+(n-2)+\dots+2+1)$ ;
  - $n(n-1)/2$  (Soma de uma série aritmética).

- Número de comparações;
- Número de trocas;
- Contabilizar todas as operações;
- Descrevendo a sua eficiência:
  - Temporal;
  - Espacial;
  - Tamanho do conjunto de dados de entrada.

- $n$  variável que descreve o tamanho de entrada de dados;
- $T(n)$  descreve a eficiência do algoritmo em relação ao tamanho dos dados de entrada.



# Comparação de algoritmos

- Como comparar dois algoritmos (a e b)?
- Se a complexidade for:
  - $T_a(n) = n^3$
  - $T_b(n) = 100 * n$
- Qual será o algoritmo mais eficiente?

# Crescimento de Funções

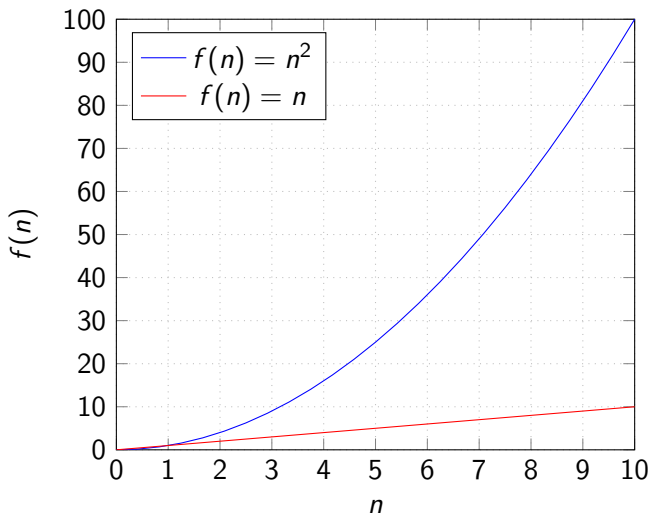


Figure: Gráfico representando o crescimento de funções  $f(n) = n^2$  e  $f(n) = n$ .

# Tipos de análise de eficiência

- Pior caso: tempo máximo de execução;
- Caso médio: tempo médio de execução (distribuição estatística dos dados de entrada);
- Melhor caso: resultado do menor tempo possível.

# Tipos de análise de eficiência

- Pesquisa sequencial;
- Encontrar um valor específico dentro de uma sequência de valores;
- Melhor caso: ?
- Pior caso: ?
- Caso médio: ?

# Tipos de análise de eficiência

- Pesquisa sequencial;
- Encontrar um valor específico dentro de uma sequência de valores;
- Melhor caso:  $T(n) = 1$ ;
- Pior caso:  $T(n) = n$ ;
- Caso médio:  $T(n) = n+1/2$ ;

# Caso Médio da Pesquisa Sequencial

- A probabilidade de encontrar o elemento em qualquer posição é  $\frac{1}{n}$ .
- Se o elemento estiver na posição  $i$  (onde  $1 \leq i \leq n$ ), então serão necessárias  $i$  comparações para encontrá-lo.

$$E[X] = \sum_{i=1}^n P(X = i) \cdot i$$

$$E[X] = \sum_{i=1}^n \frac{1}{n} \cdot i = \frac{1}{n} \sum_{i=1}^n i$$

A soma  $\sum_{i=1}^n i$  é uma série aritmética, e sua fórmula de soma é  $\frac{n \cdot (n+1)}{2}$ .

$$E[X] = \frac{1}{n} \cdot \frac{n \cdot (n+1)}{2} = \frac{n+1}{2}$$

- Princípio da indução finita;
- Princípio da indução;
- Prova matemática;
- Provar a correção:
  - Algoritmos recursivos;
  - Algoritmos iterativos;
    - Invariante de laços (Loop Invariant).



- Instrumento importante;
- Provar fatos referentes aos números naturais;
- Princípio da indução:
  - Seja  $P$  uma propriedade referente a números naturais. Se o elemento 1 atende a  $P$  e se, além disso, o fato de o número natural  $n$  atender a  $P$  implica que seu sucessor  $s(n)$  também atende, então todos os números naturais atendem a propriedade  $P$ ;

Provar que  $P(n) = \sum_{i=1}^n i = \frac{n(n+1)}{2}$ .

**Caso base:**  $P(1)$  é verdadeiro, pois  $P(1) = \frac{1(1+1)}{2} = 1$ .

**Hipótese da indução:** Assumimos que  $P(k) = \frac{k(k+1)}{2}$  é uma propriedade verdadeira para  $k \geq 1$ , então devemos provar para  $P(k+1)$ .

**Passo da indução:** Vamos provar para  $k + 1$ :

$$\begin{aligned}\sum_{i=1}^{k+1} i &= \left( \sum_{i=1}^k i \right) + (k + 1) \\&= \frac{k(k + 1)}{2} + (k + 1) \\&= \frac{k(k + 1) + 2(k + 1)}{2} \\&= \frac{k^2 + k + 2k + 2}{2} \\&= \frac{k^2 + 3k + 2}{2} \\&= \frac{(k + 1)(k + 2)}{2}\end{aligned}$$

**c.q.d.**

Provar que  $P(n) = \frac{1}{2} + \frac{1}{2^2} + \frac{1}{2^3} + \dots + \frac{1}{2^n} < 1$

**Caso base:**  $P(1)$  é verdadeiro, pois  $P(1) = \frac{1}{2^1} = \frac{1}{2} < 1$ .

**Hipótese da indução:** Assumimos que

$P(k) = \frac{1}{2} + \frac{1}{2^2} + \frac{1}{2^3} + \dots + \frac{1}{2^k} < 1$  é uma propriedade verdadeira para  $k \geq 1$ , então devemos provar para  $P(k+1)$ .

**Passo da indução:**  $P(k+1) = \frac{1}{2} + \frac{1}{2^2} + \frac{1}{2^3} + \dots + \frac{1}{2^{k+1}}$

**Passo da indução:**  $P(k+1) = \frac{1}{2} + \frac{1}{2^2} + \frac{1}{2^3} + \dots + \frac{1}{2^{k+1}}$

Reescrevendo:  $P(k+1) = \frac{1}{2} + \frac{1}{2} \left( \frac{1}{2} + \frac{1}{2^2} + \frac{1}{2^3} + \dots + \frac{1}{2^k} \right)$

**Passo da indução:**  $P(k+1) = \frac{1}{2} + \frac{1}{2^2} + \frac{1}{2^3} + \dots + \frac{1}{2^{k+1}}$

Reescrevendo:  $P(k+1) = \frac{1}{2} + \frac{1}{2} \left( \frac{1}{2} + \frac{1}{2^2} + \frac{1}{2^3} + \dots + \frac{1}{2^k} \right)$

Substituindo:  $\frac{1}{2} + \frac{1}{2^2} + \frac{1}{2^3} + \dots + \frac{1}{2^k}$  por  $\alpha$

**Passo da indução:**  $P(k+1) = \frac{1}{2} + \frac{1}{2^2} + \frac{1}{2^3} + \dots + \frac{1}{2^{k+1}}$

Reescrevendo:  $P(k+1) = \frac{1}{2} + \frac{1}{2}(\frac{1}{2} + \frac{1}{2^2} + \frac{1}{2^3} + \dots + \frac{1}{2^k})$

Substituindo:  $\frac{1}{2} + \frac{1}{2^2} + \frac{1}{2^3} + \dots + \frac{1}{2^k}$  por  $\alpha$

$$P(k+1) = \frac{1}{2} + \frac{1}{2}\alpha$$

$$P(k+1) = \frac{1}{2} + \frac{1}{2}\alpha < 1 \text{ c.q.d.}$$



Provar por indução que  $n! > 2^{n-1}$  para todo  $n \geq 3$ .

- **Base da Indução:** Para  $n = 3$ :

$$3! = 6 > 2^{3-1} = 2^2 = 4$$

- **Hipótese de Indução:** Afirmação verdadeira para algum  $k \geq 3$ , isto é,  $k! > 2^{k-1}$ .

- **Passo da Indução:** Vamos provar para  $n = k + 1$ :

$$(k + 1)! = (k + 1) \cdot k!$$

Se  $k! > 2^{k-1}$ , podemos realizar a substituição de  $k!$  por  $2^{k-1}$  (pela Hipótese de Indução).

$$(k + 1)! = (k + 1) \cdot k! > (k + 1)2^{k-1}$$

Dessa forma, temos  $(k + 1) \cdot 2^{k-1}$ .

Se substituirmos  $k + 1$  por 2, temos:

$$= 2 \cdot 2^{k-1} = 2^k = 2^{(k+1)-1}$$

Logo, concluímos que, por indução,  $n! > 2^{n-1}$  para todo  $n \geq 3$ .

- Provar que  $n! > 2^{n-1}$  para todo  $n \geq 3$

**Caso base:**

**Hipótese:**

**Passo da indução:**

- Provar que  $2^n > n$  para todo  $n \geq 1$

**Caso base:**

**Hipótese:**

**Passo da indução:**

- Provar que  $n! > n^2$  para todo  $n \geq 4$

**Caso base:**

**Hipótese:**

**Passo da indução:**

- **Grande O ( $O$ )**: Descreve o limite superior do tempo de execução.
- **Omega ( $\Omega$ )**: Descreve o limite inferior do tempo de execução.
- **Theta ( $\Theta$ )**: Descreve tanto o limite superior quanto o limite inferior.
- **Omicron ( $o$ )**: Descreve o limite superior assintótico restrito.
- **Little Omega ( $\omega$ )**: Descreve o limite inferior assintótico restrito.

- LEWIS, H. R.; PAPADIMITRIOU, C. H. **Elementos de Teoria da Computação**. 2 ed. Porto Alegre: Bookman, 2000.
- VIEIRA, N. J. **Introdução aos Fundamentos da Computação**. Editora Pioneira Thomson Learning, 2006.
- DIVERIO, T. A.; MENEZES, P. B. **Teoria da Computação: Máquinas Universais e Computabilidade**. Série Livros Didáticos Número 5, Instituto de Informática da UFRGS, Editora Sagra Luzzato, 1 ed. 1999.

# Obrigado! Dúvidas?

Guilherme Henrique de Souza Nakahata

[guilhermenakahata@gmail.com](mailto:guilhermenakahata@gmail.com)

<https://github.com/GuilhermeNakahata/UNESPAR-2024>