Aula 1

# Estrutura de Dados

Prof. Vinicius Pozzobon Borin

## Conversa Inicial

- O objetivo desta aula é introduzir os principais conceitos inerentes a esta disciplina de *Estrutura de Dados*
- Tais conceitos serão recorrentes ao longo das próximas aulas

- A estrutura de conteúdos desta aula é a seguinte:
  1. O que são estrutura de dados?
  2. Análise de complexidade de algoritmos
  3. Análise assintótica de algoritmos
  4. Recursividade e o impacto na complexidade

## Definição – estrutura de dados

- Dado atômico é aquele no qual o conjunto de dados manipulados é indivisível, ou seja, estes são tratados como sendo um único valor

```
1  Algoritmo "AULA1_Dados_Simples"
2  var
3     x, y, z: inteiro // DADOS SIMPLES
4  inicio
5     x = 5
6     y = 1
7     z = x+ y // MANIPULAÇÃO SIMPLES
8  fimalgoritmo
```

- Dados complexos são aqueles cujos elementos do conjunto de valores podem ser decompostos em partes mais simples
- Se um dado pode ser dividido, significa que ele apresenta algum tipo de organização estruturada e, portanto, é chamado de dado estruturado, o qual faz parte de uma estrutura de dados

## Estruturas de dados

- Homogêneas: são aquelas que manipulam um só tipo de dado. Exemplo: vetores e matrizes


- Heterogêneas: são capazes de manipular mais de um tipo de dados. Exemplo: registros

- Os dados estruturados, as relações entre eles, suas organizações, projetos, usos e aplicações constituem esta disciplina de *Estrutura de Dados*

## Análise de complexidade de algoritmos

- Ao longo desta disciplina, iremos aprender diversos algoritmos para solucionar distintos problemas envolvendo manipulação de estruturas de dados
- Mas... Como poderemos saber:
  - Qual deles é o mais eficiente para solucionar determinado problema?
  - Quais parâmetros de desempenho devemos analisar?

## Parâmetros de desempenho

- Tempo de execução: quanto tempo um código levou para ser executado
- Uso de memória volátil: a quantidade de espaço ocupado na memória principal do computador

- **Quanto maior o conjunto de dados de entrada, maior tenderá a ser o impacto do algoritmo em seu tempo de execução, sendo essencial um algoritmo eficaz para a execução de tal tarefa**

## Função de custo do algoritmo

- **Podemos encontrar matematicamente o custo de um algoritmo encontrando uma equação que descreve o seu comportamento em relação ao desempenho do algoritmo**

**Não será analisada nesta disciplina**

$$T(n) = T_{Tempo} + T_{Espaço}$$

**n : tamanho do conjunto de dados**

```
1  algoritmo "AULA1_Laço_Vazio"
2  var
3     i, n: inteiro
4  inicio
5     n = 10
6     para i de 0 até n faça
7        //LAÇO VAZIO
8     fimpara
9  fimalgoritmo
```

| Linha | Motivo | Total de Instruções |
|---|---|---|
| 5 | Atribuição de valor | 1 |
| 6 | Atribuição de valor (i=0) e comparação (i<n) | 2 |
| 6-loop | Comparação (i<n) e incremento (i++) | 2n |

$$T(n) = 2n + 3$$

```
1  algoritmo "AULA1_Maior_Valor"
2  var
3     i, Maior: inteiro
4     valores: vetor [1..10] de inteiro
5  inicio
6
7     Maior = 0
8     para i de 0 até 10 faça
9        se (Valores[i] >= Maior) // Testa se o valor do vetor é maior
10          Maior = Valores[i] // Atribui o valor como sendo o maior
11       fimse
12    fimpara
13
14 fimalgoritmo
```

| Linha | Motivo | Total de Instruções |
|---|---|---|
| 7 | Atribuição de valor | 1 |
| 8 | Atribuição de valor (i=0) e comparação (i<n) | 2 |
| 8-loop | Comparação (i<n) e incremento (i++) | 2n |
| 9 | Comparação | 1 |
| 10 | Atribuição de valor | 1 |

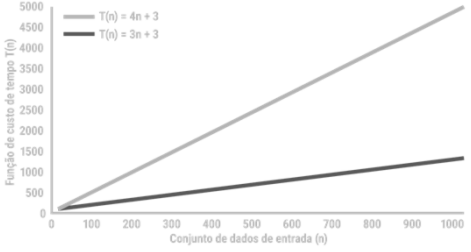
- **Vetor ordenado de forma crescente: condicional sempre verdadeira**

  $Vetor = [1, 2, 3, 4];$ $T(n) = 4n + 3$

- **Vetor ordenado de forma decrescente: condicional sempre falsa**

  $Vetor = [4, 3, 2, 1];$ $T(n) = 3n + 3$

## Comparativo gráfico da complexidade do algoritmo para o melhor e pior caso



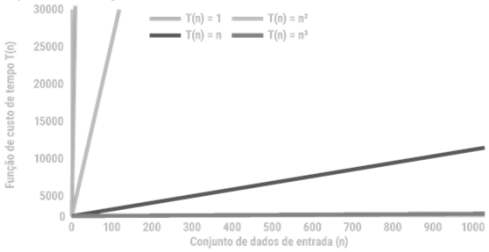
## Análise assintótica de algoritmos

- Neste tipo de análise encontramos uma curva de tendência aproximada do desempenho de um algoritmo
- A análise baseia-se na extrapolação do conjunto de dados de entrada, fazendo com que estes tendam ao infinito, de modo que seja possível negligenciar alguns termos das equações

- Descartamos das equações os termos que crescem lentamente à medida que o conjunto de dados de entrada tende ao infinito

| Função custo | Comportamento assintótico | Algoritmo |
|---|---|---|
| $T(n) = 10$ | 1 | Sequencial |
| $T(n) = 10n + 2$ | n | 1 laço |
| $T(n) = 10n^2 + 5n + 2$ | $n^2$ | 2 laços |
| $T(n) = 10n^3 + 50n^2 + n + 1$ | $n^3$ | 3 laços |

## Comparativo da complexidade do algoritmo para diferentes casos assintóticos




## Notações da análise assintótica

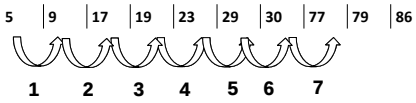
- Grande-O (Big-O):
  - Define o comportamento assintótico superior
  - É o pior caso de um algoritmo
  - Mais instruções sendo executadas
- Grande-Ômega:
  - Define o comportamento assintótico inferior
  - É o melhor caso do algoritmo (caso ótimo)
  - Menos instruções sendo executadas

## Recursividade

- A recursão é o processo de definição de algo em termos de si mesmo (Laureano, 2008)
- Um algoritmo recursivo é aquele que utiliza a si mesmo para atingir algum objetivo, ou obter algum resultado
- Em programação, a recursividade está presente quando uma função realiza chamadas a si mesma

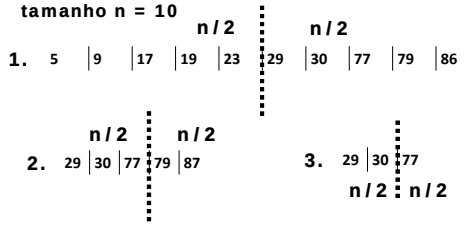
## Complexidade da recursividade

- Possibilidade 1: buscar o valor 77 no vetor de tamanho n = 10




## Complexidade da recursividade

- Possibilidade 2: buscar o valor 77 no vetor de tamanho n = 10




| Problema original | n | $n/2^x$ |
|---|---|---|
| Passo 1 | n/2 | $n/2^1$ |
| Passo 2 | n/4 | $n/2^2$ |
| Passo 3 | n/8 | $n/2^3$ |
| Passo 4 | n/16 | n/24 |

Problema inicial: $\frac{n}{2^k} = 1$

Resolvendo o problema:

$n = 2^k$

$log\, n = \log 2^k$

$log\, n = \mathrm{k}.\log 2$

$k = \frac{\log n}{\log 2}$

$log_n m = \frac{\log_x m}{\log_x n}$

$k = \frac{\log n}{\log 2} \rightarrow k = \log_2 n$

Complexidade Big-O será:

$O(\log n)$

## Exemplo clássico de recursividade – fatorial

## Fatorial iterativo

```
21 Fatorial_Iterativo (n:inteiro): inteiro
22 var
23    fat: inteiro
24 iniciofuncao
25    fat = 1
26    para i de 1 até n faça
27       fat = fat * i
28    fimpara
29
30    retorne fat
31 fimfuncao
```

$O(n)$ ⟶ 1 laço de repetição

## Fatorial recursivo

```
21 Fatorial_Recursivo (n:inteiro): inteiro
22 var
23    fat: inteiro
24 iniciofuncao
25    fat = 1
26    se (n == 0) então
27       retorne fat
28    senão
29       fat = n * Fatorial_Recursivo (n – 1)
30    fimse
31 fimfuncao
```

$O(logn)$ ⟶ recursividade

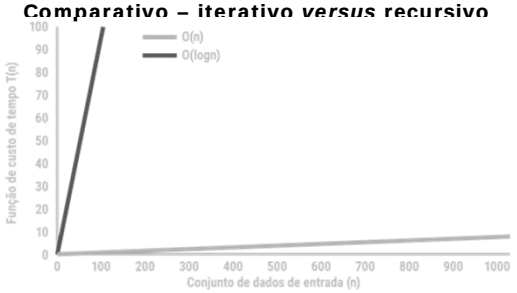
## Fatorial recursivo

- Exemplo de fatorial de 5

| Chamada | LINHA 29 |
|---|---|
| 1 | Fatorial_Recursivo(5) |
| 2 | 5 * Fatorial_Recursivo(4) |
| 3 | 5 * 4 * Fatorial_Recursivo(3) |
| 4 | 5 * 4 * 3 * Fatorial_Recursivo(2) |
| 5 | 5 * 4 * 3 * 2 * Fatorial_Recursivo(1) |
| 6 | 5 * 4 * 3 * 2 * 1 * Fatorial_Recursivo(0) |

$O(logn)$ ⟶ recursividade

## Comparativo – iterativo *versus* recursivo




## Referências


- ASCENCIO, A. F. G. Estrutura de dados: algoritmos, análise da complexidade e implementações em JAVA e C/C++. São Paulo: Pearson, 2011.
- ASCENCIO, A. F. G. Fundamentos da programação de computadores: algoritmos, Pascal, C/C++ (padrão ANSI) JAVA. 3. ed. São Paulo: Pearson, 2012.
- CORMEN, T. H. Algoritmos: teoria e prática. 3. ed. Elsevier, 2012.
- PUGA, S.; RISSETI, G. Lógica de programação e estrutura de dados. 3. ed. São Paulo: Pearson, 2016
- MIZRAHI, V. V. Treinamento em linguagem C. 2. ed. São Paulo: Pearson, 2008
- LAUREANO, M. Estrutura de dados com algoritmos E C. Brasport, 2008.