



# **ESTRUTURA DE DADOS NÃO LINEARES**

Prof. Renato Matroniani



**EDUCAÇÃO  
METODISTA**

# ESTRUTURA DE DADOS

➤ **Listas, filas, pilhas e árvores e Recursividade.**



EDUCAÇÃO  
METODISTA

# Listas

- De acordo com Puga e Risseti (2010) **“Quando falamos em listas, filas, pilhas e árvores podemos dizer que todas são na verdade 'listas de informações', cuja diferença principal está no acesso a essas 'listas' para inclusão e remoção de informações”**.





# Listas, vetores e matrizes

- Ainda segundo com Puga e Riseti (2010) “Os arranjos (vetores ou matrizes) são mais simples de implementar: o conteúdo da 'lista' é armazenado em um espaço de memória com tamanho para  $N$  elementos que serão dispostos em posições contínuas [...]. Mas, [...] os arranjos possuem limitação quanto à quantidade de elementos que o conjunto irá suportar [...], além de que pode haver a necessidade de mais espaço do que aquele que foi inicialmente reservado”.



# Listas, vetores e matrizes

- Então uma das primeiras diferenças que notamos entre as listas e outras estruturas de dados é a forma de endereçamento.
- Outras formas, como as pilhas, utilizam também, mas de forma desordenada, posições não sequenciais na RAM, e sim, **endereçamento**.



# Definição de listas

- Coleção de elementos do mesmo tipo dispostos linearmente, que podem ou não seguir uma determinada orientação.
- $[E1, E2, E3, E4, E5, \dots, E_n]$
- Exemplos: listas de chamadas de alunos, lista de compras, lista de pagamentos.

*Texto baseado em Puga e Riseti (2010).*



**EDUCAÇÃO  
METODISTA**



# Listas simples x listas encadeadas

Lista de pagamentos
Prestação do carro
Cartão de crédito
Conta de luz
Condomínio
TV a cabo
Crediário das Casas Bahia

*lista simples*

Lista de pagamentos	Ponteiro para o próximo elemento
Prestação do carro	2
Cartão de crédito	3
Conta de luz	4
Condomínio	5
TV a cabo	6
Crediário das Casas Bahia	Obs.: este é o último elemento do conjunto, então não aponta para nenhum outro.

*lista com um campo para encadeamento*

Texto baseado em Puga e Riseti (2010).



**EDUCAÇÃO  
METODISTA**

# Definição de listas

Lista de pagamentos
Prestação do carro
Cartão de crédito
Conta de luz
Condomínio
TV a cabo
Crediário das Casas Bahia

*lista simples*

- Lista implementada através de arranjo – pode-se utilizar vetor ou matriz.
- Ideia de contêiner de armazenamento de dados.
- ou....

Texto baseado em Puga e Riseti (2010).



**EDUCAÇÃO  
METODISTA**



# Definição de listas

Lista de pagamentos	Ponteiro para o próximo elemento
Prestação do carro	2
Cartão de crédito	3
Conta de luz	4
Condomínio	5
TV a cabo	6
Crediário das Casas Bahia	Obs.: este é o último elemento do conjunto, então não aponta para nenhum outro.

*lista com um campo para encadeamento*

- Lista implementada através de alocação dinâmica;
- Lista encadeada.

*Texto baseado em Puga e Riseti (2010).*



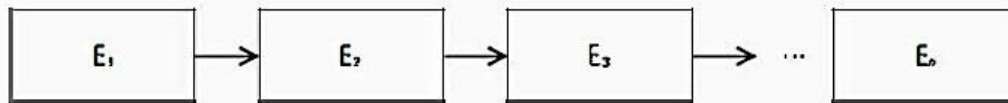
**EDUCAÇÃO  
METODISTA**

# Definição de listas

Lista de pagamentos
Prestação do carro
Cartão de crédito
Conta de luz
Condomínio
TV a cabo
Crediário das Casas Bahia

*lista simples*

- Em estrutura de dados, precisamos dizer, dentro de uma lista, qual será o próximo elemento.
- Cada elemento da lista é representado por um nó.



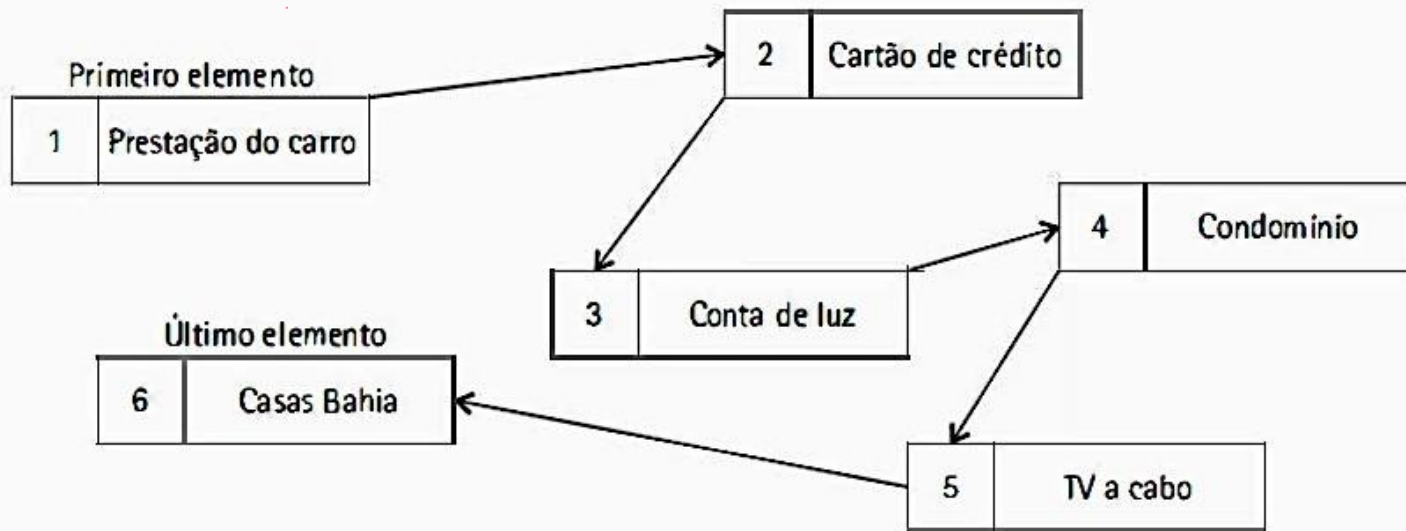
| FIGURA 10.1 | Listas encadeadas

Texto baseado em Puga e Riseti (2010).



**EDUCAÇÃO  
METODISTA**

# Listas encadeadas – exemplo hipotético



[ FIGURA 10.2 ] Listas encadeadas

Texto baseado em Puga e Riseti (2010).



**EDUCAÇÃO  
METODISTA**



# Tipos de Listas encadeadas

- **Encadeamento simples:** os elementos da lista possuem apenas um ponteiro que aponta para o elemento sucessor ou próximo (como no exemplo apresentado anteriormente).
- **Duplamente encadeadas:** cada elemento possui um campo que aponta para o seu predecessor (anterior) e outro para o seu sucessor (próximo).

*Texto baseado em Puga e Riseti (2010).*



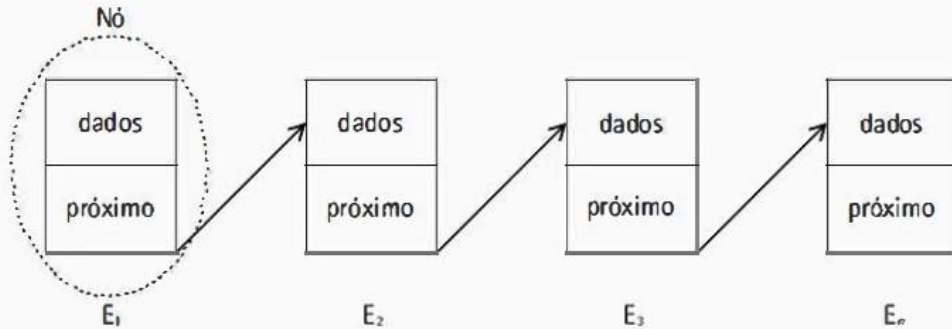
**EDUCAÇÃO  
METODISTA**

# Tipos de Listas encadeadas

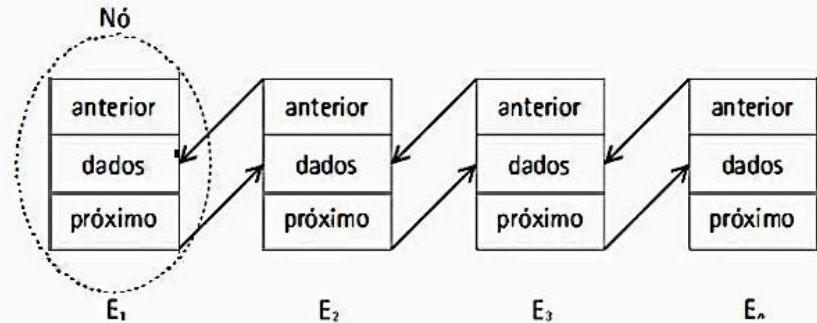
- **Ordenadas:** a ordem linear da lista corresponde à ordem linear dos elementos, isto é, quando um novo elemento é inserido na lista ele deve ser colocado em tal posição que garanta que a ordem da lista será mantida; essa ordem pode ser definida por um campo da área de dados, como por exemplo se tivermos uma lista ordenada com seguintes valores [1, 5, 7, 9] e desejarmos incluir um novo elemento com o valor 6, este valor deverá ser incluído entre os valores 5 e 7.
- **Circulares:** o ponteiro próximo do último elemento aponta para o primeiro; e o ponteiro anterior do primeiro elemento aponta para o último



# Tipos de Listas encadeadas



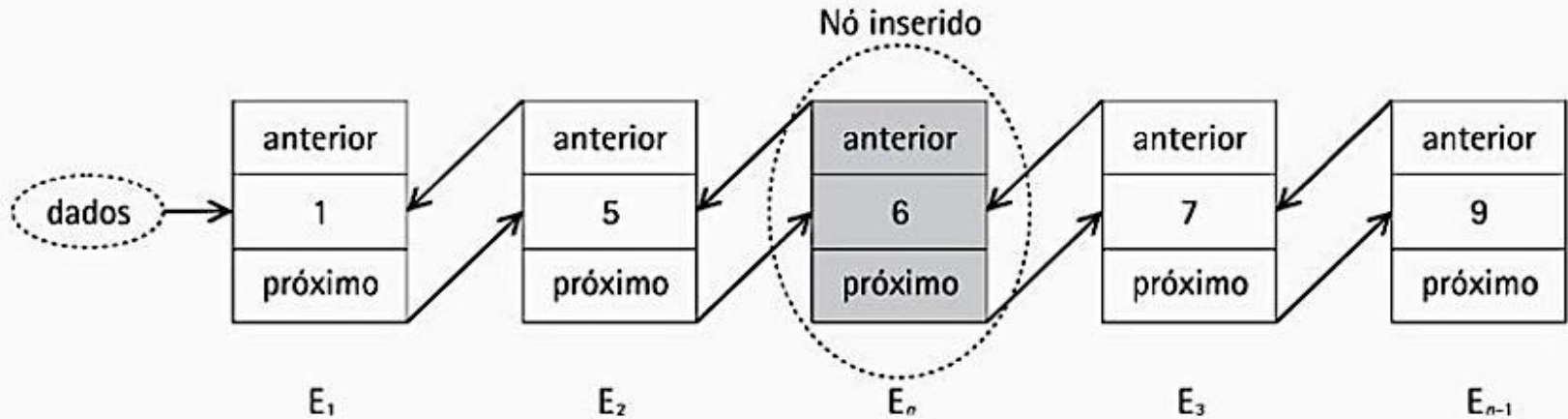
| FIGURA 10.3 | Listas com encadeamento simples



| FIGURA 10.4 | Listas duplamente encadeadas



# Tipos de Listas encadeadas



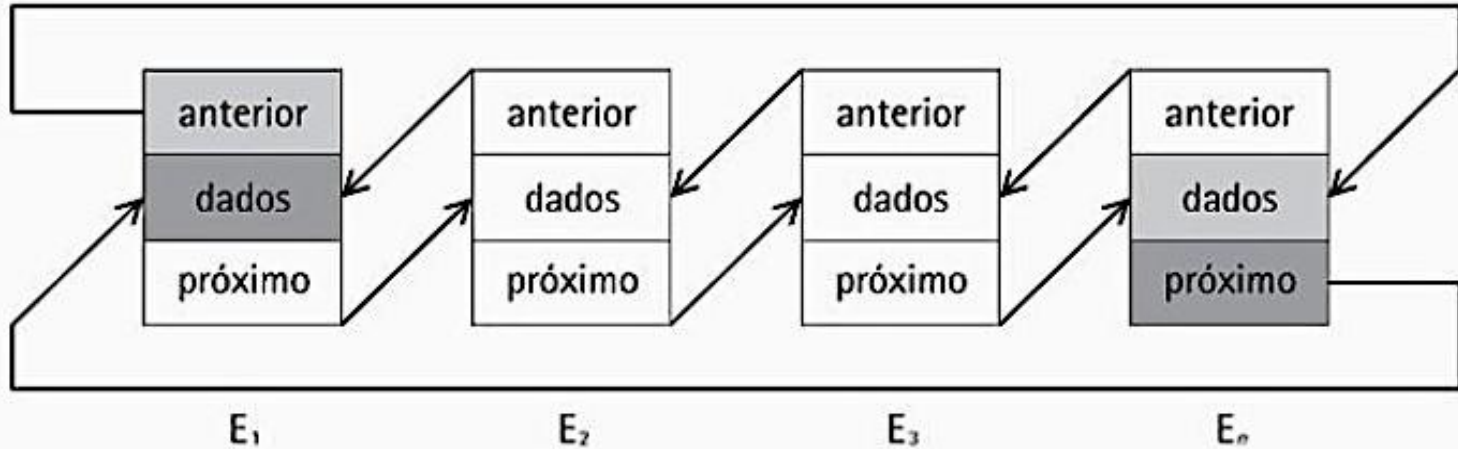
| FIGURA 10.5 | Listas ordenadas



**EDUCAÇÃO  
METODISTA**

Texto baseado em Puga e Riseti (2010).

# Tipos de Listas encadeadas



| FIGURA 10.6 | Listas circulares



**EDUCAÇÃO  
METODISTA**

Texto baseado em Puga e Riseti (2010).

# Listas duplamente encadeadas

- As listas duplamente encadeadas permitem fazer o caminho inverso, quando há a necessidade de percorrer a lista em ambas as direções.
- O que muda em relação a uma lista simples é que nas encadeadas existem uma variável que faz referência ao elemento anterior.
- No código (aqui um pseudocódigo de exemplo), observe que há a inserção de “prox” e também de “ant”.



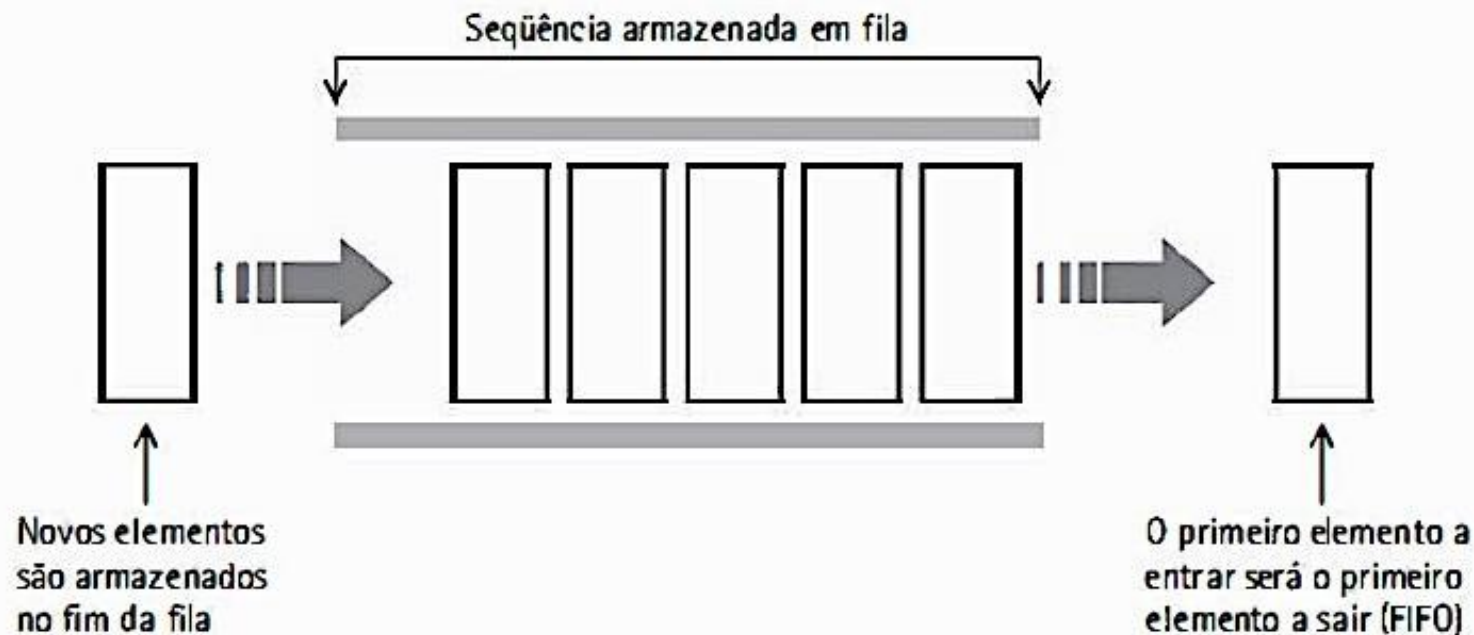


# Filas

- O conceito de filas em programação é o mesmo que o conceito “popular” de filas.
- O primeiro elemento a entrar na fila será o primeiro elemento a sair (**bem, na teoria...**) → FIFO
- As filas ou *queues*, em inglês, são listas onde as operações de inserção é feita em uma extremidade, e a de remoção, por outra.



# Filas



| FIGURA 10.11 | *Conceito de fila*



EDUCAÇÃO  
METODISTA

# Implementação de filas

- Uma fila implementada por “arranjo” precisa possuir as informações iniciais da fila, as informações finais e um vetor ou matriz, que chamamos de “contêiner”, onde cada posição na fila é indexada.



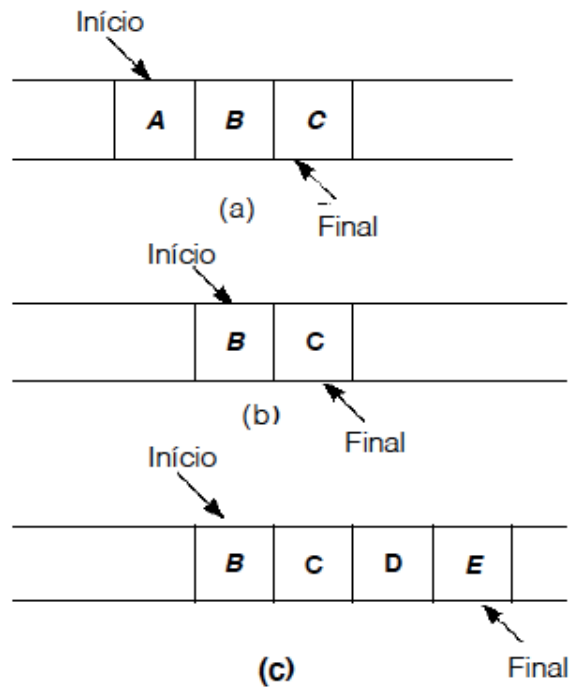


# Filas (queue /kyoo /)

- **fila:** conjunto ordenado de itens onde pode-se eliminar itens de uma extremidade e inserir itens em outra extremidade (início e final da fila, respectivamente).
- conceitos *fifo* e *lifo*.
- operações com filas: insert, remove e empty (checar se há elementos).



# Filas (continuação)



# Operações com filas

- Tratando do problema com tabelas...
- Suponha que a fila tenha capacidade para 5 elementos *int* (*variável do tipo inteiro*).

posição	0	1	2	3	4
valor	45	7	9	32	1
organização	primeiro elemento				último elemento

Texto baseado em Puga e Riseti (2010).



**EDUCAÇÃO  
METODISTA**



# Operações com filas

- Para retirar um elemento dessa fila, ele deve ser o primeiro (posição 0), e o valor 7 passa a ser então o primeiro elemento.

posição	1	2	3	4
valor	7	9	32	1
organização	primeiro elemento Início			último elemento Fim

*Texto baseado em Puga e Riseti (2010).*



**EDUCAÇÃO  
METODISTA**

# Operações com filas

- Como o tamanho da fila é 5, temos então uma posição vaga.

posição	1	2	3	4
valor	7	9	32	1
organização	primeiro elemento Início			último elemento Fim


*Texto baseado em Puga e Riseti (2010).*



**EDUCAÇÃO  
METODISTA**

# Operações com filas

- Quando um novo valor for inserido será armazenado na posição 0, que é o final da fila, e o último elemento será este.



posição	1	2	3	4	0
valor	7	9	32	1	novo valor
organização	primeiro elemento Início			último elemento	Fim

Texto baseado em Puga e Riseti (2010).



**EDUCAÇÃO  
METODISTA**



# Filas circulares

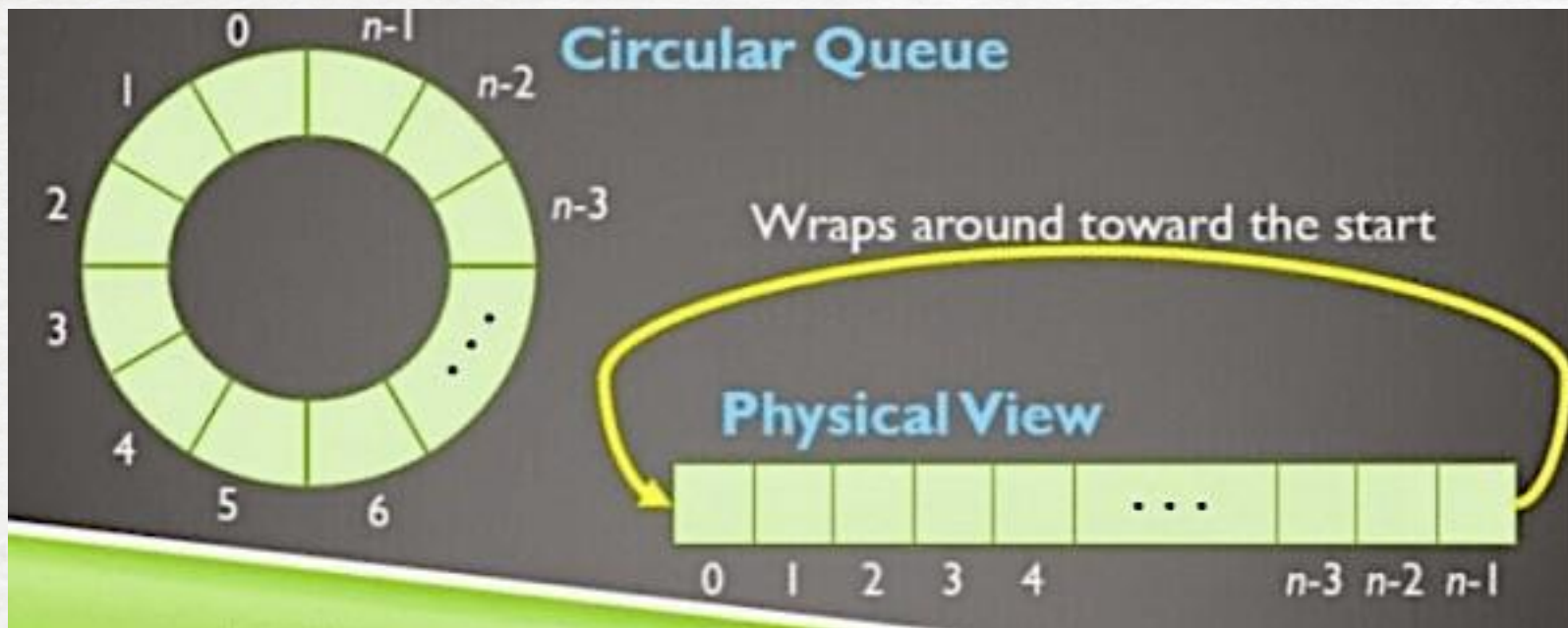
- Há situações onde as filas não apresentam uma solução completa.
- A fila pode estar cheia de “espaços vazios” devido a eliminação de elementos.
- As filas circulares resolvem esse problema e são mais eficientes, pois utilizam menos instruções.
- O motivo disso é que em uma fila circular “onde o último elemento está adjacente ao primeiro”
- Exemplo: sequências que vão de 000 a 999 e depois voltam a 000.

*Texto baseado em Puga e Riseti (2010).*



**EDUCAÇÃO  
METODISTA**

# Filas circulares



Fonte: <https://www.embarcados.com.br/fila-circular-para-sistemas-embarcados/>



**EDUCAÇÃO  
METODISTA**

# Filas circulares

- Não há comprimento fixo.
- O tamanho limite é definido pela capacidade de memória alocada para esse fim.
- Seguem o FIFO como as filas normais.
- Se pensarmos no tamanho do vetor, como trabalhar o limite dessa fila circular em um algoritmo?

*Texto baseado em Puga e Riseti (2010).*



**EDUCAÇÃO  
METODISTA**



# Pilhas – Definição e Características

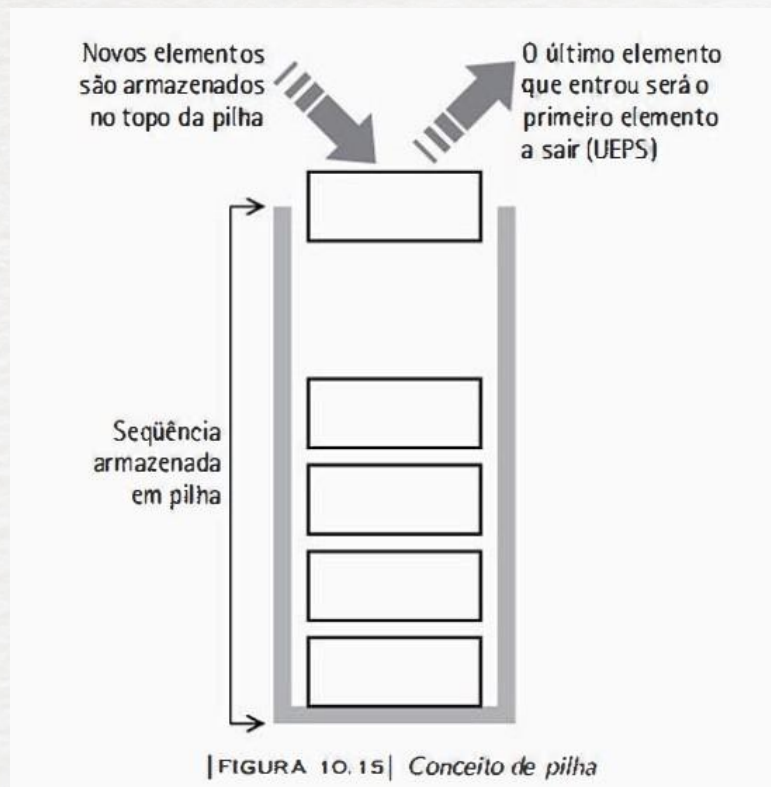
- “As pilhas também são conhecidas como lista **LIFO** (Last In, First Out), que em português significa “último a entrar e primeiro a sair” (UEPS).
- É uma lista linear em que todas as operações de inserção e remoção são feitas por um único extremo denominado topo”.
- “A operação de inserção é denominada empilhamento e a de exclusão, desempilhamento”.
- Assim como as filas, as pilhas são conjuntos dinâmicos onde o elemento removido do conjunto [delete] é especificado previamente. Mas a semelhança acaba aí, justamente pela questão FIFO x LIFO.
- A inserção ou exclusão ocorre apenas em uma das extremidades!

*Texto baseado em Puga e Riseti (2010).*



**EDUCAÇÃO  
METODISTA**

# Pilhas - Definição



# Pilhas – conceito de PUSH e POP

- A operação **INSERT** em uma pilha é frequentemente denominada **PUSH**, e a operação **DELETE**, que não toma um argumento de elemento, é frequentemente denominada **POP** → relação com as pilhas físicas.

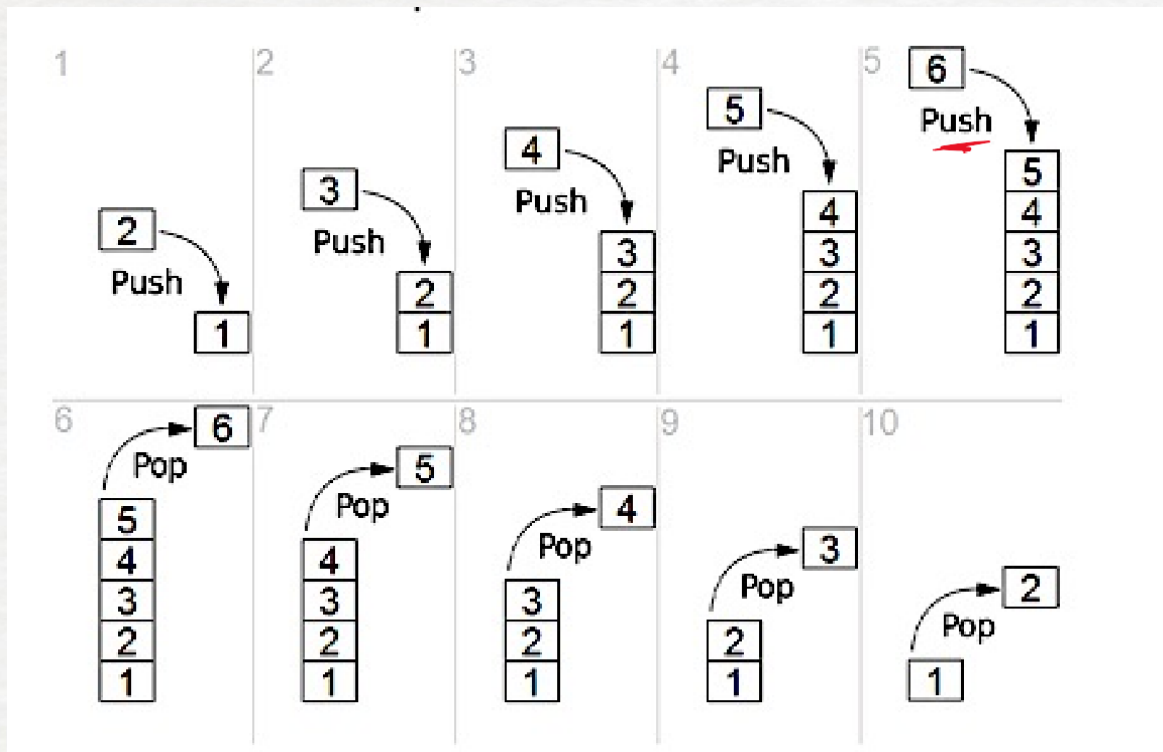


*Texto baseado em Puga e Riseti (2010).*



**EDUCAÇÃO  
METODISTA**

# Pilhas – conceito de PUSH e POP



Fonte: <https://osprogramadores.com/blog/2017/09/10/estruturas-dados-pilha/>



EDUCAÇÃO  
METODISTA



# Pilhas

- Considere uma pilha com as seguintes características:
  - número máximo de elementos  $n$ ;
  - $S.topo$  indexa o elemento mais recente inserido.  $S[1]$  seria o elemento mais abaixo na pilha, por exemplo.
  - Se  $S.topo = 0$  não contém nenhum elemento e está vazia. Se  $S.topo > n$ , a pilha “estoura”.

# Pilhas

- Algumas operações em um determinado pseudo-código:
  - `stack-empty(S)`: verifica se a pilha está vazia.
  - `push(S, x)`: inserção de elemento na pilha.
  - `pop(S)`: deletar elemento da pilha



# Pilhas – exemplificando com tabelas

posição	valor
0	45

topo

inserção

posição	valor
1	7
0	45

topo

inserção

posição	valor
4	1
3	32
2	9
1	7
0	45

topo

remoção

posição	valor
4	
3	32
2	9
1	7
0	45

topo



Texto baseado em Puga e Riseti (2010).

**EDUCAÇÃO  
METODISTA**

# ESTRUTURA DE DADOS

➤ Listas, filas, pilhas, **árvores** e Recursividade.



EDUCAÇÃO  
METODISTA



# Árvores – Definição

- Ao contrário das listas, filas e pilhas, as árvores são estruturas de dados bidimensionais, não lineares, com propriedades especiais.
- Admitem operações de conjuntos dinâmicos, como consulta, inserção e remoção.

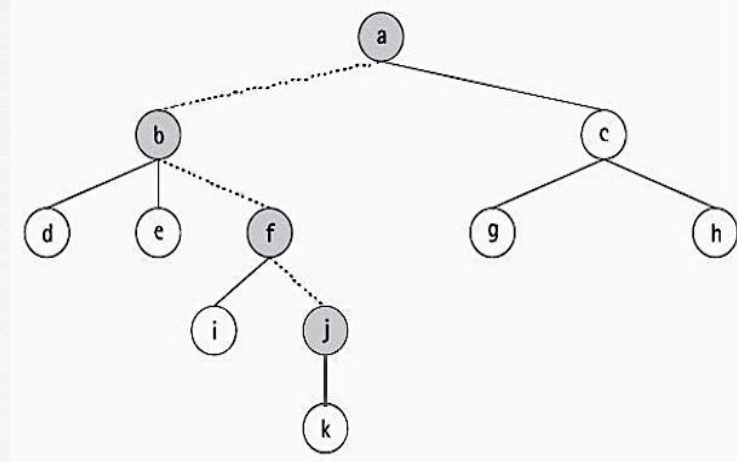
*Texto baseado em Puga e Riseti (2010).*



**EDUCAÇÃO  
METODISTA**

# Árvores – Termos e Características

- nó raiz - nó do topo da árvore, do qual descendem os demais nós. É o primeiro nó da árvore;
- nó interior - nó do interior da árvore (que possui descendentes);
- nó terminal - nó que não possui descendentes;
- trajetória- número de nós que devem ser percorridos até o nó determinado;



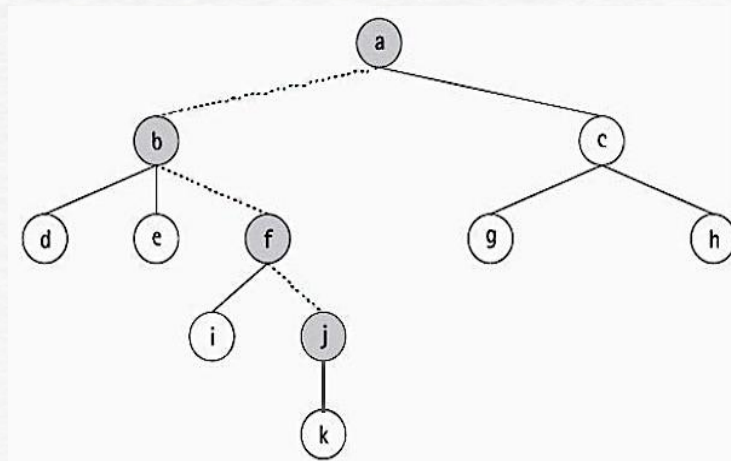
Texto baseado em Puga e Riseti (2010).



**EDUCAÇÃO  
METODISTA**

# Árvores – Termos e Características

- grau do nó - número de nós descendentes do nó, ou seja, o número de subárvores de um nó;
- grau da árvore - número máximo de subárvores de um nó;
- altura da árvore - número máximo de níveis dos seus nós;
- altura do nó - número máximo de níveis dos seus nós.



Texto baseado em Puga e Riseti (2010).



**EDUCAÇÃO  
METODISTA**

# Árvores – Tipos

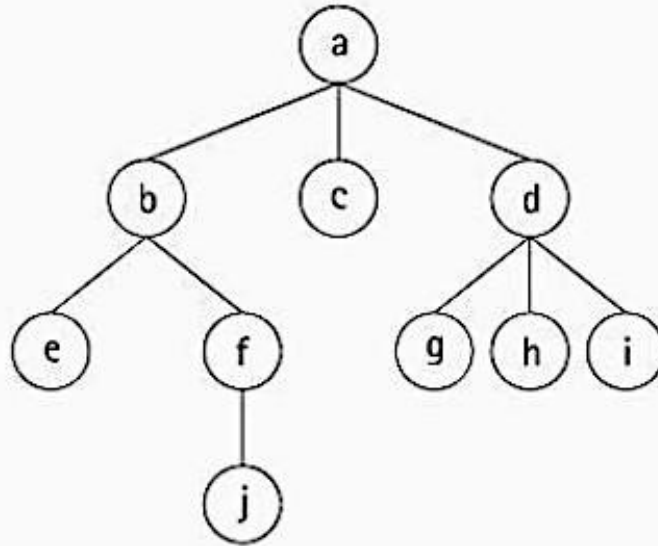
- listas generalizadas – possuem nós com grau  $\geq 0$
- binárias – possuem nós com grau  $\leq 2$





# Árvores – Tipos

Árvore como lista generalizada

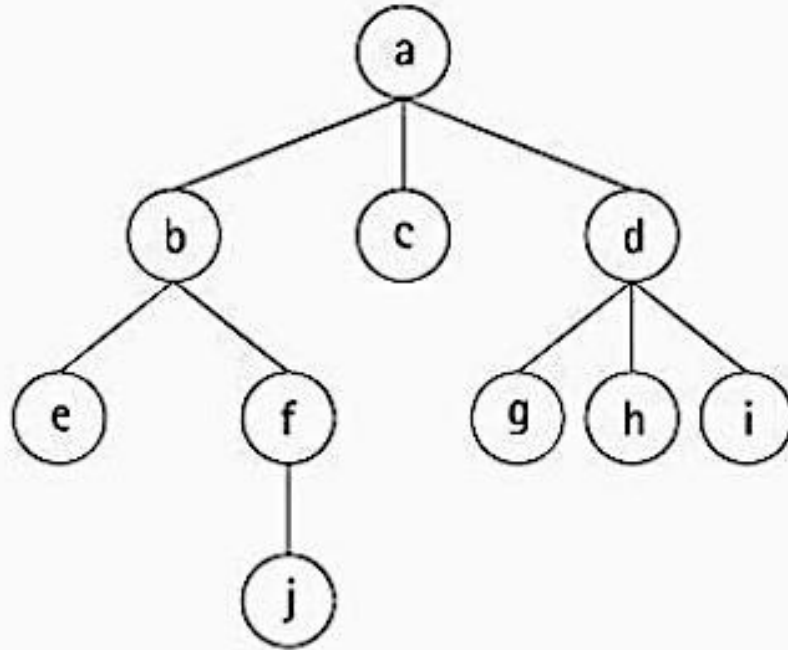


*Texto baseado em Puga e Riseti (2010).*



**EDUCAÇÃO  
METODISTA**

# Árvores – Lista generalizada

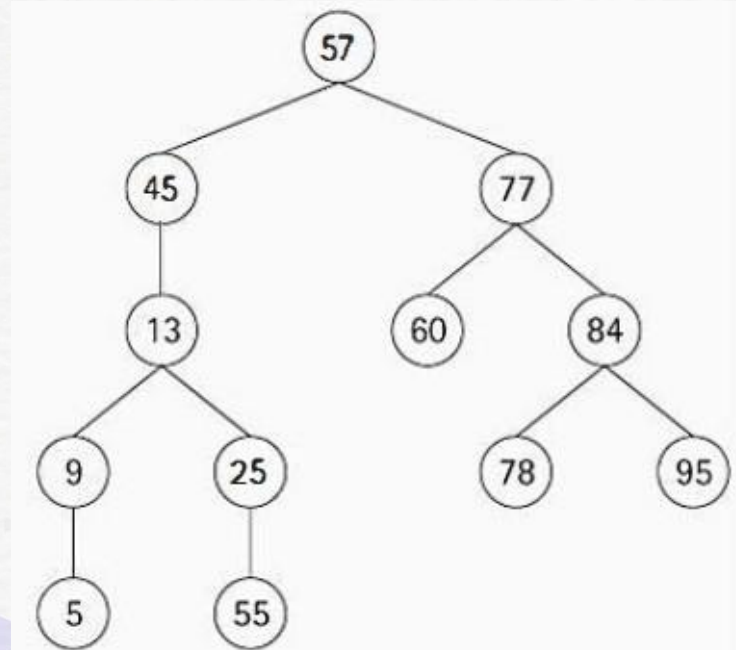
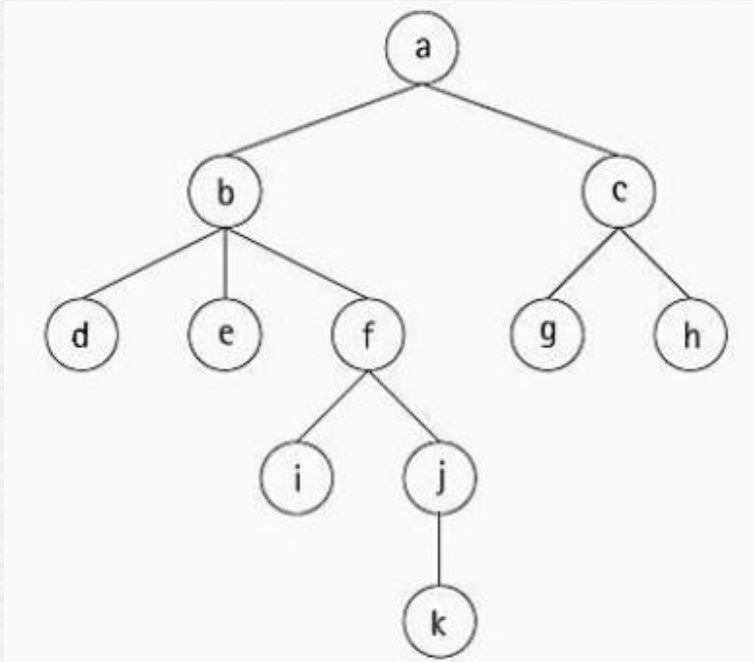


*Texto baseado em Puga e Riseti (2010).*



**EDUCAÇÃO  
METODISTA**

# Árvores – Binária

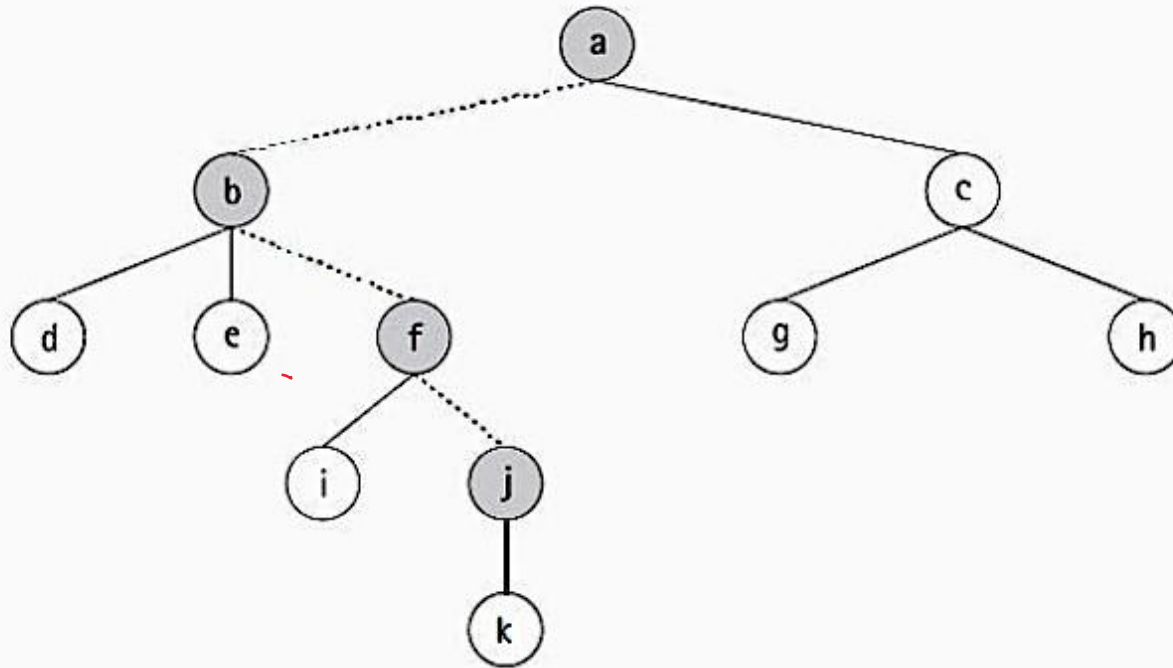


Texto baseado em Puga e Riseti (2010).



**EDUCAÇÃO  
METODISTA**

# Árvores – Entendendo a trajetória



Texto baseado em Puga e Riseti (2010).

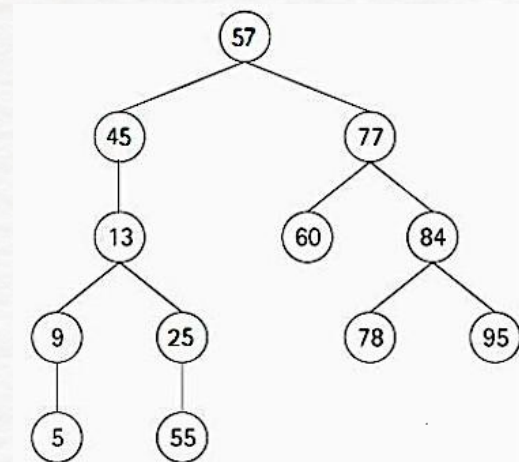


**EDUCAÇÃO  
METODISTA**



# Árvores Binárias – Características

- Nó com grau  $\leq 2$ , em outras palavras, nenhum nó possui mais que dois descendentes.
- Os nós da **direita** sempre possuem valor **superior** ao do nó pai.
- Os nós da **esquerda** sempre possuem valor **inferior** ao do nó pai.



essa árvore  
binária segue a  
regra?

Texto baseado em Puga e Riseti (2010).



**EDUCAÇÃO  
METODISTA**

# Árvores Binárias – Exemplo Java

```
1. class Exemplo106{
2.     public static void main(String[] args){
3.         BArvore arvore1 = new BArvore ();
4.         arvore1.inserirNo (14);
5.         arvore1.inserirNo (16);
6.         arvore1.inserirNo (12);
7.         arvore1.inserirNo (11);
8.         arvore1.inserirNo (17);
9.         arvore1.inserirNo (15);
10.        arvore1.exibirNo ();
11.        arvore1.inserirNo (10);
12.        arvore1.inserirNo (13);
13.        System.out.println ("");
14.        arvore1.exibirNo ();
15.    }
16. }
```

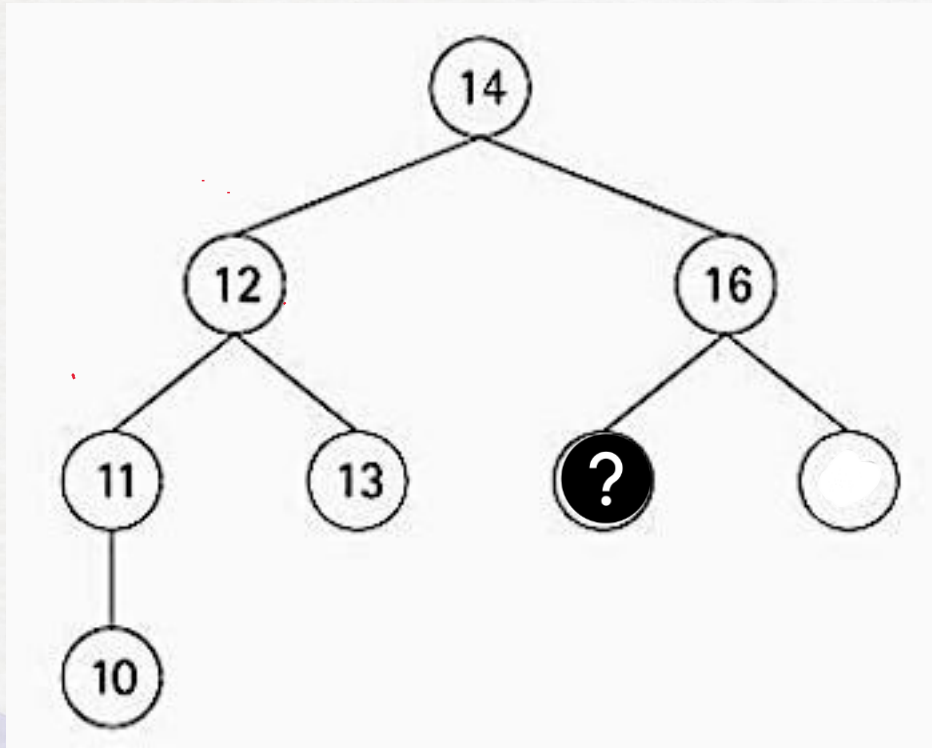
Como ficará a  
árvore gerada por  
essa classe?

*Texto baseado em Puga e Riseti (2010).*



**EDUCAÇÃO  
METODISTA**

# Árvores Binárias – Exemplo Java



Como ficará a árvore gerada por essa classe?

Qual a lógica a ser aplicada para exclusão de nós?

*Texto baseado em Puga e Riseti (2010).*

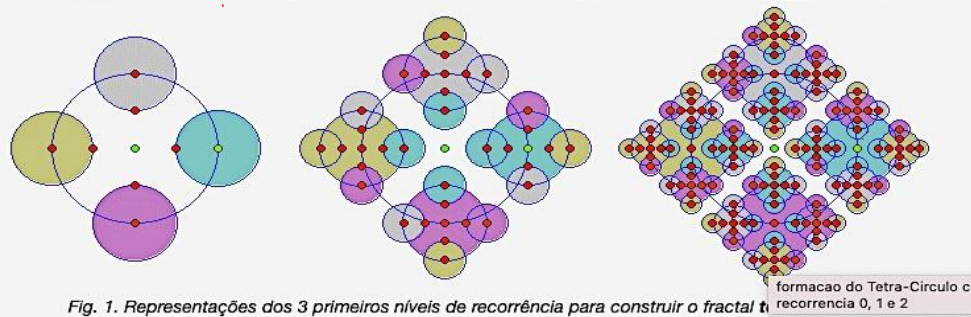


**EDUCAÇÃO  
METODISTA**



# Recursividade

- antes de mais nada, no exemplo em Java da estrutura de dados em árvore, a recursividade se dá com uma chamada interna do próprio método no qual ela está inserida.
- Definição de recursividade (aula 1): De acordo com Brandão (2019) a ideia de recursividade é a de um processo que é definido a partir de si próprio. No caso de um algoritmo, esse é definido **invocando** a si mesmo. O exemplo dado pelo autor são os fractais.





# Recursividade

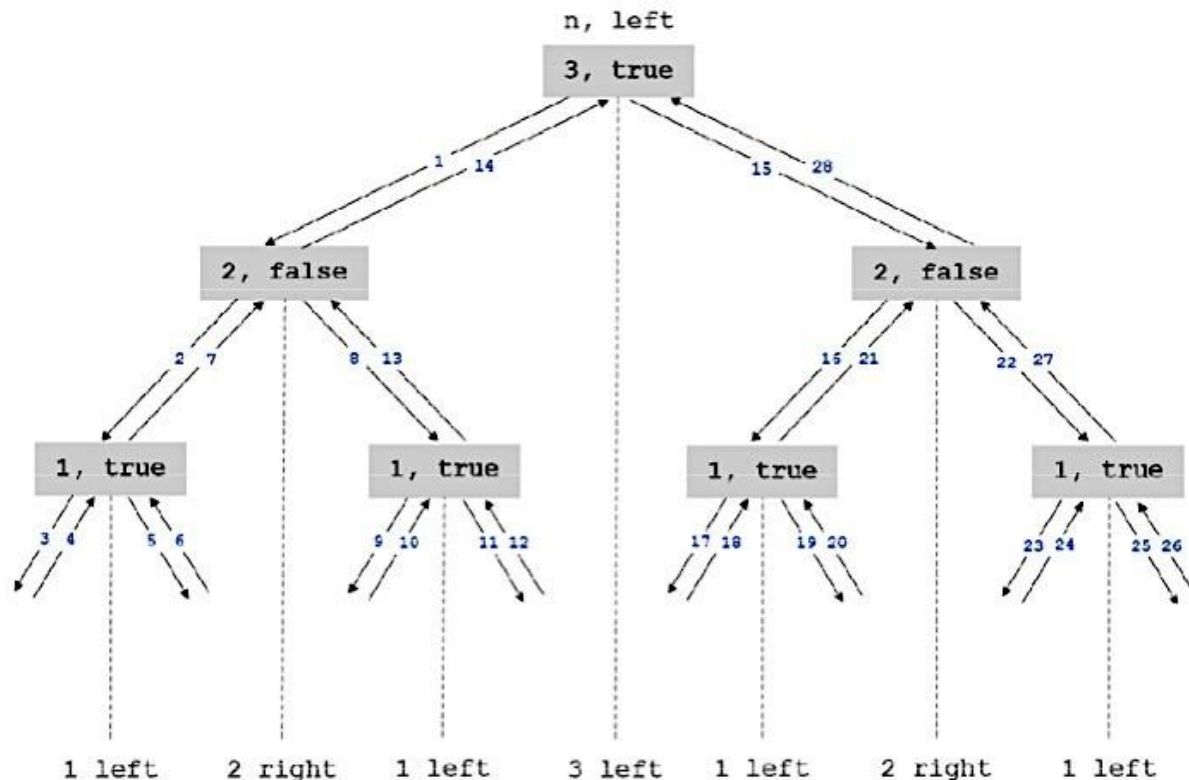
- Algoritmos iterativos e recursivos:
  - iterativos: requer a repetição explícita de um processo até que determinada condição seja satisfeita.
  - recursivos: como visto anteriormente.
- Exemplo: Cálculo de  $n!$ : ele pode ser calculado simplesmente como no exemplo  $5! = 5 * 4 * 3 * 2 * 1$  ou  $5 * 4!$
- Em uma estrutura de algoritmo, a segunda opção apresenta a necessidade de recursividade, pois:

$$n! = n * (n - 1)! = n * (n - 1) * (n - 2) * \dots * 1$$

- Esses três pontos (...) requer em um algoritmo chamar a mesma função quantas vezes forem necessárias até que a mesma seja satisfeita.

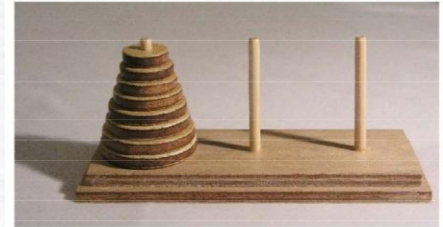


# Árvore de recursão: Torres de Hanoi



## Lenda das Torres de Hanoi

•Mundo vai acabar quando um grupo de monges conseguirem mover 64 discos de ouro em 3 pinos de diamante. A solução revela que seriam necessários  $2^n - 1$  movimentações, ou 585 milhões de anos...



EDUCAÇÃO  
METODISTA

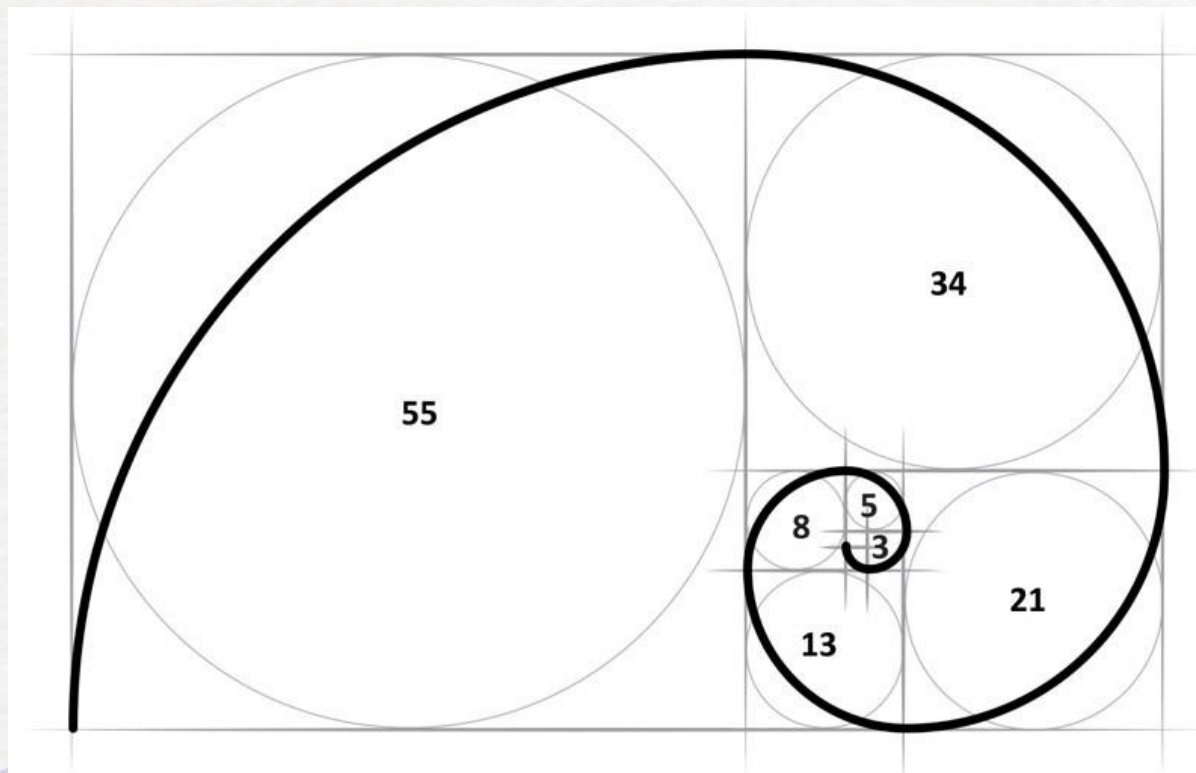
# Sequência de Fibonacci

- A sequência de Fibonacci é uma sequência de números específica que se repete em várias estruturas naturais e também feitas pelo homem (neste caso, propositalmente).
- 0 , 1 , 1 , 2 , 3 , 5 , 8 , 13 , 21 , 34...
- Cada elemento nessa sequência é a soma dos dois elementos anteriores (por exemplo,  $0 + 1 = 1$ ,  $1 + 1 = 2$ ,  $1 + 2 = 3$ ,  $2 + 3 = 5$ , ...).
- Se permitirmos que  $fib(0) = 0$ ,  $fib(1) = 1$ , e assim por diante, então poderemos definir a sequência de Fibonacci por meio da seguinte definição recursiva:

$$\begin{aligned} fib(n) &= n \text{ if } n == 0 \text{ or } n == 1 \\ fib(n) &= fib(n - 2) + fib(n - 1) \text{ if } n >= 2 \end{aligned}$$



# Sequência de Fibonacci



Texto baseado em Tenenbaum et al. (1995).



**EDUCAÇÃO  
METODISTA**



# Recursividade na prática: busca

- A recursividade é, em modos práticos computacionais, uma das atividades mais conhecidas da área: a busca;
- Qual seria o método mais eficiente para executá-la?
- O método de busca sequencial ou linear é aquele onde cada item do vetor é examinado por vez e comparado ao item procurado, até que ocorra uma coincidência.
- Mas se esta busca não for “iniciada do início” da sequência?
- No caso, se pensarmos em vetores, a busca sempre deve ser realizada dividindo esse vetor em dois, por isso, busca binária.
- “Se o item sendo procurado não for igual ao elemento do meio do vetor, as instruções serão pesquisar um subvetor usando o mesmo método”.



# ESTRUTURAS SEQUENCIAIS E ENCADEADAS

- Aqui vamos falar de listas novamente, com foco na forma dessa lista: se sequencial ou encadeada.
- Importante: pilhas, árvores e filas podem ser consideradas subtipos de listas.
- Na bibliografia podemos encontrar então pilhas sequenciais e encadeadas, por exemplo.



# ESTRUTURAS SEQUENCIAIS E ENCADEADAS

## CONCEITOS BÁSICOS

- Estruturas sequenciais: a alocação dos dados na estrutura é dada de forma consecutiva na memória.
- Estruturas encadeadas: a alocação não possui uma ordem consecutiva e os elementos podem ocupar quaisquer posições na memória.

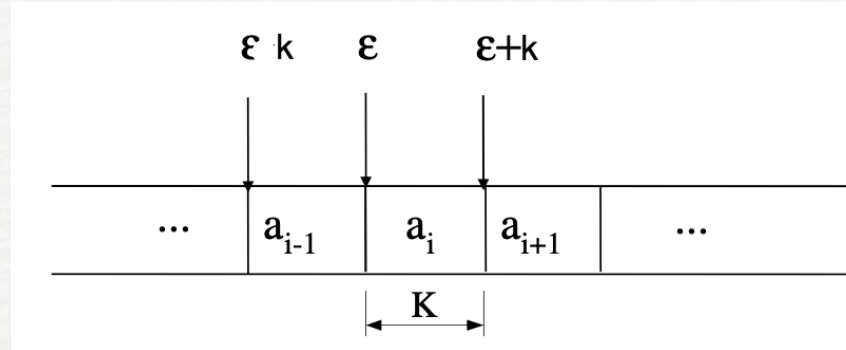




# ESTRUTURAS SEQUENCIAIS

## CARACTERÍSTICAS

- A estrutura sequencial pode ser representada como segue:



- é de fácil endereçamento (indexação);
- fácil inserção ou eliminação de elementos no final da lista.

Fonte da imagem: <http://200.17.137.109:8081/novobsi/Members/taef/graduacao/>, acesso em 08set2021.



# ESTRUTURAS SEQUENCIAIS

## CARACTERÍSTICAS

- Difícil inserção de elementos no meio da lista
- Mobilidade de elementos dificultada.
- A implementação dessas listas se dá pelo uso de vetores.
- Segundo Celes (2004), quando declaramos um vetor (escrevendo o código), é reservado um espaço contínuo na memória para armazenamento de elementos.
- A partir do primeiro elemento, e com o uso de um ponteiro, é possível acessar seus elementos.



# ESTRUTURAS SEQUENCIAIS

## ENTENDENDO O VETOR DENTRO DE UMA EST. SEQUENCIAL

- Na implementação, deve-se determinar a quantidade máxima que o vetor (string) irá armazenar.
- Veja o exemplo em C para implementação do vetor de 500 nós (onde cada nó possui a informação e o comando para o ponteiro ir ao próximo elemento):

```
#define NUMNODES 500 struct nodetype{  
int info, next; };  
struct nodetype node[NUMNODES];
```

há diferenças importantes entre um nó de uma estrutura sequencial e encadeada. como veremos aqui.



EDUCAÇÃO  
METODISTA

# ESTRUTURAS SEQUENCIAIS

## ENTENDENDO O VETOR DENTRO DE UMA EST. SEQUENCIAL

- usando Python para implementação do vetor:

```
Max = 500
```

```
Nodo = (info, next) Lista = []
```

```
for i in range(Max):
```

```
Lista += [Nodo]
```





# ESTRUTURAS SEQUENCIAIS

## ENTENDENDO O VETOR DENTRO DE UMA EST. SEQUENCIAL

- a utilização de vetores apresenta limitação pelo tamanho pré-definido inicialmente, ou seja, não é uma estrutura de dados flexível.
- para resolver o problema de uma possível necessidade de incremento no vetor, utilizam-se as listas ligadas (CELES *et al.*, 2004).
- uma lista ligada pode ser definida como um conjunto, da seguinte forma:  $V: [x_1, x_2, \dots, x_n]$  com  $n > 0$ .
- listas ligadas = listas **encadeadas**, SILVA (2007).





# ESTRUTURAS SEQUENCIAIS

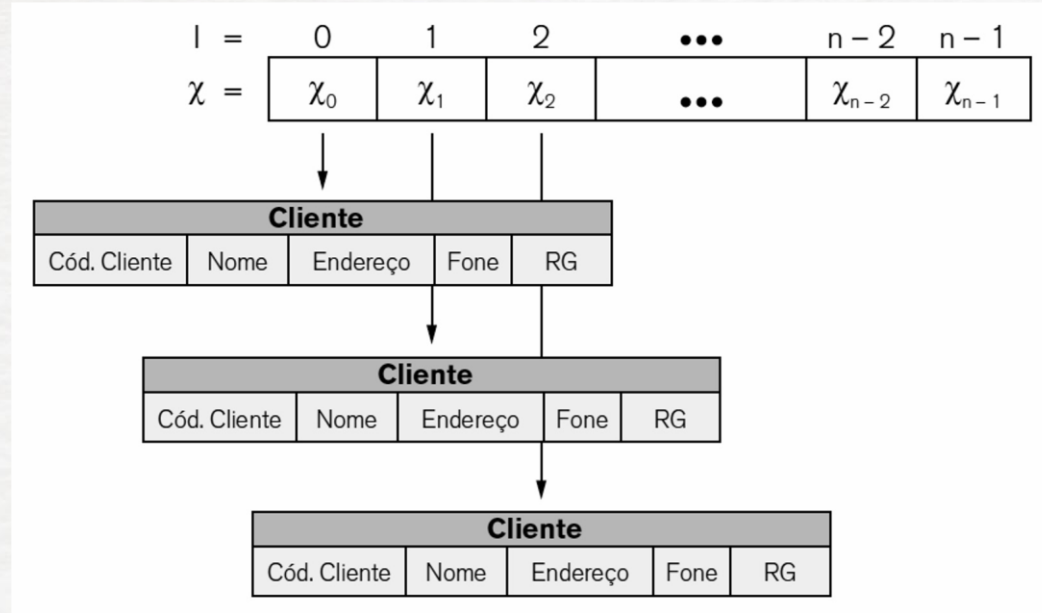
## APLICAÇÕES DAS LISTAS SEQUENCIAIS

- ideais para pequenos conjuntos de dados;
- cada nó (mencionado anteriormente), armazena os dados em formato de campos.
- Os nós da lista possui um identificador chamado de chave.



# ESTRUTURAS SEQUENCIAIS

## APLICAÇÕES DAS LISTAS SEQUENCIAIS



# ESTRUTURAS SEQUENCIAIS

## VANTAGENS DAS LISTAS SEQUENCIAIS

- facilidade de acesso aos dados através de indexação.
- tempo de acesso aos dados previsível, constante.

## • DESVANTAGENS DAS LISTAS SEQUENCIAIS

- dificuldades de inserção ou remoção de elementos.
- necessidade de “estimar” o tamanho do vetor no início da implementação ou programa.



# ESTRUTURAS ENCADEADAS

## CARACTERÍSTICAS

- Característica e vantagem sobre as sequenciais: não é necessário pré-determinar a quantidade de elementos.
- O nó da lista encadeada contém a informação ou ponteiro que indica o próximo elemento da lista, que **não necessariamente é o elemento seguinte**.
- A lista encadeada pode ser vista como um conjunto dinâmico de nós.

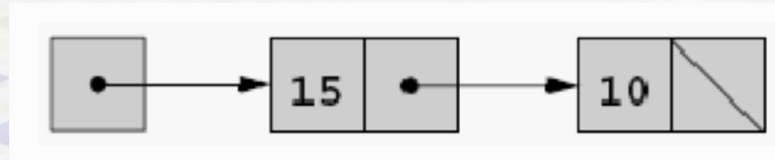
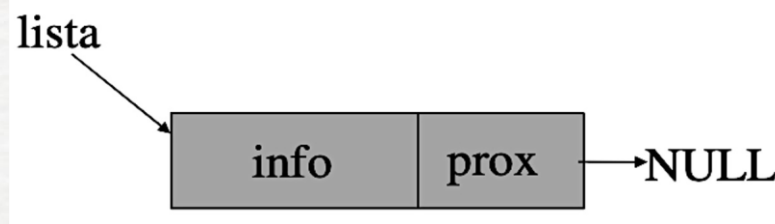




# ESTRUTURAS ENCADEADAS

## CARACTERÍSTICAS

- No caso de vetores, o armazenamento é realizado de forma adjacente, enquanto que nas listas encadeadas (ou ligadas), é realizado de forma lógica. Mais especificamente:



# ESTRUTURAS ENCADEADAS

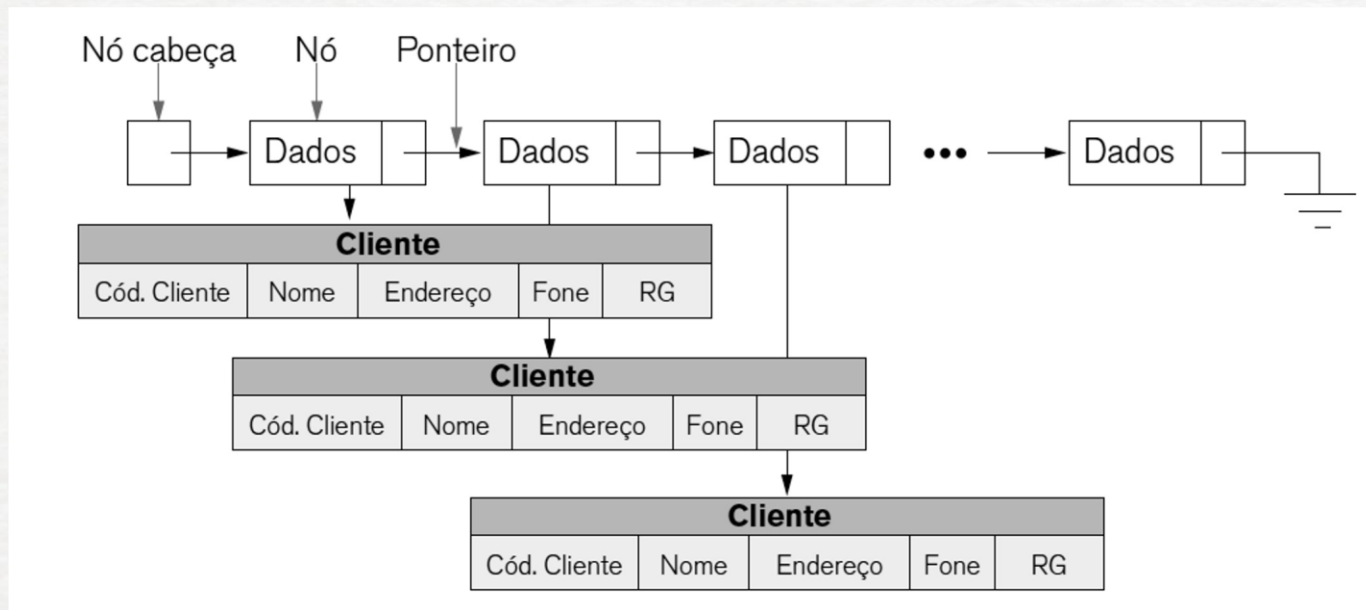
## CARACTERÍSTICAS

- A quantidade máxima de elementos na lista é determinada pela quantidade disponível de memória.
- Essa memória refere-se à memória que pode ser alocada pelo programa.
- Também chamada por alguns autores de **estrutura dinâmica**.
- A alocação de memória em estruturas dinâmicas é feita durante a execução do processo ou do programa.



# ESTRUTURAS ENCADEADAS

## ENTENDENDO UMA ESTRUTURA ENCADEADA



# ESTRUTURAS ENCADEADAS

## ENTENDENDO UMA ESTRUTURA ENCADEADA

- De acordo com Szwarcfiter e Markenzon (1994), os dados de implementação da estrutura encadeada são colocadas no nó-cabeça.
- Nó-cabeça é a definição para o nó que indica o início da lista encadeada, e que não pode ser removido.
- Desta forma, dados/registros não devem ser armazenados no nó cabeça.

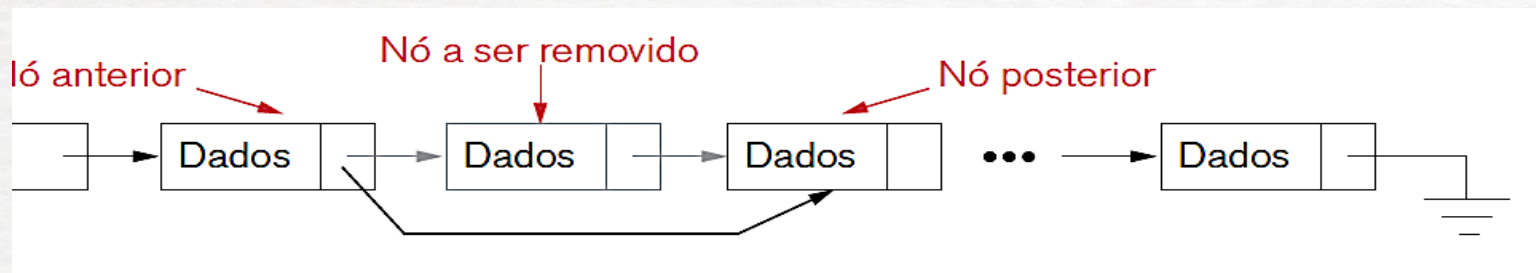




# ESTRUTURAS ENCADEADAS

## OPERAÇÕES COM ESTRUTURAS ENCADEADAS

- veremos com mais detalhes as operações com esses tipos de estrutura, mas basicamente as operações são: inserção e remoção.



# ESTRUTURAS ENCADEADAS

## VANTAGENS DAS LISTAS ENCADEADAS

- facilidade de inserir e remover elementos.
- não é necessário movimentar elementos da lista nas operações de inserção/remoção.

## • DESVANTAGENS DAS LISTAS ENCADEADAS

- maior atenção com os ponteiros.
- maior consumo de memória nas operações de busca.
- necessidade de memória para os ponteiros.

