



# Algoritmos e Estrutura de dados II

Rodrigo Adur  
rodrigoadurti@gmail.com



## ➤ **Professor Rodrigo Adur**

### ➤ **Formação**

- Bacharelado em Sistemas de Informações (FILC)
- Especialista em Sistemas de Informações (NCE/UFRJ)

### ➤ **Experiência Profissional**

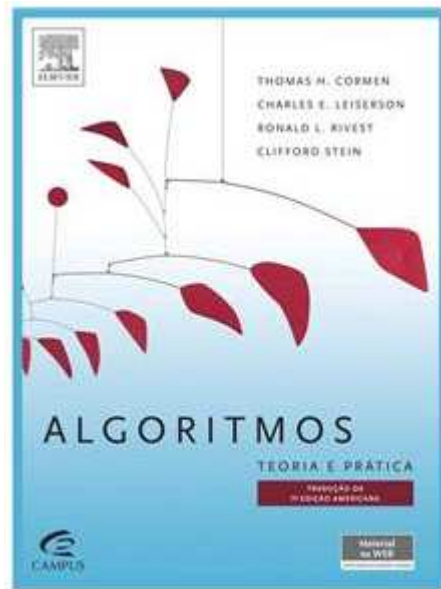
- Desenvolvimento de Sistemas
  - Desenvolvimento de biblioteca de componentes
  - Desenvolvimento de framework
- Analista de Sistemas do SERPRO
  - Arquiteto
  - Líder técnico
  - Programador



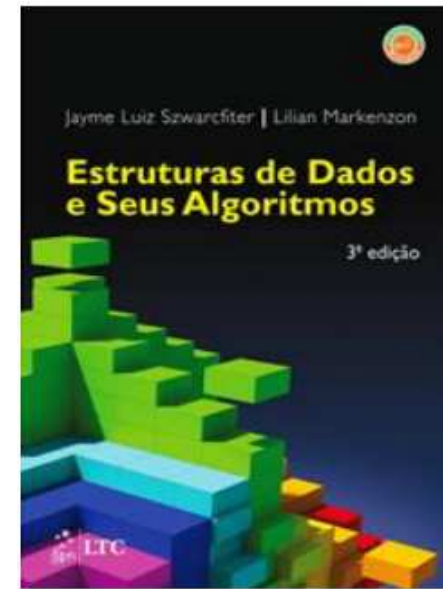
## ➤ **Módulo II**

- Estruturas de Dados Elementares e seus Algoritmos:
  - Técnicas de alocação de memória
  - Lista, Pilha, Fila e Deque.
- Árvores
  - Conceitos Básicos
  - Árvores Binárias de Busca
  - Árvore AVL
  - Árvore B
- Tabela Hash
  - Função Hash
  - Técnicas de Resolução de Colisão



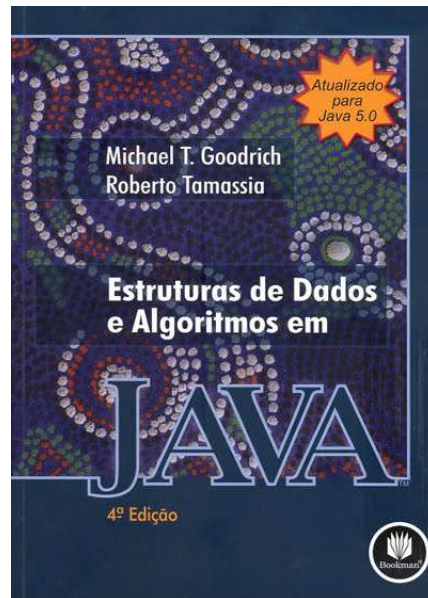


**Algoritmos – Teoria e prática**  
Thomas Cormen  
3ª Edição



**Estrutura de Dados e seus Algoritmos**  
Jayme Luiz  
3ª Edição

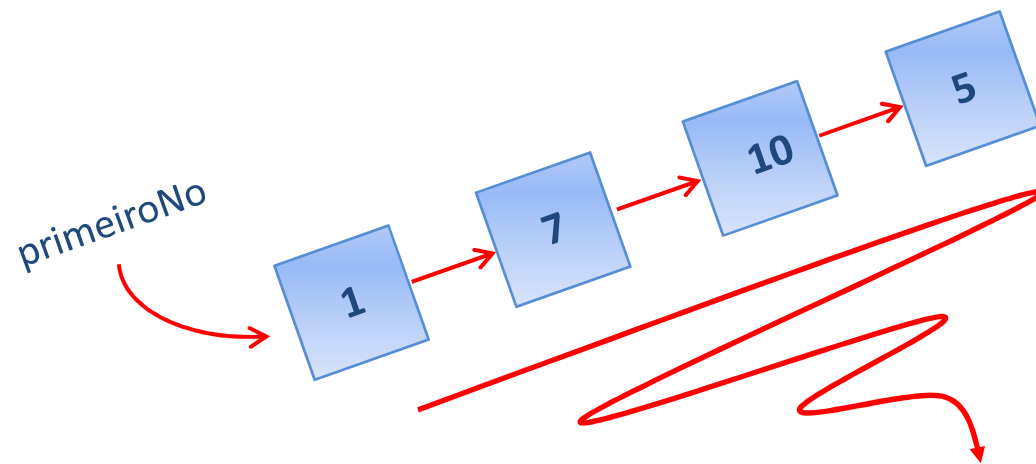




## **Estrutura de Dados e Algoritmos em Java**

Michael Goodrich  
4ª Edição





# Listas



## ➤ Conceitos Gerais

- Uma lista é uma estrutura de dados que agrupa informações referentes a um conjunto de elementos que de alguma forma se relacionam entre si.
- Cada nó da lista é formado por diversos atributos. Geralmente um desses atributos atua como identificador do nó.
- As listas podem ser mantidas na memória do computador das seguintes formas:
  - Alocação sequencial
  - Alocação encadeada



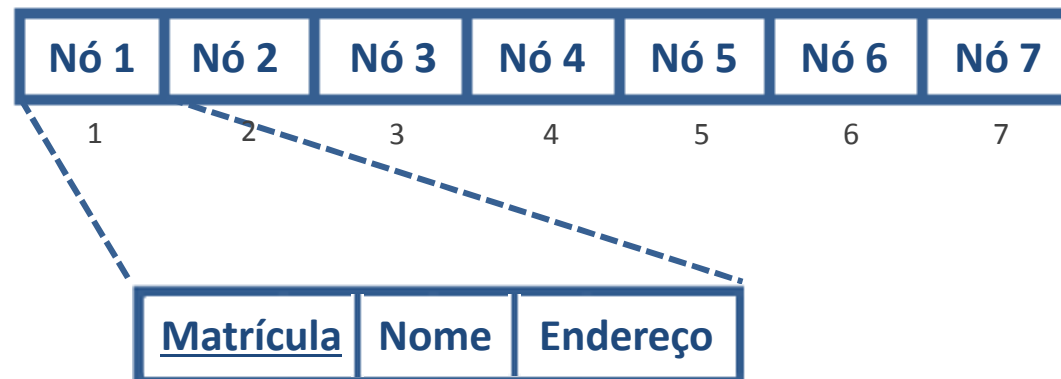
## ➤ Alocação Sequencial

- Na alocação sequencial, os elementos da lista ficam dispostos de forma contígua na memória.
- A implementação da alocação sequencial é geralmente realizada com a reserva prévia de memória (alocação estática).
- Possibilita acesso imediato a um elemento da lista.



## ➤ Alocação Sequencial

- Representação de uma lista com alocação sequencial:



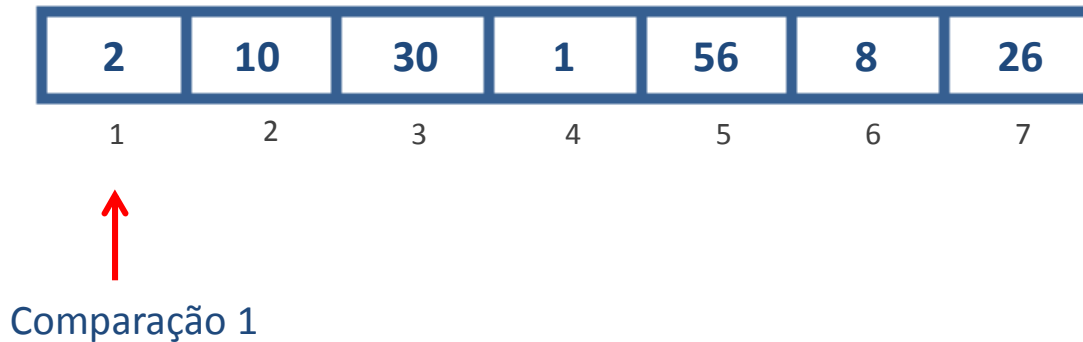


- **Alocação Sequencial – Operação Busca**
  - Para buscar um elemento em uma lista linear com alocação sequencial, podemos usar os seguintes métodos:
    - Busca sequencial
    - Busca binária



## ➤ Alocação Sequencial – Operação Busca

- Como encontrar o nó com identificador 30 na lista a seguir?





## ➤ Alocação Sequencial – Operação Busca

- Como encontrar o nó com identificador 30 na lista a seguir?

2	10	30	1	56	8	26
1	2	3	4	5	6	7



Comparação 2



## ➤ Alocação Sequencial – Operação Busca

- Como encontrar o nó com identificador 30 na lista a seguir?

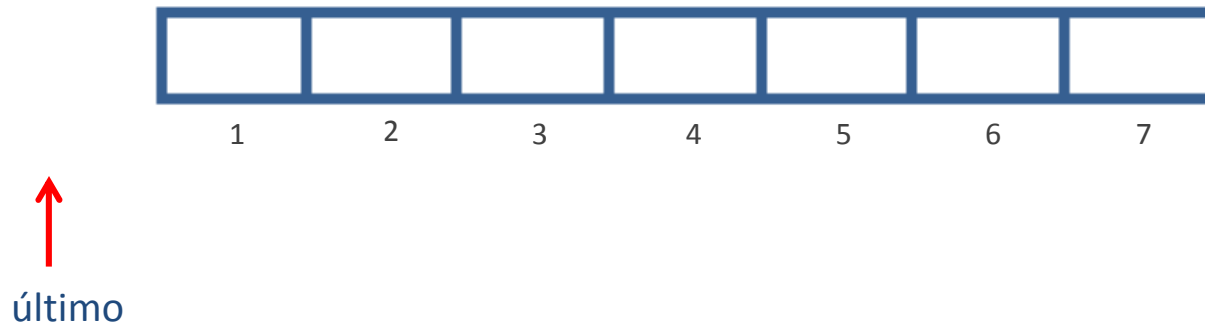
2	10	30	1	56	8	26
1	2	3	4	5	6	7



Comparação 3

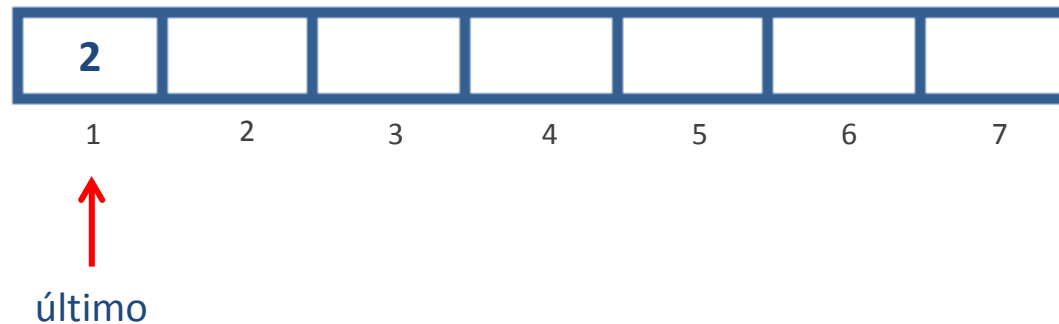


- **Alocação Sequencial – Operação Inclusão<sub>1</sub>**
  - Considerando uma lista inicialmente vazia, como a mesma ficaria após a inclusão dos seguintes elementos: 2, 10, 30, 1, 56?





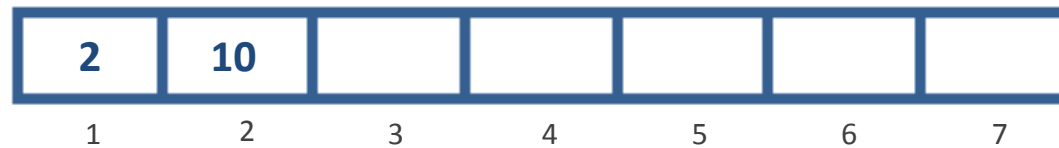
- **Alocação Sequencial – Operação Inclusão<sub>1</sub>**
  - Considerando uma lista inicialmente vazia, como a mesma ficaria após a inclusão dos seguintes elementos: **2**, 10, 30, 1, 56?



Elemento: 2



- **Alocação Sequencial – Operação Inclusão<sub>1</sub>**
  - Considerando uma lista inicialmente vazia, como a mesma ficaria após a inclusão dos seguintes elementos: 2, **10**, 30, 1, 56?

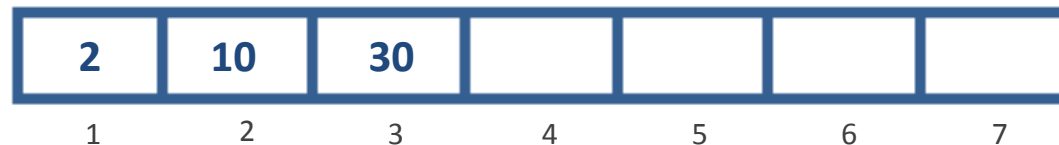


Elemento: 10

↑  
último



- **Alocação Sequencial – Operação Inclusão<sub>1</sub>**
  - Considerando uma lista inicialmente vazia, como a mesma ficaria após a inclusão dos seguintes elementos: 2, 10, **30**, 1, 56?

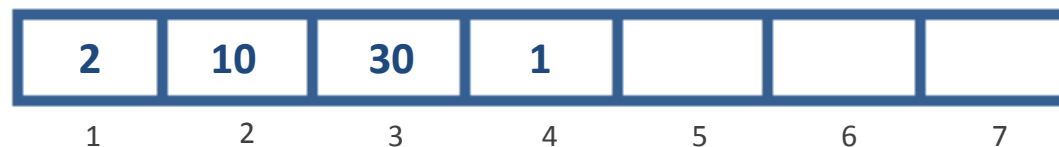


Elemento: 30

↑  
último



- **Alocação Sequencial – Operação Inclusão<sub>1</sub>**
  - Considerando uma lista inicialmente vazia, como a mesma ficaria após a inclusão dos seguintes elementos: 2, 10, 30, **1**, 56?

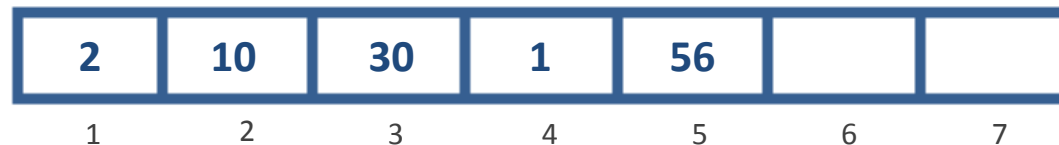


Elemento: 1

↑  
último



- **Alocação Sequencial – Operação Inclusão<sub>1</sub>**
  - Considerando uma lista inicialmente vazia, como a mesma ficaria após a inclusão dos seguintes elementos: 2, 10, 30, 1, **56**?

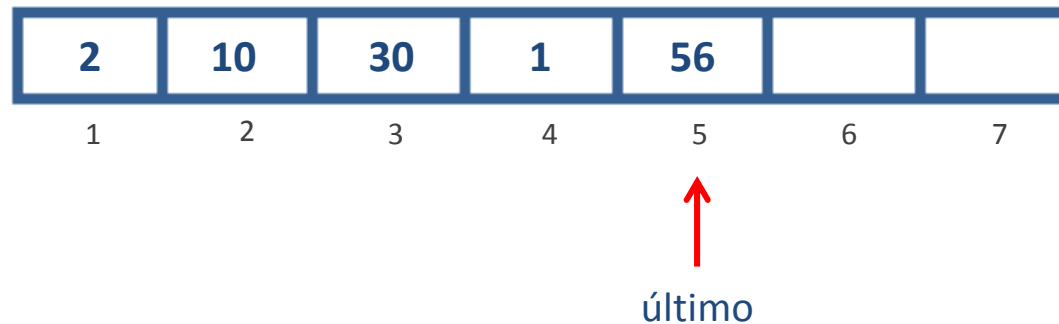


Elemento: 56

↑  
último

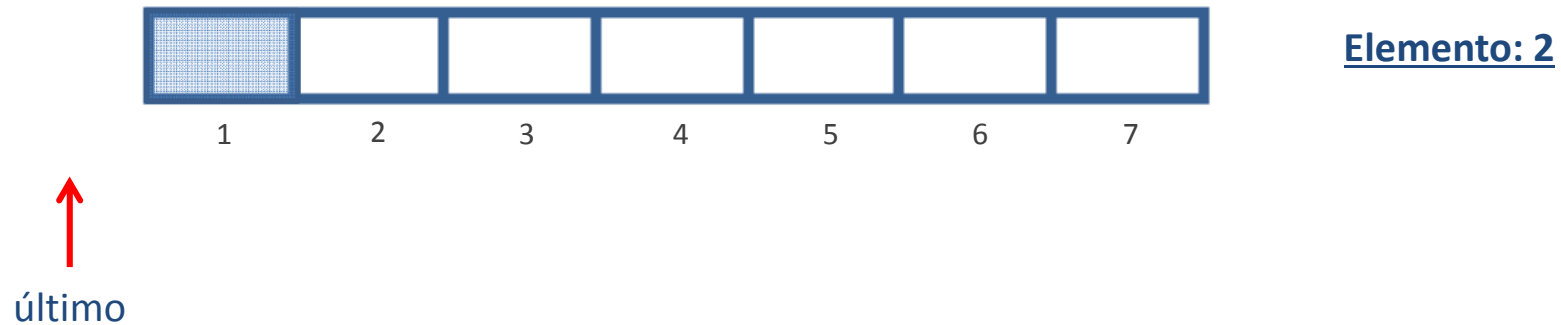


- **Alocação Sequencial – Operação Inclusão<sub>1</sub>**
  - Considerando uma lista inicialmente vazia, como a mesma ficaria após a inclusão dos seguintes elementos: 2, 10, 30, 1, 56?





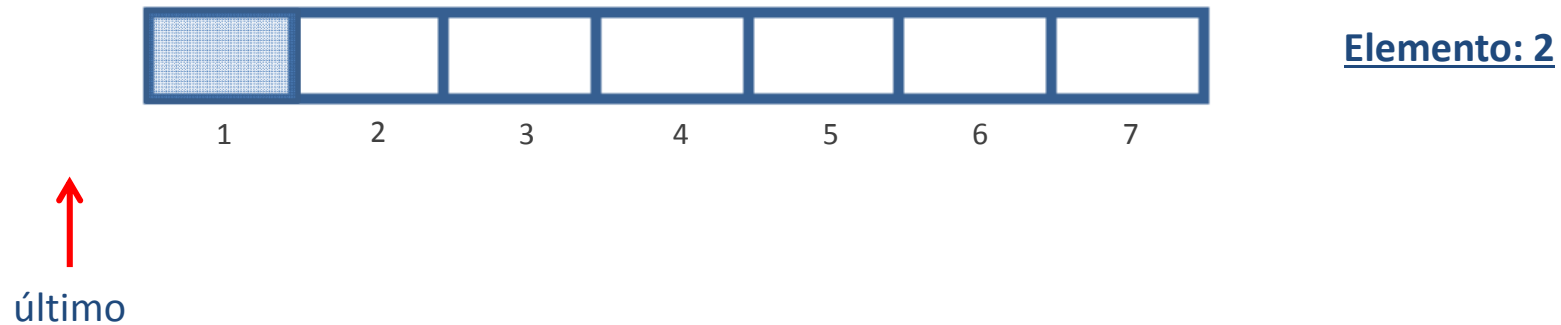
- **Alocação Sequencial – Operação Inclusão<sub>2</sub>**
  - Considerando uma lista inicialmente vazia, como a mesma ficaria após a inclusão dos seguintes elementos de forma a manter a lista ordenada: **2**, 10, 30, 1, 56?



- 1 – Identificar onde o novo elemento deve ser incluído
- 2 – Deslocar todos os elementos para frente (inclusive)
- 3 – Incluir o novo elemento



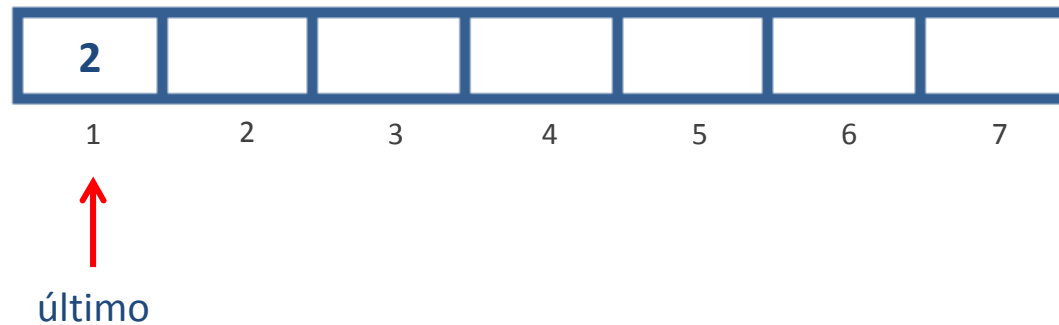
- **Alocação Sequencial – Operação Inclusão<sub>2</sub>**
  - Considerando uma lista inicialmente vazia, como a mesma ficaria após a inclusão dos seguintes elementos de forma a manter a lista ordenada: **2**, 10, 30, 1, 56?



- 1 – Identificar onde o novo elemento deve ser incluído
- 2 – **Deslocar todos os elementos para frente (inclusive)**
- 3 – Incluir o novo elemento



- **Alocação Sequencial – Operação Inclusão<sub>2</sub>**
  - Considerando uma lista inicialmente vazia, como a mesma ficaria após a inclusão dos seguintes elementos de forma a manter a lista ordenada: **2**, 10, 30, 1, 56?

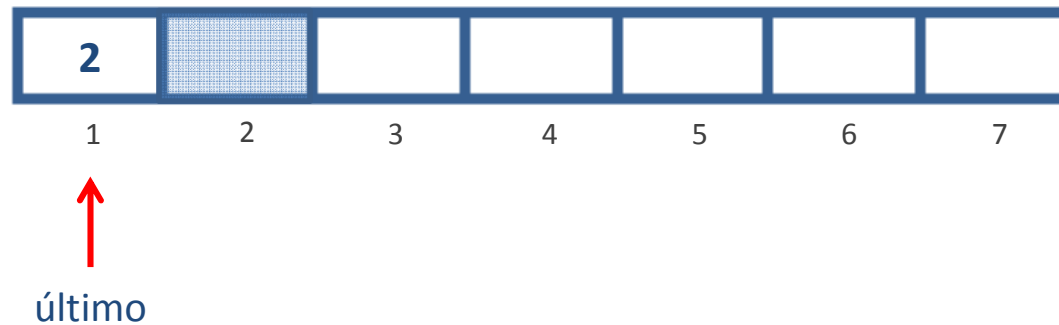


Elemento: 2

- 1 – Identificar onde o novo elemento deve ser incluído
- 2 – Deslocar todos os elementos para frente (inclusive)
- 3 – **Incluir o novo elemento**



- **Alocação Sequencial – Operação Inclusão<sub>2</sub>**
  - Considerando uma lista inicialmente vazia, como a mesma ficaria após a inclusão dos seguintes elementos de forma a manter a lista ordenada: 2, **10**, 30, 1, 56?

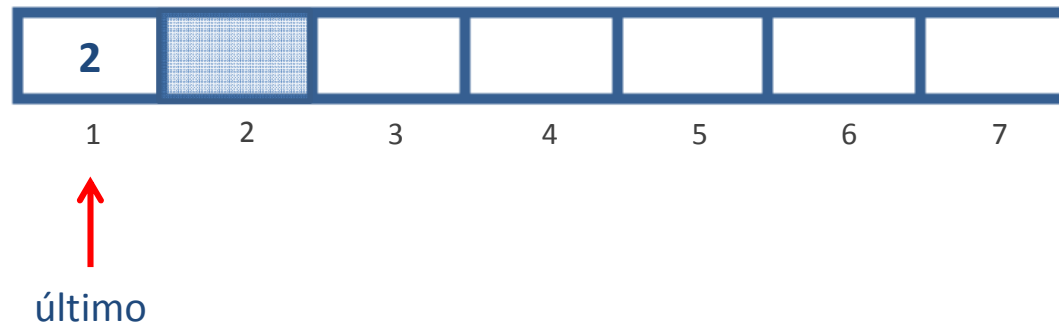


Elemento: 10

- 1 – Identificar onde o novo elemento deve ser incluído
- 2 – Deslocar todos os elementos para frente (inclusive)
- 3 – Incluir o novo elemento



- **Alocação Sequencial – Operação Inclusão<sub>2</sub>**
  - Considerando uma lista inicialmente vazia, como a mesma ficaria após a inclusão dos seguintes elementos de forma a manter a lista ordenada: 2, **10**, 30, 1, 56?

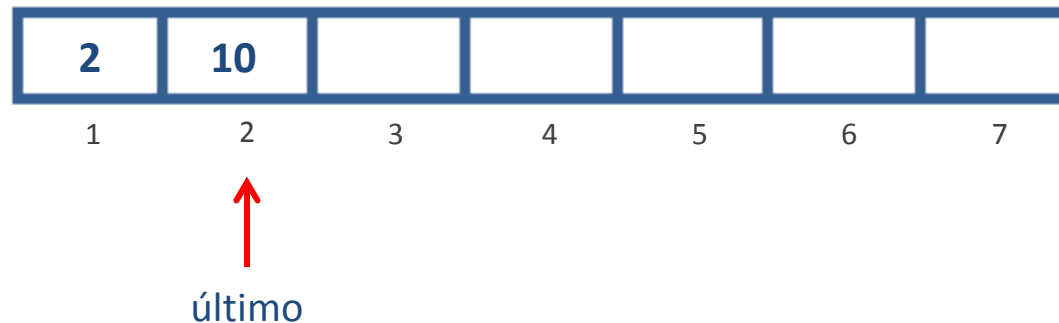


Elemento: 10

- 1 – Identificar onde o novo elemento deve ser incluído
- 2 – **Deslocar todos os elementos para frente (inclusive)**
- 3 – Incluir o novo elemento



- **Alocação Sequencial – Operação Inclusão<sub>2</sub>**
  - Considerando uma lista inicialmente vazia, como a mesma ficaria após a inclusão dos seguintes elementos de forma a manter a lista ordenada: 2, **10**, 30, 1, 56?

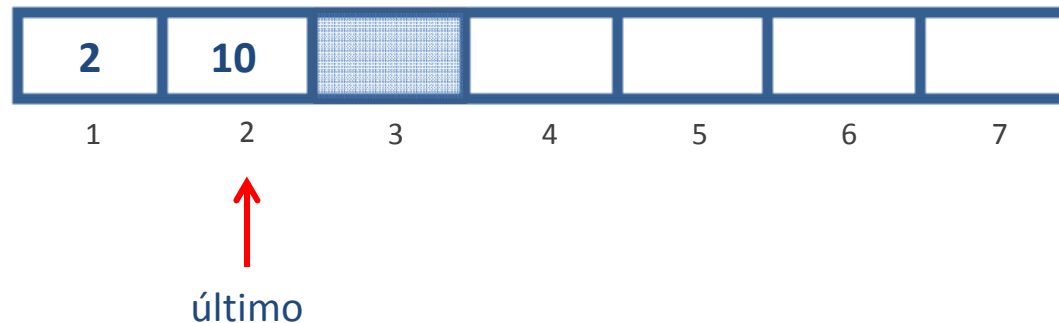


Elemento: 10

- 1 – Identificar onde o novo elemento deve ser incluído
- 2 – Deslocar todos os elementos para frente (inclusive)
- 3 – **Incluir o novo elemento**



- **Alocação Sequencial – Operação Inclusão<sub>2</sub>**
  - Considerando uma lista inicialmente vazia, como a mesma ficaria após a inclusão dos seguintes elementos de forma a manter a lista ordenada: 2, 10, **30**, 1, 56?

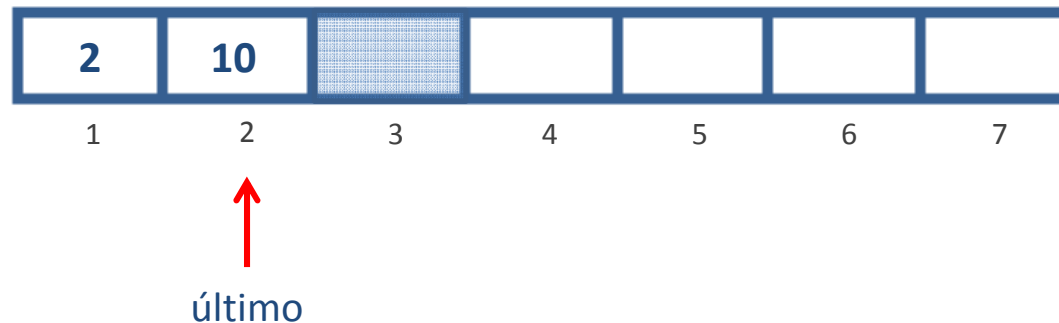


Elemento: 30

- 1 – Identificar onde o novo elemento deve ser incluído
- 2 – Deslocar todos os elementos para frente (inclusive)
- 3 – Incluir o novo elemento



- **Alocação Sequencial – Operação Inclusão<sub>2</sub>**
  - Considerando uma lista inicialmente vazia, como a mesma ficaria após a inclusão dos seguintes elementos de forma a manter a lista ordenada: 2, 10, **30**, 1, 56?



Elemento: 30

- 1 – Identificar onde o novo elemento deve ser incluído
- 2 – **Deslocar todos os elementos para frente (inclusive)**
- 3 – Incluir o novo elemento



- **Alocação Sequencial – Operação Inclusão<sub>2</sub>**
  - Considerando uma lista inicialmente vazia, como a mesma ficaria após a inclusão dos seguintes elementos de forma a manter a lista ordenada: 2, 10, **30**, 1, 56?



Elemento: 30

- 1 – Identificar onde o novo elemento deve ser incluído
- 2 – Deslocar todos os elementos para frente (inclusive)
- 3 – **Incluir o novo elemento**



- **Alocação Sequencial – Operação Inclusão<sub>2</sub>**
  - Considerando uma lista inicialmente vazia, como a mesma ficaria após a inclusão dos seguintes elementos de forma a manter a lista ordenada: 2, 10, 30, **1**, 56?

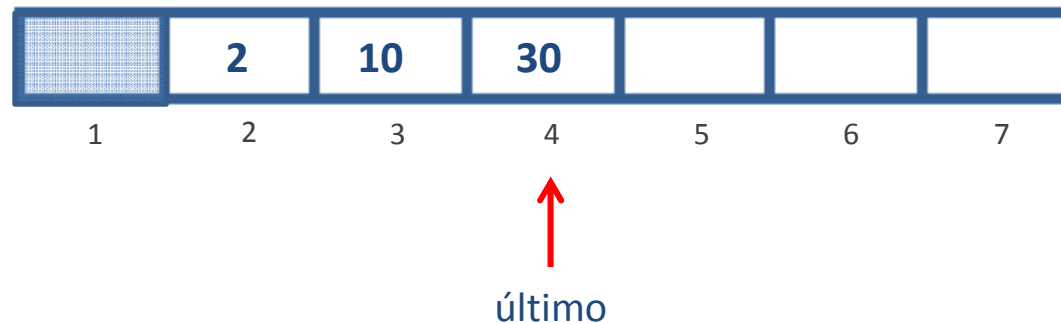


Elemento: 1

- 1 – Identificar onde o novo elemento deve ser incluído
- 2 – Deslocar todos os elementos para frente (inclusive)
- 3 – Incluir o novo elemento



- **Alocação Sequencial – Operação Inclusão<sub>2</sub>**
  - Considerando uma lista inicialmente vazia, como a mesma ficaria após a inclusão dos seguintes elementos de forma a manter a lista ordenada: 2, 10, 30, **1**, 56?

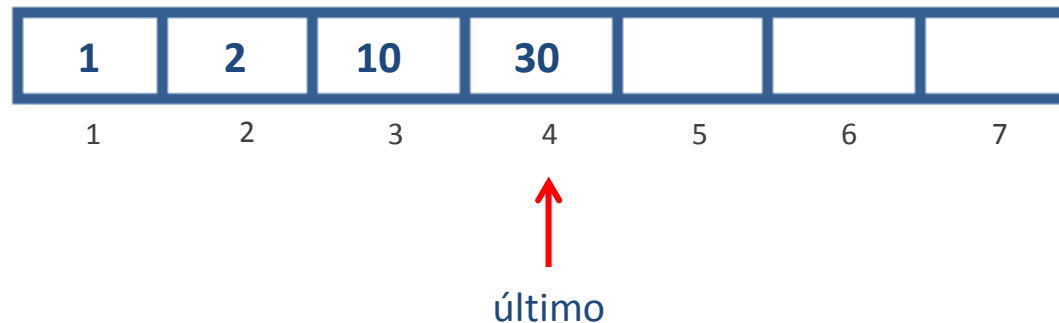


Elemento: 1

- 1 – Identificar onde o novo elemento deve ser incluído
- 2 – **Deslocar todos os elementos para frente (inclusive)**
- 3 – Incluir o novo elemento



- **Alocação Sequencial – Operação Inclusão<sub>2</sub>**
  - Considerando uma lista inicialmente vazia, como a mesma ficaria após a inclusão dos seguintes elementos de forma a manter a lista ordenada: 2, 10, 30, **1**, 56?

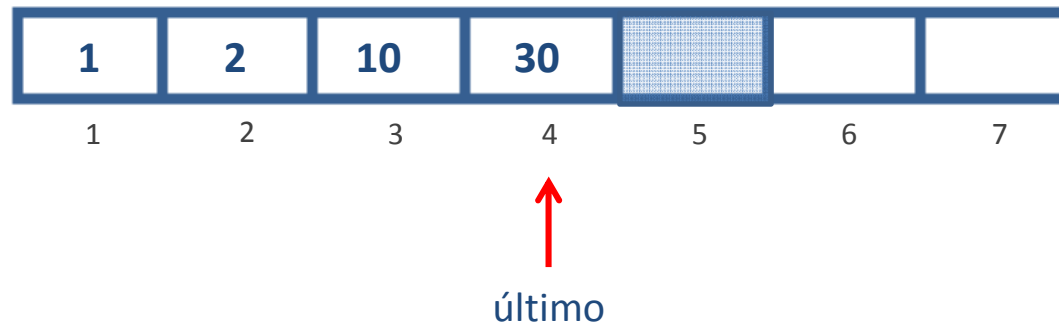


Elemento: 1

- 1 – Identificar onde o novo elemento deve ser incluído
- 2 – Deslocar todos os elementos para frente (inclusive)
- 3 – **Incluir o novo elemento**



- **Alocação Sequencial – Operação Inclusão<sub>2</sub>**
  - Considerando uma lista inicialmente vazia, como a mesma ficaria após a inclusão dos seguintes elementos de forma a manter a lista ordenada: 2, 10, 30, 1, **56**?

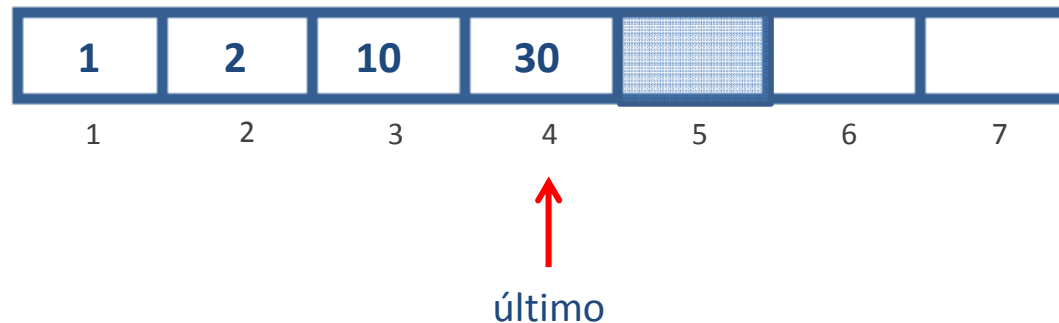


Elemento: 56

- 1 – Identificar onde o novo elemento deve ser incluído
- 2 – Deslocar todos os elementos para frente (inclusive)
- 3 – Incluir o novo elemento



- **Alocação Sequencial – Operação Inclusão<sub>2</sub>**
  - Considerando uma lista inicialmente vazia, como a mesma ficaria após a inclusão dos seguintes elementos de forma a manter a lista ordenada: 2, 10, 30, 1, **56**?

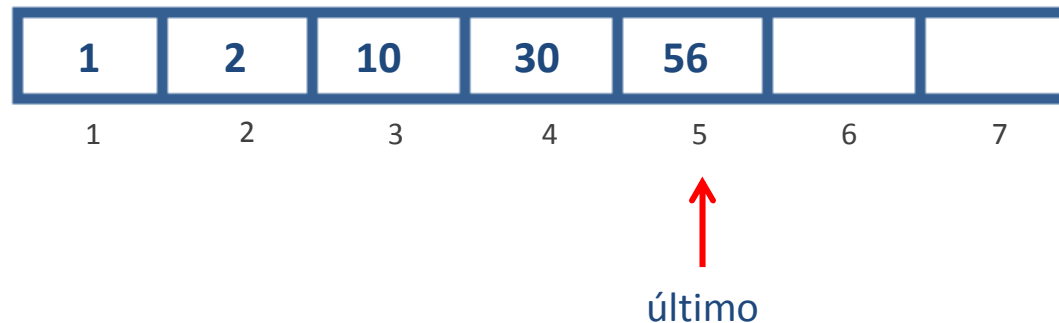


Elemento: 56

- 1 – Identificar onde o novo elemento deve ser incluído
- 2 – **Deslocar todos os elementos para frente (inclusive)**
- 3 – Incluir o novo elemento



- **Alocação Sequencial – Operação Inclusão<sub>2</sub>**
  - Considerando uma lista inicialmente vazia, como a mesma ficaria após a inclusão dos seguintes elementos de forma a manter a lista ordenada: 2, 10, 30, 1, **56**?

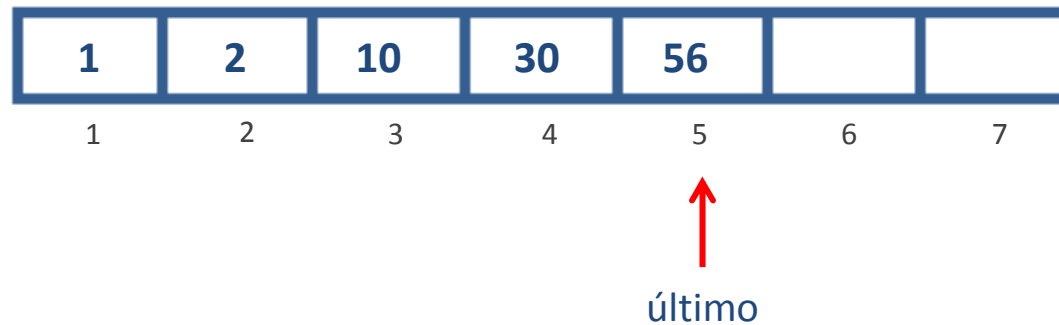


Elemento: 56

- 1 – Identificar onde o novo elemento deve ser incluído
- 2 – Deslocar todos os elementos para frente (inclusive)
- 3 – **Incluir o novo elemento**



- **Alocação Sequencial – Operação Inclusão<sub>2</sub>**
  - Considerando uma lista inicialmente vazia, como a mesma ficaria após a inclusão dos seguintes elementos de forma a manter a lista ordenada: 2, 10, 30, 1, 56?





## ➤ Alocação Sequencial – Operação Remoção

- Considerando a lista a seguir, como remover os elementos **56** e **2**?



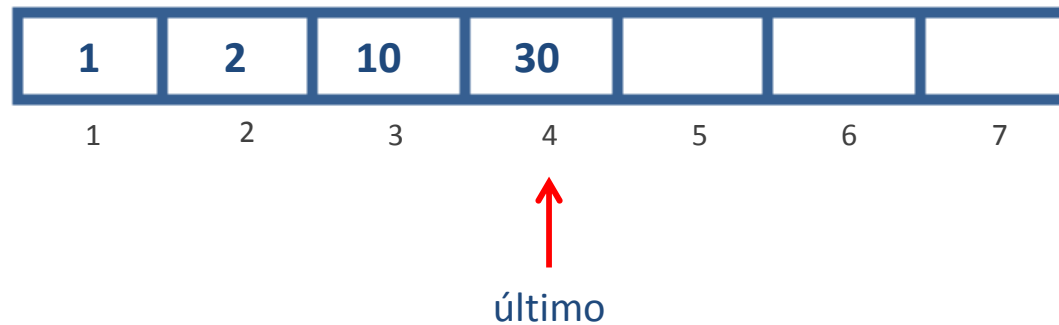
Elemento: 56

- 1 – Identificar onde se encontra o elemento a ser removido
- 2 – Remove o elemento
- 3 – desloca os seus sucessores



## ➤ Alocação Sequencial – Operação Remoção

- Considerando a lista a seguir, como remover os elementos **56** e **2**?



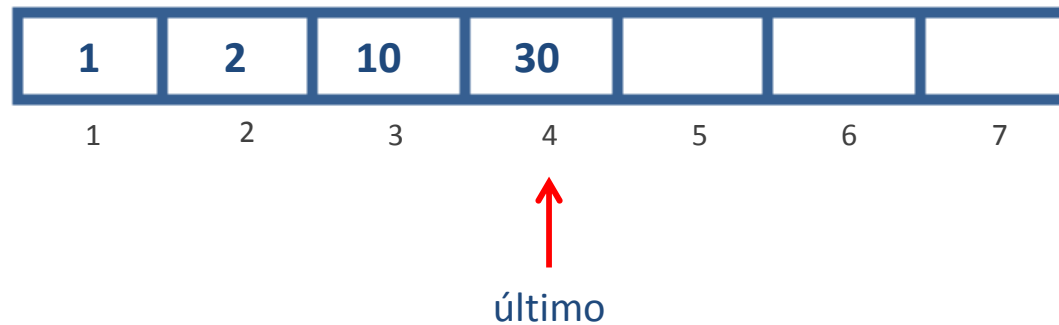
Elemento: 56

- 1 – Identificar onde se encontra o elemento a ser removido
- 2 – Remove o elemento**
- 3 – desloca os seus sucessores



## ➤ Alocação Sequencial – Operação Remoção

- Considerando a lista a seguir, como remover os elementos **56** e **2**?



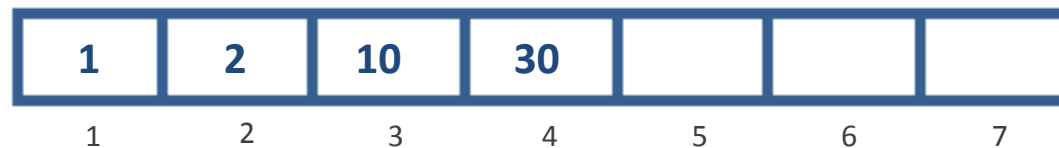
Elemento: 56

- 1 – Identificar onde se encontra o elemento a ser removido
- 2 – Remove o elemento
- 3 – **desloca os seus sucessores**



## ➤ Alocação Sequencial – Operação Remoção

- Considerando a lista a seguir, como remover os elementos 56 e 2?



Elemento: 2

↑  
elemento a ser  
removido

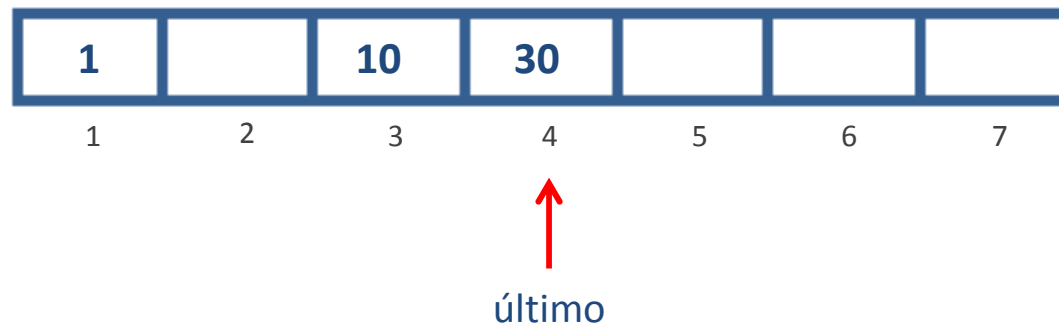
↑  
último

- 1 – Identificar onde se encontra o elemento a ser removido
- 2 – Remove o elemento
- 3 – desloca os seus sucessores



## ➤ Alocação Sequencial – Operação Remoção

- Considerando a lista a seguir, como remover os elementos 56 e 2?



Elemento: 2

- 1 – Identificar onde se encontra o elemento a ser removido
- 2 – Remove o elemento**
- 3 – desloca os seus sucessores



## ➤ Alocação Sequencial – Operação Remoção

- Considerando a lista a seguir, como remover os elementos 56 e 2?



Elemento: 2

- 1 – Identificar onde se encontra o elemento a ser removido
- 2 – Remove o elemento
- 3 – **desloca os seus sucessores**



- **Alocação Sequencial – Operação Remoção**
  - Considerando a lista a seguir, como remover os elementos 56 e 2?





## ➤ Alocação Encadeada

- Na alocação encadeada, os nós da lista são alocados/desalocados de acordo com a necessidade (alocação dinâmica).
- Os nós da lista ficam dispostos aleatoriamente na memória. Os mesmos são interligados pela referência que um nó faz ao seu sucessor.
- Quando comparado com a alocação sequencial, esses atributos extras fazem com que essa estrutura tenha um consumo maior de memória.



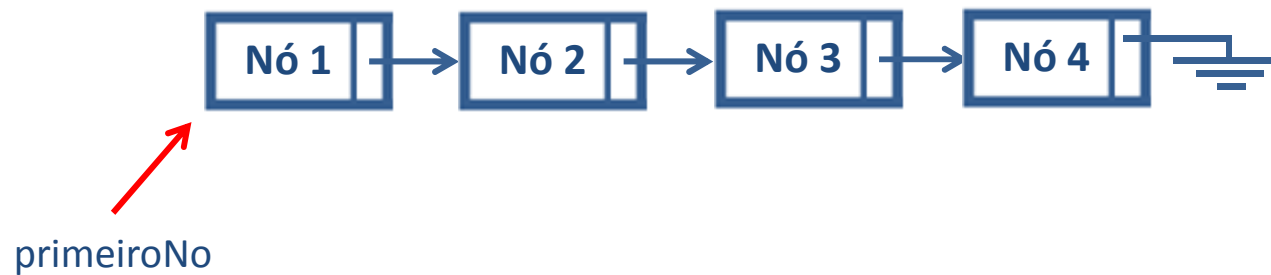
- **Alocação Encadeada**
- Estrutura mínima de um nó na linguagem java:

```
1  class No {  
2      private int id;  
3      private No proximo;  
4  
5      // Outros atribuos  
6  
7      // Construtores e métodos de acesso  
8  }
```



## ➤ Lista Simplesmente Encadeada

- Cada nó necessita de um atributo extra que tem como objetivo referenciar o próximo nó da lista.



- Essa estrutura não permite acesso direto a um nó da lista.







- **Lista Simplesmente Encadeada**
- Definição das principais operações:

```
1 interface IListaEncadeada {  
2     void insere ( No elementoAnterior, No novoElemento );  
3     No busca( int idProcurado );  
4     void remove( int idNoRemovido);  
5 }
```



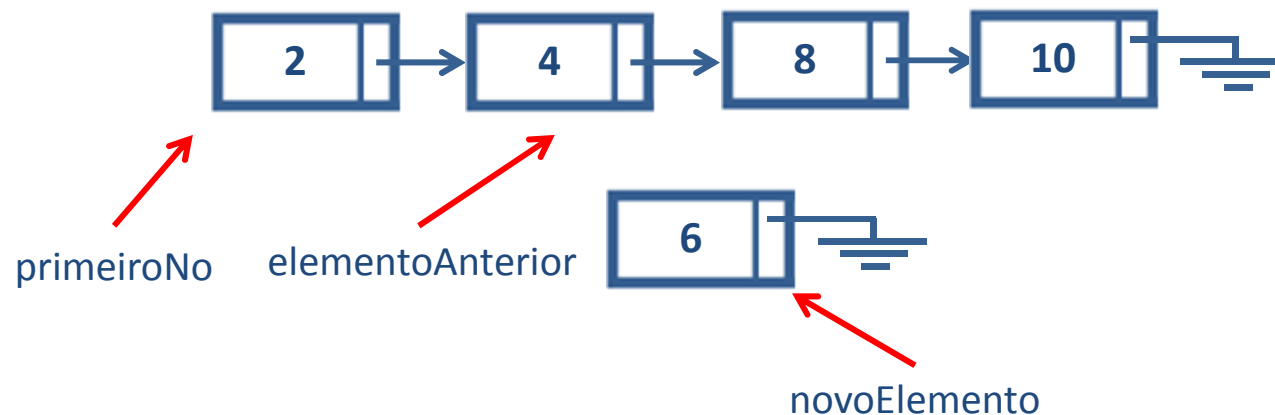
- **Lista Simplesmente Encadeada – Operação Inclusão**
- Implementação em java:

```
1 public void insere( No elementoAnterior, No novoElemento ){  
2     novoElemento.proximo = elementoAnterior.proximo;  
3     elementoAnterior.proximo = novoElemento;  
4 }
```



## ➤ Lista Simplesmente Encadeada – Operação Inclusão

```
// Execução do método  
listaEncadeada.insere( elemento4, elemento6 );
```

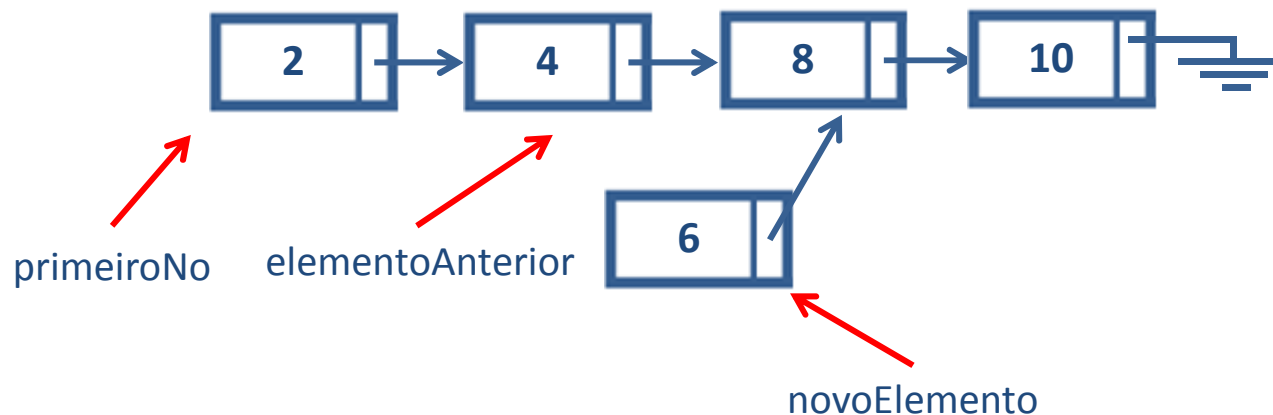


```
1 public void insere( No elementoAnterior, No novoElemento ){  
2     novoElemento.proximo = elementoAnterior.proximo;  
3     elementoAnterior.proximo = novoElemento;  
4 }
```



## ➤ Lista Simplesmente Encadeada – Operação Inclusão

```
// Execução do método  
listaEncadeada.insere( elemento4, elemento6 );
```

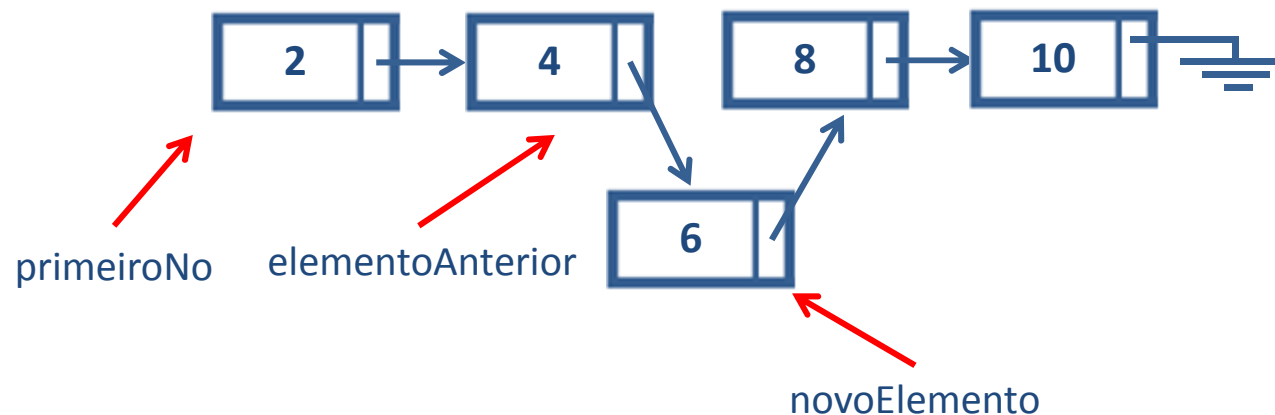


```
1 public void insere( No elementoAnterior, No novoElemento ){  
→ 2     novoElemento.proximo = elementoAnterior.proximo;  
3     elementoAnterior.proximo = novoElemento;  
4 }
```



## ➤ Lista Simplesmente Encadeada – Operação Inclusão

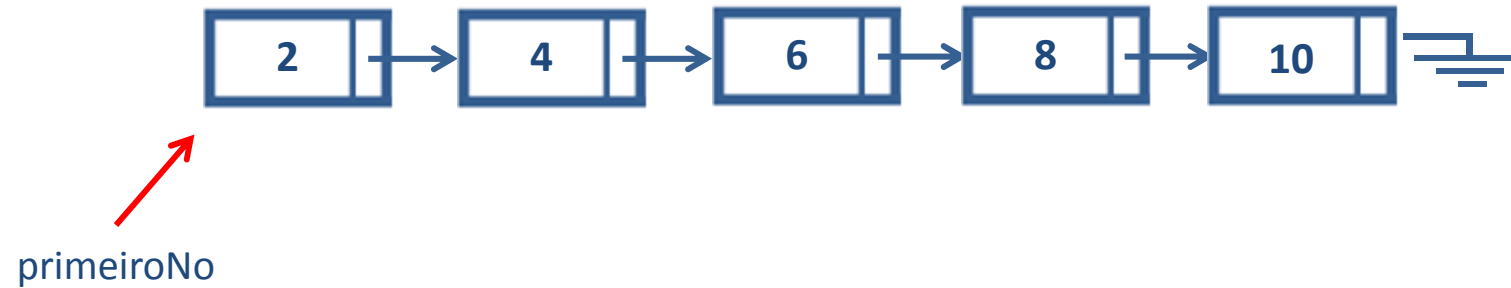
```
// Execução do método  
listaEncadeada.insere( elemento4, elemento6 );
```



```
1 public void insere( No elementoAnterior, No novoElemento ){  
2     novoElemento.proximo = elementoAnterior.proximo;  
→ 3     elementoAnterior.proximo = novoElemento;  
4 }
```



## ➤ Lista Simplesmente Encadeada – Operação Inclusão





- **Lista Simplesmente Encadeada – Operação Busca**
  - Uma vez que a lista encadeada não permite acesso direto a um nó, a mesma não admite a busca binária.
  - Para buscar um nó em uma lista encadeada devemos percorrer, a partir do primeiro elemento, todos os demais até encontrar o nó desejado.

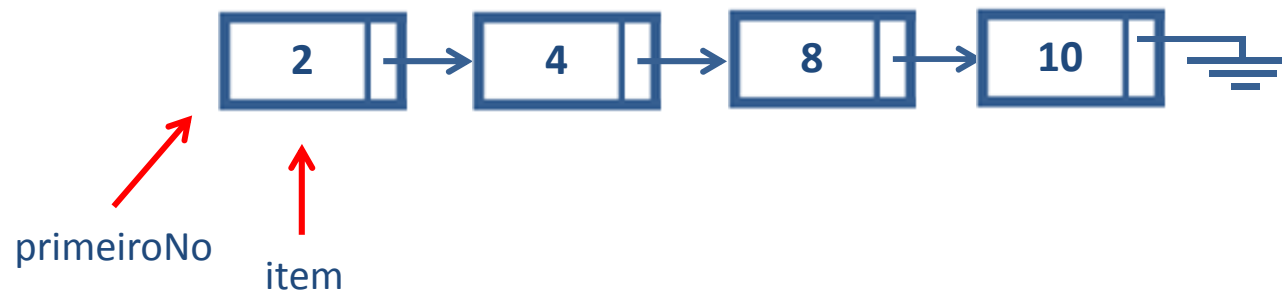


- **Lista Simplesmente Encadeada – Operação Busca**
- Implementação em java:

```
1 public No busca( No primeiroNo, int idProcurado ){
2     No item = primeiroNo;
3     while ( item != null && item.id != idProcurado ) {
4         item = item.proximo;
5     }
6
7     return item;
8 }
9 }
```



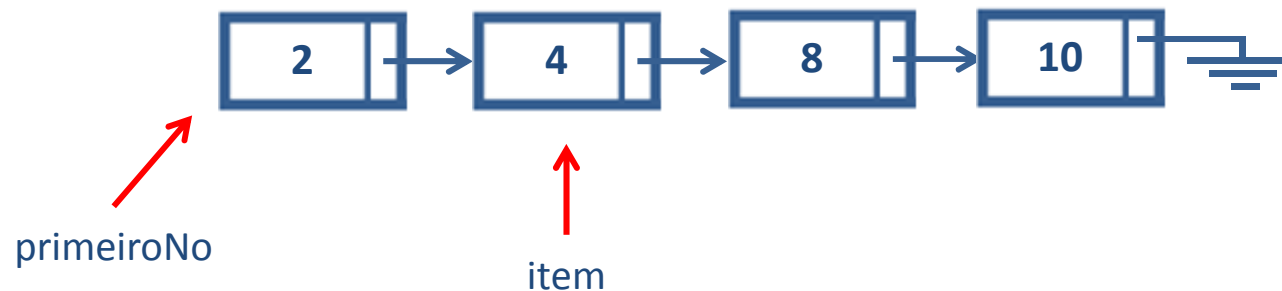
- **Lista Simplesmente Encadeada – Operação Busca**
  - Como encontrar o nó de chave 8 na lista a seguir?



```
1 public No busca( No primeiroNo, int idProcurado ){
2     No item = primeiroNo;
3     while ( item != null && item.id != idProcurado ) {
4         item = item.proximo;
5     }
6
7     return item;
8 }
9 }
```



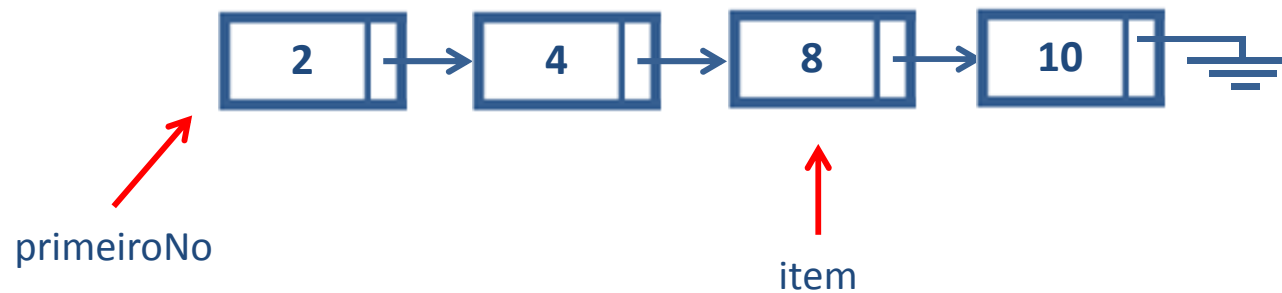
- **Lista Simplesmente Encadeada – Operação Busca**
  - Como encontrar o nó de chave 8 na lista a seguir?



```
1 public No busca( No primeiroNo, int idProcurado ){
2     No item = primeiroNo;
3     while ( item != null && item.id != idProcurado ) {
4         item = item.proximo;
5     }
6
7     return item;
8 }
9 }
```



- **Lista Simplesmente Encadeada – Operação Busca**
  - Como encontrar o nó de chave 8 na lista a seguir?



```
1 public No busca( No primeiroNo, int idProcurado ){
2     No item = primeiroNo;
3     while ( item != null && item.id != idProcurado ) {
4         item = item.proximo;
5     }
6
7     return item;
8 }
9 }
```

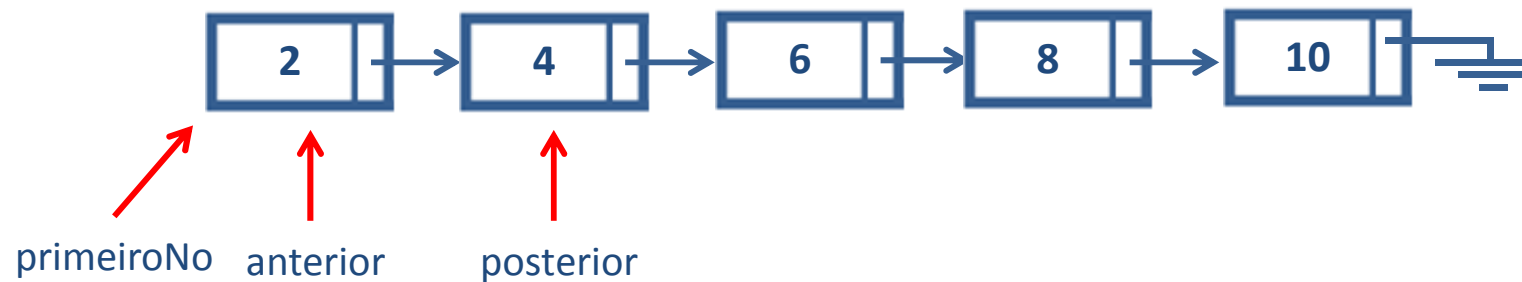


- Lista Simplesmente Encadeada – Operação Remoção
  - Implementação da remoção de um nó da lista:

```
1 public void remove( int idNoRemovido){
2
3     // Trata o caso do primeiro nó ser o removido
4     if (primeiroNo.id == idNoRemovido){
5         primeiroNo = primeiroNo.proximo;
6         return;
7     }
8
9     // trata os demais casos
10    No temp = primeiroNo;
11    while (temp!=null) {
12        No anterior = temp;
13        No posterior = temp.proximo;
14
15        if (posterior.id == idNoRemovido) {
16            anterior.proximo = posterior.proximo;
17            return;
18        }
19
20        temp = temp.proximo;
21    }
22 }
```



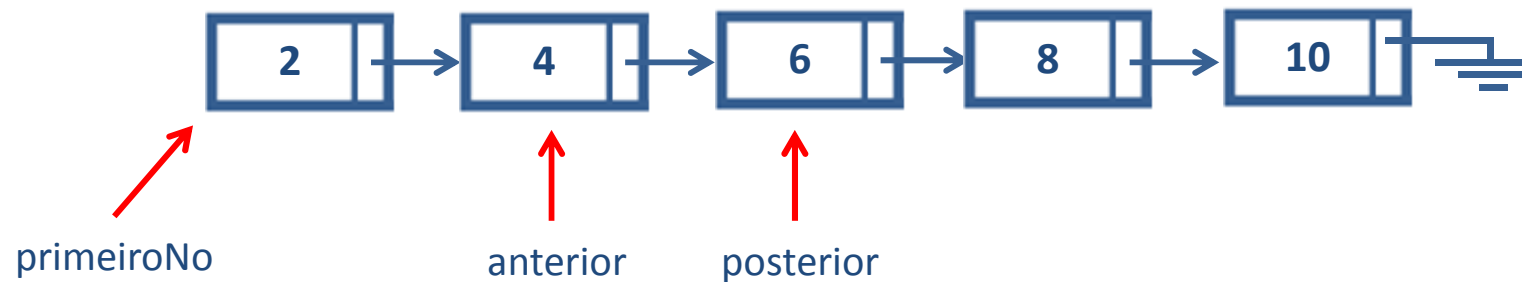
- **Lista Simplesmente Encadeada – Operação Remoção**
  - Como remover o nó de chave 8 da lista a seguir?



```
1 public void remove( int idNoRemovido){
2
3     // Trata o caso do primeiro nó ser o removido
4     if (primeiroNo.id == idNoRemovido){
5         primeiroNo = primeiroNo.proximo;
6         return;
7     }
8
9     // trata os demais casos
10    No temp = primeiroNo;
11    while (temp!=null) {
12        No anterior = temp;
13        No posterior = temp.proximo;
14
15        if (posterior.id == idNoRemovido) {
16            anterior.proximo = posterior.proximo;
17            return;
18        }
19
20        temp = temp.proximo;
21    }
22 }
```



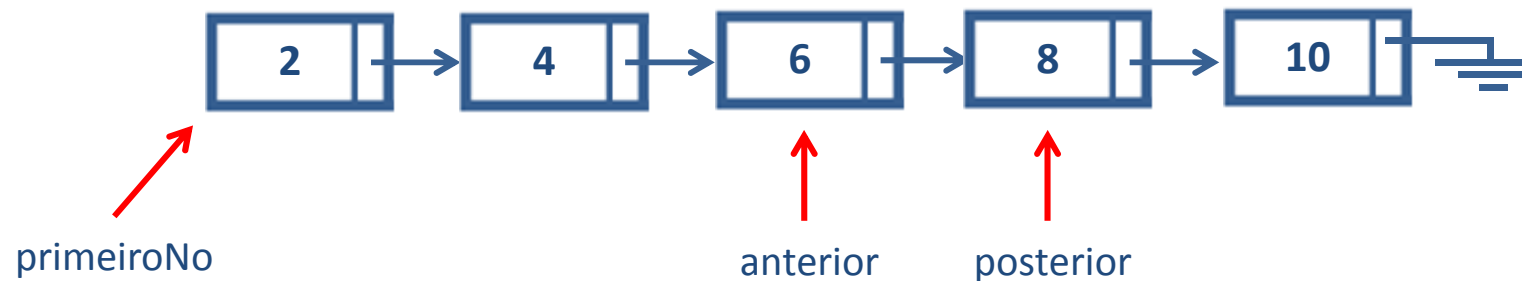
- **Lista Simplesmente Encadeada – Operação Remoção**
  - Como remover o nó de chave 8 da lista a seguir?



```
1 public void remove( int idNoRemovido){
2
3     // Trata o caso do primeiro nó ser o removido
4     if (primeiroNo.id == idNoRemovido){
5         primeiroNo = primeiroNo.proximo;
6         return;
7     }
8
9     // trata os demais casos
10    No temp = primeiroNo;
11    while (temp!=null) {
12        No anterior = temp;
13        No posterior = temp.proximo;
14
15        if (posterior.id == idNoRemovido) {
16            anterior.proximo = posterior.proximo;
17            return;
18        }
19
20        temp = temp.proximo;
21    }
22 }
```



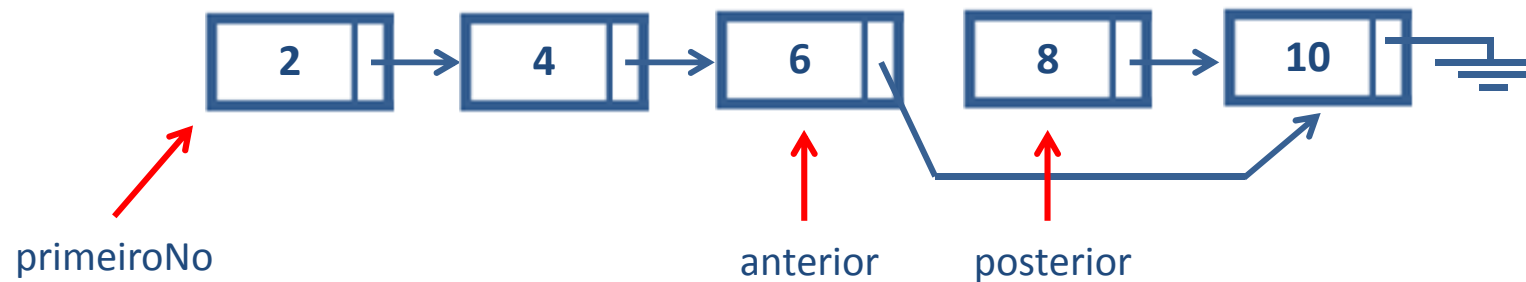
- **Lista Simplesmente Encadeada – Operação Remoção**
  - Como remover o nó de chave 8 da lista a seguir?



```
1 public void remove( int idNoRemovido){
2
3     // Trata o caso do primeiro nó ser o removido
4     if (primeiroNo.id == idNoRemovido){
5         primeiroNo = primeiroNo.proximo;
6         return;
7     }
8
9     // trata os demais casos
10    No temp = primeiroNo;
11    while (temp!=null) {
12        No anterior = temp;
13        No posterior = temp.proximo;
14
15        if (posterior.id == idNoRemovido) {
16            anterior.proximo = posterior.proximo;
17            return;
18        }
19
20        temp = temp.proximo;
21    }
22 }
```



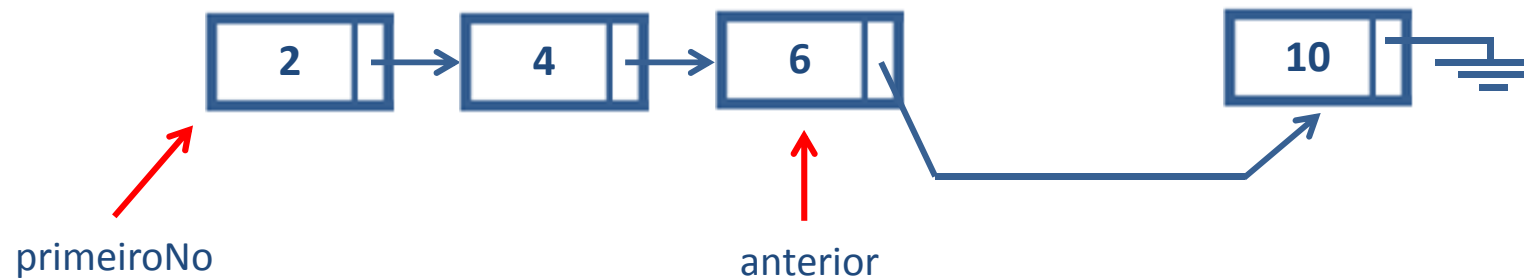
- **Lista Simplesmente Encadeada – Operação Remoção**
  - Como remover o nó de chave 8 da lista a seguir?



```
1 public void remove( int idNoRemovido){
2
3     // Trata o caso do primeiro nó ser o removido
4     if (primeiroNo.id == idNoRemovido){
5         primeiroNo = primeiroNo.proximo;
6         return;
7     }
8
9     // trata os demais casos
10    No temp = primeiroNo;
11    while (temp!=null) {
12        No anterior = temp;
13        No posterior = temp.proximo;
14
15        if (posterior.id == idNoRemovido) {
16            anterior.proximo = posterior.proximo;
17            return;
18        }
19
20        temp = temp.proximo;
21    }
22 }
```

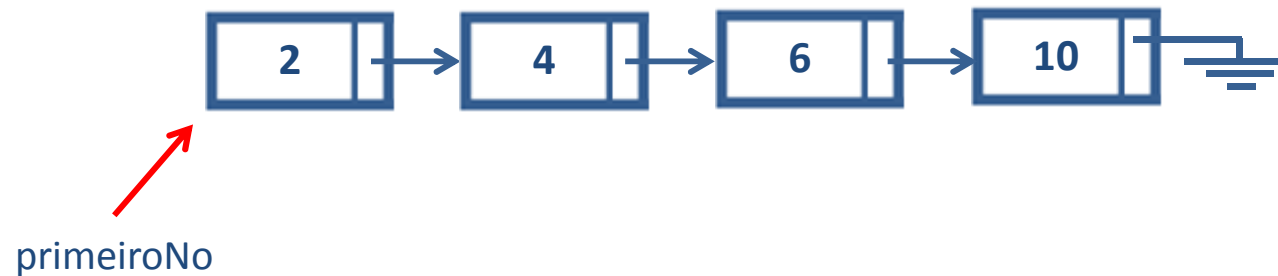


- **Lista Simplesmente Encadeada – Operação Remoção**
  - Como remover o nó de chave 8 da lista a seguir?





- **Lista Simplesmente Encadeada – Operação Remoção**
  - Como remover o nó de chave 8 da lista a seguir?

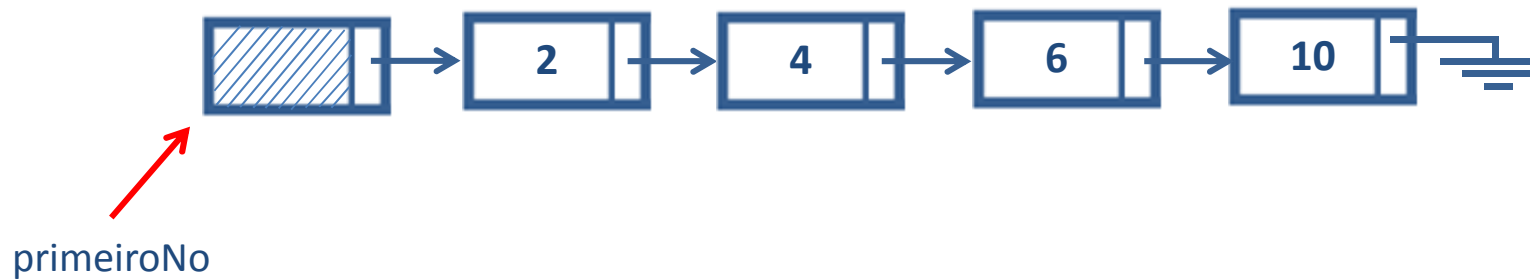




- **Lista Simplesmente Encadeada – Nó cabeça**
  - A existência de um ponteiro referenciando o início da lista obriga alguns algoritmos a apresentarem testes especiais na realização de suas operações.
  - Isso pode ser resolvido por uma pequena variação na estrutura de armazenamento: a criação de um nó-cabeça, nunca removido, que passa a ser o nó indicado pelo ponteiro de início da lista.
  - Esse nó especial não deve conter informações relacionadas a tabela propriamente dita.



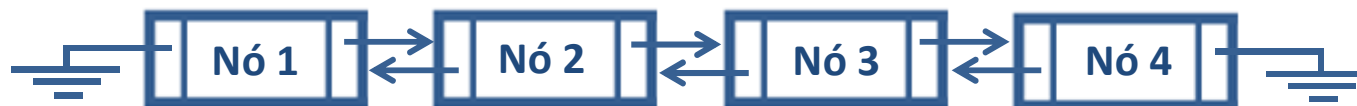
- **Lista Simplesmente Encadeada – Nó cabeça**
  - O nó cabeça também pode ser denominado sentinela ou header.





## ➤ Lista Duplamente Encadeada

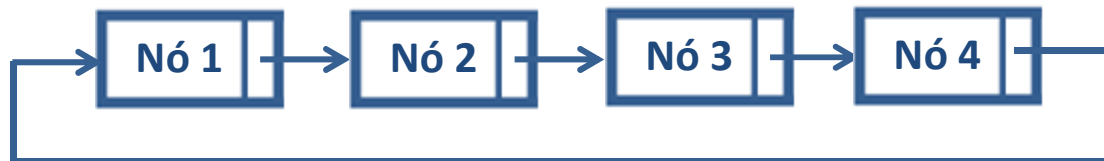
- São listas que possuem uma referência para o nó predecessor e outra para o nó sucessor.
- Essa adaptação da estrutura possibilita que a lista seja percorrida em ambas as direções.
- Comparada com a lista simplesmente encadeada, essa estrutura possui um consumo de memória ainda mais elevado.





## ➤ Lista Circular

- Uma lista circular tem como particularidade seu último nó referenciar o primeiro nó da lista.
- A lista circular pode ser simplesmente encadeada ou duplamente encadeada.





# Questão 01

## [2012 - CESPE - Banco da Amazônia - Administração de Dados]

Em algumas implementações, uma lista vazia pode ter um único nó, chamado de sentinela, nó cabeça ou header. Entre suas possíveis funções, inclui-se simplificar a implementação de algumas operações realizadas sobre a lista, como inserir novos dados, recuperar o tamanho da lista, entre outras.

Certo

Errado



# Questão 01

## [2012 - CESPE - Banco da Amazônia - Administração de Dados]

Em algumas implementações, uma lista vazia pode ter um único nó, chamado de sentinela, nó cabeça ou header. Entre suas possíveis funções, inclui-se simplificar a implementação de algumas operações realizadas sobre a lista, como inserir novos dados, recuperar o tamanho da lista, entre outras.

➡ Certo      Errado



## Questão 02

### [2004 - CESPE - TRE-AL - Analista de Tecnologia da Informação]

A atividade de programação requer conhecimento técnico de diversas formas de algoritmos e estruturas de controle e de dados.

Acerca dos elementos técnicos da atividade de programação, julgue os itens a seguir.

Quando o número de acessos randômicos a uma área de armazenamento é muito maior que o número de inserções e remoções de elementos armazenados, a organização dessa área de armazenamento por meio de uma lista encadeada resulta em desempenho melhor que o apresentado por organização feita mediante uma estrutura de array.

Certo                  Errado



## Questão 02

### [2004 - CESPE - TRE-AL - Analista de Tecnologia da Informação]

A atividade de programação requer conhecimento técnico de diversas formas de algoritmos e estruturas de controle e de dados.

Acerca dos elementos técnicos da atividade de programação, julgue os itens a seguir.

Quando o número de acessos randômicos a uma área de armazenamento é muito maior que o número de inserções e remoções de elementos armazenados, a organização dessa área de armazenamento por meio de uma lista encadeada resulta em desempenho melhor que o apresentado por organização feita mediante uma estrutura de array.

Certo      ➡ Errado



## Questão 03

### **[2012 – CESPE - Banco da Amazônia - Técnico de Administração de Dados]**

Julgue os próximos itens, relativos a tipos básicos de estruturas de dados.

O tempo de busca de um elemento em uma lista duplamente encadeada é igual à metade do tempo da busca de um elemento em uma lista simplesmente encadeada.

Certo          Errado



## Questão 03

**[2012 – CESPE - Banco da Amazônia - Técnico de Administração de Dados]**

Julgue os próximos itens, relativos a tipos básicos de estruturas de dados.

O tempo de busca de um elemento em uma lista duplamente encadeada é igual à metade do tempo da busca de um elemento em uma lista simplesmente encadeada.

Certo      ➡ Errado



## Questão 04

### [2012 - CESPE - Banco da Amazônia - Técnico de Administração de Dados]

As listas duplamente encadeadas diferenciam-se das listas simplesmente encadeadas pelo fato de, na primeira, os nós da lista formarem um anel com o último elemento ligado ao primeiro da lista.

Certo          Errado



## Questão 04

### [2012 - CESPE - Banco da Amazônia - Técnico de Administração de Dados]

As listas duplamente encadeadas diferenciam-se das listas simplesmente encadeadas pelo fato de, na primeira, os nós da lista formarem um anel com o último elemento ligado ao primeiro da lista.

Certo      ➡ Errado



## Questão 05

### [2006 - CESGRANRIO - EPE - Técnico de Nível Superior de TI]

Os registros em uma lista, duplamente encadeada com 20 elementos possuem cada um três campos:

**próximo:** um ponteiro para o próximo elemento da lista;

**valor:** informação armazenada pelo elemento;

**anterior:** um ponteiro para o elemento anterior da lista.

Sendo "Z" o décimo elemento desta lista e "X" e "Y" dois outros elementos que não pertencem à lista, com seus respectivos ponteiros "pZ", "pX" e "pY", considere o trecho de código abaixo.

```
pY↑.próximo = pX;  
pX↑.anterior = pY;  
pX↑.próximo = pZ↑.próximo;  
pZ↑.próximo↑.anterior = pX;  
pZ↑.próximo = pY;  
pY↑.anterior = pZ;
```



## Questão 05

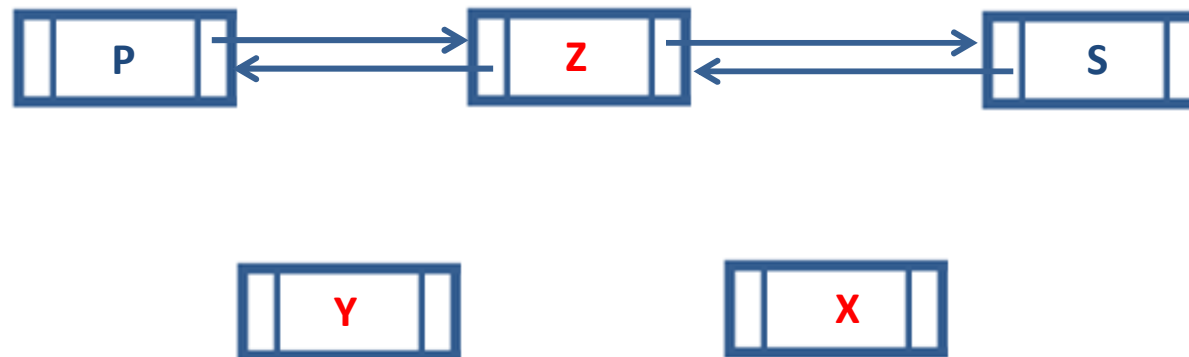
Este trecho de código é usado para inserir na lista os elementos:

- A) Y, logo após o Z, e X, logo após o Y.
- B) Y, antes do Z, e X, logo após o Z.
- C) Y, antes do Z, e X, antes do Y.
- D) X, logo após o Z, e Y, logo após o X.
- E) X, antes do Z, e Y, logo após o Z.



## Questão 05

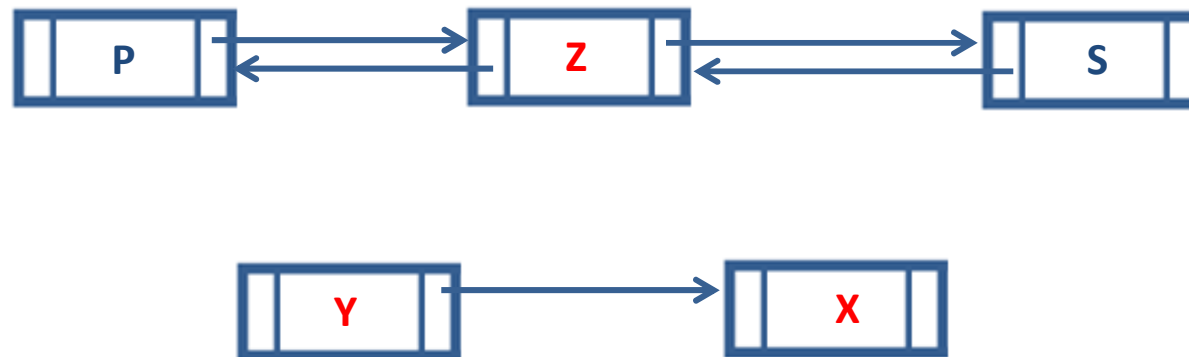
```
pY↑.próximo = pX;  
pX↑.anterior = pY;  
pX↑.próximo = pZ↑.próximo;  
pZ↑.próximo↑.anterior = pX;  
pZ↑.próximo = pY;  
pY↑.anterior = pZ;
```





## Questão 05

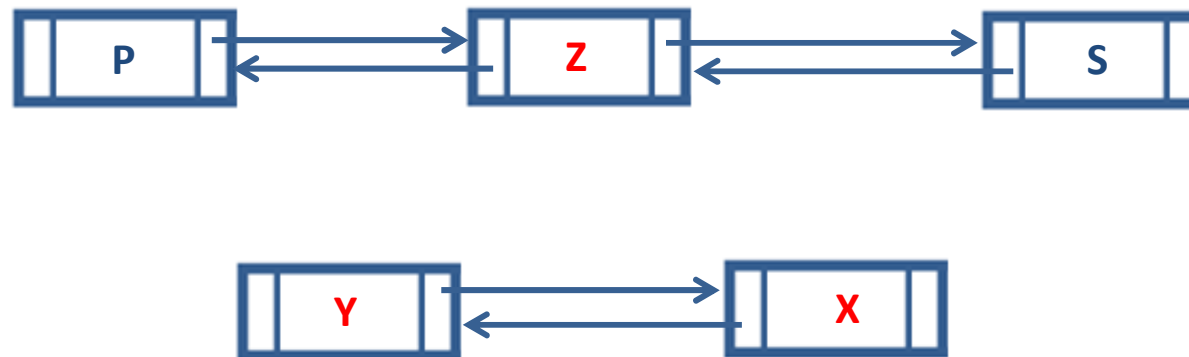
→  $pY↑.próximo = pX;$   
 $pX↑.anterior = pY;$   
 $pX↑.próximo = pZ↑.próximo;$   
 $pZ↑.próximo↑.anterior = pX;$   
 $pZ↑.próximo = pY;$   
 $pY↑.anterior = pZ;$





## Questão 05

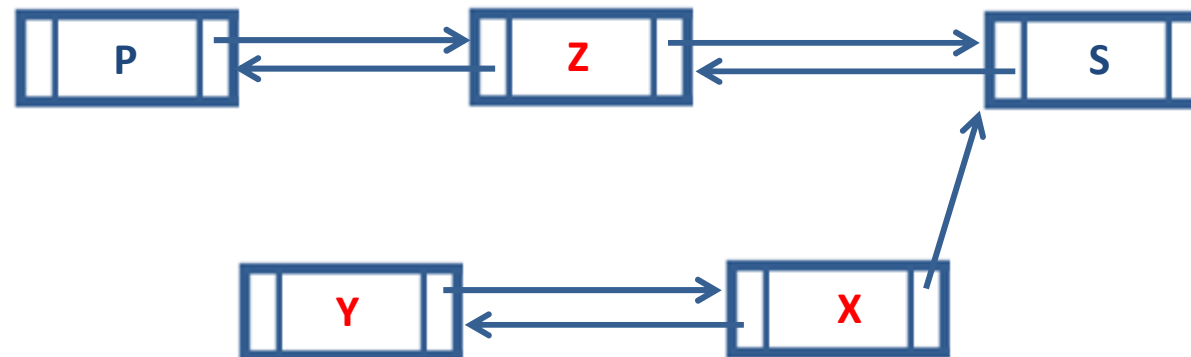
$pY \uparrow . \text{próximo} = pX;$   
 $\rightarrow pX \uparrow . \text{anterior} = pY;$   
 $pX \uparrow . \text{próximo} = pZ \uparrow . \text{próximo};$   
 $pZ \uparrow . \text{próximo} \uparrow . \text{anterior} = pX;$   
 $pZ \uparrow . \text{próximo} = pY;$   
 $pY \uparrow . \text{anterior} = pZ;$





## Questão 05

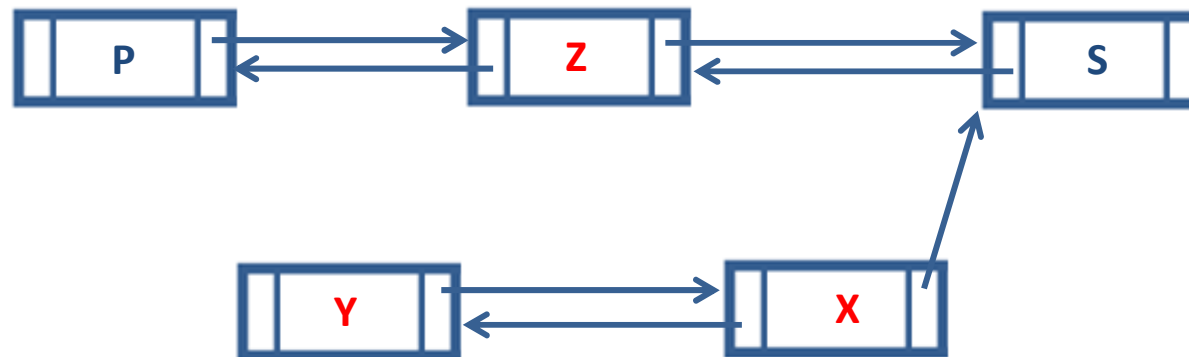
```
pY↑.próximo = pX;  
pX↑.anterior = pY;  
→ pX↑.próximo = pZ↑.próximo;  
pZ↑.próximo↑.anterior = pX;  
pZ↑.próximo = pY;  
pY↑.anterior = pZ;
```





## Questão 05

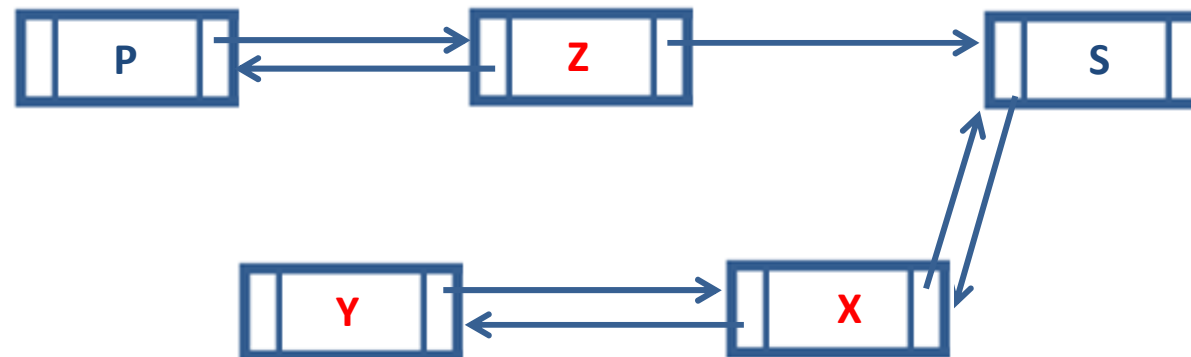
```
pY↑.próximo = pX;  
pX↑.anterior = pY;  
pX↑.próximo = pZ↑.próximo;  
→ pZ↑.próximo↑.anterior = pX;  
pZ↑.próximo = pY;  
pY↑.anterior = pZ;
```





## Questão 05

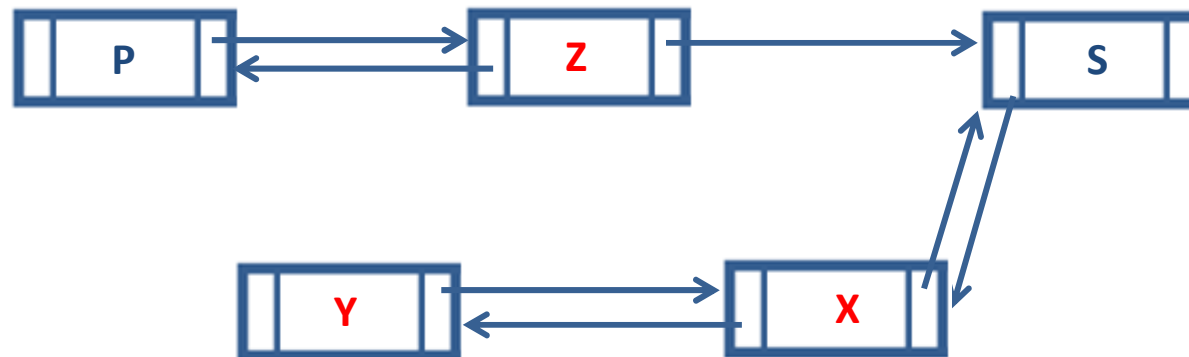
```
pY↑.próximo = pX;  
pX↑.anterior = pY;  
pX↑.próximo = pZ↑.próximo;  
→ pZ↑.próximo↑.anterior = pX;  
pZ↑.próximo = pY;  
pY↑.anterior = pZ;
```





## Questão 05

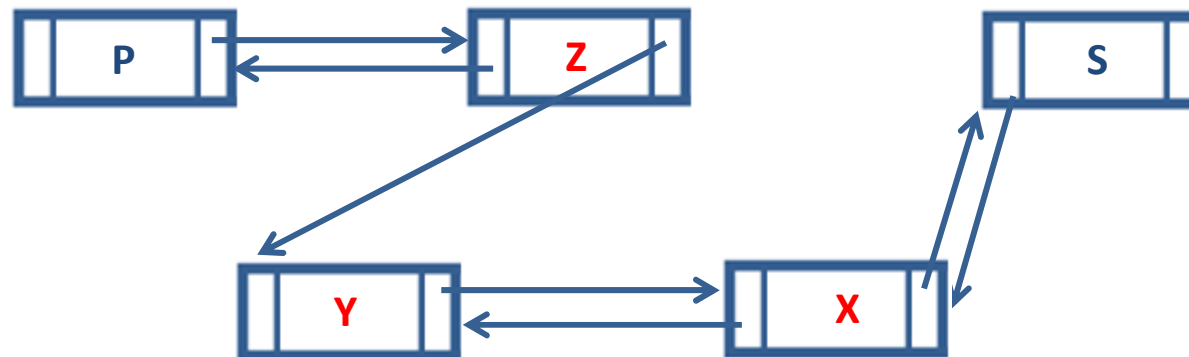
```
pY↑.próximo = pX;  
pX↑.anterior = pY;  
pX↑.próximo = pZ↑.próximo;  
pZ↑.próximo↑.anterior = pX;  
→ pZ↑.próximo = pY;  
pY↑.anterior = pZ;
```





## Questão 05

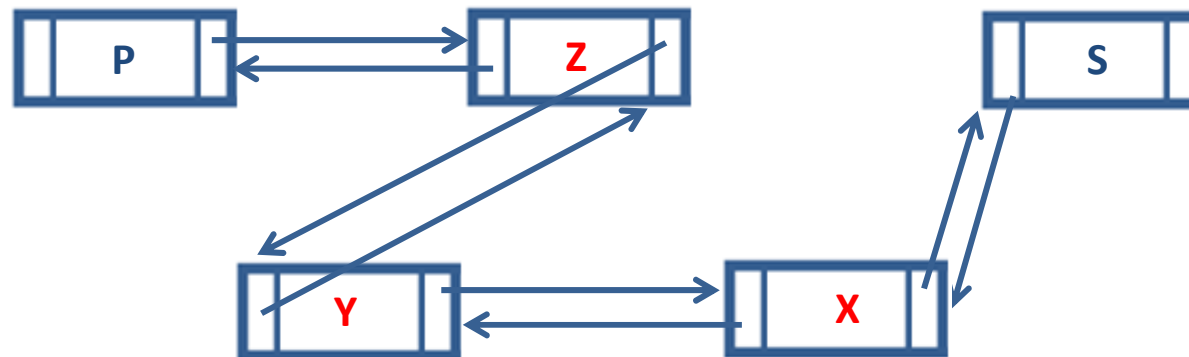
```
pY↑.próximo = pX;  
pX↑.anterior = pY;  
pX↑.próximo = pZ↑.próximo;  
pZ↑.próximo↑.anterior = pX;  
→ pZ↑.próximo = pY;  
pY↑.anterior = pZ;
```





## Questão 05

```
pY↑.próximo = pX;  
pX↑.anterior = pY;  
pX↑.próximo = pZ↑.próximo;  
pZ↑.próximo↑.anterior = pX;  
pZ↑.próximo = pY;  
→ pY↑.anterior = pZ;
```





## Questão 05

Este trecho de código é usado para inserir na lista os elementos:

- ➔ A) Y, logo após o Z, e X, logo após o Y.
- B) Y, antes do Z, e X, logo após o Z.
- C) Y, antes do Z, e X, antes do Y.
- D) X, logo após o Z, e Y, logo após o X.
- E) X, antes do Z, e Y, logo após o Z.







**01 – Certo**

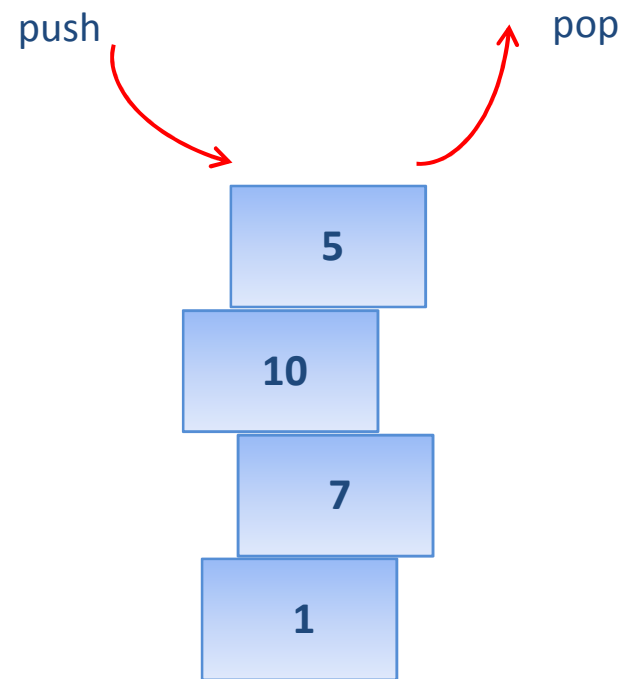
**02 – Errado**

**03 – Errado**

**04 – Errado**

**05 – A**





**Pilhas**



## ➤ Conceitos Gerais

- A pilha é uma estrutura de dados onde as inserções e as remoções são realizadas em apenas um extremo.
- O primeiro objeto a ser inserido na pilha é o último a ser removido. Por trabalhar com essa política, essa estrutura é conhecida como LIFO (last-in, first-out).
- Essa estrutura usa uma variável de controle comumente chamada de *topo* que referencia o último elemento inserido na pilha.



- Principais operações de uma pilha
  - Definição das principais operações.

```
1 interface IPilha {  
2     public boolean isPilhaVazia();  
3     public void push( Object elemento );  
4     public Object pop();  
5     public Object peak();  
6 }
```



- **Operação isPilhaVazia**

- Essa operação verifica se a pilha esta vazia.
- Implementação em java:

```
1 public boolean isPilhaVazia(){  
2     return topo==-1;  
3 }
```



## ➤ Operação push

- Essa operação insere um elemento no topo da pilha.
- Implementação em java:

```
1 public void push( Object elemento ){  
2     if (topo!=COMPRIMENTO_PILHA-1) {  
3         elementos[++topo] = elemento;  
4     } else {  
5         System.out.println("Estouro de pilha: OverFlow");  
6     }  
7 }
```



- **Operação push**

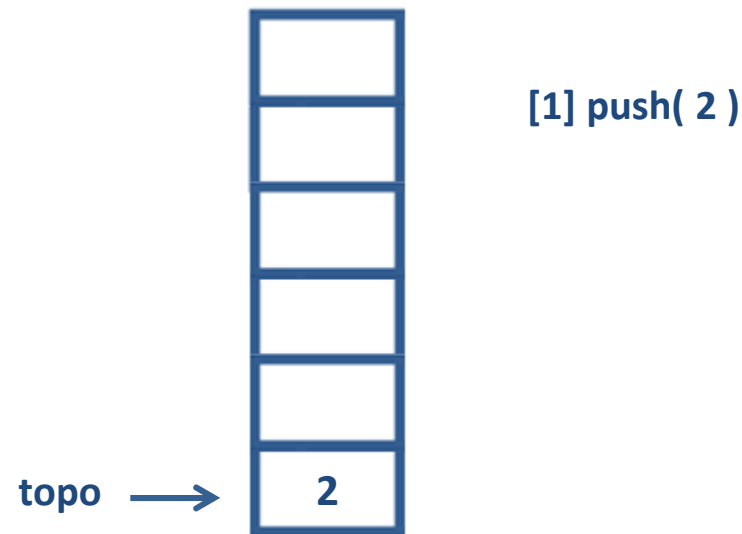
- Como ficaria uma pilha, inicialmente vazia, após a inclusão dos elementos: 2, 10, 30, 1, 56?





## ➤ Operação push

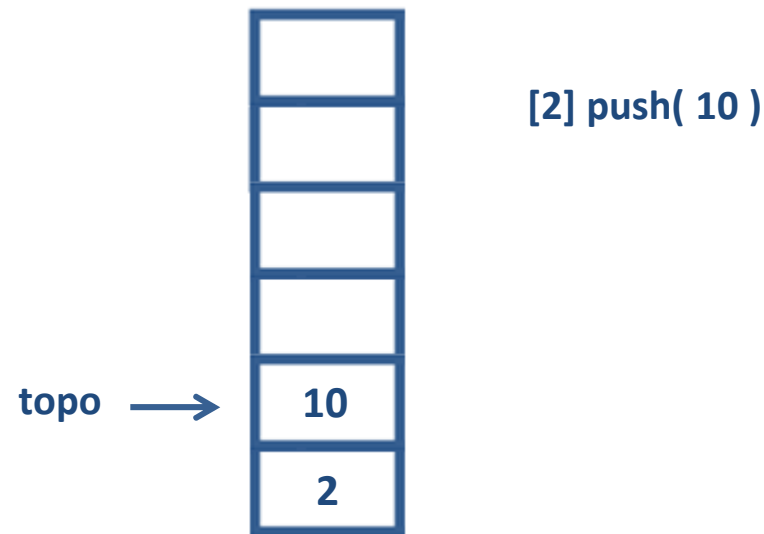
- Como ficaria uma pilha, inicialmente vazia, após a inclusão dos elementos: **2**, 10, 30, 1, 56?





- **Operação push**

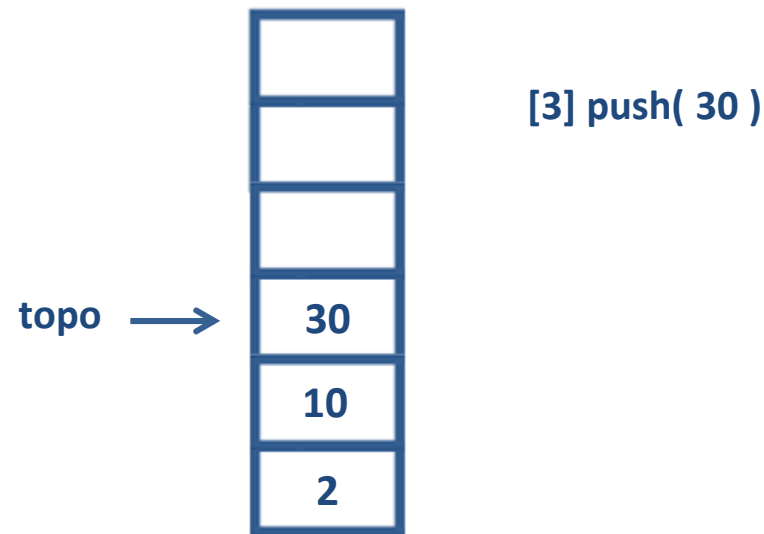
- Como ficaria uma pilha, inicialmente vazia, após a inclusão dos elementos: 2, **10**, 30, 1, 56?





- **Operação push**

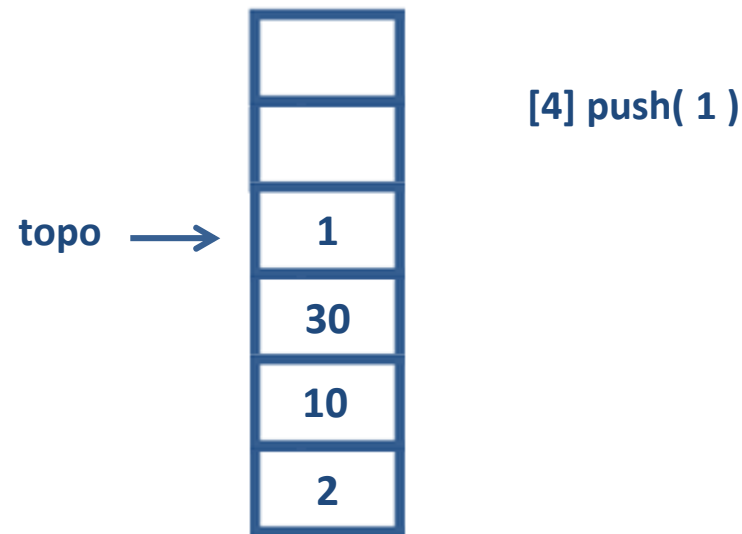
- Como ficaria uma pilha, inicialmente vazia, após a inclusão dos elementos: 2, 10, **30**, 1, 56?





## ➤ Operação push

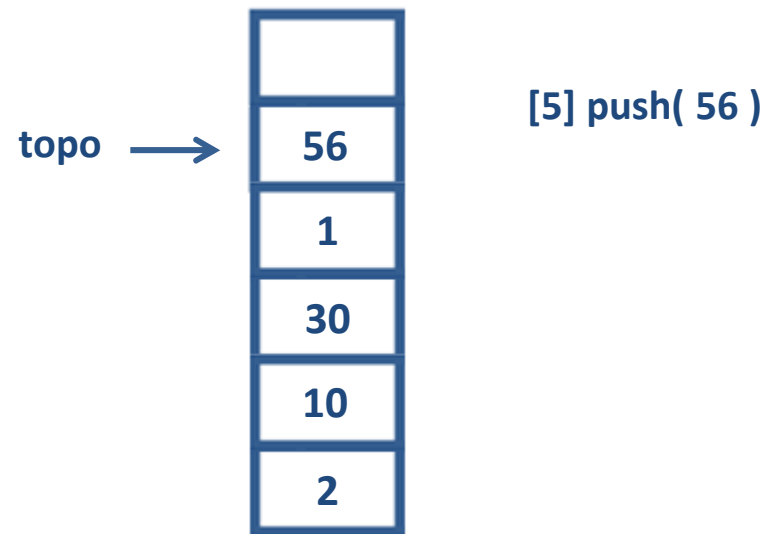
- Como ficaria uma pilha, inicialmente vazia, após a inclusão dos elementos: 2, 10, 30, **1**, 56?





## ➤ Operação push

- Como ficaria uma pilha, inicialmente vazia, após a inclusão dos elementos: 2, 10, 30, 1, **56**?





## ➤ Operação pop

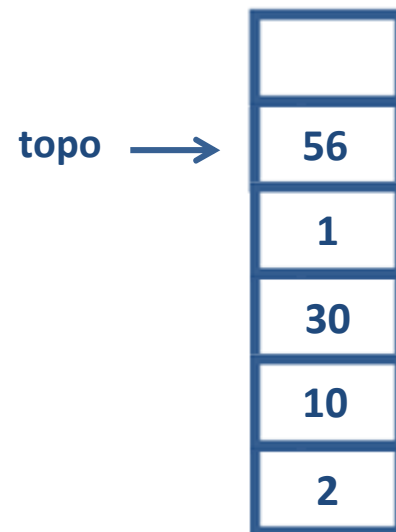
- Essa operação remove o elemento do topo da pilha.
- Implementação em java:

```
1 public Object pop(){
2     if (!isPilhaVazia()){
3         Object object = elementos[topo];
4         elementos[topo--] = null;
5         return object;
6     } else {
7         System.out.println( "Estouro de pilha: UnderFlow" );
8         return null;
9     }
10 }
```



## ➤ Operação pop

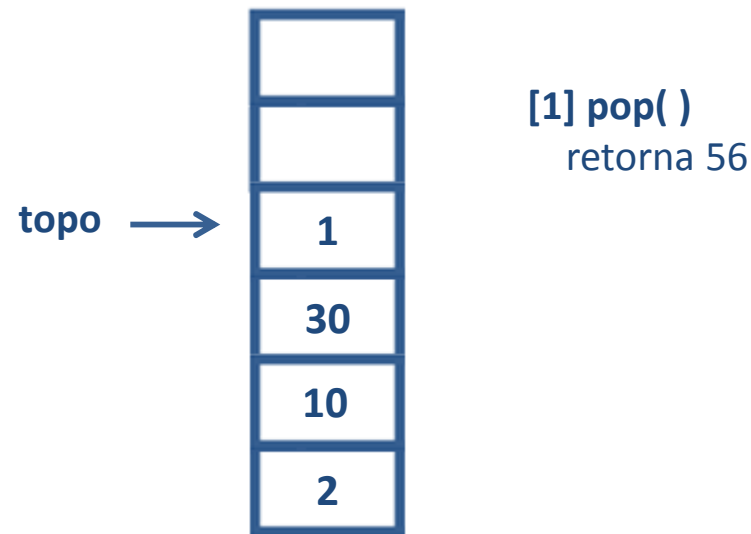
- Como ficaria essa pilha após a execução da operação *pop* três vezes seguidas?





## ➤ Operação pop

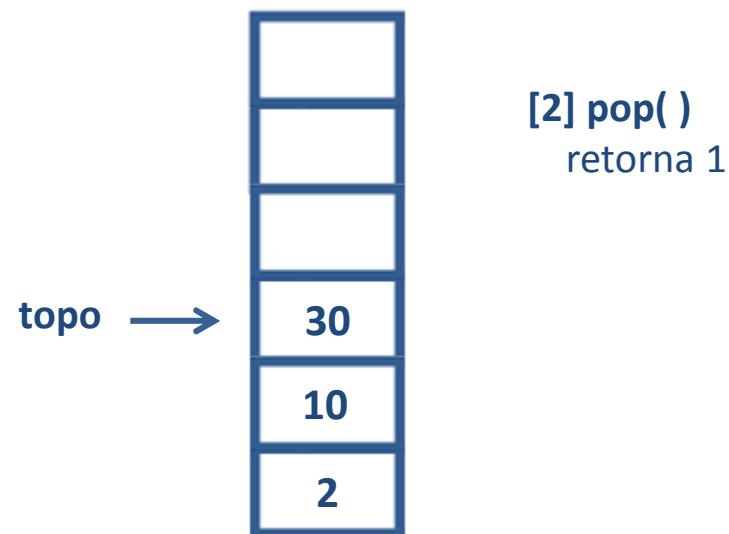
- Como ficaria essa pilha após a execução da operação *pop* três vezes seguidas?





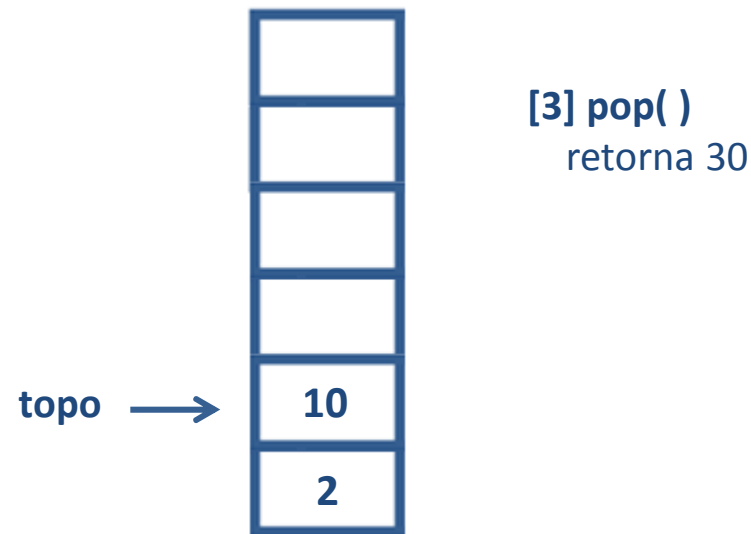
## ➤ Operação pop

- Como ficaria essa pilha após a execução da operação *pop* três vezes seguidas?





- **Operação pop**
  - Como ficaria essa pilha após a execução da operação *pop* três vezes seguidas?



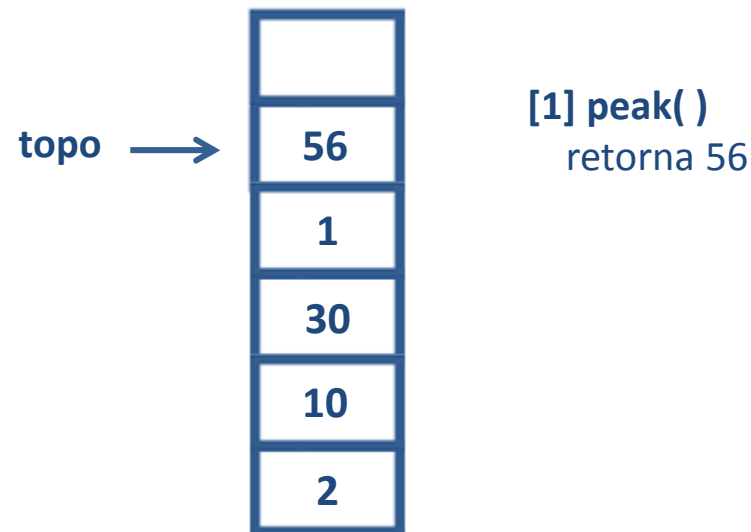


- **Operação peak (top)**
  - Essa operação permite consultar o elemento do topo da pilha sem a remoção do mesmo.
  - Implementação em java:

```
1 public Object peak(){  
2     if (!isPilhaVazia()){  
3         return elementos[topo];  
4     }  
5  
6     return null;  
7 }
```

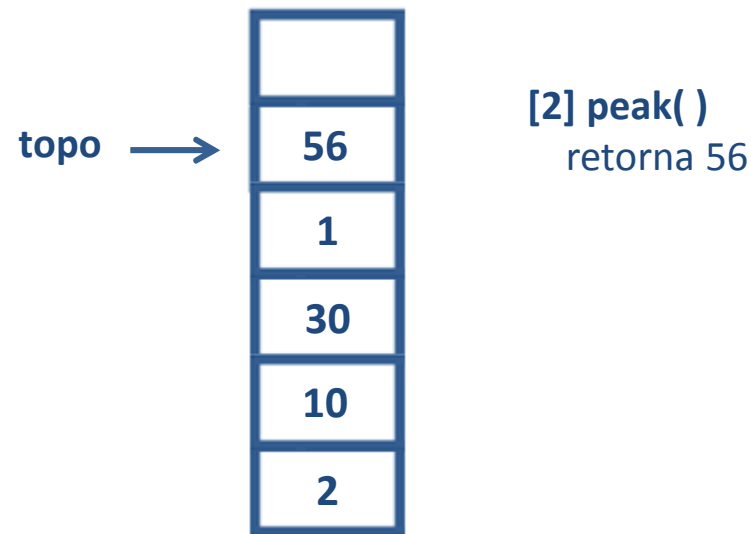


- **Operação peak / top**
  - Como ficaria essa pilha após a execução da operação *peak()* duas vezes seguidas?





- **Operação peak / top**
  - Como ficaria essa pilha após a execução da operação *peak()* duas vezes seguidas?





## Questão 06

### [2013 - ESAF - DNIT - Analista Tecnologia da Informação]

Assinale a opção correta relativa às operações básicas suportadas por pilhas.

- A) Push: insere um novo elemento no final da pilha.
- B) Pop: adiciona elementos ao topo da pilha.
- C) Pull: insere um novo elemento no interior da pilha.
- D) Top: transfere o último elemento para o topo da pilha.
- E) Top: acessa o elemento posicionado no topo da pilha.



## Questão 06

### [2013 - ESAF - DNIT - Analista Tecnologia da Informação]

Assinale a opção correta relativa às operações básicas suportadas por pilhas.

- A) Push: insere um novo elemento no final da pilha.
- B) Pop: adiciona elementos ao topo da pilha.
- C) Pull: insere um novo elemento no interior da pilha.
- D) Top: transfere o último elemento para o topo da pilha.
- ➡ E) Top: acessa o elemento posicionado no topo da pilha.



## Questão 07

### [2012 - FCC - TRE-CE - Programador de computador]

Sobre pilhas é correto afirmar:

- A) Uma lista LIFO (Last-In/First-Out) é uma estrutura estática, ou seja, é uma coleção que não pode aumentar e diminuir durante sua existência.
- B) Os elementos na pilha são sempre removidos na mesma ordem em que foram inseridos.
- C) Uma pilha suporta apenas duas operações básicas, tradicionalmente denominadas push (insere um novo elemento no topo da pilha) e pop (remove um elemento do topo da pilha).
- D) Cada vez que um novo elemento deve ser inserido na pilha, ele é colocado no seu topo e, em qualquer momento, apenas aquele posicionado no topo da pilha pode ser removido.
- E) Sendo P uma pilha e x um elemento qualquer, a operação Push(P,x) diminui o tamanho da pilha P, removendo o elemento x do seu topo.



## Questão 07

### [2012 - FCC - TRE-CE - Programador de computador]

Sobre pilhas é correto afirmar:

- A) Uma lista LIFO (Last-In/First-Out) é uma estrutura estática, ou seja, é uma coleção que não pode aumentar e diminuir durante sua existência.
- B) Os elementos na pilha são sempre removidos na mesma ordem em que foram inseridos.
- C) Uma pilha suporta apenas duas operações básicas, tradicionalmente denominadas push (insere um novo elemento no topo da pilha) e pop (remove um elemento do topo da pilha).
- ➡ D) Cada vez que um novo elemento deve ser inserido na pilha, ele é colocado no seu topo e, em qualquer momento, apenas aquele posicionado no topo da pilha pode ser removido.
- E) Sendo P uma pilha e x um elemento qualquer, a operação Push(P,x) diminui o tamanho da pilha P, removendo o elemento x do seu topo.



## Questão 08

**[2012 - CESPE - Banco da Amazônia - Técnico de Administração de Dados]**

O uso de alocação dinâmica de memória é essencial na criação de uma pilha de dados.

Certo      Errado



## Questão 08

**[2012 - CESPE - Banco da Amazônia - Técnico de Administração de Dados]**

O uso de alocação dinâmica de memória é essencial na criação de uma pilha de dados.

Certo ➡ Errado



## Questão 09

### **[2010 - CESPE - Banco da Amazônia - Técnico de Administração de Dados]**

Na representação física de uma pilha sequencial, é necessário uso de uma variável ponteiro externa que indique a extremidade da lista linear onde ocorrem as operações de inserção e retirada de nós.

Certo      Errado



## Questão 09

### [2010 - CESPE - Banco da Amazônia - Técnico de Administração de Dados]

Na representação física de uma pilha sequencial, é necessário uso de uma variável ponteiro externa que indique a extremidade da lista linear onde ocorrem as operações de inserção e retirada de nós.

➡ Certo      Errado



## Questão 10

### [2012 - CESGRANRIO - Petrobrás - Técnico de Exploração de Petróleo]

Os algoritmos abaixo apresentam uma versão muito simples de uma estrutura de dados conhecida. Para isso, é utilizado um vetor e não há preocupações com possíveis erros de operação ou de limites ultrapassados.

```
VETOR[1..MÁXIMO] É UM VETOR DE NÚMEROS
TOPO É UM NÚMERO INTEIRO COM VALOR INICIAL 0
TEMP É UM NÚMERO INTEIRO COM VALOR INICIAL 0

FUNÇÃO COLOCA (ENTRADA : NÚMERO) NÃO RETORNA VALOR
    TOPO := TOPO + 1
    VETOR[TOPO] := ENTRADA
FIM

FUNÇÃO RETIRA() RETORNA NÚMERO
    TEMP := VETOR[TOPO]
    TOPO := TOPO - 1
    RETORNA TEMP
FIM
```

Qual a denominação da estrutura de dados implementada?

- A) Árvore binária
- B) Fila
- C) Lista encadeada
- D) Pilha
- E) Registro



## Questão 10

### [2012 - CESGRANRIO - Petrobrás - Técnico de Exploração de Petróleo]

Os algoritmos abaixo apresentam uma versão muito simples de uma estrutura de dados conhecida. Para isso, é utilizado um vetor e não há preocupações com possíveis erros de operação ou de limites ultrapassados.

```
VETOR[1..MÁXIMO] É UM VETOR DE NÚMEROS
TOPO É UM NÚMERO INTEIRO COM VALOR INICIAL 0
TEMP É UM NÚMERO INTEIRO COM VALOR INICIAL 0

FUNÇÃO COLOCA (ENTRADA : NÚMERO) NÃO RETORNA VALOR
    TOPO := TOPO + 1
    VETOR[TOPO] := ENTRADA
FIM

FUNÇÃO RETIRA() RETORNA NÚMERO
    TEMP := VETOR[TOPO]
    TOPO := TOPO - 1
    RETORNA TEMP
FIM
```

Qual a denominação da estrutura de dados implementada?

- A) Árvore binária
- B) Fila
- C) Lista encadeada
- ➡ D) Pilha
- E) Registro



# Questão 11

## [2009 - FGV - MEC - Analista de Sistemas]

A figura abaixo mostra uma aplicação da estrutura de dados pilha denominada MEC, inicialmente vazia, suportando três operações básicas, conforme definidas no Quadro I. Observe que o Quadro II apresenta uma seqüência de operações sobre a estrutura.

Quadro I		SIGNIFICADO
OPERAÇÃO	Push(PILHA,x)	Insere um elemento <u>x</u> na pilha
	Pop(PILHA)	Remove o elemento <u>x</u> do topo da pilha <u>PILHA</u>
	Top(PILHA)	Acessa, sem remover, o elemento de topo da pilha <u>PILHA</u>

Quadro II
SEQUÊNCIA DE OPERAÇÕES → Push(MEC,operacional) Push(MEC,gerencial) Push(MEC,organizacional) Push(MEC,tatico) Top(MEC,) Push(MEC,Pop(MEC)) Push(MEC,estrategico) Push(MEC,Top(MEC)) Pop(MEC) Pop(MEC)

Ao final das operações, o elemento que se encontra no topo da pilha é:

- A) organizacional
- B) operacional
- C) estrategico
- D) gerencial
- E) tatico



# Questão 11

```
Push(MEC,operacional)
Push(MEC,gerencial)
Push(MEC,organizacional)
Push(MEC,tatico)
Top(MEC,)
Push(MEC,Pop(MEC))
Push(MEC,estrategico)
Push(MEC,Top(MEC))
Pop(MEC)
Pop(MEC)
```



topo →



# Questão 11

→ Push(MEC,operacional)  
Push(MEC,gerencial)  
Push(MEC,organizacional)  
Push(MEC,tatico)  
Top(MEC,)  
Push(MEC,Pop(MEC))  
Push(MEC,estrategico)  
Push(MEC,Top(MEC))  
Pop(MEC)  
Pop(MEC)

topo →

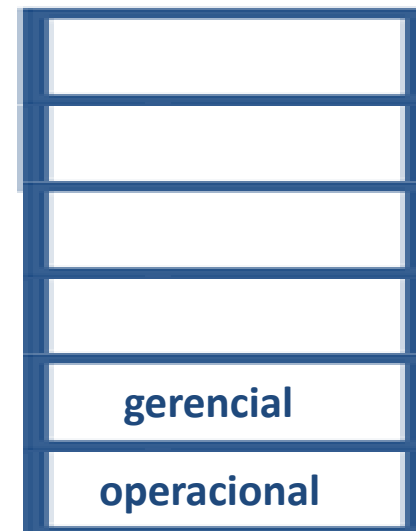




# Questão 11

→ Push(MEC,operacional)  
Push(MEC,gerencial)  
Push(MEC,organizacional)  
Push(MEC,tatico)  
Top(MEC,)  
Push(MEC,Pop(MEC))  
Push(MEC,estrategico)  
Push(MEC,Top(MEC))  
Pop(MEC)  
Pop(MEC)

topo →





# Questão 11

```
Push(MEC,operacional)
Push(MEC,gerencial)
→ Push(MEC,organizacional)
Push(MEC,tatico)
Top(MEC,)
Push(MEC,Pop(MEC))
Push(MEC,estrategico)
Push(MEC,Top(MEC))
Pop(MEC)
Pop(MEC)
```

topo →





# Questão 11

```
Push(MEC,operacional)
Push(MEC,gerencial)
Push(MEC,organizacional)
→ Push(MEC,tatico)
Top(MEC,)
Push(MEC,Pop(MEC))
Push(MEC,estrategico)
Push(MEC,Top(MEC))
Pop(MEC)
Pop(MEC)
```

topo →





# Questão 11





# Questão 11

```
Push(MEC,operacional)
Push(MEC,gerencial)
Push(MEC,organizacional)
Push(MEC,tatico)
Top(MEC,)
→ Push(MEC,Pop(MEC))
Push(MEC,estrategico)
Push(MEC,Top(MEC))
Pop(MEC)
Pop(MEC)
```

topo →





# Questão 11

```
Push(MEC,operacional)
Push(MEC,gerencial)
Push(MEC,organizacional)
Push(MEC,tatico)
Top(MEC,)
Push(MEC,Pop(MEC))
→ Push(MEC,estrategico)
Push(MEC,Top(MEC))
Pop(MEC)
Pop(MEC)
```

topo →





# Questão 11





# Questão 11

```
Push(MEC,operacional)
Push(MEC,gerencial)
Push(MEC,organizacional)
Push(MEC,tatico)
Top(MEC,)
Push(MEC,Pop(MEC))
Push(MEC,estrategico)
Push(MEC,Top(MEC))
→ Pop(MEC)
Pop(MEC)
```

topo →





# Questão 11





# Questão 11

## [2009 - FGV - MEC - Analista de Sistemas]

A figura abaixo mostra uma aplicação da estrutura de dados pilha denominada MEC, inicialmente vazia, suportando três operações básicas, conforme definidas no Quadro I. Observe que o Quadro II apresenta uma seqüência de operações sobre a estrutura.

Quadro I		SIGNIFICADO
OPERAÇÃO	Push(PILHA,x)	Insere um elemento <u>x</u> na pilha
	Pop(PILHA)	Remove o elemento <u>x</u> do topo da pilha <u>PILHA</u>
	Top(PILHA)	Acessa, sem remover, o elemento de topo da pilha <u>PILHA</u>

Quadro II
SEQÜÊNCIA DE OPERAÇÕES → Push(MEC,operacional) Push(MEC,gerencial) Push(MEC,organizacional) Push(MEC,tatico) Top(MEC,) Push(MEC,Pop(MEC)) Push(MEC,estrategico) Push(MEC,Top(MEC)) Pop(MEC) Pop(MEC)

Ao final das operações, o elemento que se encontra no topo da pilha é:

- A) organizacional
- B) operacional
- C) estrategico
- D) gerencial

➔ E) tatico





**06 – E**

**07 – D**

**08 – Errado**

**09 – Certo**

**10 – D**

**11 – E**





# Filas



## ➤ Conceitos gerais

- É uma estrutura de dados onde as inserções são realizadas em um extremo e as remoções em outro.
- O primeiro objeto a ser inserido na fila é o primeiro a ser removido da mesma. Por trabalhar com essa política, essa estrutura é conhecida como FIFO (first-in, first-out).
- Essa estrutura usa duas variáveis de controle para referenciar o *inicio* e o *fim* da fila.



- Principais operações de uma fila
  - Definição das principais operações em java.

```
1 interface IFila {  
2     public boolean isFilaVazia();  
3     public void enfileirar( Object elemento) ;  
4     public Object desenfileirar();  
5 }
```



- **Operação isFilaVazia**

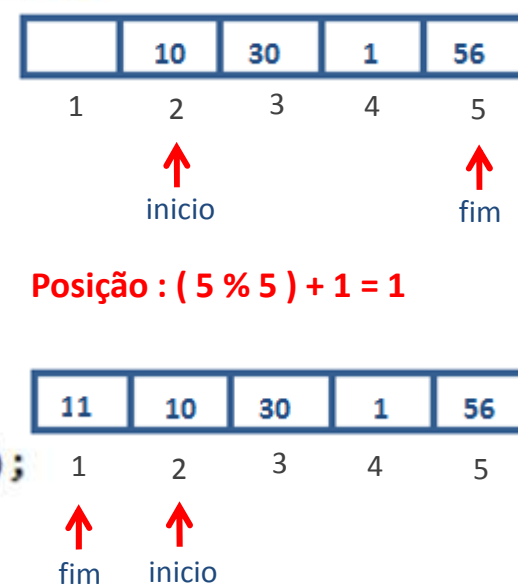
- Essa operação verifica se a fila esta vazia.
- Implementação em java:

```
1 public boolean isFilaVazia(){  
2     return (inicio == fim) && (fim == 0);  
3 }
```



- **Operação enfileirar**
  - Essa operação insere um elemento na fila.
  - Implementação em java:

```
1 public void enfileirar( Object elemento) {
2     int posicao = (fim % COMPRIMENTO) + 1;
3
4     if (posicao != inicio) {
5         fim = posicao;
6         elementos[fim] = elemento;
7         if (inicio == 0) {
8             inicio = 1;
9         }
10    } else {
11        // overFlow
12        System.out.println("overFlow");
13    }
14 }
```



	10	30	1	56
1	2	3	4	5

↑ inicio ↑ fim

Posição : ( 5 % 5 ) + 1 = 1

11	10	30	1	56
1	2	3	4	5

↑ fim ↑ inicio



## ➤ Operação enfileirar

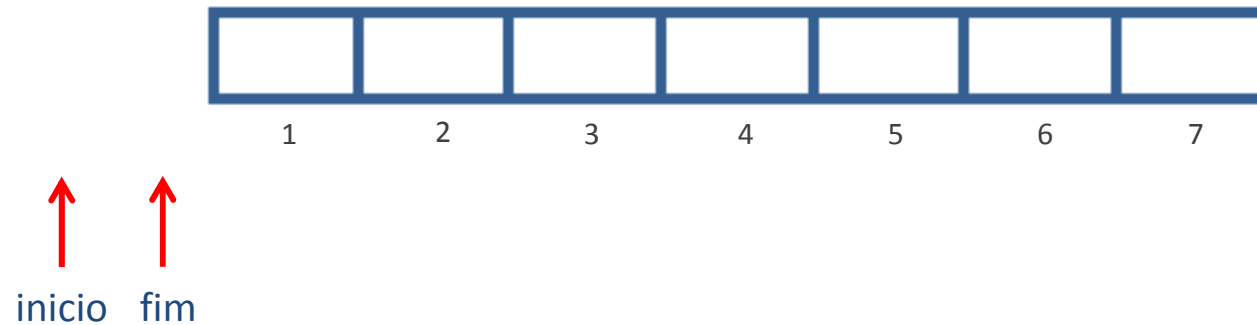
- Essa operação insere um elemento na fila.
- Implementação em java:

```
1  public void enfileirar( Object elemento) {  
2      int posicao = (fim % COMPRIMENTO) + 1;  
3  
4      if (posicao != inicio) {  
5          fim = posicao;  
6          elementos[fim] = elemento;  
7          if (inicio == 0) {  
8              inicio = 1;  
9          }  
10     } else {  
11         // overFlow  
12         System.out.println("overFlow");  
13     }  
14 }
```



## ➤ Operação enfileirar

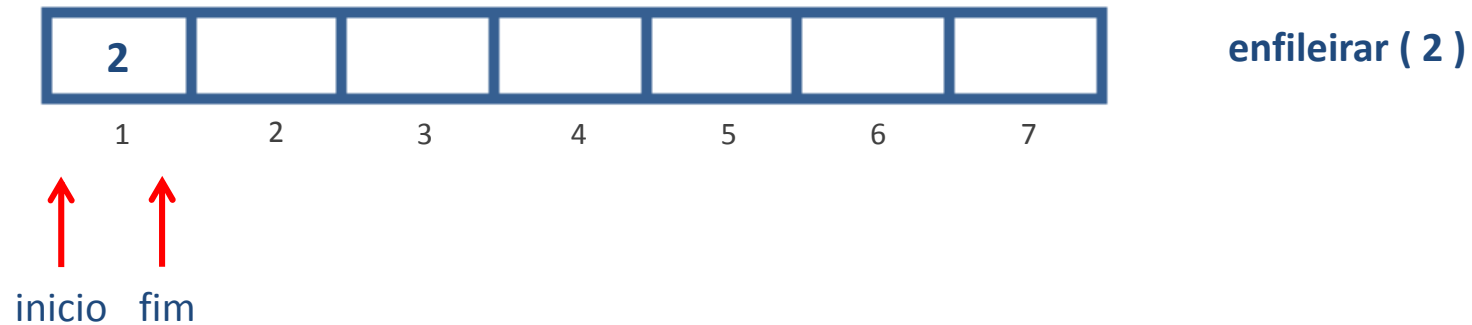
- Como ficaria uma fila, inicialmente vazia, após a inclusão dos elementos: 2, 10, 30, 1, 56?





## ➤ Operação enfileirar

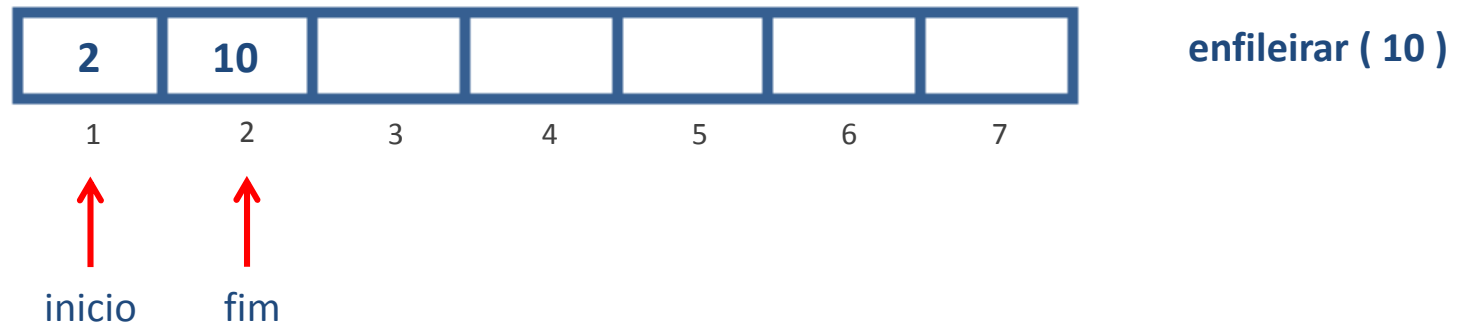
- Como ficaria uma fila, inicialmente vazia, após a inclusão dos elementos: **2**, 10, 30, 1, 56?





## ➤ Operação enfileirar

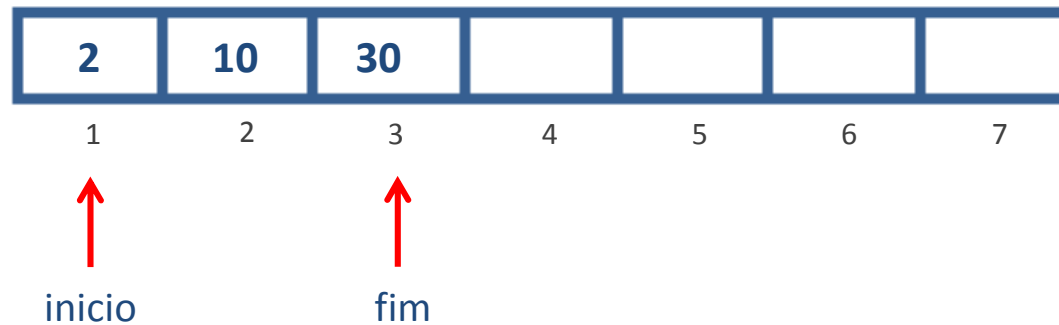
- Como ficaria uma fila, inicialmente vazia, após a inclusão dos elementos: 2, **10**, 30, 1, 56?





## ➤ Operação enfileirar

- Como ficaria uma fila, inicialmente vazia, após a inclusão dos elementos: 2, 10, **30**, 1, 56?

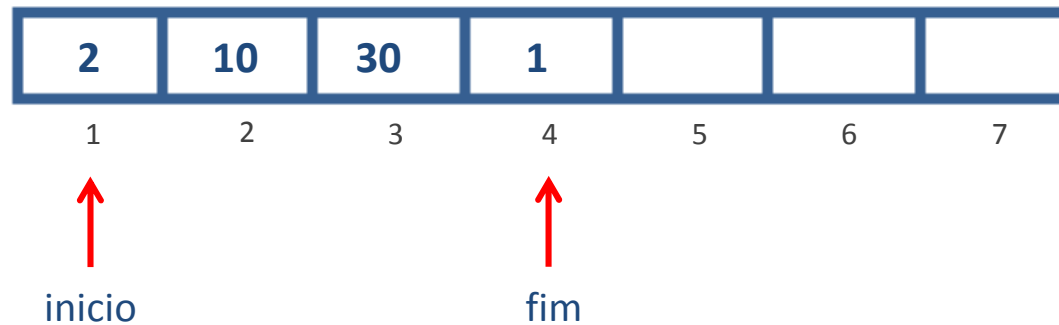


enfileirar ( 30 )



## ➤ Operação enfileirar

- Como ficaria uma fila, inicialmente vazia, após a inclusão dos elementos: 2, 10, 30, **1**, 56?

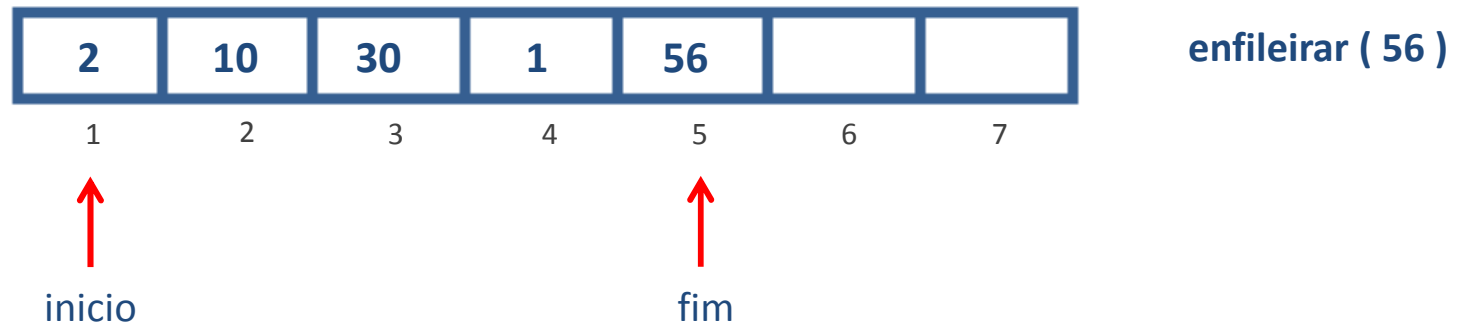


enfileirar ( 1 )



## ➤ Operação enfileirar

- Como ficaria uma fila, inicialmente vazia, após a inclusão dos elementos: 2, 10, 30, 1, **56**?



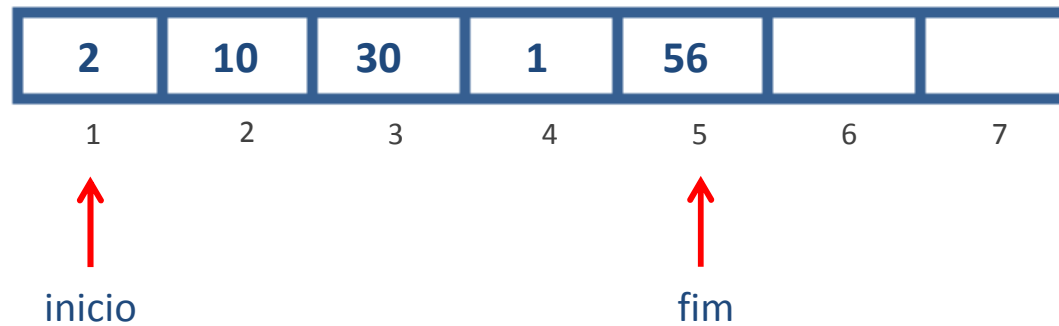


- **Operação desenfileirar**
  - Essa operação remove um elemento da fila.
  - Implementação em java:

```
1 public Object desenfileirar(){
2     Object elementoRecuperado = null;
3
4     if ( !isFilaVazia() {
5         elementoRecuperado = elementos[inicio];
6         elementos[inicio] = null;
7
8         if (inicio == fim) {
9             inicio = fim = 0;
10        } else {
11            inicio = (inicio % COMPRIMENTO) + 1;
12        }
13    } else {
14        // underFlow
15        System.out.println("underFlow");
16    }
17
18    return elementoRecuperado;
19 }
```

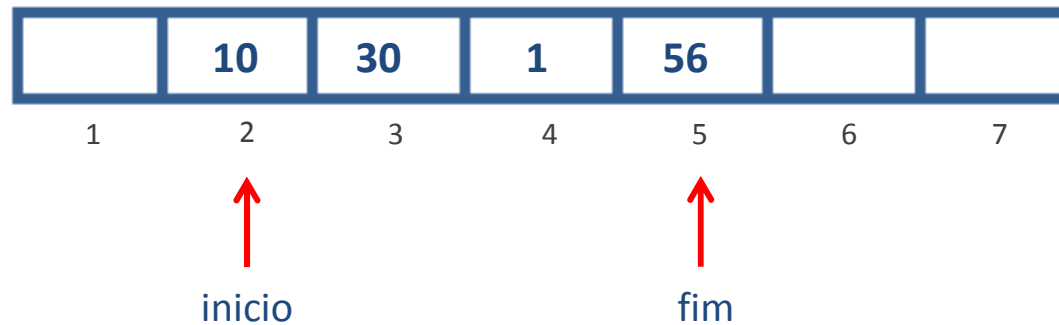


- **Operação desenfileirar**
  - Como ficaria a fila a seguir após desenfileirar 3 elementos?





- **Operação desenfileirar**
  - Como ficaria a fila a seguir após desenfileirar 3 elementos?



[1] desenfileirar( )  
retorna 2



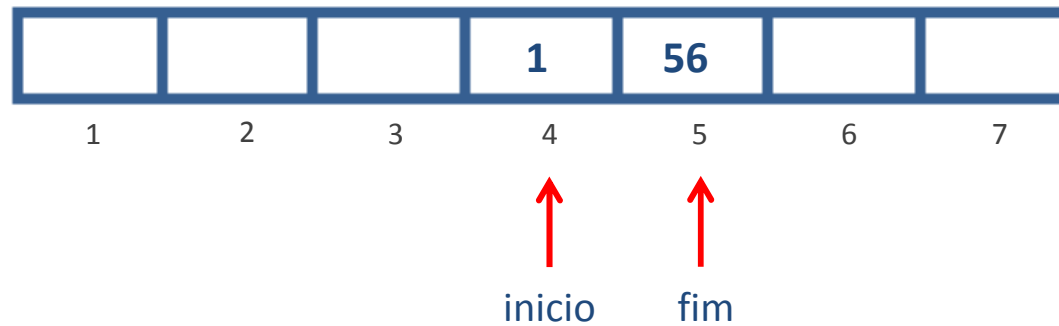
- **Operação desenfileirar**
  - Como ficaria a fila a seguir após desenfileirar 3 elementos?



[2] desenfileirar( )  
retorna 10



- **Operação desenfileirar**
  - Como ficaria a fila a seguir após desenfileirar 3 elementos?



**[3] desenfileirar( )**  
retorna 30



# Questão 12

## [2012 - CESPE - TRE-RJ - Programador de computador]

Julgue os itens a seguir, referentes a estrutura de dados e organização de arquivos.

As filas são estruturas com base no princípio LIFO (last in, first out), no qual os dados que forem inseridos primeiro na fila serão os últimos a serem removidos. Existem duas funções que se aplicam a todas as filas: PUSH, que insere um dado no topo da fila, e POP, que remove o item no topo da fila.

Certo      Errado



# Questão 12

## [2012 - CESPE - TRE-RJ - Programador de computador]

Julgue os itens a seguir, referentes a estrutura de dados e organização de arquivos.

As filas são estruturas com base no princípio LIFO (last in, first out), no qual os dados que forem inseridos primeiro na fila serão os últimos a serem removidos. Existem duas funções que se aplicam a todas as filas: PUSH, que insere um dado no topo da fila, e POP, que remove o item no topo da fila.

Certo ➡ Errado



## Questão 13

### [2012 - INSTITUTO CIDADES - TCM-GO - Auditor de Controle Externo TI]

A melhor definição para a estrutura de dados chamada FILA é(são):

- A) É uma estrutura de dados linear, que também pode ser linear e dinâmica. É composta por nós que apontam para o próximo elemento.
- B) São estruturas baseadas no princípio FIFO (first in, first out), em que os elementos que foram inseridos no início são os primeiros a serem removidos.
- C) São estruturas baseadas no princípio LIFO (last in, first out), na qual os dados que foram inseridos por último na pilha serão os primeiros a serem removidos.
- D) É uma estrutura de dados em que cada elemento tem um ou mais elementos associados.
- E) São estruturas de dados lineares e estáticas, isto é, são compostas por um número fixo (finito) de elementos de um determinado tipo de dados. O tempo de acesso aos elementos é muito rápido porém, a remoção de elementos pode ser custosa se não for desejável que haja espaços "vazios" no meio da estrutura.



# Questão 13

## [2012 - INSTITUTO CIDADES - TCM-GO - Auditor de Controle Externo TI]

A melhor definição para a estrutura de dados chamada FILA é(são):

- A) É uma estrutura de dados linear, que também pode ser linear e dinâmica. É composta por nós que apontam para o próximo elemento.
- ➡ B) São estruturas baseadas no princípio FIFO (first in, first out), em que os elementos que foram inseridos no início são os primeiros a serem removidos.
- C) São estruturas baseadas no princípio LIFO (last in, first out), na qual os dados que foram inseridos por último na pilha serão os primeiros a serem removidos.
- D) É uma estrutura de dados em que cada elemento tem um ou mais elementos associados.
- E) São estruturas de dados lineares e estáticas, isto é, são compostas por um número fixo (finito) de elementos de um determinado tipo de dados. O tempo de acesso aos elementos é muito rápido porém, a remoção de elementos pode ser custosa se não for desejável que haja espaços "vazios" no meio da estrutura.



# Questão 14

## [2010 - CESPE - DETRAN-ES - Analista de Sistemas]

No armazenamento de dados pelo método FIFO (first in - first out), a estrutura de dados é representada por uma fila, em cuja posição final ocorrem inserções e, na inicial, retiradas.

Certo      Errado



# Questão 14

## [2010 - CESPE - DETRAN-ES - Analista de Sistemas]

No armazenamento de dados pelo método FIFO (first in - first out), a estrutura de dados é representada por uma fila, em cuja posição final ocorrem inserções e, na inicial, retiradas.

➡ Certo      Errado



# Questão 15

## [2013 - FCC - TRT - 9ª REGIÃO (PR) - Técnico Judiciário TI]

Insira os dados de entrada numa fila. Em seguida retire cada dado da fila e insira numa pilha. Mostre a pilha. Depois retire os dados da pilha e insira na fila. Mostre a fila. Dados de entrada: 11, 12, 23, 14, 25, 50, 8, 18, 29, 10

As estruturas mostradas ficam:

I. Pilha: (topo) 10 - 29 - 18 - 8 - 50 - 25 - 14 - 23 - 12 - 11

II. Fila: (começo) 11 - 12 - 23 - 14 - 25 - 50 - 8 - 18 - 29 - 10 (fim)

III. Fila: (começo) 10 - 29 - 18 - 8 - 50 - 25 - 14 - 23 - 12 - 11 (fim)

IV. Pilha: (topo) 11 - 12 - 23 - 14 - 25 - 50 - 8 - 18 - 29 - 10

V. A fila mostrada fica com os elementos em ordem invertida dos dados de entrada

Está correto o que se afirma APENAS em

A) III e IV.

B) II e IV.

C) I, II e III.

D) I, III e V.

E) I, IV e V.



# Questão 15

## [2013 - FCC - TRT - 9ª REGIÃO (PR) - Técnico Judiciário TI]

Insira os dados de entrada numa fila. Em seguida retire cada dado da fila e insira numa pilha. Mostre a pilha. Depois retire os dados da pilha e insira na fila. Mostre a fila. Dados de entrada: 11, 12, 23, 14, 25, 50, 8, 18, 29, 10

**Fila:** 11 | 12 | 23 | 14 | 25 | 50 | 8 | 18 | 29 | 10

↑  
início

↑  
fim

**Pilha:** 11 | 12 | 23 | 14 | 25 | 50 | 8 | 18 | 29 | 10

↑  
topo

**Fila:** 10 | 29 | 18 | 8 | 50 | 25 | 14 | 23 | 12 | 11

↑  
início

↑  
fim

- I. Pilha: (topo) 10 - 29 - 18 - 8 - 50 - 25 - 14 - 23 - 12 - 11
- II. Fila: (começo) 11 - 12 - 23 - 14 - 25 - 50 - 8 - 18 - 29 - 10 (fim)
- III. Fila: (começo) 10 - 29 - 18 - 8 - 50 - 25 - 14 - 23 - 12 - 11 (fim)
- IV. Pilha: (topo) 11 - 12 - 23 - 14 - 25 - 50 - 8 - 18 - 29 - 10
- V. A fila mostrada fica com os elementos em ordem invertida dos dados de entrada



# Questão 15

## [2013 - FCC - TRT - 9ª REGIÃO (PR) - Técnico Judiciário TI]

Insira os dados de entrada numa fila. Em seguida retire cada dado da fila e insira numa pilha. Mostre a pilha. Depois retire os dados da pilha e insira na fila. Mostre a fila. Dados de entrada: 11, 12, 23, 14, 25, 50, 8, 18, 29, 10

As estruturas mostradas ficam:

I. Pilha: (topo) 10 - 29 - 18 - 8 - 50 - 25 - 14 - 23 - 12 - 11

II. Fila: (começo) 11 - 12 - 23 - 14 - 25 - 50 - 8 - 18 - 29 - 10 (fim)

III. Fila: (começo) 10 - 29 - 18 - 8 - 50 - 25 - 14 - 23 - 12 - 11 (fim)

IV. Pilha: (topo) 11 - 12 - 23 - 14 - 25 - 50 - 8 - 18 - 29 - 10

V. A fila mostrada fica com os elementos em ordem invertida dos dados de entrada

Está correto o que se afirma APENAS em

A) III e IV.

B) II e IV.

C) I, II e III.

➡ D) I, III e V.

E) I, IV e V.





**12 – Errado**

**14 – Certo**

**13 – B**

**15 – D**





# Deque



- **Conceitos gerais**

- É uma estrutura de dados similar a uma fila, no entanto, suporta inserção e remoção em ambas extremidades da estrutura.
- Essa estrutura usa duas variáveis de controle, uma para referenciar o início e outra para referenciar o fim da estrutura.



- Principais operações de um deque
  - Definição das principais operações.

```
1 interface IDeque {  
2     void inserirPrimeiro( Object elemento );  
3     void inserirUltimo( Object elemento );  
4     Object removerPrimeiro();  
5     Object removerUltimo();  
6 }
```



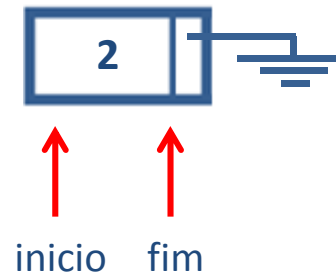
- **Operação inserirPrimeiro**

- Insere um novo elemento no inicio do deque.
- Como ficaria um deque, inicialmente vazio, após a inclusão dos elementos: 2, 10 e 30?



## ➤ Operação inserirPrimeiro

- Insere um novo elemento no início do deque.
- Como ficaria um deque, inicialmente vazio, após a inclusão dos elementos: **2**, 10 e 30?

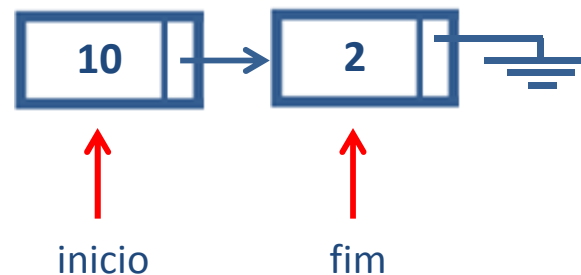


`inserirPrimeiro( 2 )`



## ➤ Operação inserirPrimeiro

- Insere um novo elemento no início do deque.
- Como ficaria um deque, inicialmente vazio, após a inclusão dos elementos: 2, **10** e 30?

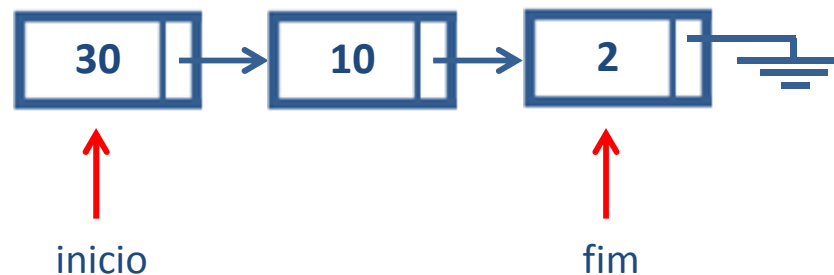


inserirPrimeiro( 10 )



## ➤ Operação inserirPrimeiro

- Insere um novo elemento no início do deque.
- Como ficaria um deque, inicialmente vazio, após a inclusão dos elementos: 2, 10 e **30**?

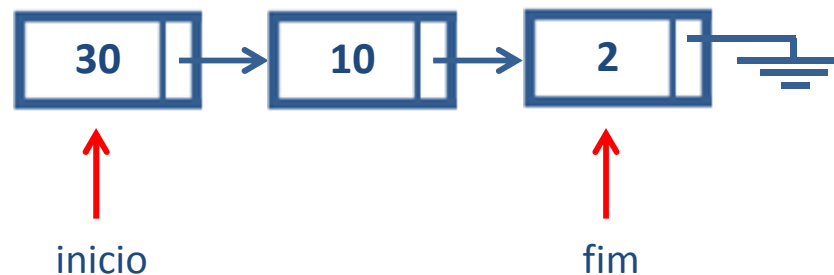


inserirPrimeiro( 30 )



## ➤ Operação inserirUltimo

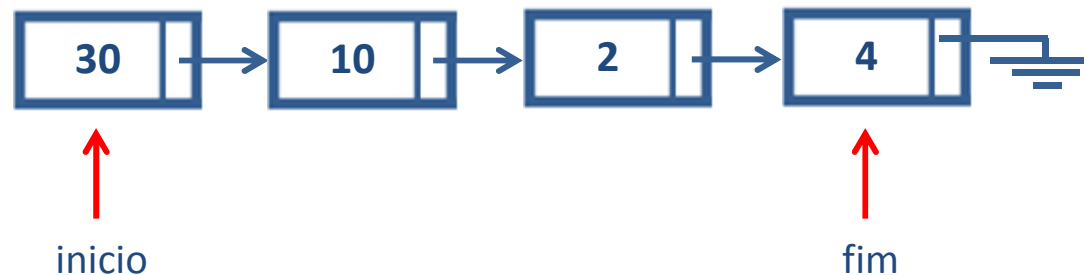
- Insere um novo elemento no fim do deque.
- Como ficaria o deque abaixo após a inserção dos elementos: 4 e 6?





## ➤ Operação `inserirUltimo`

- Insere um novo elemento no fim do deque.
- Como ficaria o deque abaixo após a inserção dos elementos: **4** e **6**?

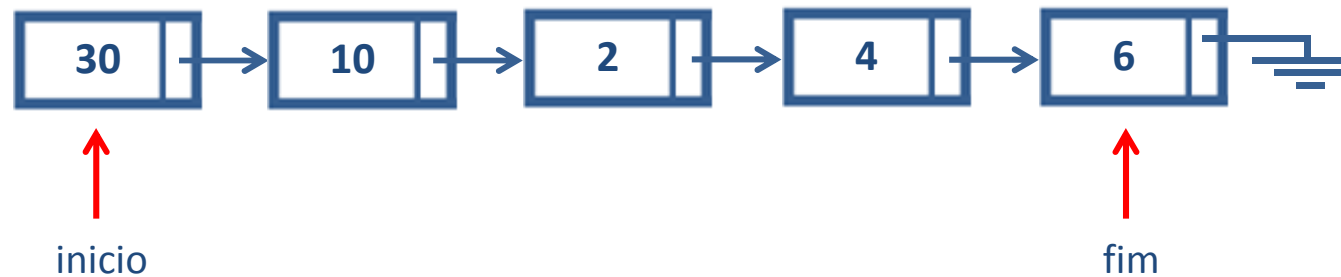


`inserirUltimo( 4 )`



## ➤ Operação **inserirUltimo**

- Insere um novo elemento no fim do deque.
- Como ficaria o deque abaixo após a inserção dos elementos: 4 e 6?

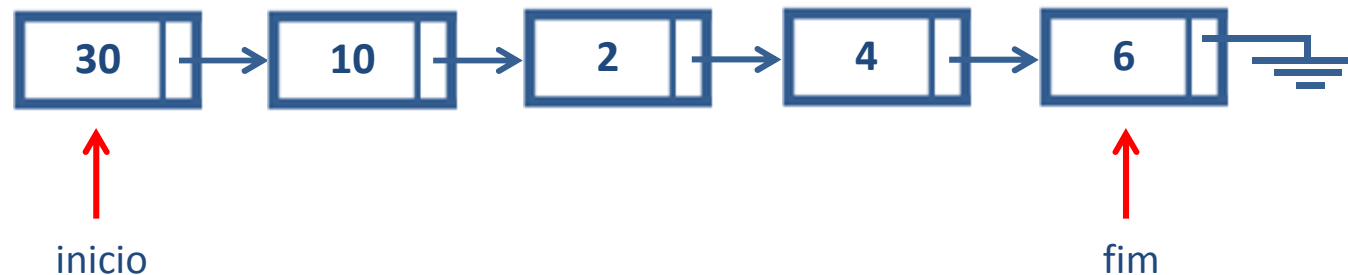


**inserirUltimo( 6 )**



## ➤ Operação removerPrimeiro

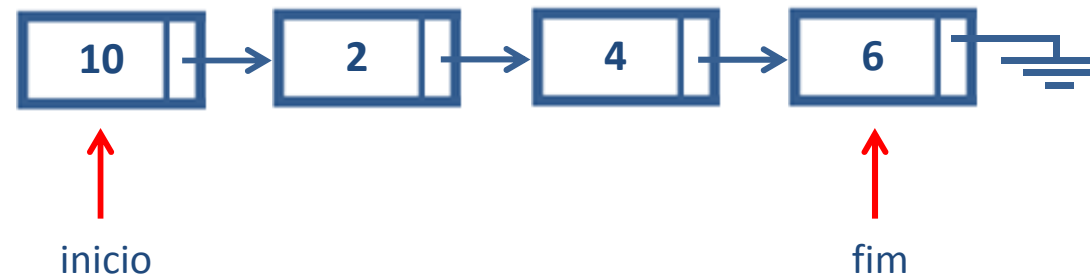
- Remove e retorna o primeiro elemento do deque.
- Como ficaria o deque abaixo após a execução da operação removerPrimeiro duas vezes?





## ➤ Operação removerPrimeiro

- Remove e retorna o primeiro elemento do deque.
- Como ficaria o deque abaixo após a execução da operação removerPrimeiro duas vezes?

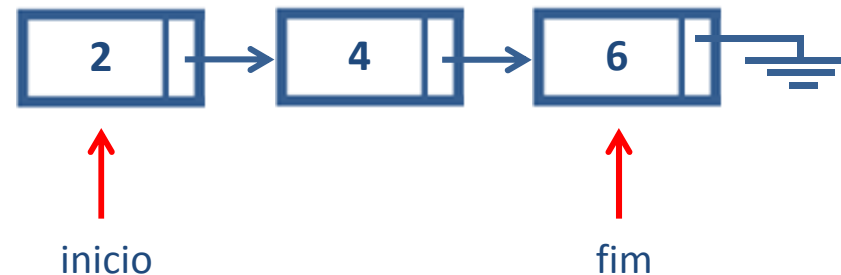


[1] removerPrimeiro( )  
retorna 30



## ➤ Operação removerPrimeiro

- Remove e retorna o primeiro elemento do deque.
- Como ficaria o deque abaixo após a execução da operação removerPrimeiro duas vezes?

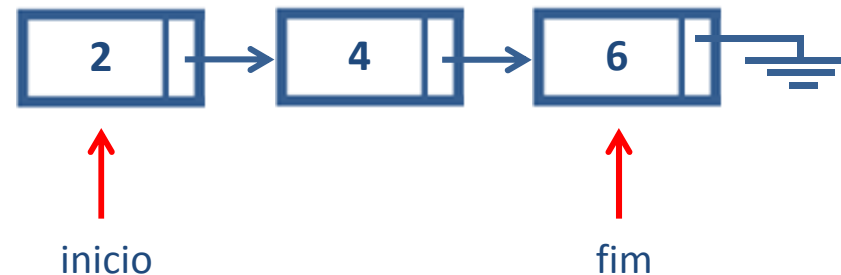


**[2] removerPrimeiro( )**  
retorna 10



## ➤ Operação removerUltimo

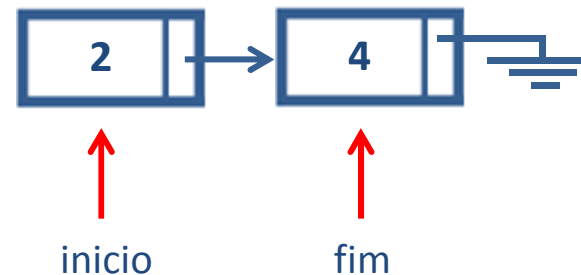
- Remove e retorna o último elemento do deque.
- Como ficaria o deque abaixo após a execução da operação removerUltimo duas vezes?





## ➤ Operação removerUltimo

- Remove e retorna o último elemento do deque.
- Como ficaria o deque abaixo após a execução da operação removerUltimo duas vezes?

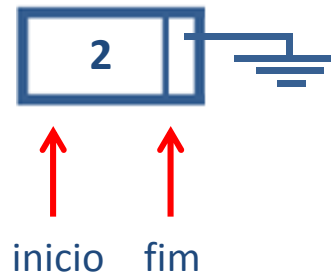


**[1] removerUltimo()**  
retorna 6



## ➤ Operação removerUltimo

- Remove e retorna o último elemento do deque.
- Como ficaria o deque abaixo após a execução da operação removerUltimo duas vezes?



**[2] removerUltimo()**  
retorna 4



# Questão 16

## [2010 - FCC - TRT - 22ª Região (PI) - Analista Judiciário TI]

Uma fila duplamente terminada, isto é, uma estrutura linear que permite inserir e remover de ambos os extremos é chamada

- A) Árvore.
- B) Shift-and.
- C) Autômato.
- D) Deque.
- E) Boyer-Moore.



# Questão 16

## [2010 - FCC - TRT - 22ª Região (PI) - Analista Judiciário TI]

Uma fila duplamente terminada, isto é, uma estrutura linear que permite inserir e remover de ambos os extremos é chamada

- A) Árvore.
- B) Shift-and.
- C) Autômato.
- ➡ D) Deque.
- E) Boyer-Moore.



# Questão 17

## [2012 - FCC - TRE-SP - Análise de Sistemas]

No que se refere a estruturas de dados é INCORRETO afirmar:

- A) Numa fila dupla, os elementos podem ser inseridos e removidos de qualquer um dos extremos da fila.
- B) Em qualquer situação é possível usar uma única fila dupla para representar duas filas simples.
- C) A implementação de uma fila dupla normalmente é mais eficiente com uma lista duplamente encadeada que com uma encadeada simples.
- D) Pela definição de fila, se os elementos são inseridos por um extremo da lista linear, eles só podem ser removidos pelo outro.
- E) Numa lista singularmente encadeada, para acessar o último nodo é necessário partir do primeiro e ir seguindo os campos de ligação até chegar ao final da lista.



# Questão 17

## [2012 - FCC - TRE-SP - Análise de Sistemas]

No que se refere a estruturas de dados é INCORRETO afirmar:

- A) Numa fila dupla, os elementos podem ser inseridos e removidos de qualquer um dos extremos da fila.
- ➡ B) Em qualquer situação é possível usar uma única fila dupla para representar duas filas simples.
- C) A implementação de uma fila dupla normalmente é mais eficiente com uma lista duplamente encadeada que com uma encadeada simples.
- D) Pela definição de fila, se os elementos são inseridos por um extremo da lista linear, eles só podem ser removidos pelo outro.
- E) Numa lista singularmente encadeada, para acessar o último nodo é necessário partir do primeiro e ir seguindo os campos de ligação até chegar ao final da lista.



# Questão 18

## [2010 - CESPE - DETRAN-ES - Analista de Sistemas]

Com relação à programação, algoritmos e estrutura de dados, julgue os itens seguintes.

Na implementação de um deque sequencial, é necessário ter, em cada extremidade, uma variável de ponteiro externa, por meio da qual as inserções e retiradas sejam efetuadas.

Certo      Errado



# Questão 18

## [2010 - CESPE - DETRAN-ES - Analista de Sistemas]

Com relação à programação, algoritmos e estrutura de dados, julgue os itens seguintes.

Na implementação de um deque sequencial, é necessário ter, em cada extremidade, uma variável de ponteiro externa, por meio da qual as inserções e retiradas sejam efetuadas.

➡ Certo      Errado



# Gabarito



**16 – D**

**17 – B**

**18 – Certo**





# Árvores



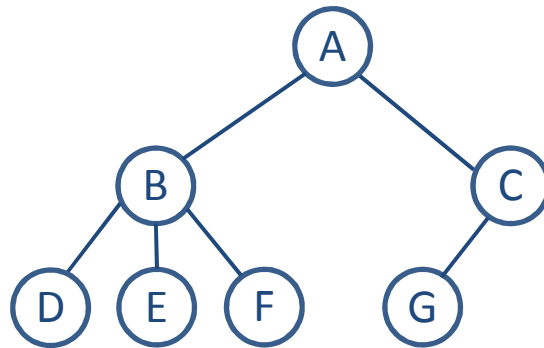
## ➤ Definição

- Uma árvore pode ser definida com um conjunto de nós que se relacionam de forma hierárquica mantendo as seguintes propriedades:
  - Caso a árvore não possua nenhum elemento, ela é considerada vazia.
  - Se a árvore não é vazia, ela tem um nó especial chamado raiz que não possui pai.
  - Todos os outros nós, com exceção da raiz, possuem um único pai e zero ou mais filhos.

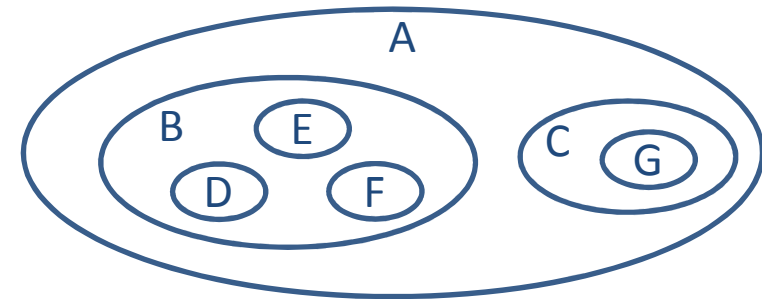


- **Representações**

- Representação hierárquica



- Diagrama de inclusão



- Parênteses aninhados

( A ( B (D) (E) (F) ) ( C (G) ) )



- **Conceitos gerais**
  - **Nó**
    - É um vértice ou elemento da árvore.
  - **Pai**
    - Todo nó, com exceção do nó raiz, tem exatamente um pai.
  - **Filho**
    - É um descendente direto de um nó.
  - **Subárvore**
    - Cada filho de um nó da origem a uma subárvore. Os princípios que se aplicam a uma árvore também se aplicam a qualquer uma de suas subárvores.



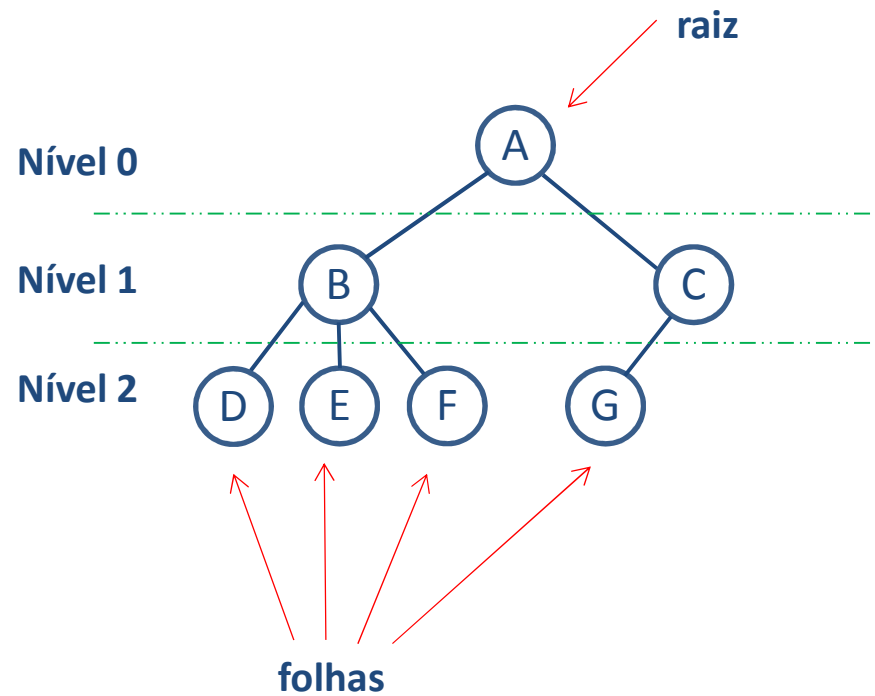
- **Conceitos gerais**
  - **Nó folha (nó terminal)**
    - Nome dado ao nó que não tem filho.
  - **Nó interior (nó não terminal)**
    - Nome dado ao nó que possui pelo menos um filho.
  - **Grau de um nó**
    - É indicado pelo número de filhos que o nó possui.
  - **Grau de uma árvore**
    - É indicado pelo nó que tem o maior grau na árvore.



- **Conceitos gerais**
- **Nível de um nó**
  - É determinado pelo comprimento do caminho da raiz ao nó específico. A raiz tem nível 0 (zero).
- **Caminho da árvore**
  - É o nome dado a uma sequencia de nós distintos tal que existe sempre entre nós consecutivos a relação “é filho de” ou “é pai de”.
- **Altura de um nó**
  - É o maior comprimento do nó até a folha.
- **Floresta**
  - Nome dado ao conjunto de zero ou mais árvores.



## ➤ Conceitos gerais



### Grau

- B: 3
- A: 2
- C: 1
- D, E, F, G: 0

### Altura

- A: 2
- B, C: 1
- D, E, F, G: 0

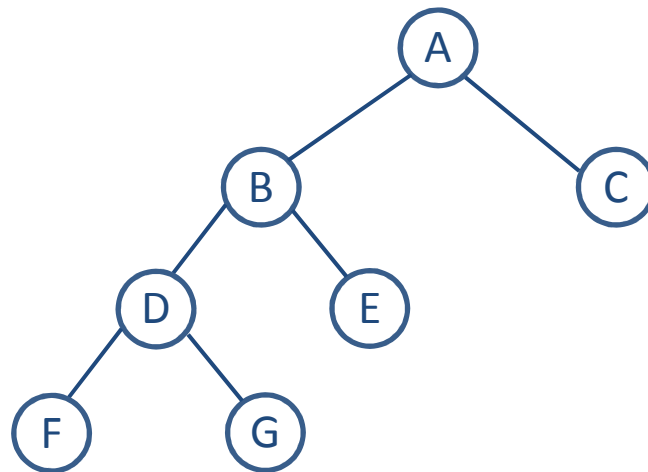


## ➤ Árvores Binárias

- É uma árvore que possui um nó especial chamado raiz e outros dois nós denominados subárvore esquerda e subárvore direita.

## ➤ Classificação de árvores binárias:

- Árvore estritamente binária - é uma árvore binária em que cada nó possui 0 ou 2 filhos.

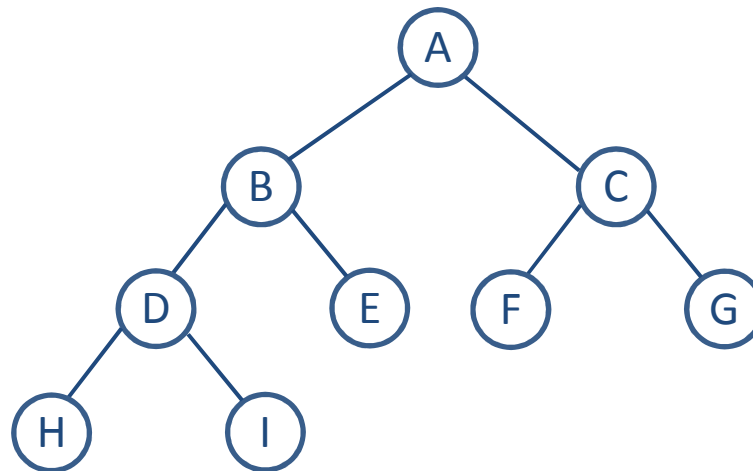




- **Árvores Binárias**

- **Classificação de árvores binárias:**

- Árvore binária completa – é uma árvore binária onde somente os nós do último ou do penúltimo nível da árvore podem ter uma subárvore vazia. Sempre possui altura mínima.

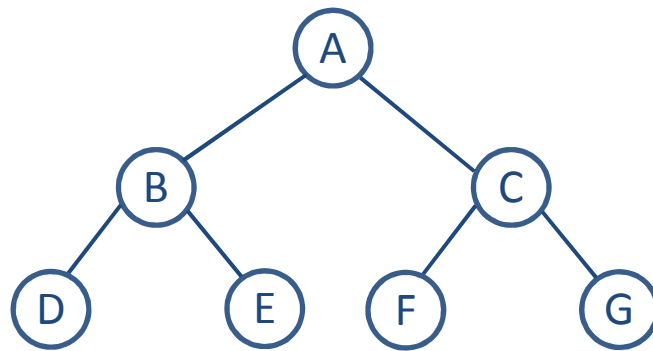




- **Árvores Binárias**

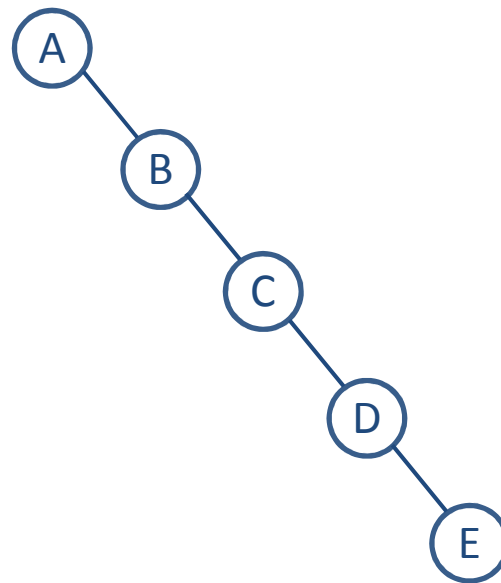
- **Classificação de árvores binárias:**

- Árvore binária cheia – é uma árvore binária onde somente o último nível pode ter uma subárvore vazia. Sempre possui altura mínima. Toda árvore binária cheia é completa e estritamente binária.





- **Árvores Binárias**
- **Classificação de árvores binárias:**
  - Árvore ziguezague – são árvores binárias que possuem altura máxima. Seus nós interiores possuem exatamente uma subárvore vazia.

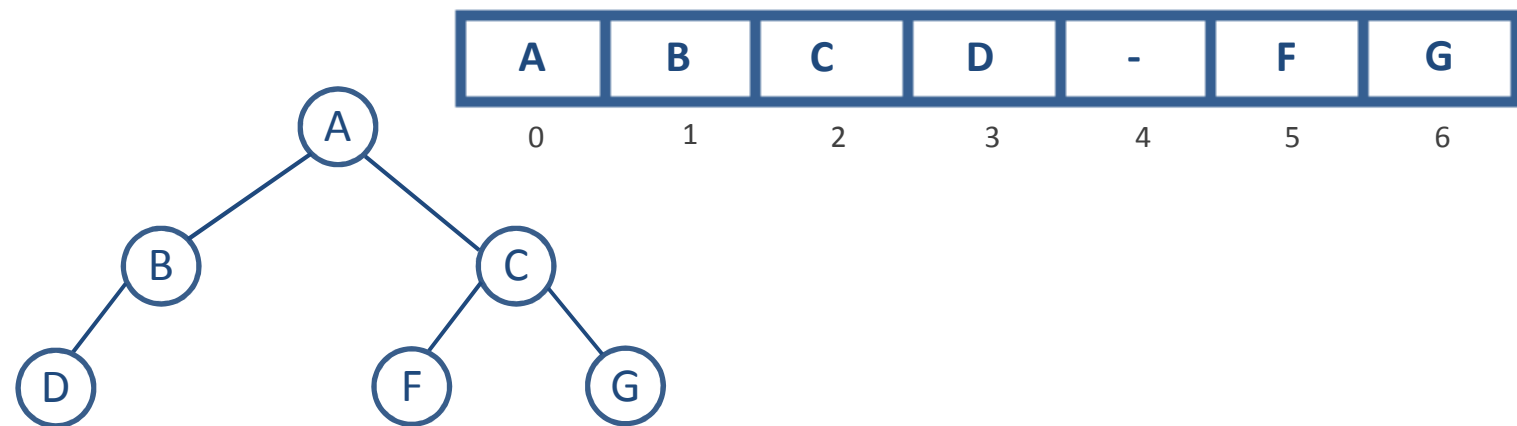




## ➤ Árvores Binárias - Implementação

### ➤ Alocação estática

- Implementação realizada com um vetor.
- O mapeamento da árvore para o vetor ocorre a partir do nível zero, seguindo pelo nível 1 e assim sucessivamente. Sempre iniciando pelo nó da esquerda.
- O vetor deve ter  $2^{(\text{qtdNíveis})} - 1$  elementos.

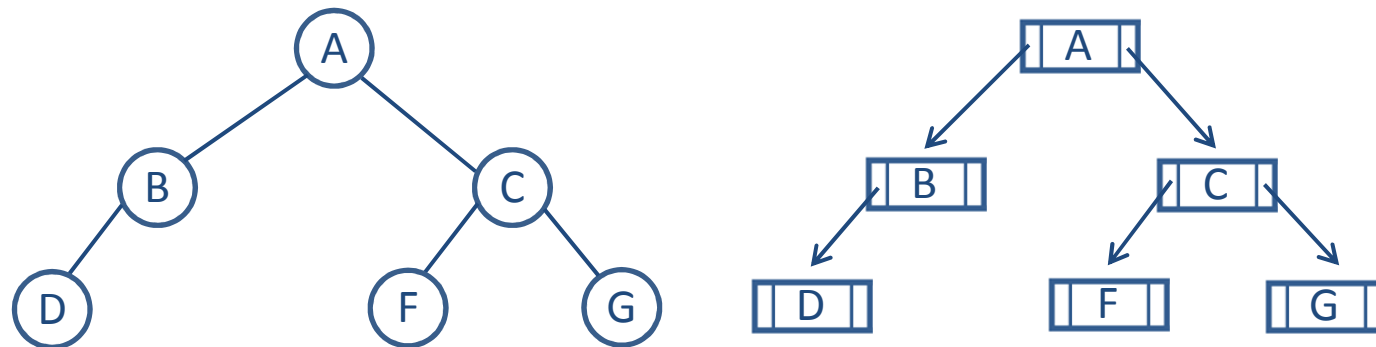




## ➤ Árvores Binárias - Implementação

### ➤ Alocação dinâmica

- Implementação realizada com uma lista encadeada.
- Cada elemento da lista será formado pelos dados do nó e a inclusão de mais duas referências: subárvore esquerda e subárvore direita.
- Nessa representação teremos uma quantidade de elementos na lista correspondente a quantidade de nós da árvore.





- **Percursos em árvores binárias**
  - Ao percorrer uma árvore fazemos uma visita sistemática em cada um de seus nós.
  - A seguir as principais formas de se percorrer os nós de uma árvore:
    - Em-ordem
    - Pré-ordem
    - Pós-ordem

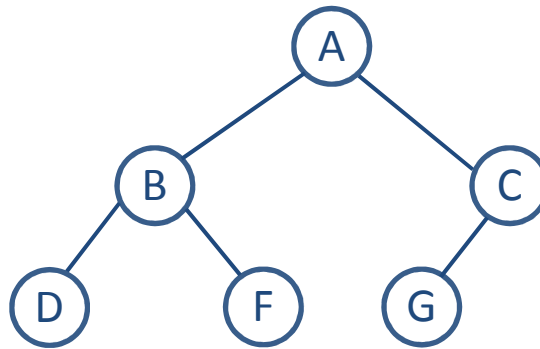


- **Percursos em árvores binárias – Em-ordem**
- O percurso em ordem segue recursivamente os seguintes passos para cada subárvore da árvore:
  - Percorrer a subárvore esquerda em-ordem
  - Visitar a raiz
  - Percorrer a subárvore direita em-ordem
- Implementação em java:

```
1 public void emOrdem(ArvoreBinaria no) {  
2     if (no != null) {  
3         emOrdem( no.subarvoreEsquerda );  
4         visitar( no );  
5         emOrdem( no.subarvoreDireita );  
6     }  
7 }
```



- **Percursos em árvores binárias – Em-ordem**
- Percorra a árvore a seguir com o percurso em-ordem:



Em ordem: D B F A G C

```
1 public void emOrdem(ArvoreBinaria no) {  
2     if (no != null) {  
3         emOrdem( no.subarvoreEsquerda );  
4         visitar( no );  
5         emOrdem( no.subarvoreDireita );  
6     }  
7 }
```

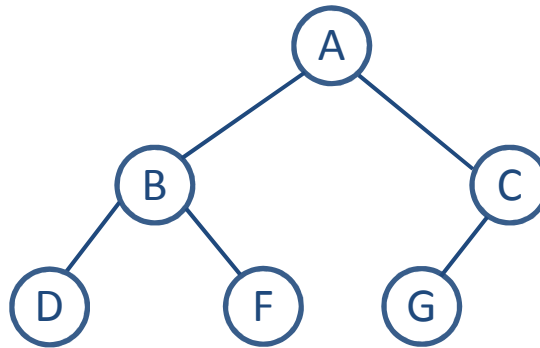


- **Percursos em árvores binárias – Pré-ordem**
- O percurso pré-ordem segue recursivamente os seguintes passos para cada subárvore da árvore:
  - Visitar a raiz
  - Percorrer a subárvore esquerda em pré-ordem
  - Percorrer a subárvore direita em pré-ordem
- Implementação em java:

```
1 public void preOrdem(ArvoreBinaria no) {  
2     if (no != null) {  
3         visitar( no );  
4         preOrdem( no.subarvoreEsquerda );  
5         preOrdem( no.subarvoreDireita );  
6     }  
7 }
```



- Percursos em árvores binárias – Pré-ordem
  - Percorra a árvore a seguir com o percurso pré-ordem:



**Pré-ordem:** A B D F C G

```
1 public void preOrdem(ArvoreBinaria no) {  
2     if (no != null) {  
3         visitar( no );  
4         preOrdem( no.subarvoreEsquerda );  
5         preOrdem( no.subarvoreDireita );  
6     }  
7 }
```

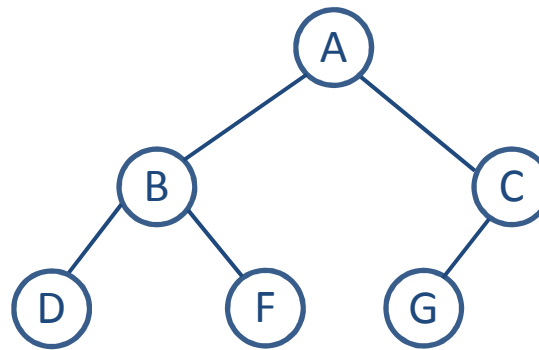


- **Percursos em árvores binárias – Pós-ordem**
- O percurso pós-ordem segue recursivamente os seguintes passos para cada subárvore da árvore:
  - Percorrer a subárvore esquerda em pós-ordem
  - Percorrer a subárvore direita em pós-ordem
  - Visitar a raiz
- Implementação em java:

```
1 public void posOrdem(ArvoreBinaria no) {  
2     if (no != null) {  
3         posOrdem( no.subarvoreEsquerda );  
4         posOrdem( no.subarvoreDireita );  
5         visitar( no );  
6     }  
7 }
```



- **Percursos em árvores binárias – Pós-ordem**
- Percorra a árvore a seguir com o percurso pós-ordem:



**Pós-ordem:** D F B G C A

```
1 public void posOrdem(ArvoreBinaria no) {  
2     if (no != null) {  
3         posOrdem( no.subarvoreEsquerda );  
4         posOrdem( no.subarvoreDireita );  
5         visitar( no );  
6     }  
7 }
```



## ➤ Árvores Binárias de Busca

- São estruturas adequadas à solução de problemas de busca. Elas possuem as seguintes características:
  - Dado um nó  $n$  da árvore, qualquer outro nó  $n1$  de sua subárvore esquerda não possui chave superior a chave de  $n$ .
  - Dado um nó  $n$  da árvore, qualquer outro nó  $n2$  de sua subárvore direita não tem chave inferior a chave de  $n$ .



- Principais operações de uma árvore binária de busca
  - Definição das principais operações.

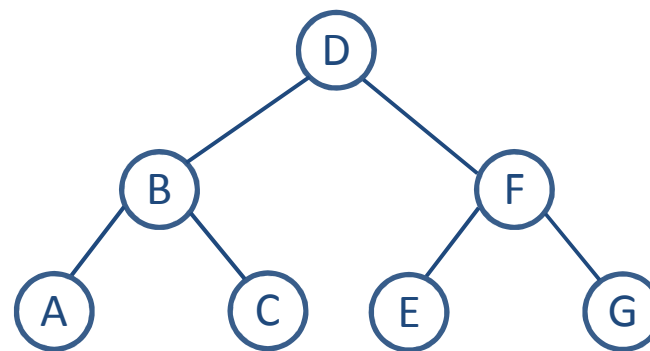
```
1 interface IArvoreBuscaBinaria {  
2     No busca( char id );  
3     void inserir( No elemento );  
4     No remove( char id );  
5 }  
-
```



- **Árvore Binária de Busca – Operação busca**
  - Ao buscar um elemento  $x$  na árvore, iniciamos a busca pelo nó raiz.
  - Caso não haja correspondência, continuamos a busca pelo:
    - Filho esquerdo, caso  $x < \text{chave}(\text{raiz})$
    - Filho direito, caso  $x > \text{chave}(\text{raiz})$
  - Esse procedimento ocorre recursivamente até o nó ser encontrado ou até a constatação que a chave procurada não existe.
  - A complexidade dessa operação é proporcional a altura da árvore.



- **Árvore Binária de Busca – Operação busca**
  - Como encontrar o elemento G na árvore a seguir?



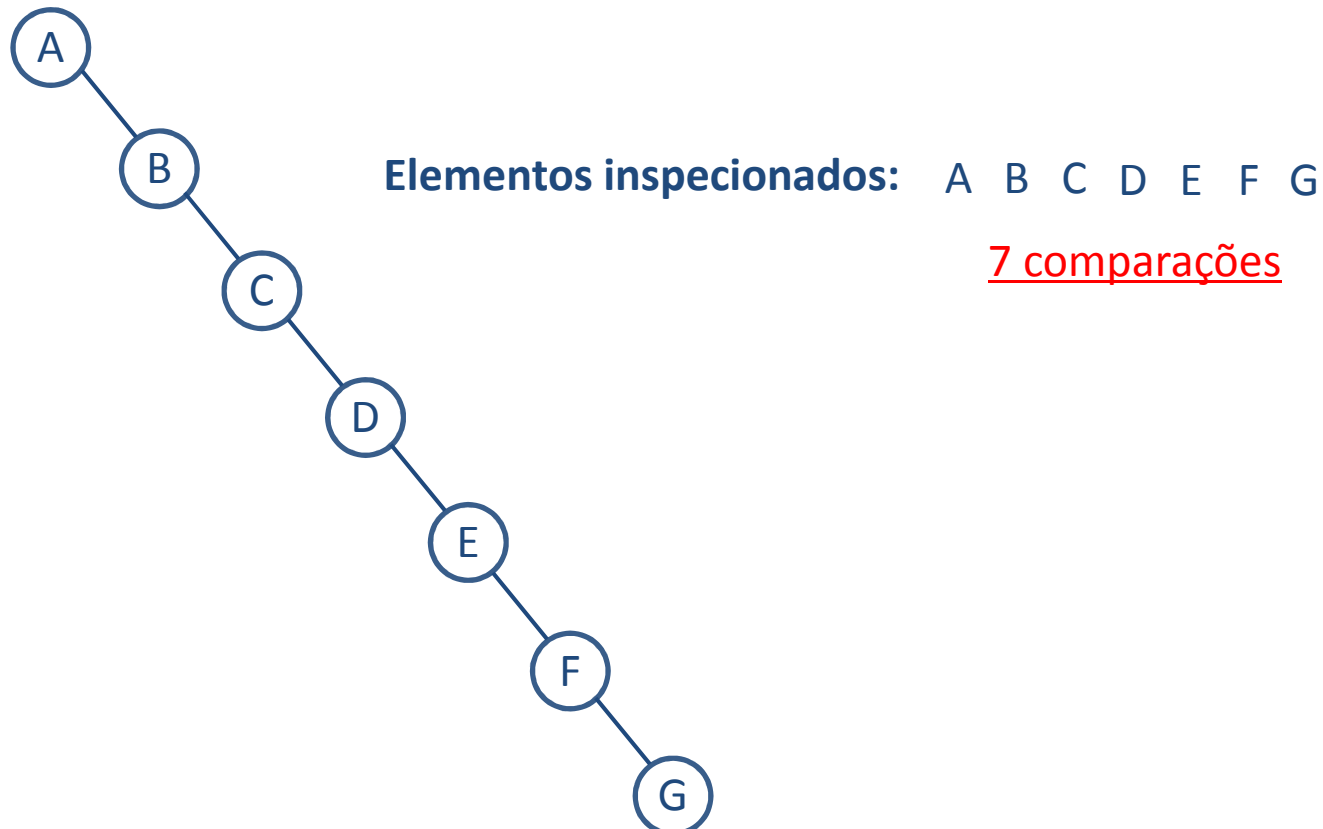
**Elementos inspecionados:** D F G

3 comparações

- A busca em uma árvore binária com altura mínima tem complexidade  $O(\log n)$ .



- **Árvore Binária de Busca – Operação busca**
- Como encontrar o elemento G na árvore a seguir?



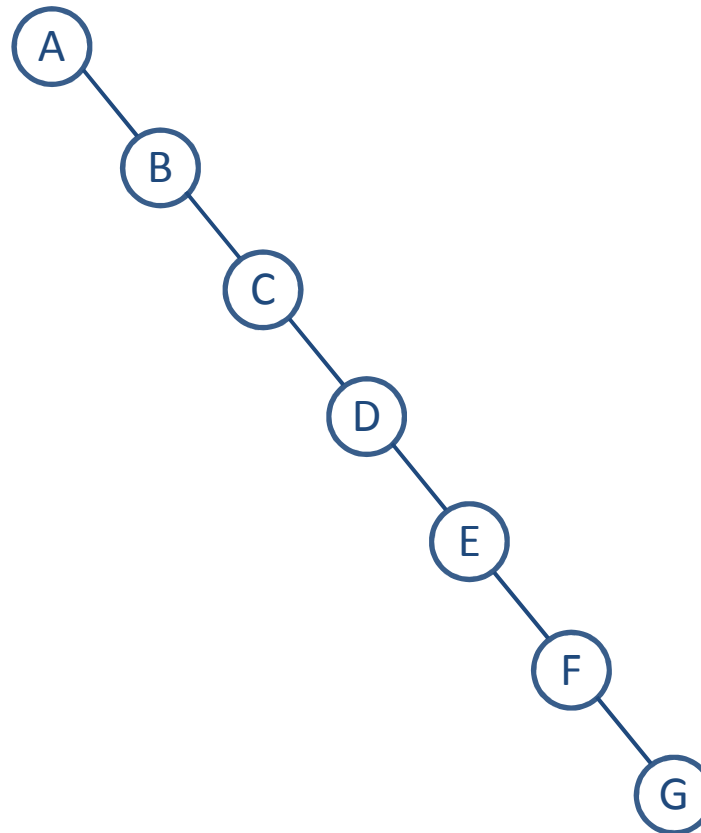
- A busca em uma árvore binária com altura máxima tem complexidade  $O(n)$ .



- **Árvore Binária de Busca – Operação inserção**
  - Primeiramente devemos identificar onde o elemento deve ser incluído. Em seguida, basta a inserção do elemento.
  - A disposição final da árvore depende da ordem em que os nós forem incluídos.



- **Árvore Binária de Busca – Operação inserção**
  - Como ficaria uma árvore após a inclusão dos elementos: A, B, C, D, E, F e G.

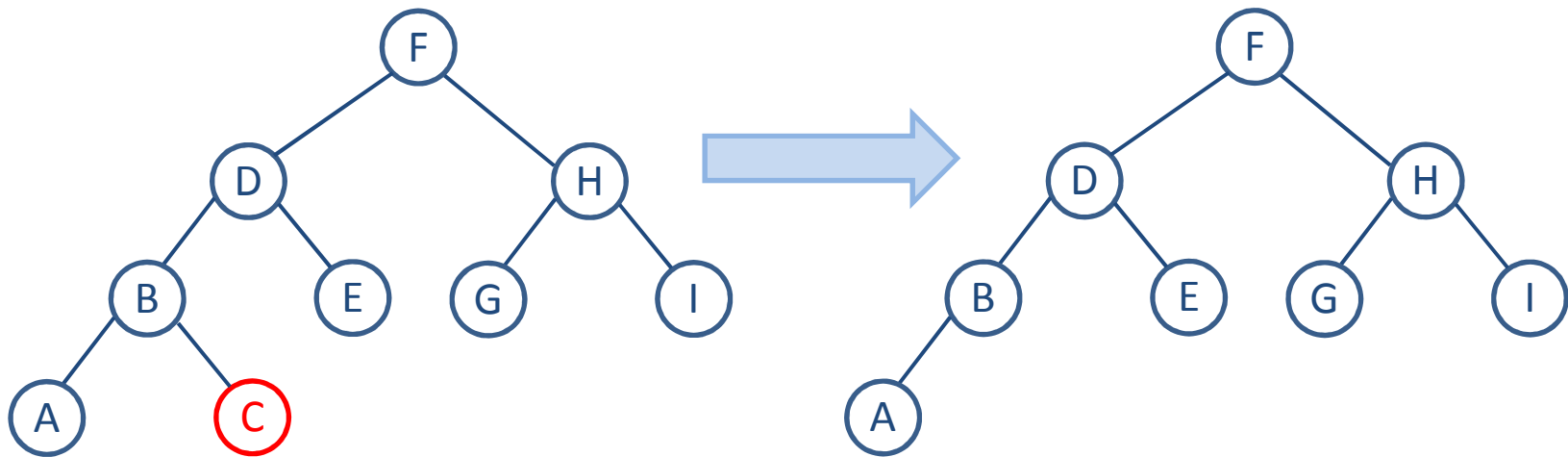




- **Árvore Binária de Busca – Operação remoção**
- Para remover um elemento da árvore binária de busca devemos considerar três cenários:
  - Elemento a ser removido não possui filho
  - Elemento a ser removido possui 1 filho
  - Elemento a ser removido possui 2 filhos

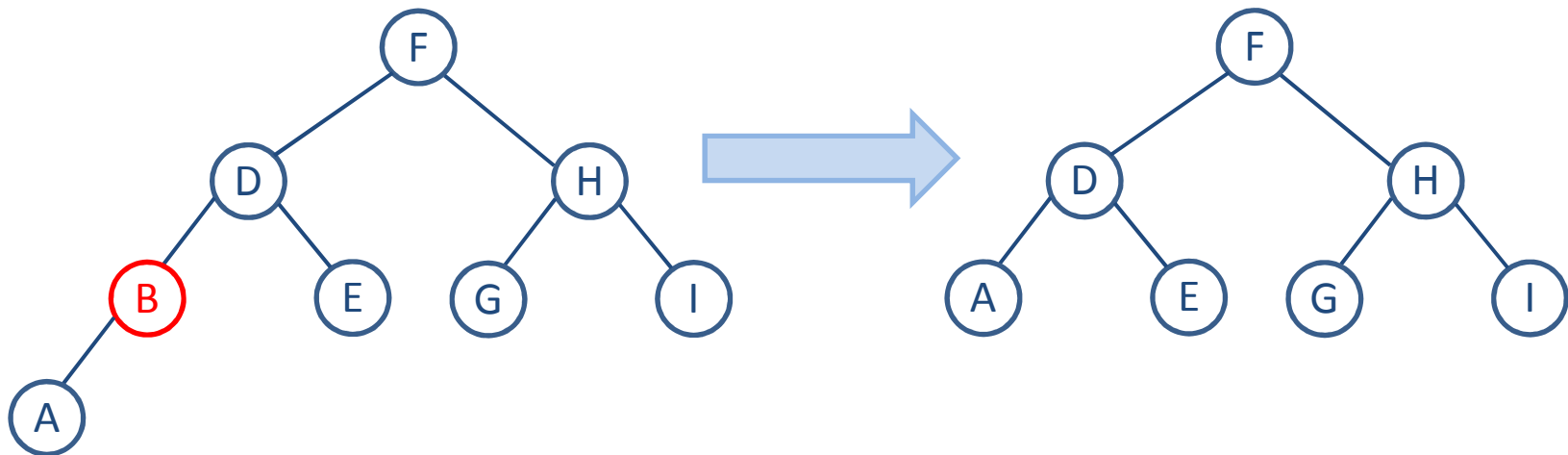


- **Árvore Binária de Busca – Operação remoção**
  - **Elemento a ser removido não possui filho**
    - Nesse cenário basta remover o elemento.



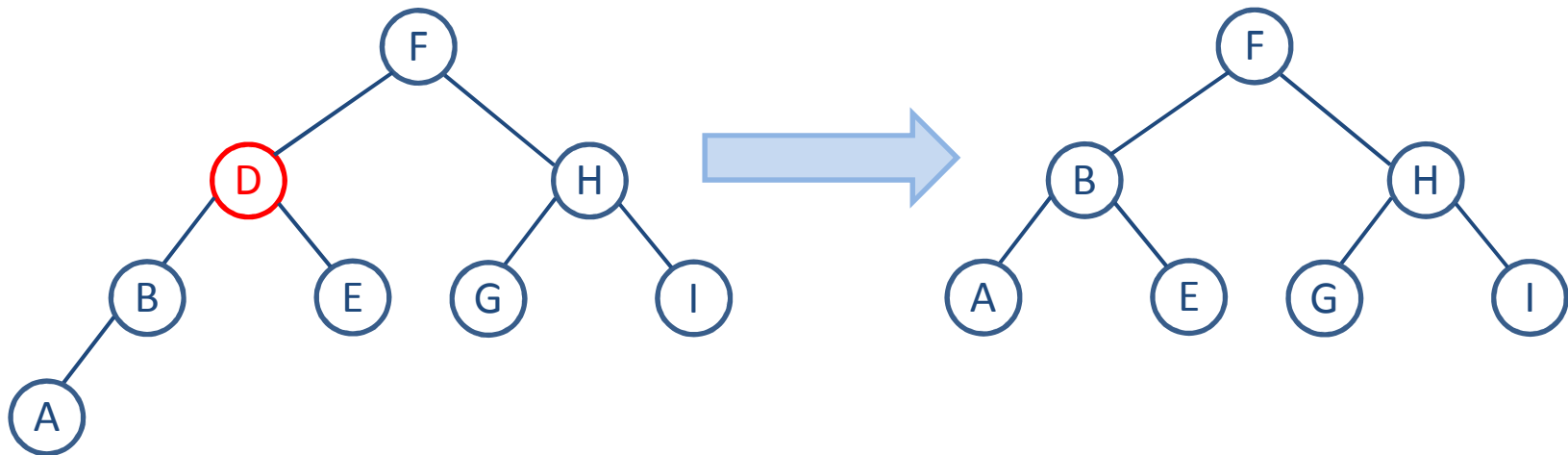


- **Árvore Binária de Busca – Operação remoção**
  - **Elemento a ser removido possui 1 filho**
    - Nesse cenário, o filho do nó a ser removido passa a ser filho do pai do nó a ser removido.





- **Árvore Binária de Busca – Operação remoção**
  - **Elemento a ser removido possui 2 filhos**
    - Nesse cenário o elemento pode ser substituído pelo:
      - Maior valor da subárvore esquerda.
      - Menor valor da subárvore direita.





# Questão 19

## [2012 – CESPE - Banco da Amazônia - Administração de Dados]

O tipo de dados árvore representa organizações hierárquicas entre dados.

Certo      Errado



# Questão 19

**[2012 – CESPE - Banco da Amazônia - Administração de Dados]**

O tipo de dados árvore representa organizações hierárquicas entre dados.

➡ Certo      Errado



## Questão 20

### [2012 – CESPE - Banco da Amazônia - Administração de Dados]

Acerca do tipo de dados árvore, julgue os próximos itens.

Quando não é possível prever antecipadamente quantos nós serão necessários para a criação e utilização de uma árvore binária, utilizam-se, na sua implementação, listas encadeadas.

Certo

Errado



## Questão 20

### [2012 – CESPE - Banco da Amazônia - Administração de Dados]

Acerca do tipo de dados árvore, julgue os próximos itens.

Quando não é possível prever antecipadamente quantos nós serão necessários para a criação e utilização de uma árvore binária, utilizam-se, na sua implementação, listas encadeadas.

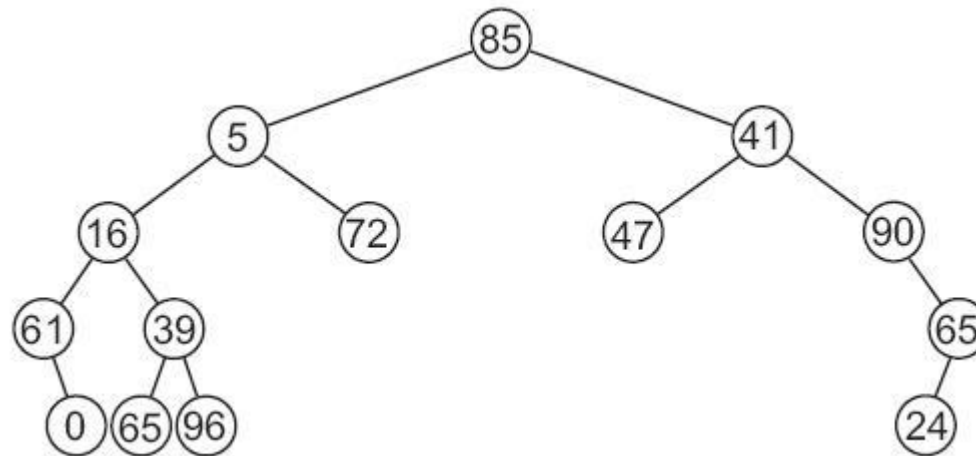
➡ Certo      Errado



## Questão 21

[2009 - CESGRANRIO - BNDES - Análise de Sistemas – Desenvolvimento]

Observe a árvore binária a seguir.



No percurso em pré-ordem dessa árvore, o quarto elemento a ser visitado é o

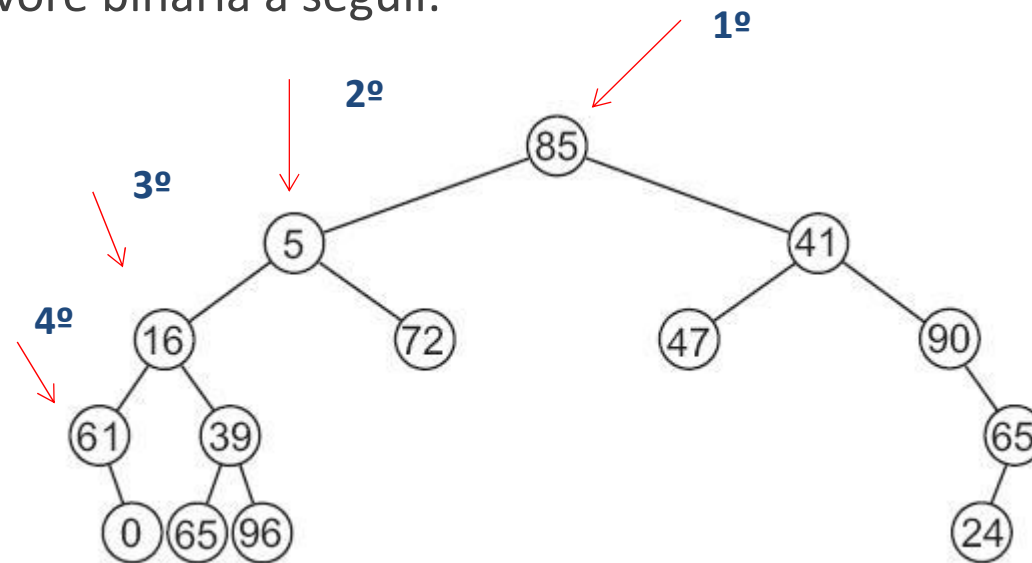
- A) 24.
- B) 39.
- C) 61.
- D) 85.
- E) 90.



# Questão 21

[2009 - CESGRANRIO - BNDES - Análise de Sistemas – Desenvolvimento]

Observe a árvore binária a seguir.



No percurso em pré-ordem dessa árvore, o quarto elemento a ser visitado é o

- A) 24.
- B) 39.
- ➡ C) 61.
- D) 85.
- E) 90.

```
1 public void preOrdem(ArvoreBinaria no) {  
2     if (no != null) {  
3         visitar( no );  
4         preOrdem( no.subarvoreEsquerda );  
5         preOrdem( no.subarvoreDireita );  
6     }  
7 }
```



## Questão 22

### [2012 – FCC - TRE-CE - Programação de Sistemas]

Com relação a árvores binárias é INCORRETO afirmar:

- A) Uma árvore binária é uma coleção finita de  $n > 0$  nodos que não pode ser nula.
- B) Uma árvore binária, cuja raiz armazena o elemento R, é denominada árvore de busca binária se todo elemento armazenado na subárvore esquerda é menor que R, nenhum elemento armazenado na subárvore direita é menor que R e as subárvores esquerda e direita também são árvores de busca binária.
- C) É um caso especial de árvore em que nenhum nodo tem grau superior a 2, isto é, nenhum nodo tem mais que dois filhos.
- D) Existe um nodo especial denominado raiz e os demais nodos são particionados em T1 e T2 estruturas disjuntas de árvores binárias. T1 é denominado subárvore esquerda e T2 subárvore direita da raiz.
- E) É uma árvore que pode ser nula



## Questão 22

### [2012 – FCC - TRE-CE - Programação de Sistemas]

Com relação a árvores binárias é INCORRETO afirmar:

- ➡ A) Uma árvore binária é uma coleção finita de  $n > 0$  nodos que não pode ser nula.
- B) Uma árvore binária, cuja raiz armazena o elemento R, é denominada árvore de busca binária se todo elemento armazenado na subárvore esquerda é menor que R, nenhum elemento armazenado na subárvore direita é menor que R e as subárvores esquerda e direita também são árvores de busca binária.
- C) É um caso especial de árvore em que nenhum nodo tem grau superior a 2, isto é, nenhum nodo tem mais que dois filhos.
- D) Existe um nodo especial denominado raiz e os demais nodos são particionados em T1 e T2 estruturas disjuntas de árvores binárias. T1 é denominado subárvore esquerda e T2 subárvore direita da raiz.
- E) É uma árvore que pode ser nula



# Gabarito



**19 – Certo**

**20 – Certo**

**21 – C**

**22 – A**



## ➤ Árvores Balanceadas

- As árvores balanceadas objetivam manter o custo de suas operações em  $O(\log n)$ .
- O custo das operações deve se manter durante toda manipulação da árvore.
- Para alcançar essa meta, a estrutura deve ser frequentemente alterada de forma a se moldar aos novos dados.
- Uma estrutura que opera com essas características é denominada balanceada.



- **O conceito de balanceamento**
  - As árvores completas minimizam o número de comparações efetuadas no pior caso para uma busca.
  - O objetivo é ter uma árvore cuja altura seja da mesma ordem de grandeza de uma árvore completa com o mesmo número de nós.
  - Uma árvore que satisfaça essa condição é denominada árvore balanceada.



## ➤ **Árvore AVL**

- Uma árvore binária T é denominada AVL quando todos os seus nós estão regulados.
- O nó de uma árvore binária é considerado regulado quando as alturas de suas subárvores esquerda e direita diferem em até uma unidade. Caso contrário, consideramos que o nó está desregulado.



## ➤ **Árvore AVL**

- Um nó regulado tem fator de balanceamento igual a: -1, 0 ou 1.
- Calculando o fator de balanceamento de um nó  $v$ :
  - $\text{balanço}(v) = h_{\text{SD}}(v) - h_{\text{SE}}(v)$
- Para aumentar o desempenho das operações realizadas sobre a estrutura, o valor do fator de balanceamento é pré-calculado e mantido nos nós correspondentes. Esses valores são atualizados durante a realização das operações de inserção e remoção.



- **Árvore AVL – Principais operações**
  - Definição das principais operações:

```
1 interface IArvoreAVL {  
2     No busca( char id );  
3     void inserir( No elemento );  
4     No remove( char id );  
5 }
```



## ➤ Árvores AVL – Principais operações

- As operações realizadas nas árvores AVL seguem os mesmos princípios das operações realizadas nas árvores binárias de busca, com as seguintes variações:
  - Após inserir ou remover um elemento na árvore, deve-se garantir que todos os nós da árvore continuem regulados.
  - Para garantir essa propriedade, rotações são executadas na estrutura.



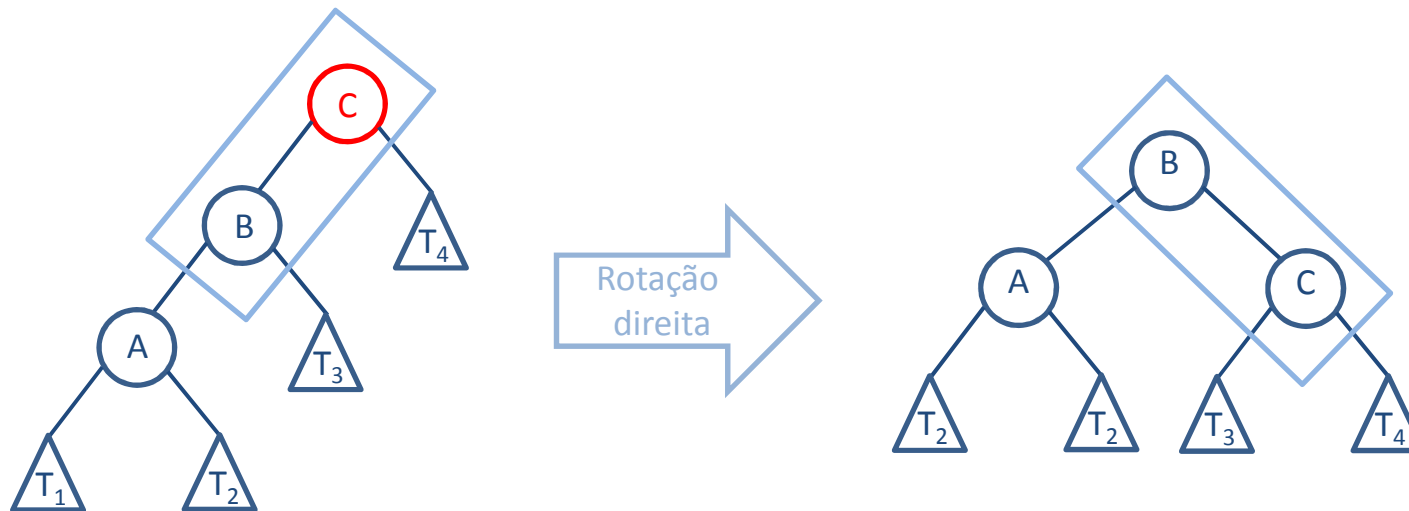
## ➤ Árvores AVL - Rotações

- Rotações são transformações aplicadas na estrutura da árvore de forma a manter a mesma regulada.
- Tipos de rotações:
  - Rotação à direita
  - Rotação à esquerda
  - Rotação dupla à direita
  - Rotação dupla à esquerda



## ➤ Árvores AVL – Rotação à direita

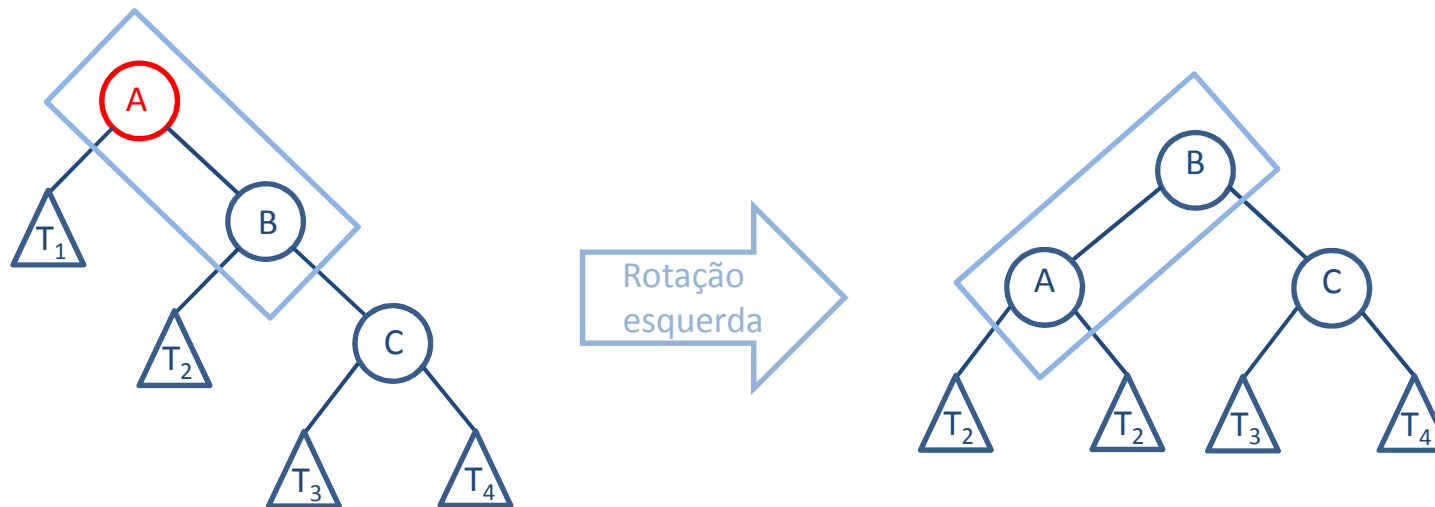
$$\text{balanço}(c) = h_{SD}(c) - h_{SE}(c)$$
$$\text{balanço}(c) = 1 - 3 = -2$$





## ➤ Árvores AVL – Rotação à esquerda

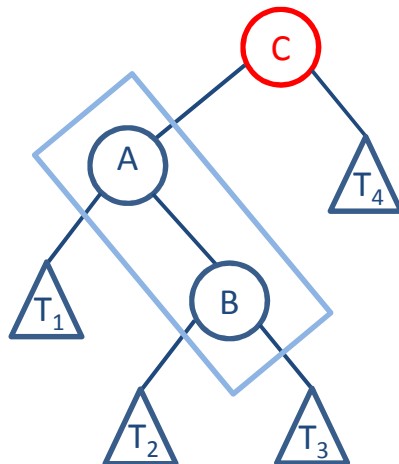
$$\text{balanço}(c) = h_{SD}(c) - h_{SE}(c)$$
$$\text{balanço}(c) = 3 - 1 = 2$$



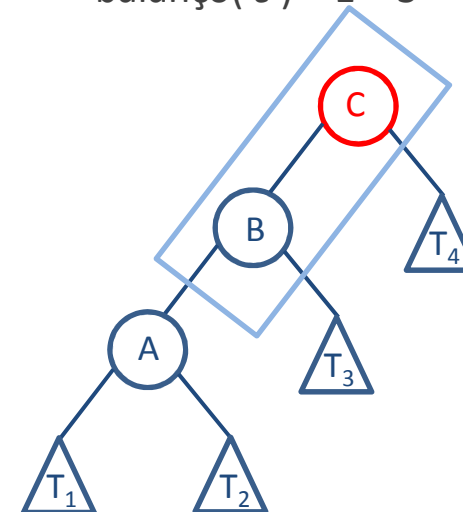


## ➤ Árvores AVL – Rotação dupla à direita

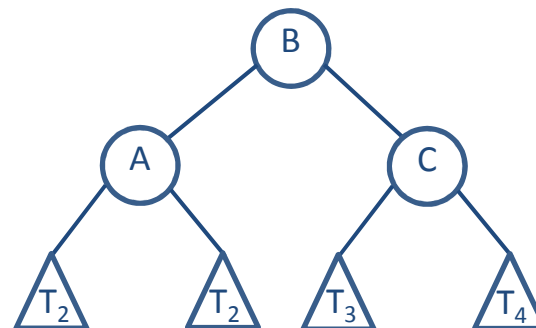
$$\text{balanço}(c) = h_{SD}(c) - h_{SE}(c)$$
$$\text{balanço}(c) = 1 - 3 = -2$$



[1] Rotação  
esquerda



[2] - Rotação  
direita

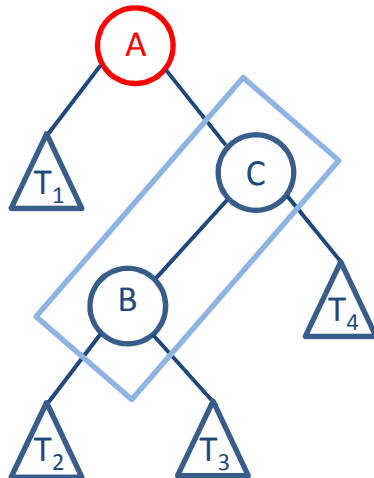




## ➤ Árvores AVL – Rotação dupla à esquerda

$$\text{balanço}(c) = h_{SD}(c) - h_{SE}(c)$$

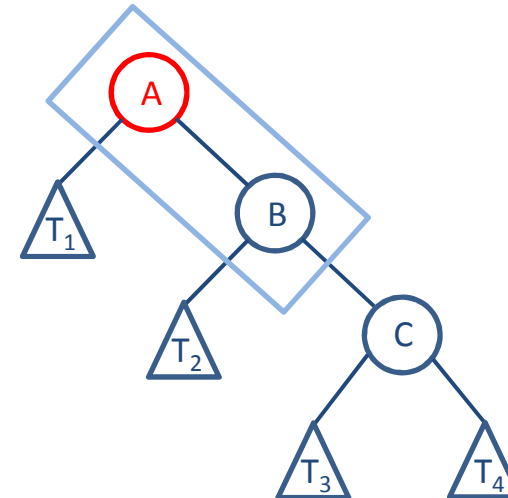
$$\text{balanço}(c) = 3 - 1 = 2$$



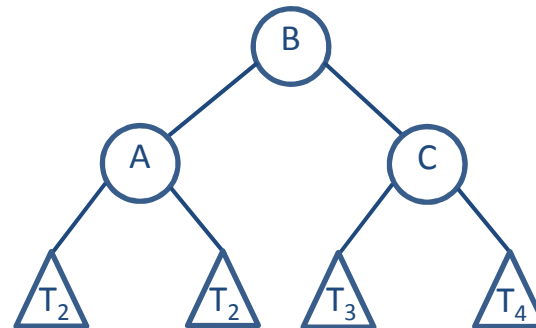
[1] - Rotação direita

$$\text{balanço}(c) = h_{SD}(c) - h_{SE}(c)$$

$$\text{balanço}(c) = 3 - 1 = 2$$



[2] - Rotação esquerda





## ➤ Árvores AVL

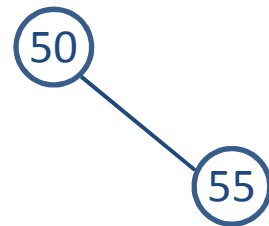
- Como ficaria uma árvore AVL, inicialmente vazia, após a inserção dos seguintes elementos: **50**, 55, 60, 70, 52 e 51.

50



## ➤ Árvores AVL

- Como ficaria uma árvore AVL, inicialmente vazia, após a inserção dos seguintes elementos: 50, **55**, 60, 70, 52 e 51.



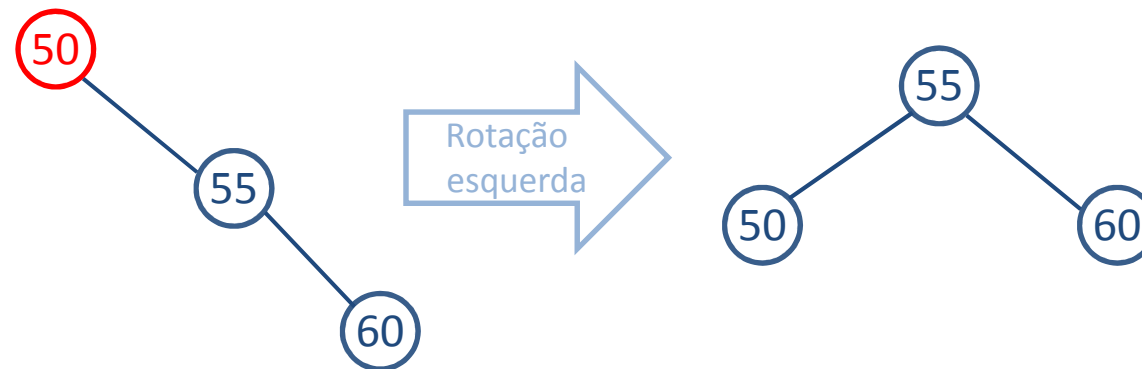


## ➤ Árvores AVL

- Como ficaria uma árvore AVL, inicialmente vazia, após a inserção dos seguintes elementos: 50, 55, **60**, 70, 52 e 51.

$$\text{balanço}(c) = h_{SD}(c) - h_{SE}(c)$$

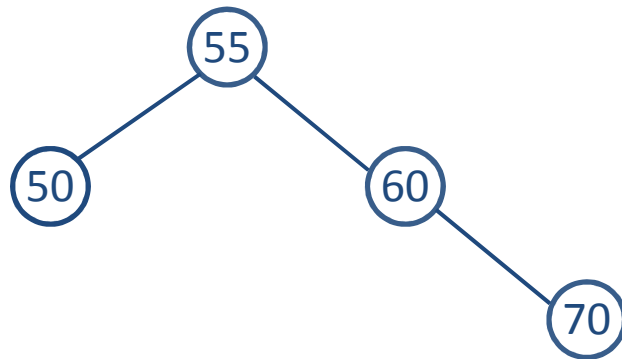
$$\text{balanço}(c) = 2 - 0 = 2$$





## ➤ Árvores AVL

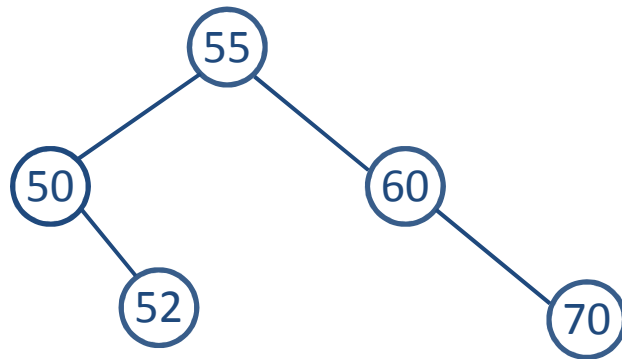
- Como ficaria uma árvore AVL, inicialmente vazia, após a inserção dos seguintes elementos: 50, 55, 60, **70**, 52 e 51.





## ➤ Árvores AVL

- Como ficaria uma árvore AVL, inicialmente vazia, após a inserção dos seguintes elementos: 50, 55, 60, 70, **52** e 51.

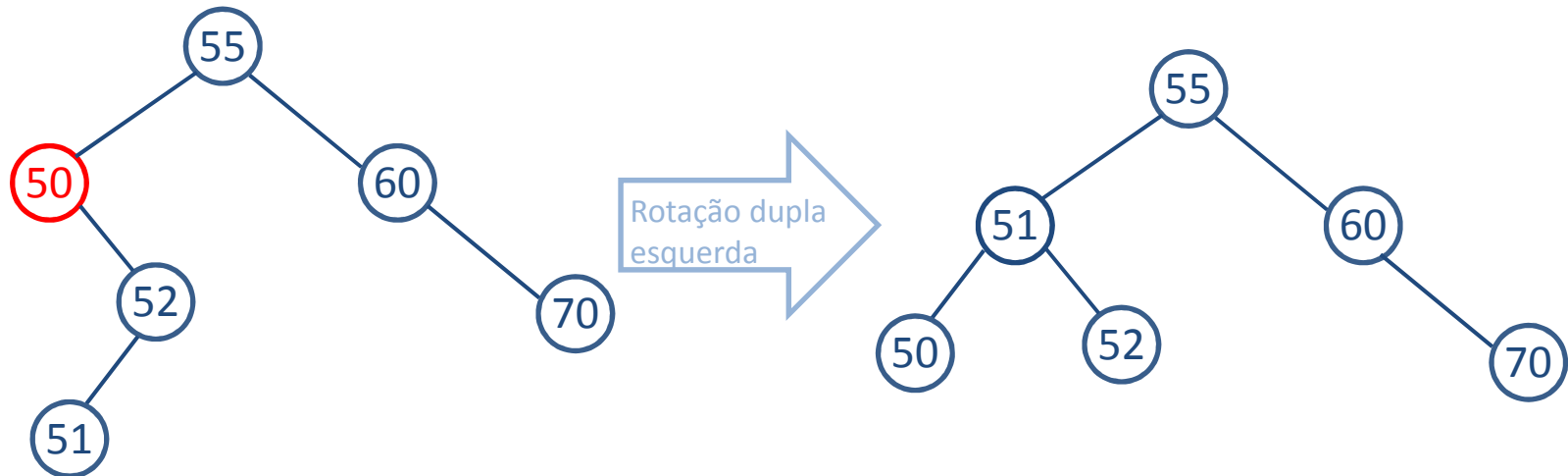




## ➤ Árvores AVL

- Como ficaria uma árvore AVL, inicialmente vazia, após a inserção dos seguintes elementos: 50, 55, 60, 52 e **51**.

$$\text{balanço}(c) = h_{SD}(c) - h_{SE}(c)$$
$$\text{balanço}(c) = 2 - 0 = 2$$





## Questão 23

### [2011 – FCC - TRE-RN - Técnico Programação de Sistemas]

Uma estrutura de dados onde cada nó mantém uma informação adicional, chamada fator de balanceamento, que indica a diferença de altura entre as subárvores esquerda e direita, é conhecida por árvore:

- A) hiberbólica.
- B) de busca binária.
- C) ordenada.
- D) AVL.
- E) binária.



## Questão 23

### [2011 – FCC - TRE-RN - Técnico Programação de Sistemas]

Uma estrutura de dados onde cada nó mantém uma informação adicional, chamada fator de balanceamento, que indica a diferença de altura entre as subárvores esquerda e direita, é conhecida por árvore:

- A) hiberbólica.
- B) de busca binária.
- C) ordenada.
- ➡ D) AVL.
- E) binária.



# Questão 24

## [2012 – CESPE - TRE-RJ - Técnico Judiciário - Programação de Sistemas]

Julgue os itens a seguir, referentes a estrutura de dados e organização de arquivos.

Na raiz de uma árvore balanceada, o número de descendentes da esquerda e de descendentes da direita é igual.

Certo          Errado



# Questão 24

## [2012 – CESPE - TRE-RJ - Técnico Judiciário - Programação de Sistemas]

Julgue os itens a seguir, referentes a estrutura de dados e organização de arquivos.

Na raiz de uma árvore balanceada, o número de descendentes da esquerda e de descendentes da direita é igual.

Certo      ➡ Errado



## Questão 25

### [2010 – CESGRANRIO – EPE - Analista de Gestão Corporativa - TI]

Um programador decidiu utilizar, em determinado sistema de análise estatística, uma árvore AVL como estrutura de dados. Considerando-se  $n$  a quantidade de elementos dessa árvore, o melhor algoritmo de pesquisa, com base em comparações, possui complexidade de tempo, no pior caso, igual a:

- A)  $O(1)$
- B)  $O(\log n)$
- C)  $\Omega(n)$
- D)  $\Omega(n \log n)$
- E)  $\Omega(n^2)$



## Questão 25

### [2010 – CESGRANRIO – EPE - Analista de Gestão Corporativa - TI]

Um programador decidiu utilizar, em determinado sistema de análise estatística, uma árvore AVL como estrutura de dados. Considerando-se  $n$  a quantidade de elementos dessa árvore, o melhor algoritmo de pesquisa, com base em comparações, possui complexidade de tempo, no pior caso, igual a:

- A)  $O(1)$
- ➔ B)  $O(\log n)$
- C)  $\Omega(n)$
- D)  $\Omega(n \log n)$
- E)  $\Omega(n^2)$





**23 – D**

**24 – Errado**

**25 – B**



## ➤ **Árvore B**

- A Árvore B é uma estrutura balanceada e largamente utilizada como forma de armazenamento secundário. Ela minimiza o tempo de acesso para buscas, inserções e remoções.
- Um nó de uma Árvore B é denominado página.
- É possível manter mais de uma chave em cada página da estrutura.



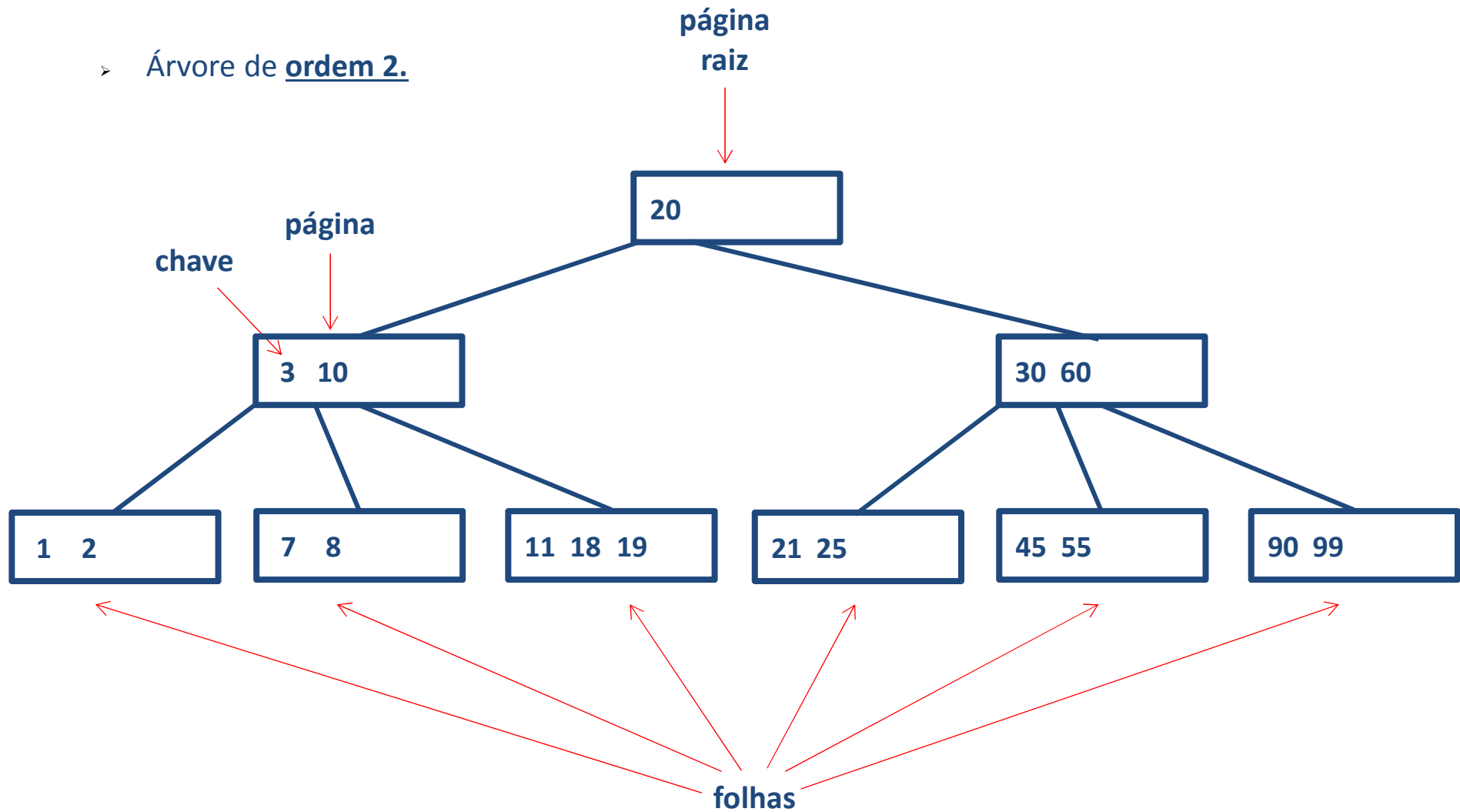
## ➤ **Árvore B**

- Uma Árvore B de ordem  $d$  é uma árvore ordenada que é vazia, ou satisfaz as seguintes condições:
  - Cada página possui entre  $d$  e  $2d$  chaves, exceto a página raiz que possui entre  $1$  e  $2d$  chaves;
  - Seja  $m$  o número de chaves em uma página  $P$ , então:
    - $P$  tem  $m + 1$  ponteiros para filhos;
    - Se  $P$  for uma folha, todos os ponteiros apontam para NULL.
    - Se  $P$  não for uma folha, todos os ponteiros referenciam uma página.
  - Todas as folhas estão no mesmo nível.



## ➤ Árvore B

- Árvore de ordem 2.





- Principais operações de uma Árvore B
  - Definição das principais operações:

```
1 interface IArvoreB {  
2     Elemento busca( int chave );  
3     void inserir( Elemento elemento );  
4     Elemento remove( int chave );  
5 }
```



## ➤ **Árvore B – Operação busca**

- Passos para buscar um elemento em uma Árvore B:

(A busca inicia pela página raiz)

**Passo 1:** Busca-se o nó na página corrente.

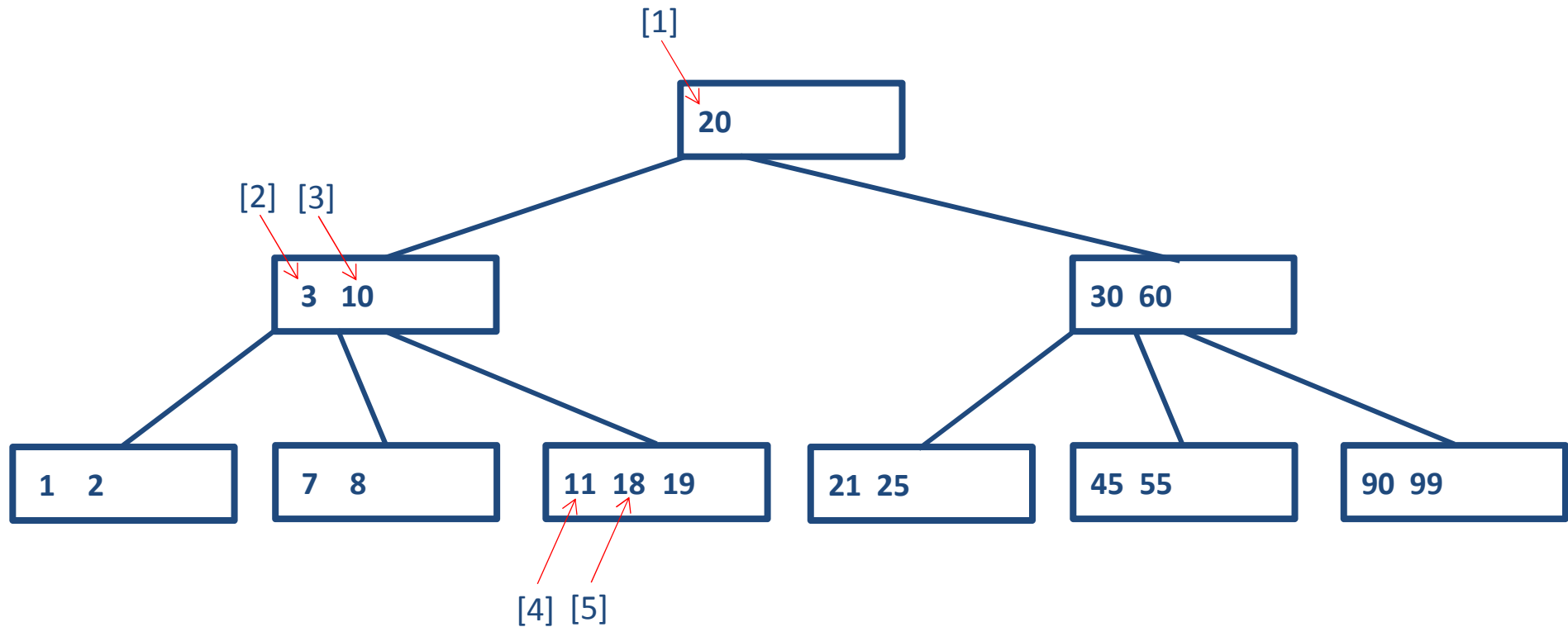
**Passo 2:** Caso o valor seja encontrado, a operação é encerrada.

**Passo 3:** Caso contrário, identificamos a próxima página a ser pesquisada e voltamos para o passo 1.

- Essa operação tem complexidade  $O(\log_d n)$ , onde  $d$  é a ordem da árvore.



- **Árvore B – Operação busca**
  - Como localizar o elemento cuja chave é 18?





## ➤ **Árvore B – Operação inserção**

- Para inserir um elemento em uma Árvore B devemos, inicialmente, identificar onde o elemento deve ser incluído. A operação busca auxilia nessa tarefa.
- A seguir incluímos o elemento na posição apropriada.
- Se após a inserção uma página ficar com  $2d + 1$  elementos, então será necessário realizar uma cisão.



## ➤ **Árvore B – Operação inserção**

- Como ficaria uma Árvore B (ordem 2) após a inclusão dos seguintes elementos: **60**, 90, 30, 55, 99, 45, 10, 29, 7, 3, 20, 18, 11, 25, 8, 2, 1 e 19?

60



## ➤ **Árvore B – Operação inserção**

- Como ficaria uma Árvore B (ordem 2) após a inclusão dos seguintes elementos: 60, **90**, 30, 55, 99, 45, 10, 29, 7, 3, 20, 18, 11, 25, 8, 2, 1 e 19?

60 90



## ➤ **Árvore B – Operação inserção**

- Como ficaria uma Árvore B (ordem 2) após a inclusão dos seguintes elementos: 60, 90, **30**, 55, 99, 45, 10, 29, 7, 3, 20, 18, 11, 25, 8, 2, 1 e 19?

30 60 90



## ➤ **Árvore B – Operação inserção**

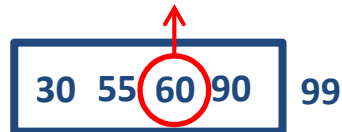
- Como ficaria uma Árvore B (ordem 2) após a inclusão dos seguintes elementos: 60, 90, 30, **55**, 99, 45, 10, 29, 7, 3, 20, 18, 11, 25, 8, 2, 1 e 19?

30 55 60 90



## ➤ Árvore B – Operação inserção

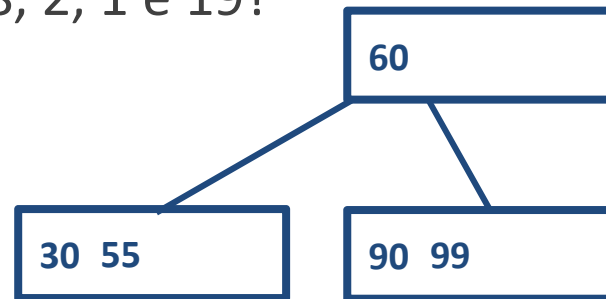
- Como ficaria uma Árvore B (ordem 2) após a inclusão dos seguintes elementos: 60, 90, 30, 55, **99**, 45, 10, 29, 7, 3, 20, 18, 11, 25, 8, 2, 1 e 19?





## ➤ **Árvore B – Operação inserção**

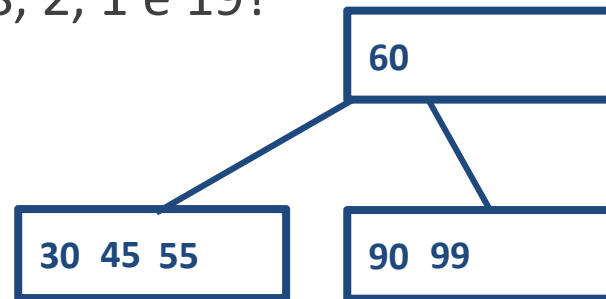
- Como ficaria uma Árvore B (ordem 2) após a inclusão dos seguintes elementos: 60, 90, 30, 55, **99**, 45, 10, 29, 7, 3, 20, 18, 11, 25, 8, 2, 1 e 19?





## ➤ **Árvore B – Operação inserção**

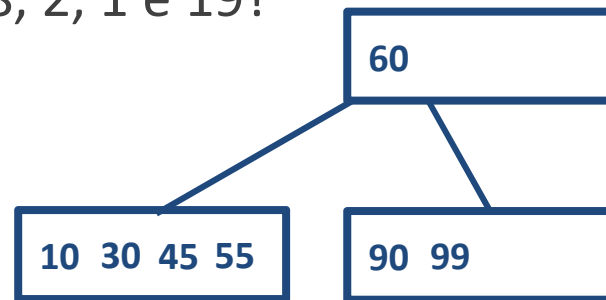
- Como ficaria uma Árvore B (ordem 2) após a inclusão dos seguintes elementos: 60, 90, 30, 55, 99, **45**, 10, 29, 7, 3, 20, 18, 11, 25, 8, 2, 1 e 19?





## ➤ **Árvore B – Operação inserção**

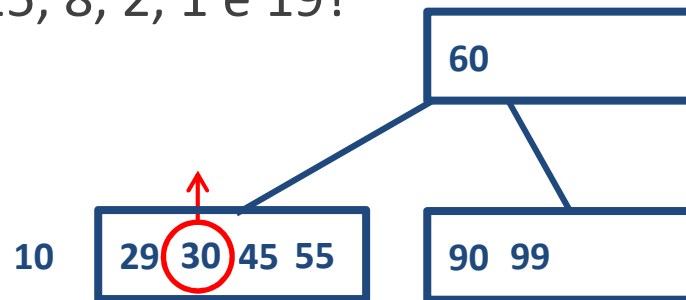
- Como ficaria uma Árvore B (ordem 2) após a inclusão dos seguintes elementos: 60, 90, 30, 55, 99, 45, **10**, 29, 7, 3, 20, 18, 11, 25, 8, 2, 1 e 19?





## ➤ Árvore B – Operação inserção

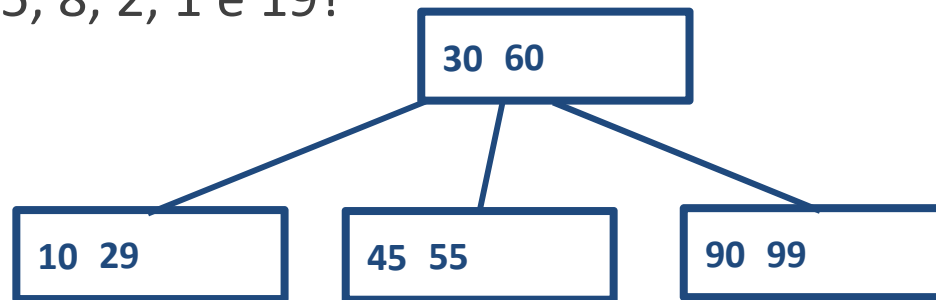
- Como ficaria uma Árvore B (ordem 2) após a inclusão dos seguintes elementos: 60, 90, 30, 55, 99, 45, 10, **29**, 7, 3, 20, 18, 11, 25, 8, 2, 1 e 19?





## ➤ Árvore B – Operação inserção

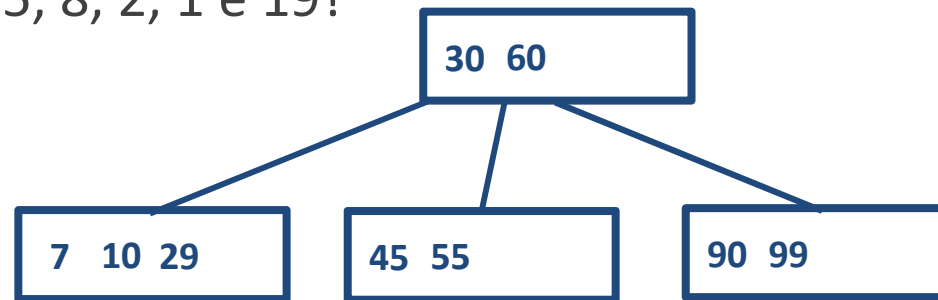
- Como ficaria uma Árvore B (ordem 2) após a inclusão dos seguintes elementos: 60, 90, 30, 55, 99, 45, 10, **29**, 7, 3, 20, 18, 11, 25, 8, 2, 1 e 19?





## ➤ **Árvore B – Operação inserção**

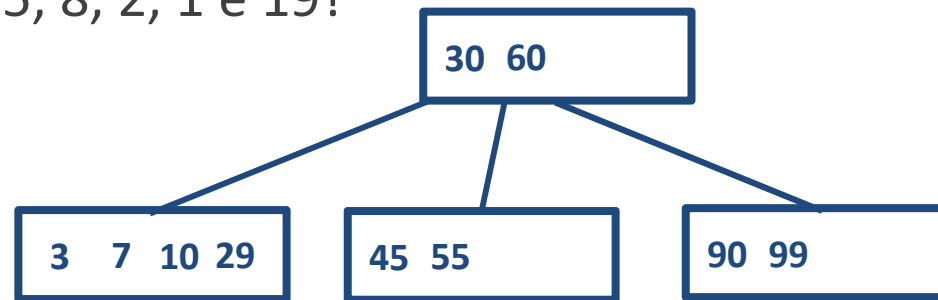
- Como ficaria uma Árvore B (ordem 2) após a inclusão dos seguintes elementos: 60, 90, 30, 55, 99, 45, 10, 29, **7**, 3, 20, 18, 11, 25, 8, 2, 1 e 19?





## ➤ **Árvore B – Operação inserção**

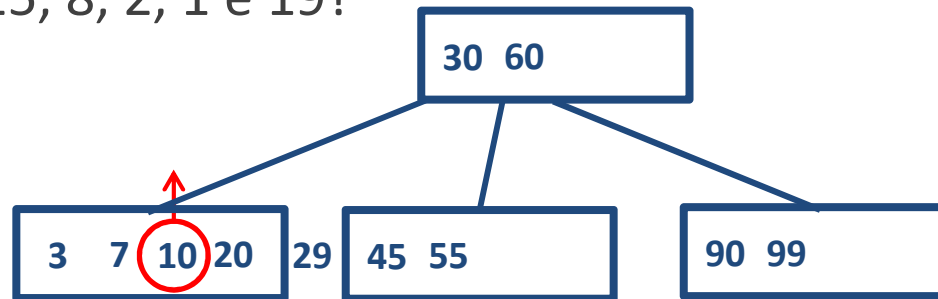
- Como ficaria uma Árvore B (ordem 2) após a inclusão dos seguintes elementos: 60, 90, 30, 55, 99, 45, 10, 29, 7, **3**, 20, 18, 11, 25, 8, 2, 1 e 19?





## ➤ Árvore B – Operação inserção

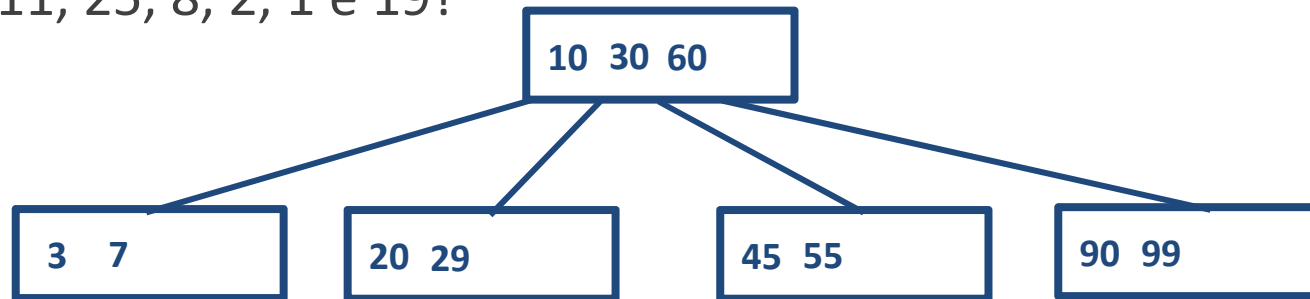
- Como ficaria uma Árvore B (ordem 2) após a inclusão dos seguintes elementos: 60, 90, 30, 55, 99, 45, 10, 29, 7, 3, **20**, 18, 11, 25, 8, 2, 1 e 19?





## ➤ **Árvore B – Operação inserção**

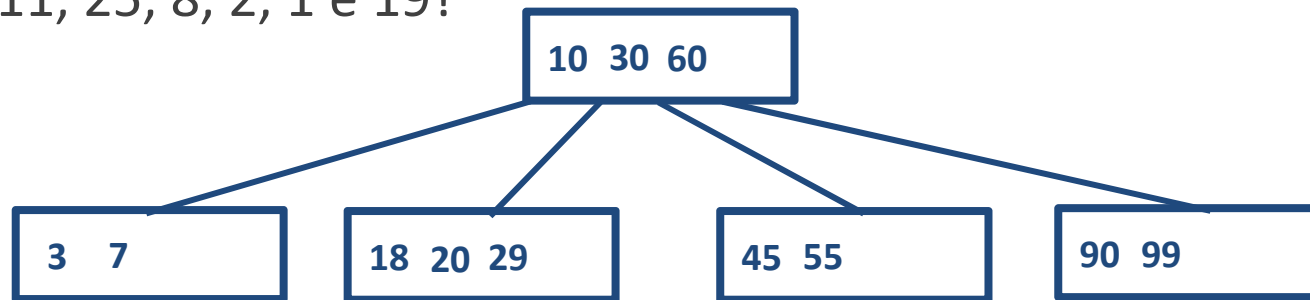
- Como ficaria uma Árvore B (ordem 2) após a inclusão dos seguintes elementos: 60, 90, 30, 55, 99, 45, 10, 29, 7, 3, **20**, 18, 11, 25, 8, 2, 1 e 19?





## ➤ **Árvore B – Operação inserção**

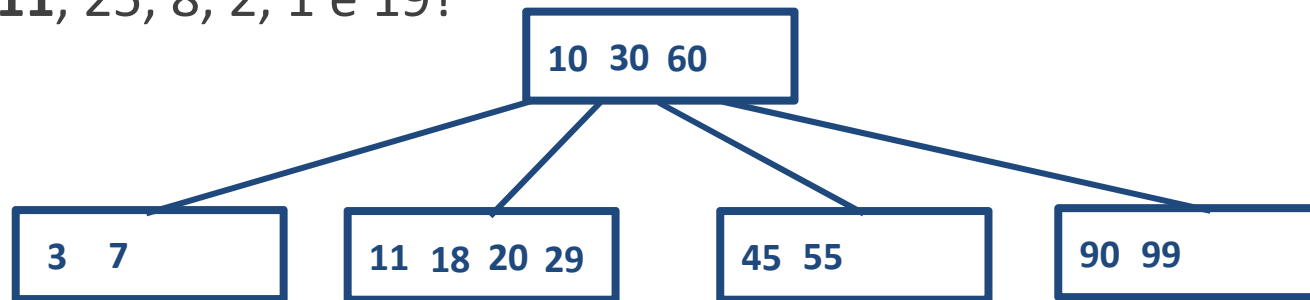
- Como ficaria uma Árvore B (ordem 2) após a inclusão dos seguintes elementos: 60, 90, 30, 55, 99, 45, 10, 29, 7, 3, 20, **18**, 11, 25, 8, 2, 1 e 19?





## ➤ **Árvore B – Operação inserção**

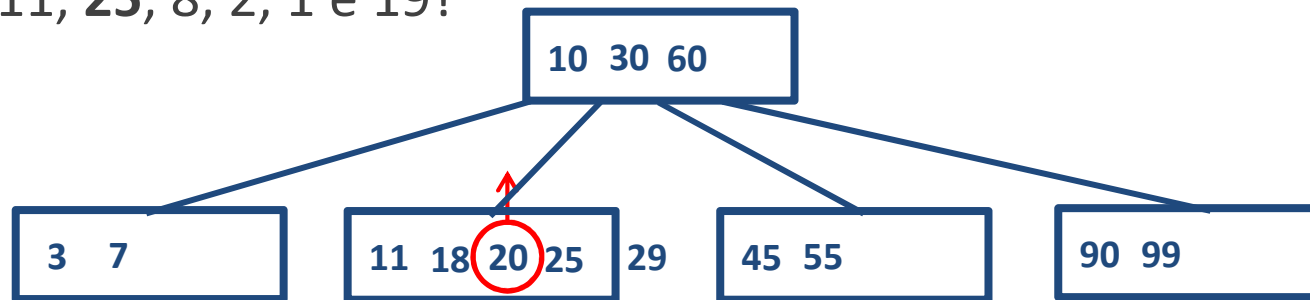
- Como ficaria uma Árvore B (ordem 2) após a inclusão dos seguintes elementos: 60, 90, 30, 55, 99, 45, 10, 29, 7, 3, 20, 18, **11**, 25, 8, 2, 1 e 19?





## ➤ **Árvore B – Operação inserção**

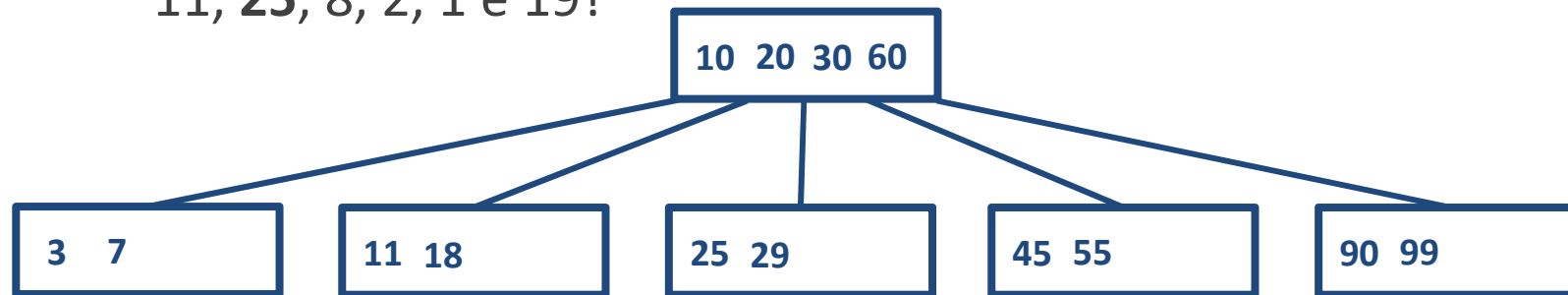
- Como ficaria uma Árvore B (ordem 2) após a inclusão dos seguintes elementos: 60, 90, 30, 55, 99, 45, 10, 29, 7, 3, 20, 18, 11, **25**, 8, 2, 1 e 19?





## ➤ **Árvore B – Operação inserção**

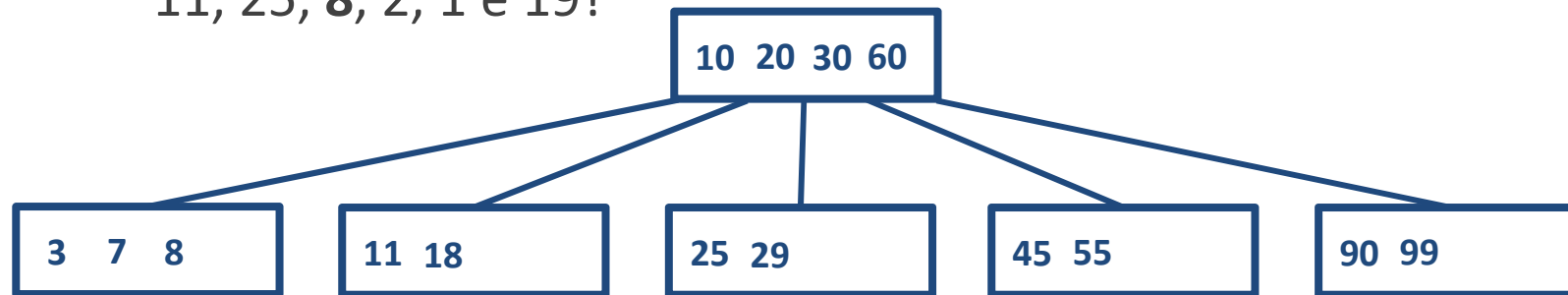
- Como ficaria uma Árvore B (ordem 2) após a inclusão dos seguintes elementos: 60, 90, 30, 55, 99, 45, 10, 29, 7, 3, 20, 18, 11, **25**, 8, 2, 1 e 19?





## ➤ **Árvore B – Operação inserção**

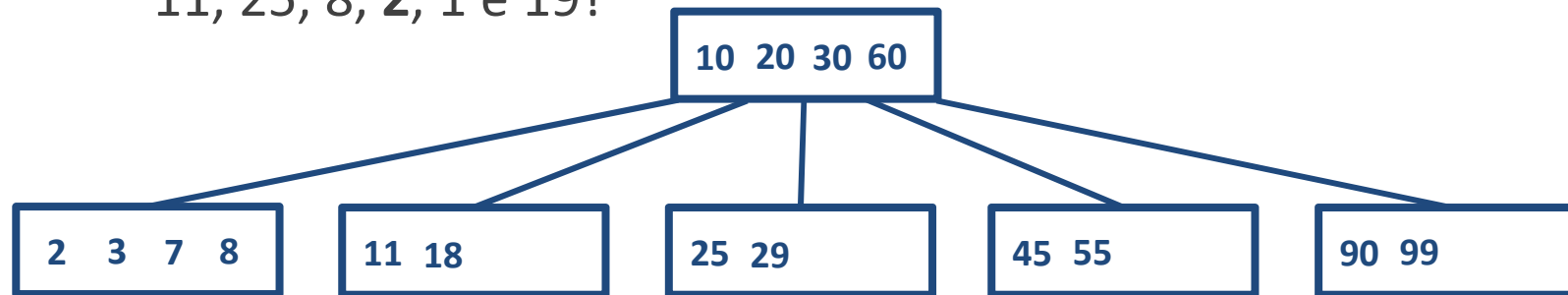
- Como ficaria uma Árvore B (ordem 2) após a inclusão dos seguintes elementos: 60, 90, 30, 55, 99, 45, 10, 29, 7, 3, 20, 18, 11, 25, **8**, 2, 1 e 19?





## ➤ **Árvore B – Operação inserção**

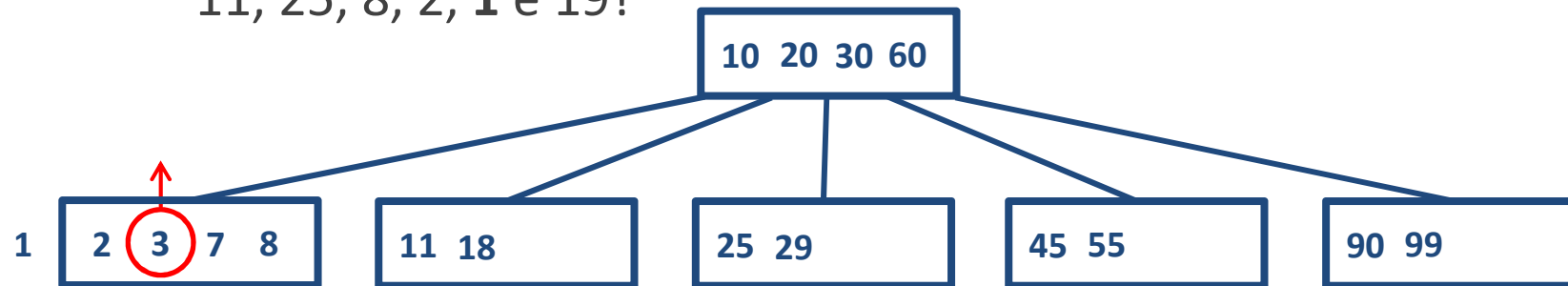
- Como ficaria uma Árvore B (ordem 2) após a inclusão dos seguintes elementos: 60, 90, 30, 55, 99, 45, 10, 29, 7, 3, 20, 18, 11, 25, 8, **2**, 1 e 19?





## ➤ **Árvore B – Operação inserção**

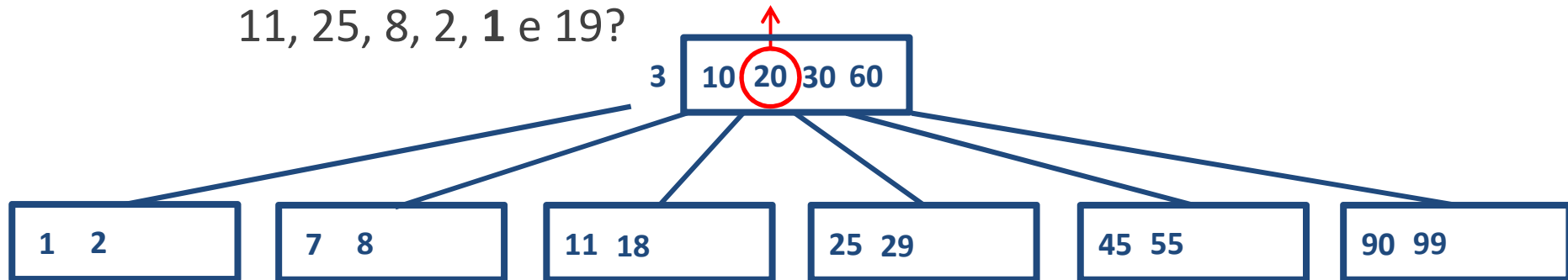
- Como ficaria uma Árvore B (ordem 2) após a inclusão dos seguintes elementos: 60, 90, 30, 55, 99, 45, 10, 29, 7, 3, 20, 18, 11, 25, 8, 2, 1 e 19?





## ➤ Árvore B – Operação inserção

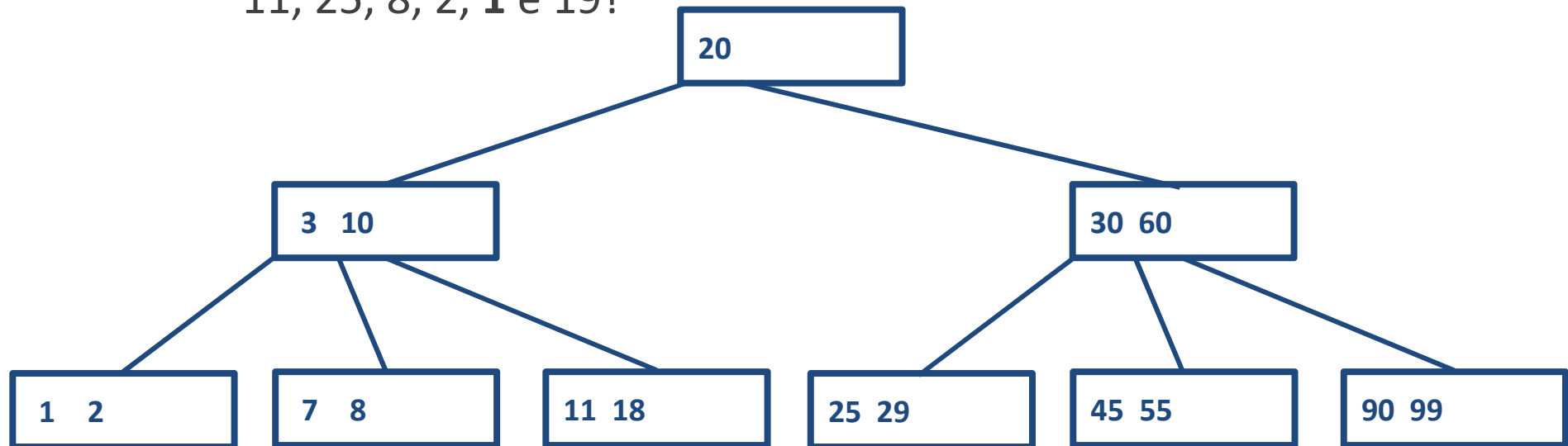
- Como ficaria uma Árvore B (ordem 2) após a inclusão dos seguintes elementos: 60, 90, 30, 55, 99, 45, 10, 29, 7, 3, 20, 18, 11, 25, 8, 2, 1 e 19?





## ➤ **Árvore B – Operação inserção**

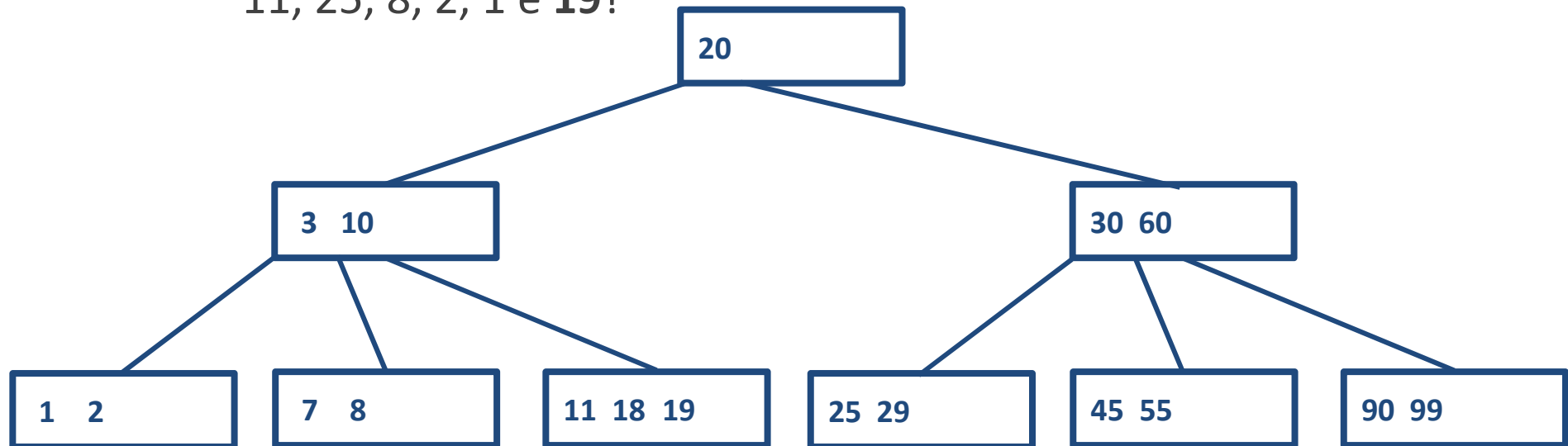
- Como ficaria uma Árvore B (ordem 2) após a inclusão dos seguintes elementos: 60, 90, 30, 55, 99, 45, 10, 29, 7, 3, 20, 18, 11, 25, 8, 2, 1 e 19?





## ➤ Árvore B – Operação inserção

- Como ficaria uma Árvore B (ordem 2) após a inclusão dos seguintes elementos: 60, 90, 30, 55, 99, 45, 10, 29, 7, 3, 20, 18, 11, 25, 8, 2, 1 e **19**?





- **Árvore B – Operação inserção**
  - Uma cisão transforma uma página com excesso de chaves em duas.
  - A cisão de páginas é propagável, podendo atingir a raiz da árvore.
  - Uma inserção pode alterar a altura da árvore.



## ➤ **Árvore B – Operação remoção**

- Na remoção devem ser considerados dois casos:
  - Se o elemento a ser removido se encontra na folha, é simplesmente retirado.
  - Caso contrário, ele é substituído pelo valor imediatamente maior, que se encontra em outra página.
- Se após a remoção alguma página fica com quantidade de chaves inferior a  $d$ , a estrutura deve ser ajustada com uma das seguintes técnicas:
  - Concatenação
  - Redistribuição



## ➤ **Árvore B – Operação remoção**

### ➤ **Concatenação**

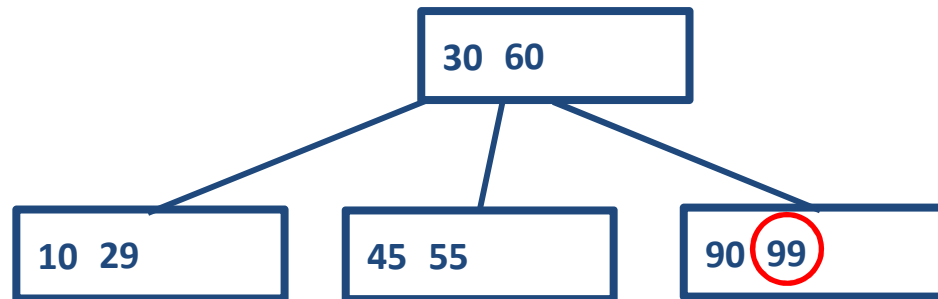
- Duas páginas adjacentes podem ser concatenadas se a soma de suas chaves for inferior a  $2d$ .
- A chave pai, que fica entre os dois ponteiros das páginas adjacentes, passa a fazer parte dessa nova página concatenada.
- Esse procedimento pode se propagar, podendo afetar a altura da árvore.



- **Árvore B – Operação remoção**

- **Concatenação**

- Como ficaria a árvore abaixo (ordem 2) após a remoção do elemento 99?

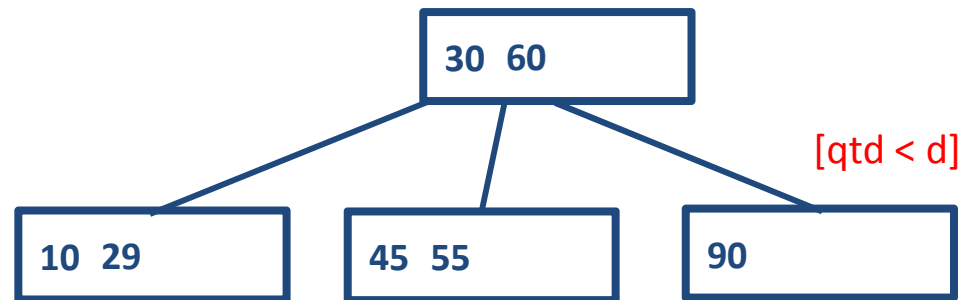




- **Árvore B – Operação remoção**

- **Concatenação**

- Como ficaria a árvore abaixo (ordem 2) após a remoção do elemento 99?

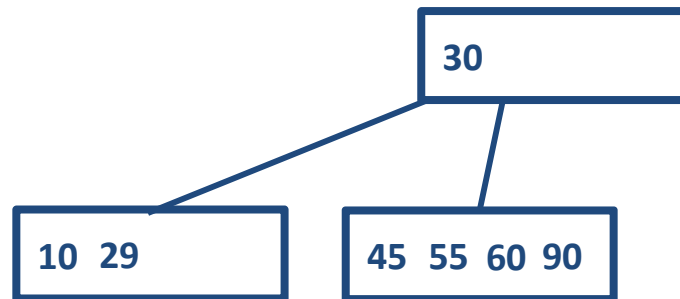




- **Árvore B – Operação remoção**

- **Concatenação**

- Como ficaria a árvore abaixo (ordem 2) após a remoção do elemento 99?





## ➤ **Árvore B – Operação remoção**

### ➤ **Redistribuição**

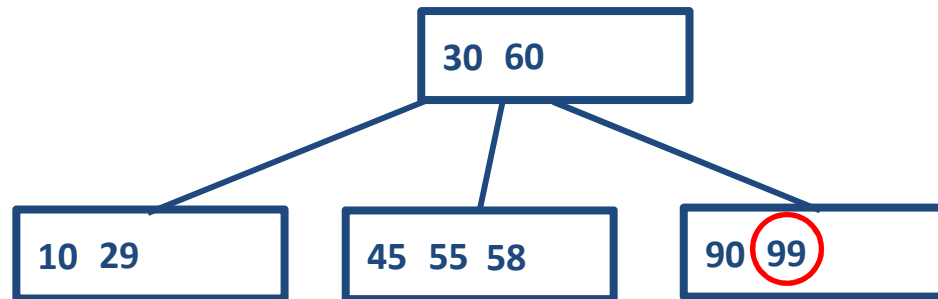
- No caso das duas páginas adjacentes possuírem em conjunto  $2d$  ou mais chaves essas podem ser equilibradamente distribuídas:
  - Concatena-se as duas páginas (descendo inclusive as chaves entre os ponteiros adjacentes).
  - Em seguida efetua-se uma cisão.
  - Esse procedimento não pode se propagar. A página pai é modificada, mas seu número de chaves permanece o mesmo.



- **Árvore B – Operação remoção**

- **Redistribuição**

- Como ficaria a árvore abaixo (ordem 2) após a remoção do elemento 99?

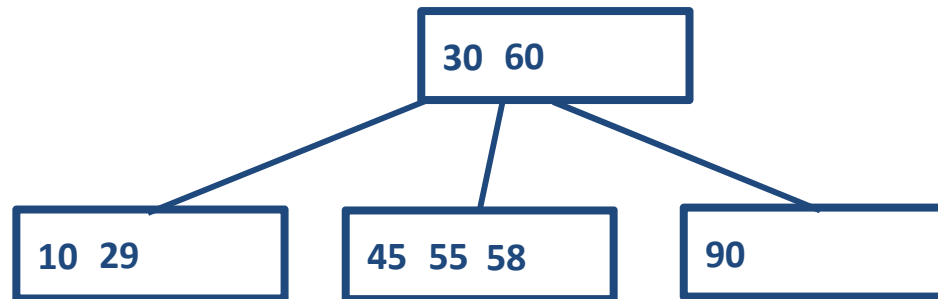




- **Árvore B – Operação remoção**

- **Redistribuição**

- Como ficaria a árvore abaixo (ordem 2) após a remoção do elemento 99?

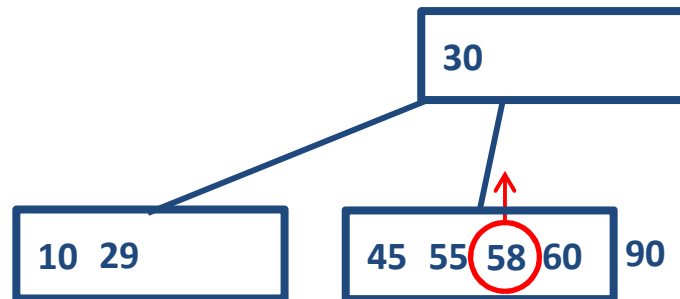




- **Árvore B – Operação remoção**

- **Redistribuição**

- Como ficaria a árvore abaixo (ordem 2) após a remoção do elemento 99?

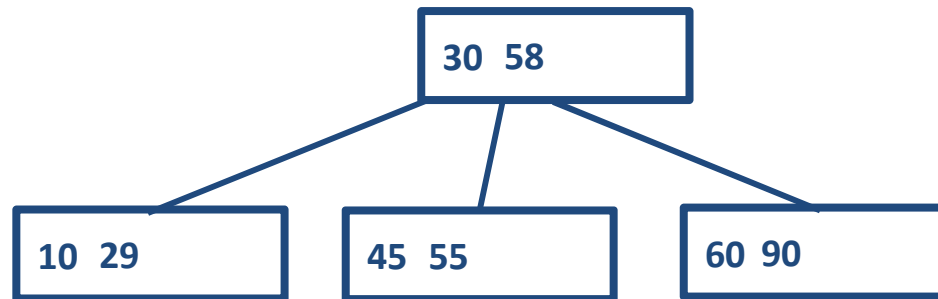




- **Árvore B – Operação remoção**

- **Redistribuição**

- Como ficaria a árvore abaixo (ordem 2) após a remoção do elemento 99?





- **Árvore B – Definições de outros autores**
  - Dada uma árvore B com grau mínimo igual a **t**:
    - Toda página diferente da raiz deve ter pelo menos **t** filhos.
    - Uma página pode ter no máximo **2t** filhos.
  - Como consequência:
    - Cada página teria uma quantidade mínima de chaves igual a **t - 1**;
    - Todo nó pode ter no máximo **2t - 1** chaves.



### [2010 – CESGRANRIO – Petrobras - Analista Processos de Negócios]

Uma árvore B é um tipo de árvore que se mantém balanceada com o decorrer do tempo. Para tanto, ela usa uma série de operações que garantem a manutenção de uma série de propriedades importantes, uma das quais é a ordem da árvore que pode ser definida como o número máximo de elementos que podem ser armazenados em um nó da árvore. Com base nesses conceitos, qual das situações a seguir representa uma propriedade das árvores B?

- A) Em uma árvore B de ordem maior do que 1, não é permitido que uma folha armazene apenas um elemento.
- B) Em uma árvore B de ordem  $d$ , a raiz armazena um número de elementos  $n$  tal que  $d \leq n \leq 2d$ .
- C) Em uma árvore B de ordem  $d$ , pode haver folhas em alturas diferentes da árvore até que tenham sido inseridos, pelo menos,  $2d+1$  elementos.
- D) Em um nó de uma árvore B que contenha  $n$  elementos não vazios, podem-se ter, no máximo,  $n/2$  ponteiros apontando para vazio (nil ou null).
- E) Em um nó interno de uma árvore B que contenha  $n$  elementos, têm-se exatamente  $n+1$  ponteiros que não apontam para vazio (nil ou null).



## Questão 26

### [2010 – CESGRANRIO – Petrobras - Analista Processos de Negócios]

Uma árvore B é um tipo de árvore que se mantém balanceada com o decorrer do tempo. Para tanto, ela usa uma série de operações que garantem a manutenção de uma série de propriedades importantes, uma das quais é a ordem da árvore que pode ser definida como o número máximo de elementos que podem ser armazenados em um nó da árvore. Com base nesses conceitos, qual das situações a seguir representa uma propriedade das árvores B?

- A) Em uma árvore B de ordem maior do que 1, não é permitido que uma folha armazene apenas um elemento.
- B) Em uma árvore B de ordem  $d$ , a raiz armazena um número de elementos  $n$  tal que  $d \leq n \leq 2d$ .
- C) Em uma árvore B de ordem  $d$ , pode haver folhas em alturas diferentes da árvore até que tenham sido inseridos, pelo menos,  $2d+1$  elementos.
- D) Em um nó de uma árvore B que contenha  $n$  elementos não vazios, podem-se ter, no máximo,  $n/2$  ponteiros apontando para vazio (nil ou null).
- ➡ E) Em um nó interno de uma árvore B que contenha  $n$  elementos, têm-se exatamente  $n+1$  ponteiros que não apontam para vazio (nil ou null).



## Questão 27

### [2008 – CESGRANRIO – BNDES - Análise de Sistemas - Desenvolvimento]

Considere uma árvore B de ordem 2 inicialmente vazia.

Os números abaixo são inseridos na seguinte ordem:

10, 15, 8, 3, 4, 12, 20, 9.

Que número(s) compõe(m) o nó raiz?

- A) 8
- B) 10
- C) 4 e 15
- D) 8 e 12
- E) 9 e 10



## Questão 27

Árvore B de ordem 2 inicialmente vazia.

Os números são inseridos na seguinte ordem: **10**, 15, 8, 3, 4, 12, 20, 9.

10



## Questão 27

Árvore B de ordem 2 inicialmente vazia.

Os números são inseridos na seguinte ordem: 10, **15**, 8, 3, 4, 12, 20, 9.

10 15



## Questão 27

Árvore B de ordem 2 inicialmente vazia.

Os números são inseridos na seguinte ordem: 10, 15, **8**, 3, 4, 12, 20, 9.

8 10 15



## Questão 27

Árvore B de ordem 2 inicialmente vazia.

Os números são inseridos na seguinte ordem: 10, 15, 8, **3**, 4, 12, 20, 9.

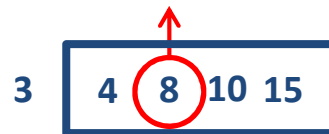
3	8	10	15
---	---	----	----



## Questão 27

Árvore B de ordem 2 inicialmente vazia.

Os números são inseridos na seguinte ordem: 10, 15, 8, 3, 4, 12, 20, 9.

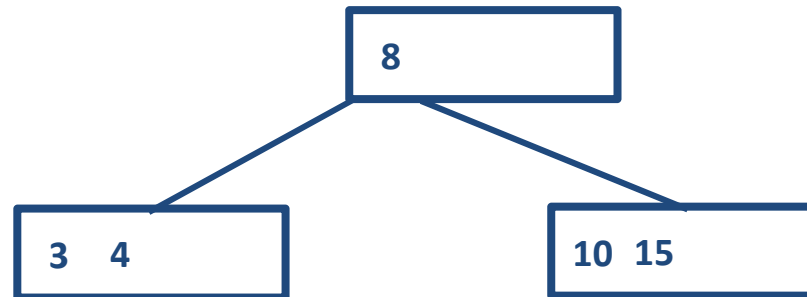




## Questão 27

Árvore B de ordem 2 inicialmente vazia.

Os números são inseridos na seguinte ordem: 10, 15, 8, 3, 4, 12, 20, 9.

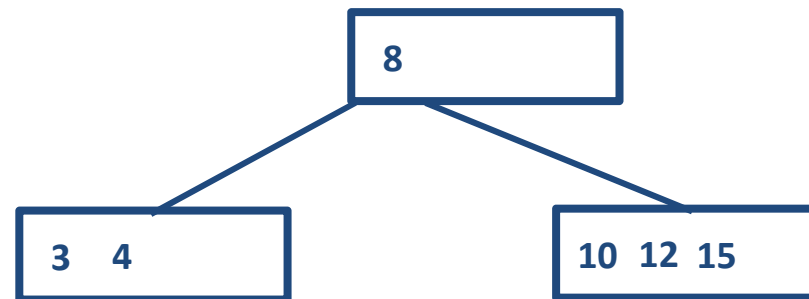




## Questão 27

Árvore B de ordem 2 inicialmente vazia.

Os números são inseridos na seguinte ordem: 10, 15, 8, 3, 4, **12**, 20, 9.

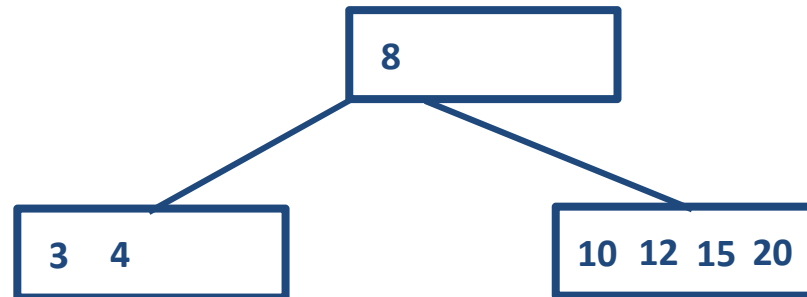




## Questão 27

Árvore B de ordem 2 inicialmente vazia.

Os números são inseridos na seguinte ordem: 10, 15, 8, 3, 4, 12, **20**, 9.

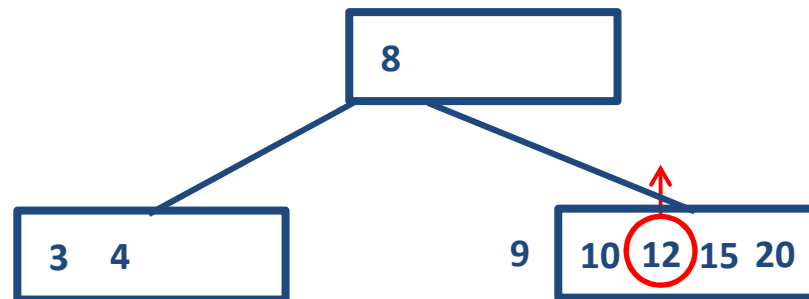




## Questão 27

Árvore B de ordem 2 inicialmente vazia.

Os números são inseridos na seguinte ordem: 10, 15, 8, 3, 4, 12, 20, **9**.

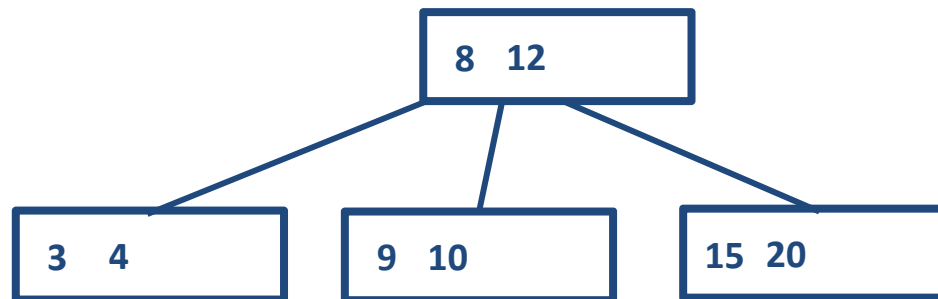




## Questão 27

Árvore B de ordem 2 inicialmente vazia.

Os números são inseridos na seguinte ordem: 10, 15, 8, 3, 4, 12, 20, **9**.





## Questão 27

### [2008 – CESGRANRIO – BNDES - Análise de Sistemas - Desenvolvimento]

Considere uma árvore B de ordem 2 inicialmente vazia.

Os números abaixo são inseridos na seguinte ordem:

10, 15, 8, 3, 4, 12, 20, 9.

Que número(s) compõe(m) o nó raiz?

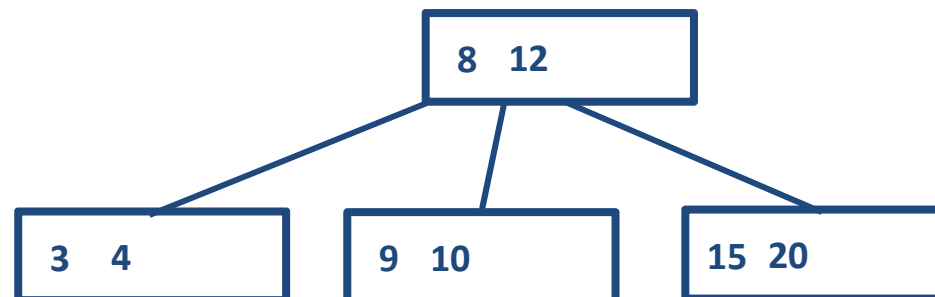
A) 8

B) 10

C) 4 e 15

➔ D) 8 e 12

E) 9 e 10



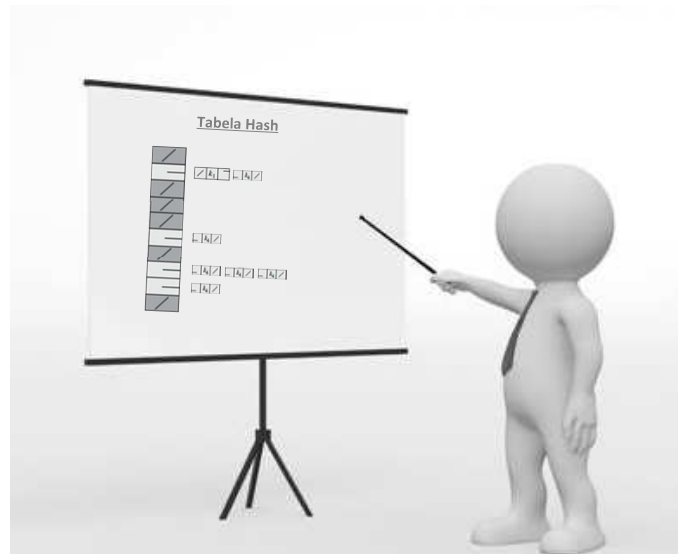




**26 – E**

**27 – D**





# Tabela Hash



## ➤ Conceitos gerais

- A tabela hash é uma estrutura de dados eficaz para implementar dicionários.
- Um elemento é armazenado em uma posição indicada por  $h(k)$ . A função hash  $h$  calcula a posição onde o elemento será armazenado dentro da tabela hash.



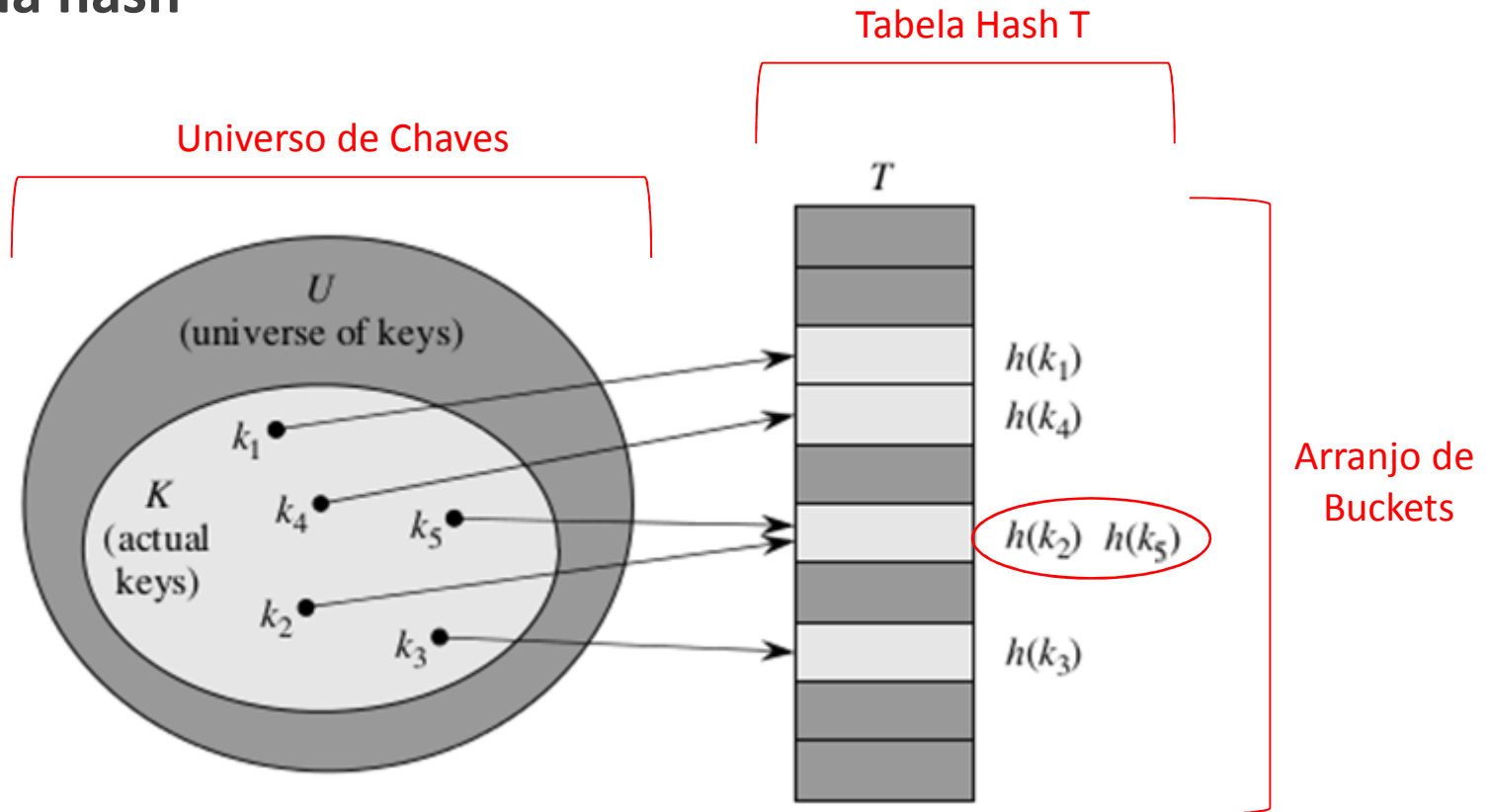
## ➤ Função hash

- Mapeia o universo de chaves para as posições de uma tabela hash.
- Bucket é o nome dado ao lugar onde os elementos são armazenados na tabela hash.
- A função hash reduz a faixa de índices possíveis e, conseqüentemente, o tamanho da tabela hash.
- Uma função hash pode mapear duas chaves distintas para uma mesma posição da tabela. Essa situação é chamada de colisão.



# Tabela Hash

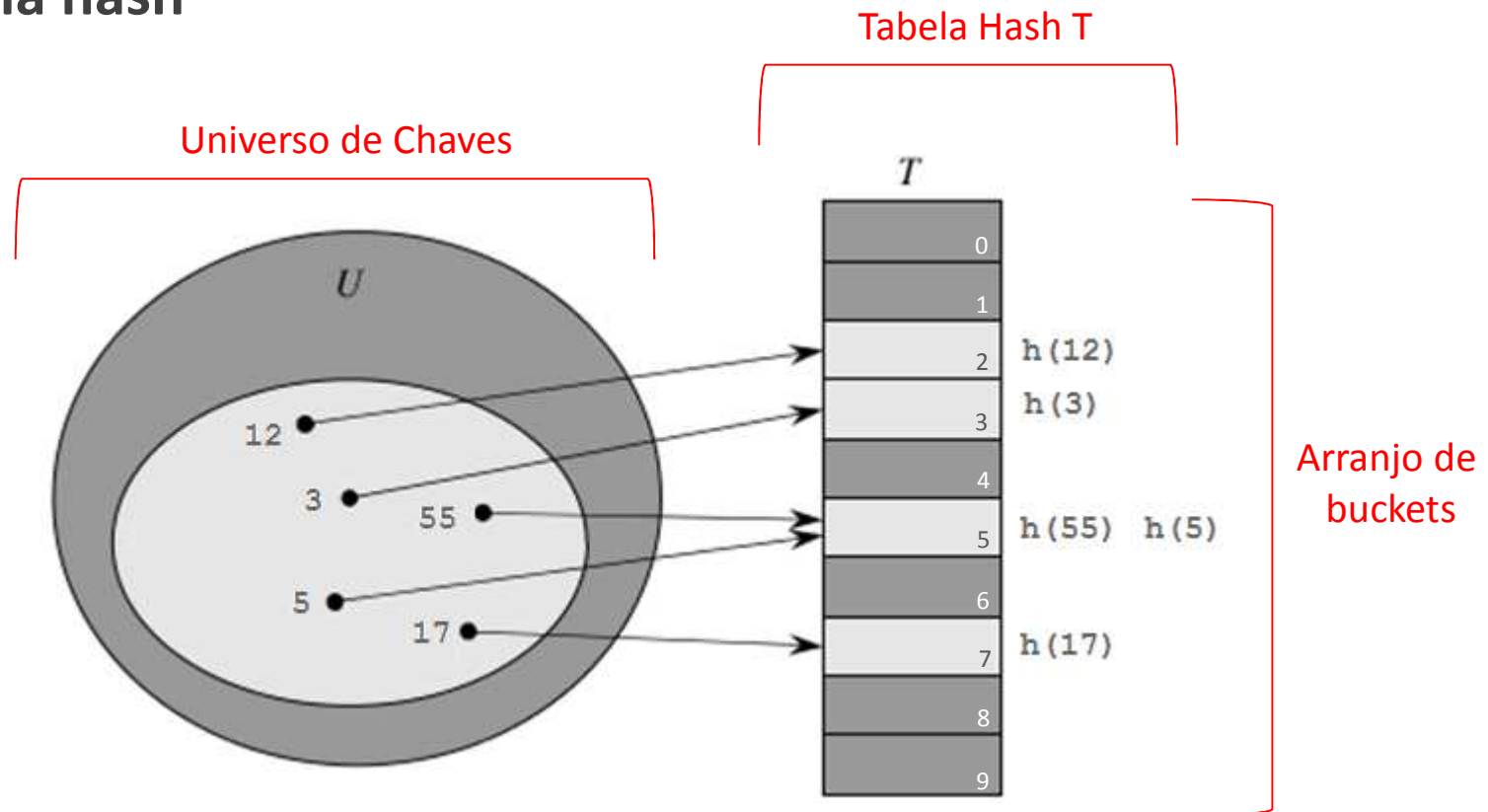
## ➤ Tabela hash





# Tabela Hash

## ➤ Tabela hash



→  $h(12) = 12 \bmod 10 = 2$

→  $h(3) = 3 \bmod 10 = 3$

→  $h(55) = 5 \bmod 10 = 5$

→  $h(5) = 5 \bmod 10 = 5$

→  $h(17) = 17 \bmod 10 = 7$



- **Função hash**
  - Uma função hash  $h$  deve ser determinística, no sentido de que uma dada entrada  $k$  sempre deve produzir a mesma saída  $h(k)$ .
  - Alguns tipos comuns de funções hash:
    - Método da divisão
    - Método da multiplicação



- **Função hash – Método da divisão**

- O método de divisão mapeia uma chave  $k$  para uma das  $m$  posições da tabela hash, tomando o resto da divisão de  $k$  por  $m$ :

$$H(k) = k \bmod m$$

- Onde:
  - $k$  – representa a chave do elemento manipulado.
  - $m$  – indica o tamanho do arranjo de buckets.



- **Função hash – Método de multiplicação**
- Esse método funciona em duas etapas:
  - Multiplicamos a chave  $k$  por uma constante  $A$  na faixa  $0 < A < 1$  e extraímos a parte fracionária de  $kA$ .
  - Em seguida, multiplicamos esse valor por  $m$  e tomamos o piso do resultado.

$$H(k) = \lfloor m ( \underbrace{kA \bmod 1}_{\text{Parte fracionária de } KA} ) \rfloor$$

Parte fracionária  
de  $KA$

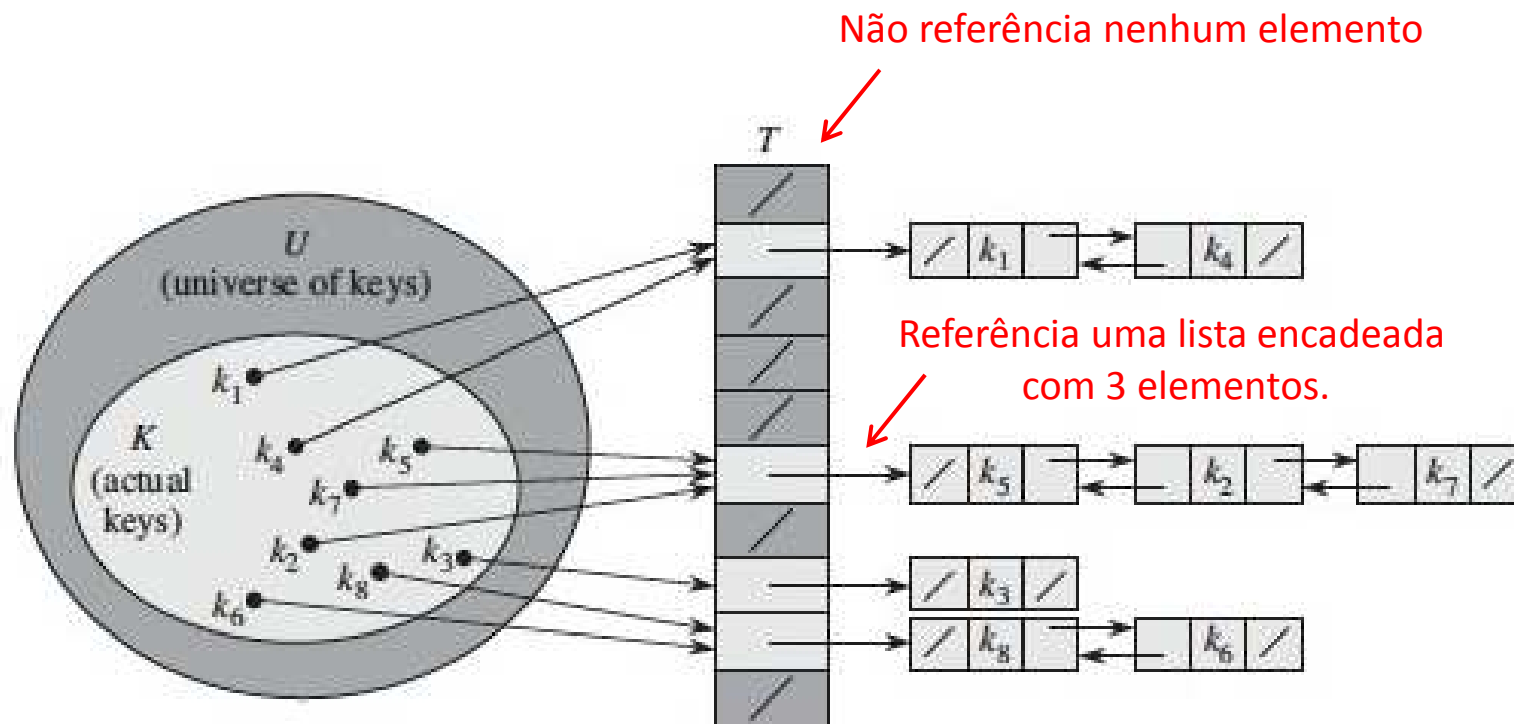
- Onde:
  - $k$  – representa a chave do elemento manipulado.
  - $m$  – indica o tamanho do arranjo de buckets.



- **Técnica de resolução de colisão: Encadeamento Separado**
  - Essa é a técnica mais simples onde todos os elementos resultantes do hash vão para a mesma posição em uma lista encadeada.
  - Cada entrada da tabela hash contém um ponteiro para o início de uma lista encadeada ou tem valor NULL.
  - Outras estruturas de dados podem ser usadas no lugar da lista encadeada.

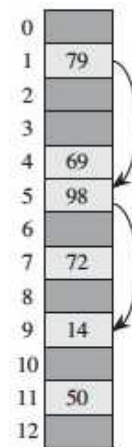


## ➤ Técnica de resolução de colisão: Encadeamento Separado





- **Técnica de resolução de colisão: Endereçamento Aberto**
  - Nessa técnica, todos o elementos ficam na própria tabela hash.
  - Cada entrada da tabela hash contém um elemento do conjunto dinâmico ou NULL.
  - A vantagem do endereçamento aberto é que ele evita por completo os ponteiros, consumindo menos memória.
  - Em vez de seguir ponteiros, calculamos a sequencia de posições a examinar.





## ➤ Função hash - Colisão

- A função hash menos eficiente seria aquela que mapeia todos os elementos para um mesmo bucket.
- A função hash mais eficiente seria aquela que distribui de maneira mais uniforme possível os elementos na tabela de espalhamento.
- Para ter uma função hash eficiente é necessário o conhecimento da natureza das chaves manipuladas.



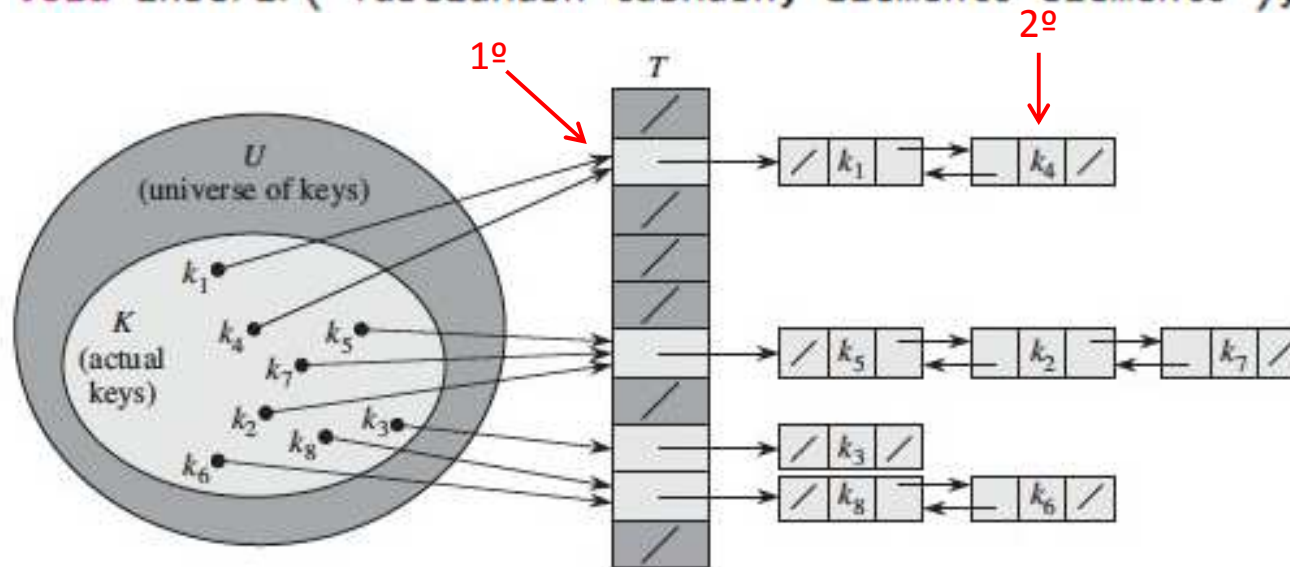
## ➤ Principais operações de uma tabela hash

```
1 public interface ITabelaHash {  
2     void inserir( TabelaHash tabHash, Elemento elemento );  
3     Elemento buscar( TabelaHash tabHash, Chave chave );  
4     Elemento remover ( TabelaHash tabHash, Elemento elemento );  
5 }
```



## ➤ Operação Inserção

```
void inserir( TabelaHash tabHash, Elemento elemento );
```



Passos a serem executados:

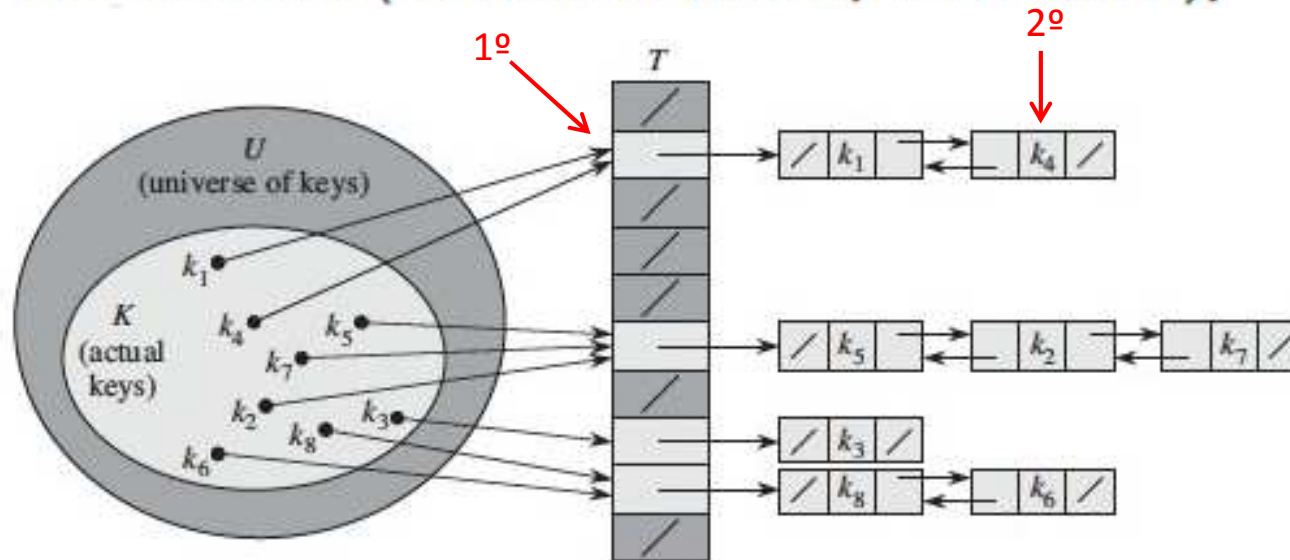
1º passo: Computar a função hash a partir da chave do elemento.

2º passo: Incluir elemento na lista encadeada.



## ➤ Operação Busca

Elemento buscar( TabelaHash tabHash, Chave chave );



Passos a serem executados:

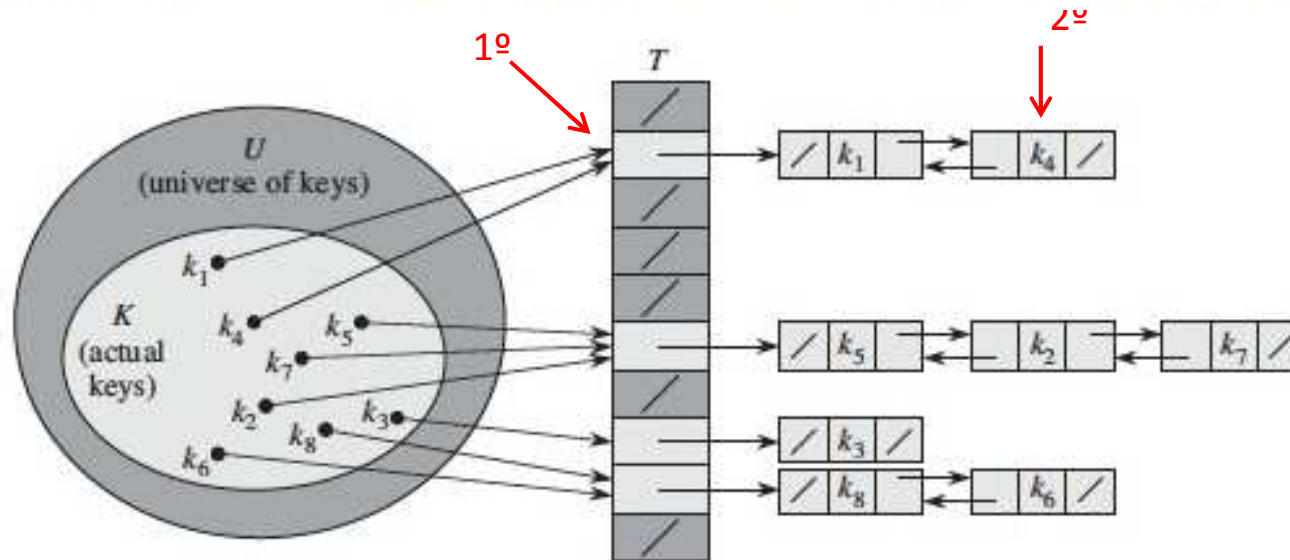
1º passo: Computar a função hash a partir da chave do elemento.

2º passo: Busca elemento na lista encadeada.



## ➤ Operação Remoção

Elemento remover ( TabelaHash tabHash, Elemento elemento );



Passos a serem executados:

1º passo: Computar a função hash a partir da chave do elemento.

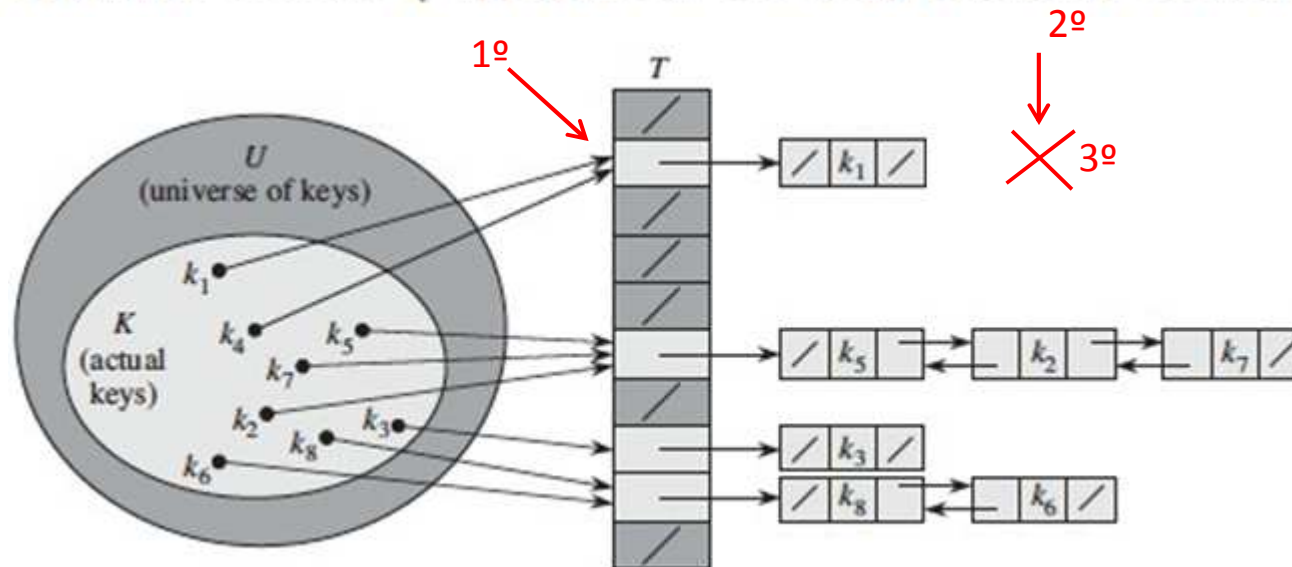
2º passo: Busca elemento na lista encadeada.

3º passo: Remove o elemento da lista encadeada.



## ➤ Operação Remoção

Elemento remover ( TabelaHash tabHash, Elemento elemento );



Passos a serem executados:

1º passo: Computar a função hash a partir da chave do elemento.

2º passo: Busca elemento na lista encadeada.

3º passo: Remove o elemento da lista encadeada.



## Questão 28

### **[2012 - CESPE - Banco da Amazônia - Técnico Administração de Dados]**

Com relação a métodos de pesquisa de dados, julgue os itens subsecutivos.

Listas encadeadas não são utilizadas na busca que emprega tabelas hash.

Certo      Errado



## Questão 28

### [2012 - CESPE - Banco da Amazônia - Técnico Administração de Dados]

Com relação a métodos de pesquisa de dados, julgue os itens subsecutivos.

Listas encadeadas não são utilizadas na busca que emprega tabelas hash.

Certo ➡ Errado



## Questão 29

### **[2012 - CESPE - Banco da Amazônia - Técnico Administração de Dados]**

A busca que utiliza uma tabela hash realiza comparação das chaves para encontrar a posição do elemento que está sendo buscado.

Certo      Errado



## Questão 29

### [2012 - CESPE - Banco da Amazônia - Técnico Administração de Dados]

A busca que utiliza uma tabela hash realiza comparação das chaves para encontrar a posição do elemento que está sendo buscado.

Certo ➡ Errado



## Questão 30

### **[2012 - CESPE - Banco da Amazônia - Técnico Administração de Dados]**

As colisões ocorrem na utilização de tabela hash porque várias chaves podem resultar na mesma posição.

Certo      Errado



## Questão 30

**[2012 - CESPE - Banco da Amazônia - Técnico Administração de Dados]**

As colisões ocorrem na utilização de tabela hash porque várias chaves podem resultar na mesma posição.

➡ Certo      Errado



# Questão 31

## [2011 - CESPE - FUB - Analista de Tecnologia da Informação]

Julgue os próximos itens em relação às estruturas de dados.

No uso de estruturas de transformação de chave (hashing), a solução de colisões usando encadeamento tem como principal característica o fato de nunca transbordar. Adicionalmente, o tempo de busca na lista ligada pode ser reduzido se uma lista duplamente encadeada for utilizada.

Certo      Errado



## Questão 31

### [2011 - CESPE - FUB - Analista de Tecnologia da Informação]

Julgue os próximos itens em relação às estruturas de dados.

No uso de estruturas de transformação de chave (hashing), a solução de colisões usando encadeamento tem como principal característica o fato de nunca transbordar. Adicionalmente, o tempo de busca na lista ligada pode ser reduzido se uma lista duplamente encadeada for utilizada.

Certo ➡ Errado





**28– Errado**

**31– Errado**

**29– Errado**

**30– Certo**



**FIM**

Rodrigo Adur  
rodrigoadurti@gmail.com