Estruturas de Dados

Filas

Universidade Federal do Rio Grande do Norte

Fila

Definição

Conjunto **ordenado de dados (ordem de entrada)** no qual os dados podem ser **inseridos** partir de uma extremidade **chamada fim da fila** e **removidos** a partida da outra extremidade chamada **início da fila**.

Fila

- Chamada de “queue” em inglês.
- É possível dizer que uma fila é um tipo abstrato de dados baseado no princípio de **F**irst-**I**n, **F**irst-**O**ut
- FIFO ou “o primeiro a entrar é o primeiro a sair”.
- Quando um elemento é inserido na fila ele é alocado no **final da fila**. O elemento que é removido da fila é sempre aquele que **está no início da fila**.
- A operação de **INSERIR** sobre uma fila é geralmente chamada de **ENQUEUE(ENFILEIRAR)** e possui como parâmetro o dado que será inserido na fila.
- A operação de **REMOVER** sobre uma fila é geralmente chamada de **DEQUEUE(DESENFILIERAR)** e não possui parâmetros.

Fila

- Se uma fila vazia sofre uma operação de remoção (dequeue) dizemos que a fila tem um **estouro negativo** (ou *queue underflow*)
- Se uma fila cheia sofre uma operação de inserção (enqueue) dizemos que a fila tem um **estouro positivo** (ou *queue overflow*)
- Ambas as situações devem ser tratadas como **erro**.

Principais Operações

Enqueue (dado)

Adiciona um dado à fila.

Dequeue()

Remove um dado da fila.

Front ()

Retorna o dado que está no inicio da fila.

isEmpty()

Verifica se uma fila está vazia.

Outras Operações

Clear ()

Remove todos os dados da fila.

Length()

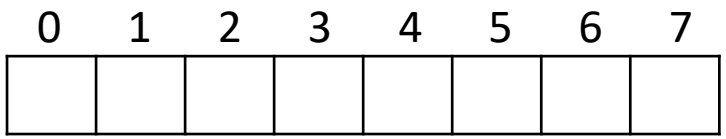
Retorna a quantidade de dados presentes na fila.

toString()

Retorna em formato de texto o conteúdo da fila.

Fila com array “infinito”

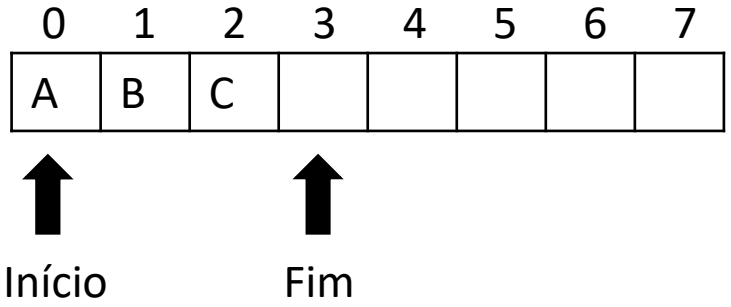
- É possível implementar uma fila de ***n*** elementos com um **array** para armazenar os dados e dois atributos, chamados *início e fim*, para armazenar o **índice** para o **primeiro e o último elemento da fila**, respectivamente.

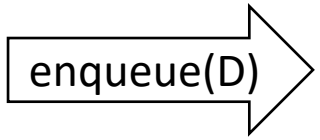


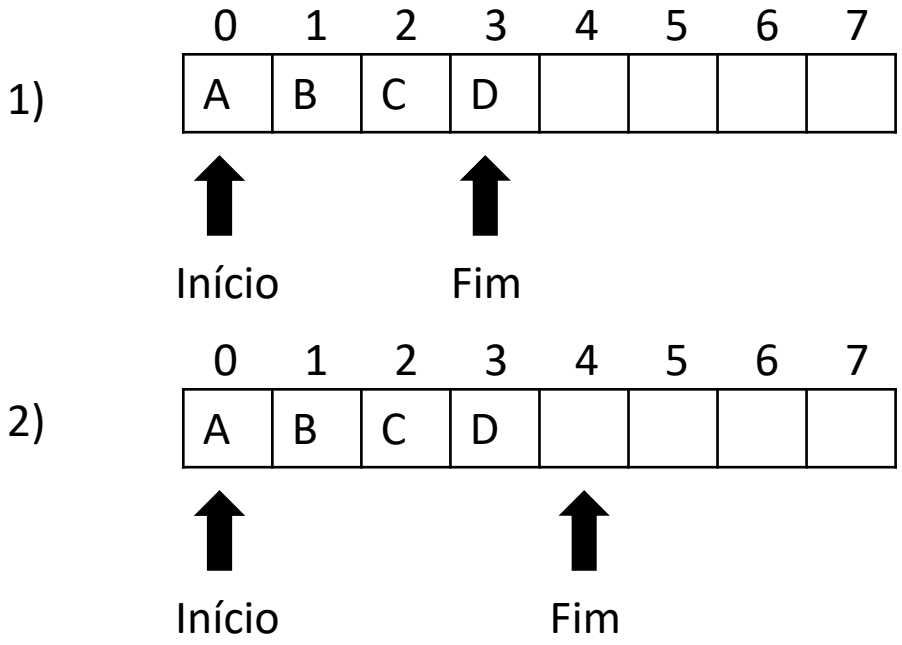
- Quando inicio == fim a fila não contém nenhum elemento e está **vazia.**
- Quando fim – início == n, podemos dizer que a fila está **cheia.**

Fila com array “infinito”

- Utilizando essa abordagem as operação de **enfileirar** pode ser feita em dois passos:
  1) Adicionar o dado inserido no índice apontado pelo fim da fila;
  2) Incrementar o fim da fila;

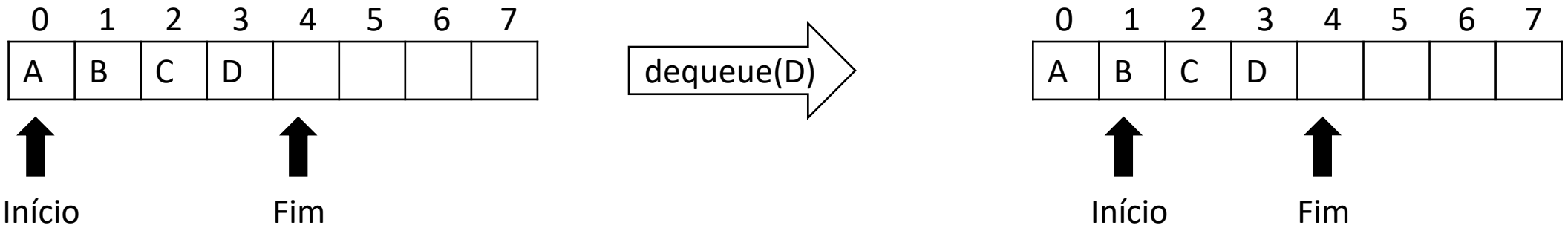






Fila com array “infinito”

- Utilizando essa abordagem as operação de **desenfileirar** pode ser feita em um passo
  1) Incrementar o início da fila;

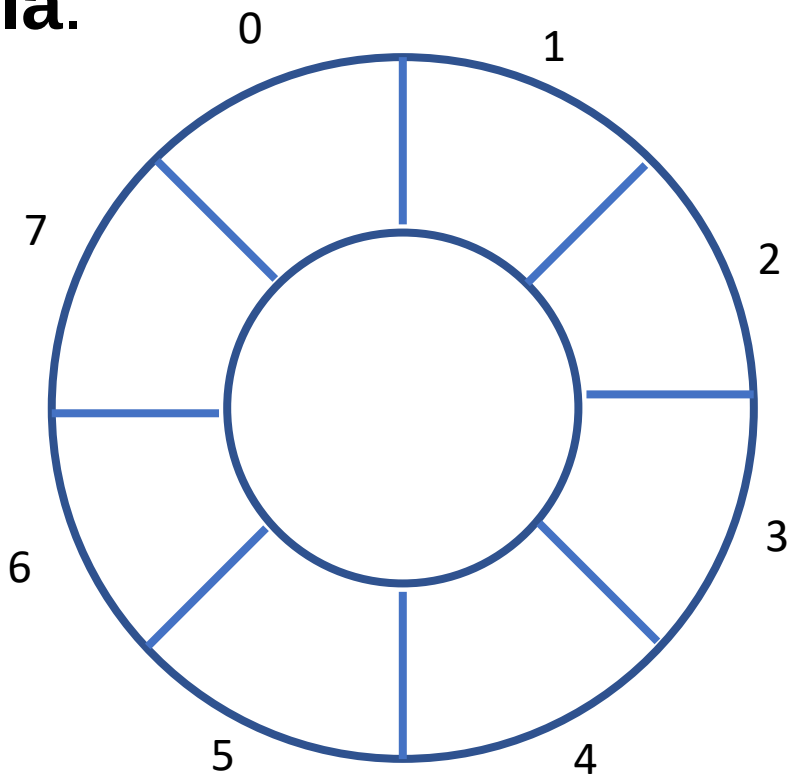


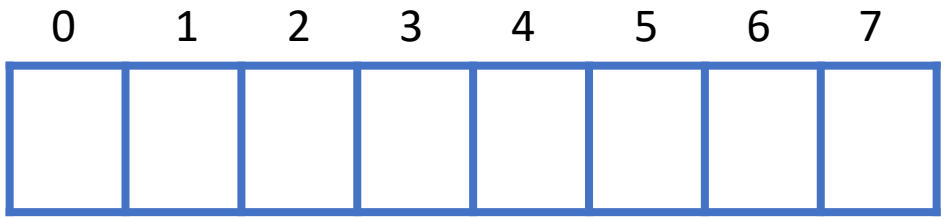
Fila com array “infinito”

- Essa abordagem é **eficiente em termos de tempo de execução** mas é **ineficiente em termos de uso de memória**.
- As posições do array que não estão mais na fila **permanecem alocadas na memória. Ou seja, um problema grave.**
- Uma solução mais eficiente é a implementação de uma **Fila com Array Circular ou Fila Circular**

Fila Circular

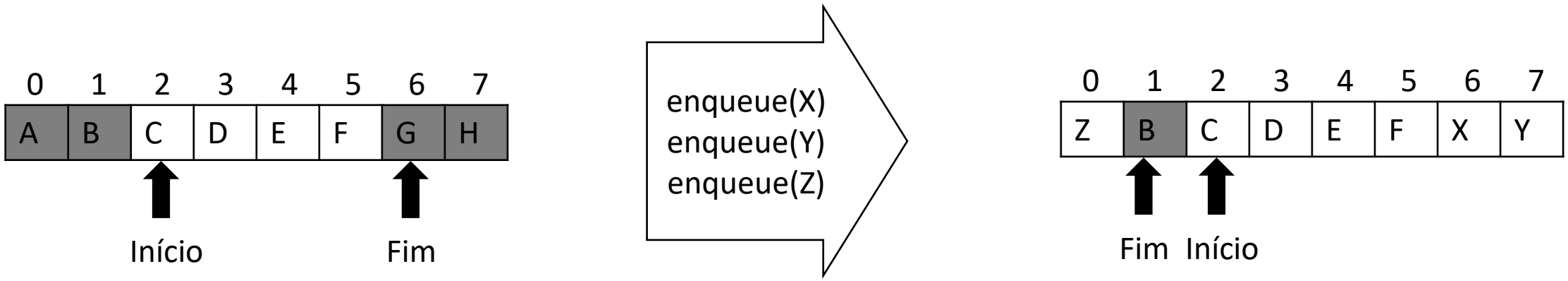
- Quando o atributo de **fim** atinge a **capacidade** da fila, será verificado se existem posições **desocupadas no início da fila**.





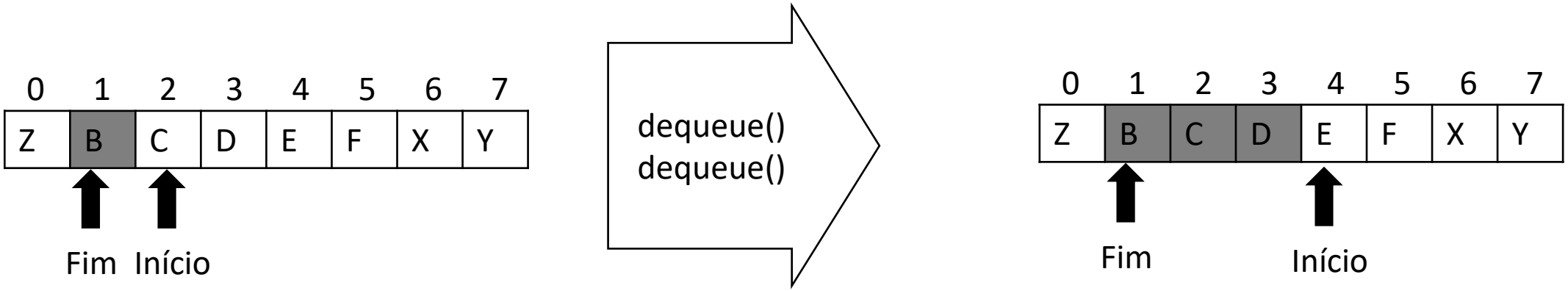
Fila Circular - Enfileirar

- Se a fila não está **cheia** então
  - Se **fim** da fila atingiu o valor da ***capacidade n*** então
    - fim = 0
  - Se não
    - fim = fim +1

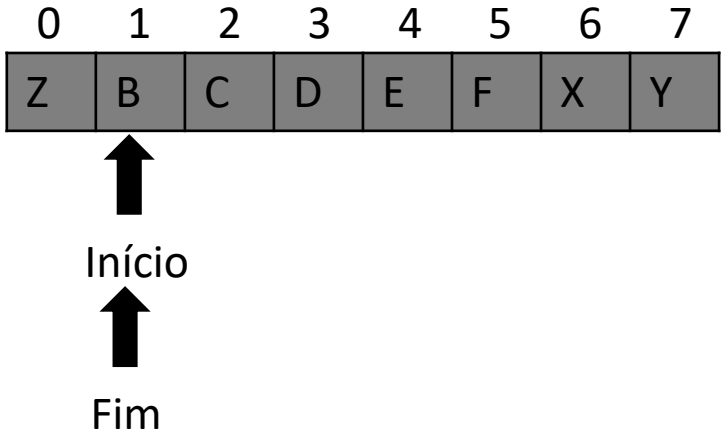


Fila Circular - Desenfileirar

- Se a fila não está **vazia** então
  - Se **inicio** da fila atingiu o valor da ***capacidade n*** então
    - inicio = 0
  - Se não
    - inicio = inicio +1

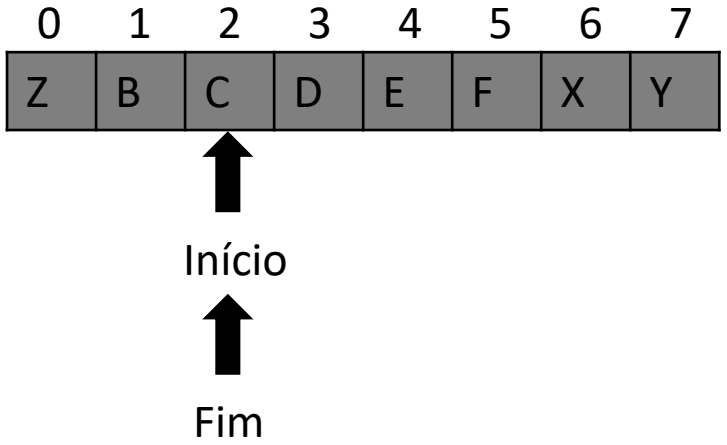


Implementação da Fila Circular



- Fila inicializada

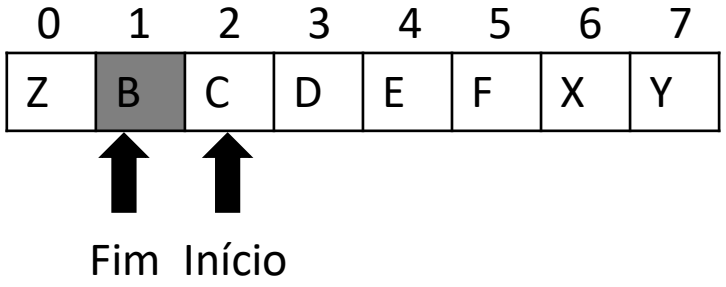
```
inicio = fim = 1
```



- Fila vazia

```
início == fim
```

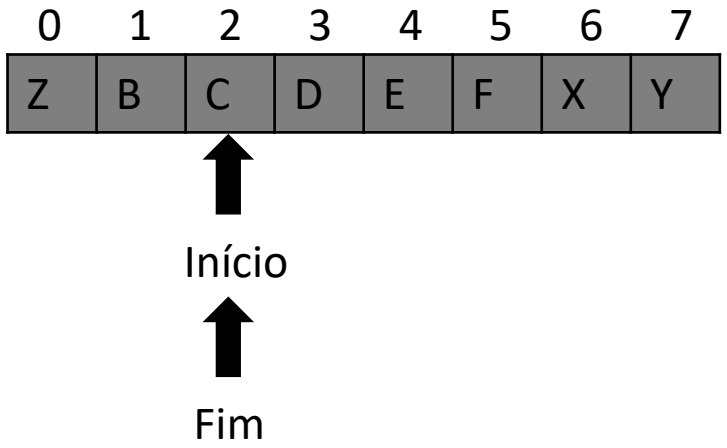
Implementação da Fila Circular



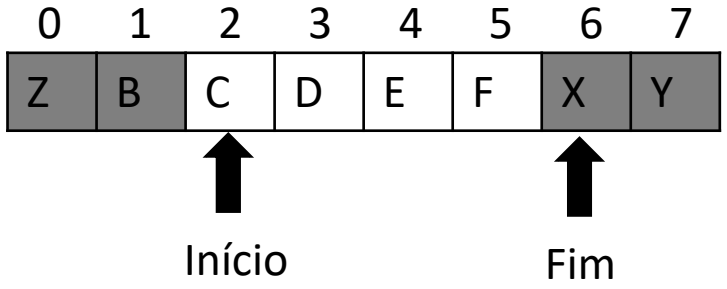
- Fila cheia

`início == fim +1`

Implementação da Fila Circular

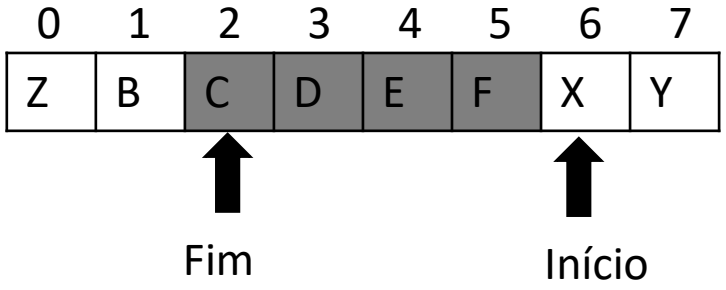


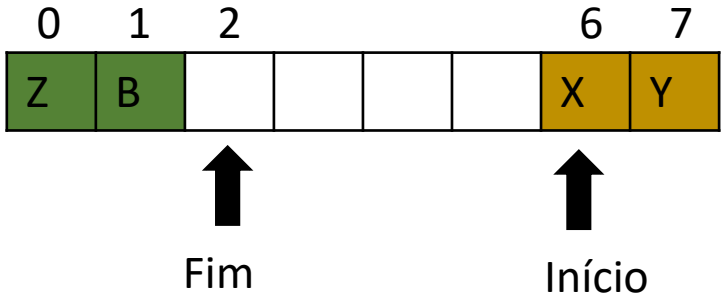
- Se a fila está vazia o tamanho é 0.



- Se início < fim o tamanho é fim – início

Implementação da Fila Circular





- Se inicio > fim o tamanho é a soma de:
  - `Fim – 1 + 0`
  - `Capacidade + 1 – inicio`
- Simplificando
  - Capacidade - inicio + fim + 1

Bibliografia Básica

- CORMEN, Thomas H et al. **Algoritmos: teoria e prática.** Rio de Janeiro: Elsevier, 2012. 926 p. ISBN: 9788535236996.
- ASCENCIO, Ana Fernanda Gomes. **Estruturas de dados: algoritmos, análise da complexidade e implementações em Java e C/C++.** São Paulo: Pearson, c2010. 432 p. ISBN: 9788576052216, 97885760581 6.
- PIVA JÚNIOR, Dilermando (et al). **Estrutura de dados e técnicas de programação**. 1. ed. Rio de Janeiro, RJ: Campus, 2014. 399 p. ISBN: 9788535274370.

Bibliografia Complementar

- FERRARI, Roberto et al. **Estruturas de dados com jogos**. 1. ed. Rio de Janeiro: Elsevier, 2014. 259p. ISBN: 9788535278040.
- GRONER, Loiane. **Estruturas de dados e algoritmos em Javascript**: aperfeiçoe suas habilidades conhecendo estruturas de dados e algoritmos clássicos em JavaScript. São Paulo: Novatec, 2017. 302 p. ISBN: 9788575225530.
- SZWARCFITER, Jayme Luiz; MARKENZON, Lilian. **Estruturas de dados e seus algoritmos**. 3. ed. Rio de Janeiro: LTC, 2010. xv, 302 p. ISBN: 9788521617501.
- GOODRICH, Michael T; TAMASSIA, Roberto. **Estruturas de dados e algoritmos em Java**. 5. ed. Porto Alegre: Bookman, 2013. xxii, 713 p. ISBN: 9788582600184.
- GUIMARÃES, Ângelo M. **Algoritmos e estruturas de dados**. LTC, 1994.