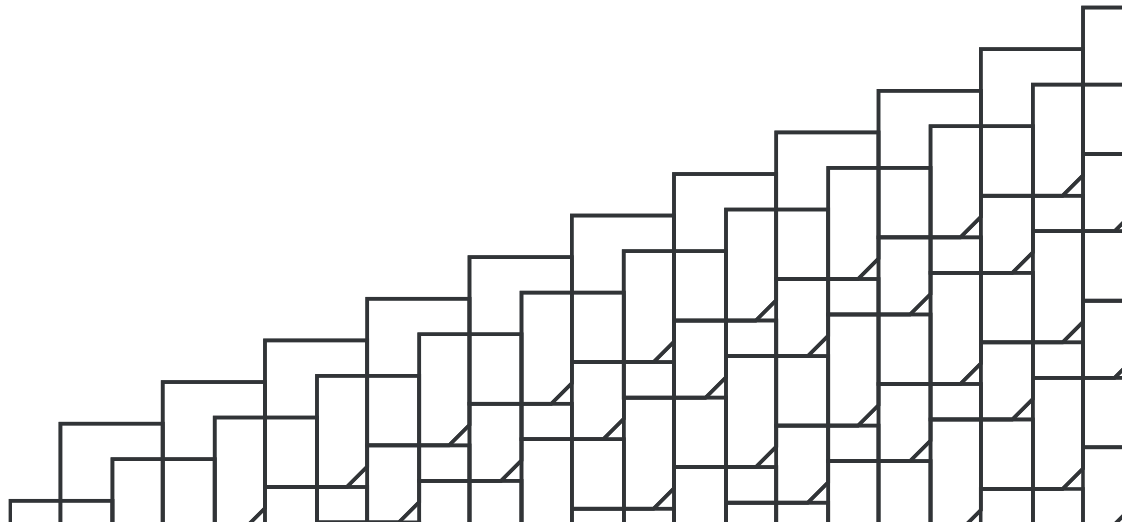
Universidade Presbiteriana Mackenzie | 150 anos 1870 - 2020

FCI 50 ANOS

Breve Revisão de Estrutura de Dados I

Estrutura de Dados II

Prof. Dr. Jean M. Laine



Estrutura de Dados

Uma forma particular de organizar e armazenar dados em um computador de modo que esses dados possam ser usados de forma eficiente

- Para cada estrutura de dados deve ser possível:
  - Implementá-la
  - Inserir itens
  - Remover itens
  - Obter itens
  - Percorrê-la
  - Etc
- Categorias de estruturas de dados: lineares e não lineares
- Quando for escolher uma estrutura de dados deve considerar as seguintes questões:
  - Quais operações você precisa realizar com mais frequência?
  - É necessário manter alguma ordenação dos itens?
  - Existe alguma restrição de espaço ou de tempo?
  - Posso escolher um estrutura diferente capaz de melhorar o desempenho do meu programa?
- Exemplos de EDs: array, stack, queue, linked list, tree, hash etc

ARRAY

Uma coleção de itens indexados

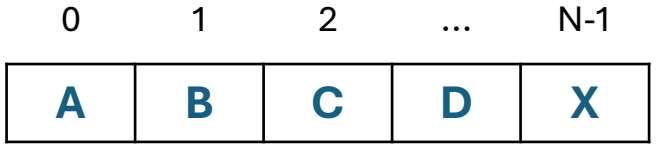
- ❑ Estrutura de dados linear, estática e indexada
- ❑ Armazena dados de um mesmo tipo (ED homogênea)
- ❑ Ocupa um espaço fixo em memória
- ❑ Pode ser usada para implementar outras estruturas de dados: pilha e fila, por exemplo
- ❑ Existem Arrays multidimensionais:
  - 2D
  - 3D

Complexidade de Tempo

(Pior Caso)

- Inserção: O(n)
- Remoção: O(n)
- Acesso: O(1)
- Busca: O(n)

Ilustração



Problemas e/ou Aplicações:

- Ordenação de itens (sorting)
- Busca binária
- Representação de imagens
- Criação de estruturas tabulares

STACK

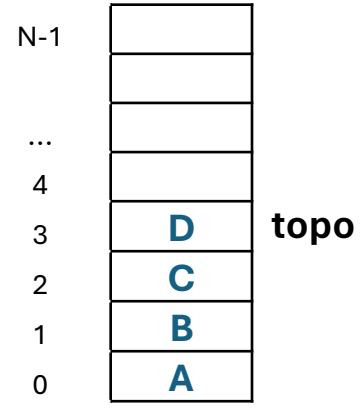
É uma estrutura de dados LIFO (Last in First Out)

- ❑ Estrutura de dados linear
- ❑ Pode ser implementada com array (estática) ou linked list (dinâmica)
- ❑ Armazena dados de um mesmo tipo (ED homogênea)
- ❑ Se implementada com linked list, a estrutura pode alterar seu espaço/tamanho sob demanda
- ❑ Dados só podem ser acessados, inseridos ou removidos do topo da pilha (stack)

Complexidade de Tempo
(Pior Caso)

- push: O(1)
- pop: O(1)
- peek: O(1)
- search: O(n)

Ilustração



Problemas e/ou Aplicações:

- Browser
- Editores
- Recursão
- Parsing

QUEUE

É uma estrutura de dados FIFO (First in First Out)

- ❑ Estrutura de dados linear
- ❑ Pode ser implementada com array (estática) ou linked list (dinâmica)
- ❑ Armazena dados de um mesmo tipo (ED homogênea)
- ❑ Se implementada com linked list, a estrutura pode alterar seu espaço/tamanho sob demanda
- ❑ Dados são inseridos no final da estrutura e removidos do início
- ❑ É comum adicionar marcadores para a posição inicial e final da estrutura
- ❑ Mantém os dados organizados de acordo com a ordem que foram inseridos

Complexidade de Tempo

(Pior Caso)

- enqueue: O(1)
- dequeue: O(1)
- search: O(n)

Ilustração

Problemas e/ou Aplicações:

- Spool de impressão
- Envio e recepção de msgs em rede
- Controle de pedidos e atendimento

LINKED LIST

É uma estrutura de dados composta por uma sequência de células ligadas ou encadeadas umas às outras

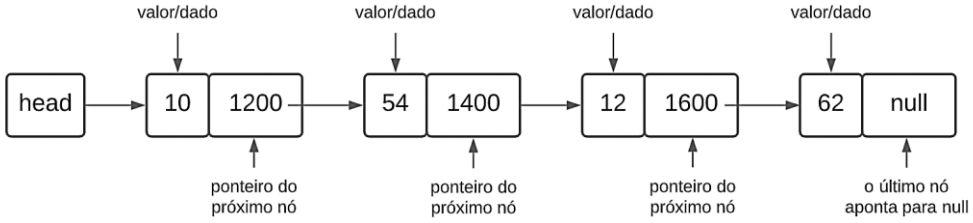
- ❑ Estrutura de dados linear e dinâmica
- ❑ Armazena dados de um mesmo tipo (ED homogênea)
- ❑ É comum adicionar marcadores para a posição inicial e final da estrutura
- ❑ Há diversos modelos de lista ligadas como lista encadeada simples, listas duplamente ligadas e listas encadeadas circulares.

Complexidade de Tempo

(Pior Caso)

- Inserção: O(1)
- Remoção: O(n)
- Acesso: O(n)
- Busca: O(n)

Ilustração



Problemas e/ou Aplicações:

- Implementação de TADs
- Alocação de arquivos
- Gerenciamento de memória
- Representação de matrizes esparsas

EXERCÍCIO TEÓRICO

ENADE 2021 - QUESTÃO 13

Considere que as variáveis **pilha** e **fila** correspondem, respectivamente, às estruturas de dados do tipo Pilha e Fila. Para testar as duas estruturas, um programador realizou a série de operações a seguir.

Pilha **pilha** = new Pilha();
Fila **fila** = new Fila();
pilha.push('A');
pilha.push('B');
pilha.push('C');
fila.enqueue(pilha.top());
fila.enqueue(pilha.top());
fila.enqueue('D');
pilha.push(fila.dequeue());
fila.enqueue(fila.dequeue());
fila.enqueue(pilha.pop());
pilha.push('E');
fila.enqueue('E');
pilha.pop();

Após essas operações, ao imprimir o conteúdo de pilha e fila, respectivamente, seria exibido:

A) **pilha**: topo → C → A → E.
fila: início → D → A → A → E.

B) **pilha**: topo →A.
fila: início → D → B → C → E.

C) **pilha**: topo → C → B → A.
fila: início → D → C → C → E.

D) **pilha**: topo → B → A.
fila: início → D → B → C → E.

E) **pilha**: topo → C → B → A.
fila: início → D → B → C → E.

ENADE 2021 - QUESTÃO 13

Considere que as variáveis **pilha** e **fila** correspondem, respectivamente, às estruturas de dados do tipo Pilha e Fila. Para testar as duas estruturas, um programador realizou a série de operações a seguir.

```
Pilha pilha = new Pilha();
Fila fila = new Fila();
pilha.push('A');
pilha.push('B');
pilha.push('C');
fila.enqueue(pilha.top());
fila.enqueue(pilha.top());
fila.enqueue('D');
pilha.push(fila.dequeue());
fila.enqueue(fila.dequeue());
fila.enqueue(pilha.pop());
pilha.push('E');
fila.enqueue('E');
pilha.pop();
```

Após essas operações, ao imprimir o conteúdo de pilha e fila, respectivamente, seria exibido:

A) **pilha**: topo → C → A → E.
fila: início → D → A → A → E.

B) **pilha**: topo →A.
fila: início → D → B → C → E.

C) **pilha**: topo → C → B → A.
fila: início → D → C → C → E.

D) **pilha**: topo → B → A.
fila: início → D → B → C → E.

E) **pilha**: topo → C → B → A.
fila: início → D → B → C → E.

GABARITO:

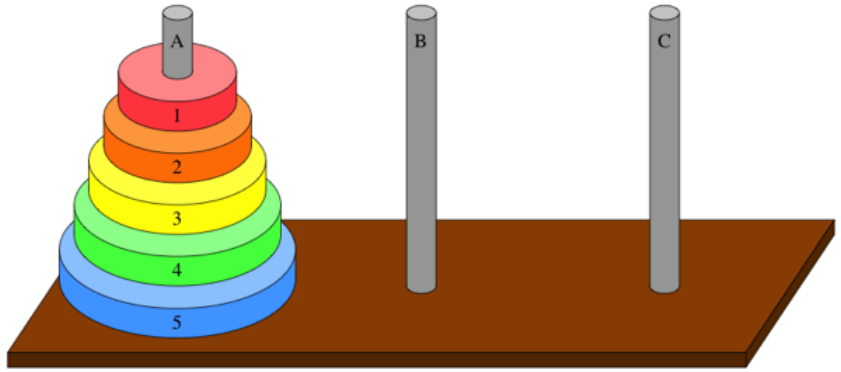
C) **pilha**: topo → C → B → A.
fila: início → D → C → C → E.

QUESTÃO 51 – ENADE 2005. No famoso jogo da Torre de Hanoi, é dada uma torre com discos de raios diferentes, empilhados por tamanho decrescente em um dos três pinos dados, como ilustra a figura. O objetivo do jogo é transportar-se toda a torre para um dos outros pinos, de acordo com as seguintes regras:

- apenas um disco pode ser deslocado por vez, e, em todo instante, todos os discos precisam estar em um dos três pinos;
- além disso, em nenhum momento, um disco pode ser colocado sobre um disco de raio menor que o dele;
- é claro que o terceiro pino pode ser usado como local temporário para os discos.

Imaginando que se tenha uma situação em que a torre inicial tenha um conjunto de 5 discos, qual o número mínimo de movimentações de discos que deverão ser realizadas para se atingir o objetivo do jogo?

a) 25
b) 28
c) 31
d) 34
e) 38

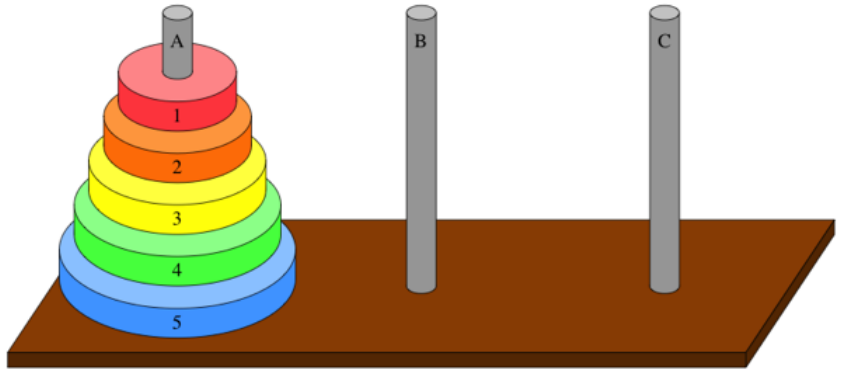


QUESTÃO 51 – ENADE 2005. No famoso jogo da Torre de Hanoi, é dada uma torre com discos de raios diferentes, empilhados por tamanho decrescente em um dos três pinos dados, como ilustra a figura. O objetivo do jogo é transportar-se toda a torre para um dos outros pinos, de acordo com as seguintes regras:

- apenas um disco pode ser deslocado por vez, e, em todo instante, todos os discos precisam estar em um dos três pinos;
- além disso, em nenhum momento, um disco pode ser colocado sobre um disco de raio menor que o dele;
- é claro que o terceiro pino pode ser usado como local temporário para os discos.

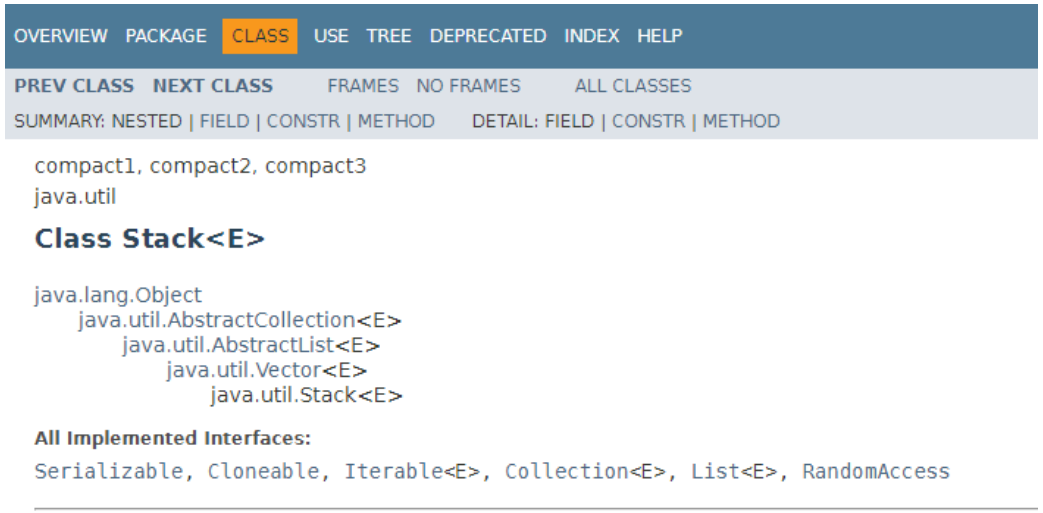
Imaginando que se tenha uma situação em que a torre inicial tenha um conjunto de 5 discos, qual o número mínimo de movimentações de discos que deverão ser realizadas para se atingir o objetivo do jogo?

a) 25
b) 28
c) 31
d) 34
e) 38



EXERCÍCIO PRÁTICO

- Agora, implemente em java uma solução para o quebra-cabeças Torre de Hanoi.
- Tente fazer um projeto em Java utilizando a classe “Stack”, nativa do Java (java.util.satck). Use o exemplo dado.



OVERVIEW PACKAGE CLASS USE TREE DEPRECATED INDEX HELP

PREV CLASS NEXT CLASS FRAMES NO FRAMES ALL CLASSES

SUMMARY: NESTED | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD

compact1, compact2, compact3

java.util

Class Stack<E>

java.lang.Object
  java.util.AbstractCollection<E>
    java.util.AbstractList<E>
      java.util.Vector<E>
        java.util.Stack<E>

All Implemented Interfaces:

Serializable, Cloneable, Iterable<E>, Collection<E>, List<E>, RandomAccess

ExemploStack.java

```java
import java.util.Stack;

public class ExemploStack {

    public static void main(String[] args) {
        // Criando uma pilha (stack) de strings
        Stack<String> pilha = new Stack<>();

        // Adicionando elementos à pilha usando push
        pilha.push("Estrutura de Dados");
        pilha.push("Análise de Algoritmos");
        pilha.push("Banco de Dados");

        // Imprimindo a pilha
        System.out.println("Pilha após push: " + pilha);

        // Obtendo o elemento no topo da pilha sem removê-lo usando peek
        String elementoTopo = pilha.peek();
        System.out.println("Elemento no topo da pilha: " + elementoTopo);

        // Removendo elementos da pilha usando pop
        String elementoRemovido = pilha.pop();
        System.out.println("Elemento removido da pilha: " + elementoRemovido);

        // Imprimindo a pilha após pop
        System.out.println("Pilha após pop: " + pilha);

        // Verificando se a pilha está vazia
        if (pilha.isEmpty()) {
            System.out.println("A pilha está vazia.");
        } else {
            System.out.println("A pilha não está vazia.");
        }
    }
}
```

Pesquise sobre:

- ❑ Class Stack
- ❑ Interface Queue <E>
- ❑ Class LinkedList <E>

presentes na linguagem Java.

OVERVIEW PACKAGE CLASS USE TREE DEPRECATED INDEX HELP

PREV CLASS NEXT CLASS FRAMES NO FRAMES ALL CLASSES

SUMMARY: NESTED | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD

compact1, compact2, compact3
java.util

Interface Queue<E>

Type Parameters:

`E - the type of elements held in this collection`

All Superinterfaces:

`Collection<E>, Iterable<E>`

All Known Subinterfaces:

`BlockingDeque<E>, BlockingQueue<E>, Deque<E>, TransferQueue<E>`

OVERVIEW PACKAGE CLASS USE TREE DEPRECATED INDEX HELP

PREV CLASS NEXT CLASS FRAMES NO FRAMES ALL CLASSES

SUMMARY: NESTED | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD

compact1, compact2, compact3
java.util

Class LinkedList<E>

java.lang.Object
  java.util.AbstractCollection<E>
    java.util.AbstractList<E>
      java.util.AbstractSequentialList<E>
        java.util.LinkedList<E>

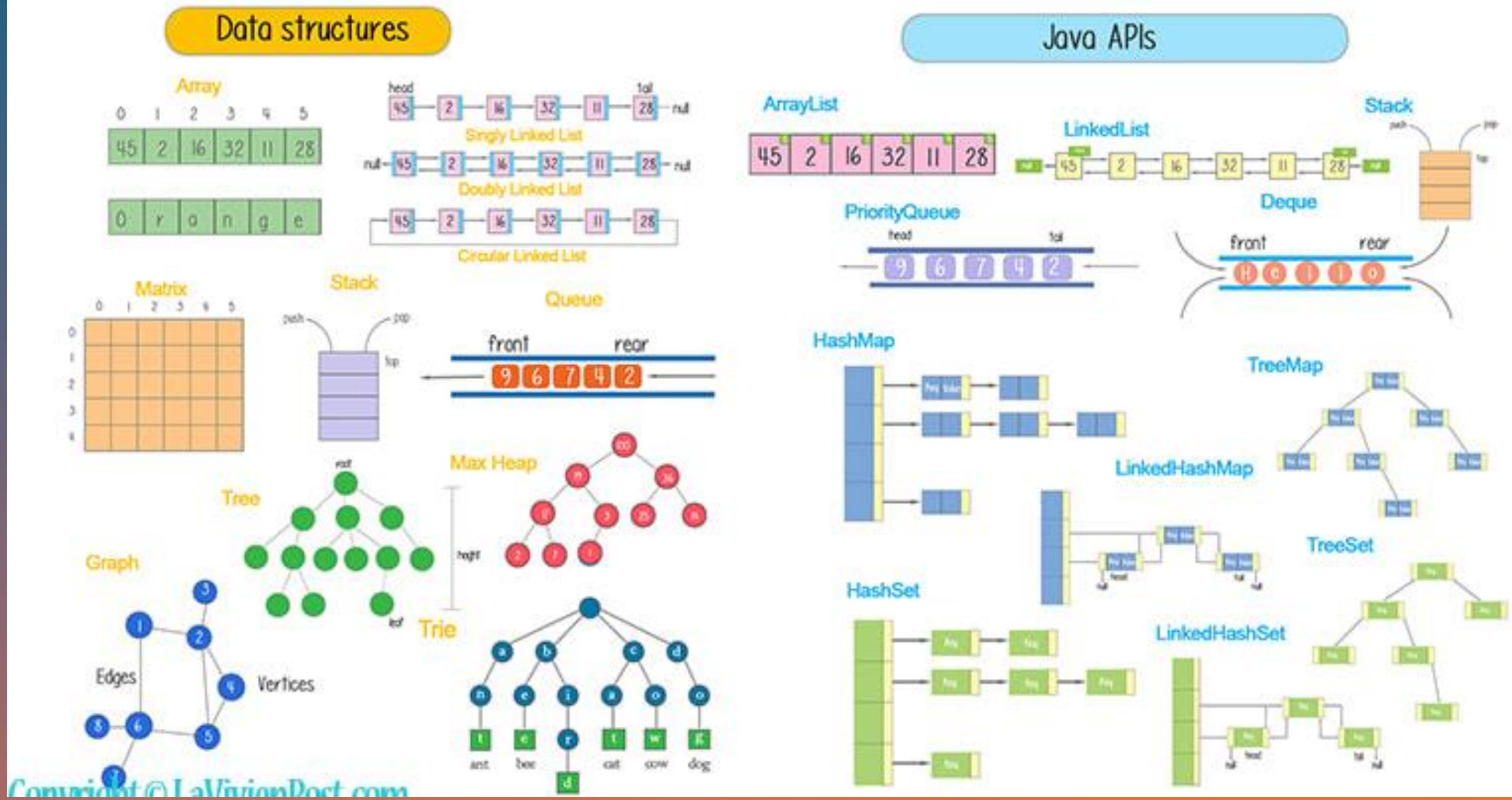
Type Parameters:

`E - the type of elements held in this collection`

All Implemented Interfaces:

`Serializable, Cloneable, Iterable<E>, Collection<E>, Deque<E>, List<E>, Queue<E>`

```
public class LinkedList<E>
extends AbstractSequentialList<E>
implements List<E>, Deque<E>, Cloneable, Serializable
```



Fonte: https://www.lavivienpost.com/data-structures-and-java-collections/


Universidade Presbiteriana
Mackenzie
150 anos
1870 - 2020


Faculdade de
Computação e Informática
FCI
50
ANOS