Estrutura de Dados

Algoritmos e Solução de Problemas

Profa. Eliane Oliveira Santiago

Conteúdo Programático (1)

1. Algoritmos e a Solução de Problemas.
   a. Introdução;
   b. Nivelamento de algoritmos;
   c. Conceito de análise de algoritmos;
   d. Revisão de arranjos. Representação linear de matrizes.

O que são algoritmos?

Um algoritmo é qualquer procedimento computacional bem definido que toma algum valor ou conjunto de valores como **entrada** e produz algum valor ou conjunto de valores como **saída**.

Uma ferramenta para resolver um problema computacional bem especificado.

O enunciado do problema especifica em termos gerais a entrada e a saída do problema.

Problema: ordenar o vetor V

Entrada:

V[]={31, 41, 59, 26, 41, 58}

Instância do problema

Entrada que satisfaz a quaisquer restrições impostas no enunciado do problema.

Saída esperada:

V[]={26, 31, 41, 41, 58, 59}

Um algoritmo é dito CORRETO se gerar uma saída correta para cada instância do problema.

Algoritmos eficientes.

Medida de eficiência: *velocidade (quanto tempo um algoritmo demora para produzir seu resultado).*

Problemas de instância

Encontrar a média dos elementos de um vetor A[1 . . n] de números.

Uma das instâncias deste problema consiste em encontrar a média dos elementos do vetor A[]= {876, -145, 323, 112, 221}.

Outra instância deste problema consiste em encontrar a média dos elementos do vetor B[]={100, 120, 350, 440, 50, 600, 200}.

O tamanho de uma instância de um problema é a quantidade de dados necessária para descrever a instância. Neste caso, diz-se que

$$A=|5| \text{ e } B=|7|.$$

Estrutura de Dados

É um meio para armazenar e organizar dados com o objetivo de facilitar o acesso e as modificações.

Nenhuma ED única funciona bem para todos os propósitos. Portanto, é importante conhecer os pontos fortes e as limitações de várias delas.

Discussão

1. Quais estruturas de dados você conhece?
2. Discuta seus pontos fortes e suas limitações.

Revisão de arranjos (array). Vetores e Matrizes

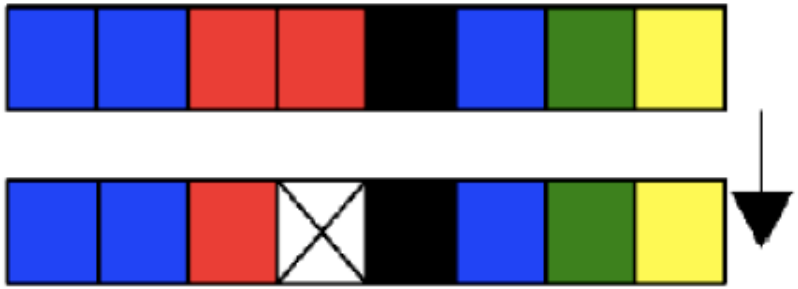
Em computação um Vetor (Array) ou arranjo é o nome de uma **matriz** unidimensional considerada a mais simples das **estruturas de dados.**

Vetor (array unidimensional) é uma variável que armazena várias variáveis do mesmo tipo. No problema apresentado anteriormente, nós podemos utilizar um vetor de 50 posições para armazenar os nomes dos 50 alunos.

Matriz (array multidimensional) é um vetor de vetores. Imagine uma matriz para armazenar as 4 notas de 50 alunos. Ou seja, um vetor de 50 posições, e em cada posição do vetor, há outro vetor com 4 posições. Isso é uma matriz.

Cada item do vetor (ou matriz) é acessado por um (ou dois) número(s) chamado(s) de índice(s).

Limitações dos Arrays



Retire a quarta Conta

conta[3] = **null**;

Supondo que os dados armazenados representem contas, o que acontece quando precisarmos inserir uma nova conta no banco?

- Precisaremos procurar por um espaço vazio?
- Guardaremos em alguma estrutura de dados externa, as posições vazias?
- E se não houver espaço vazio?
- Teríamos de criar um array maior e copiar os dados do antigo para ele?

Declaração de Matriz

```
int[][] M1 = new int[2][3];
```

	0	1	2
0	11	12	13
1	21	22	23

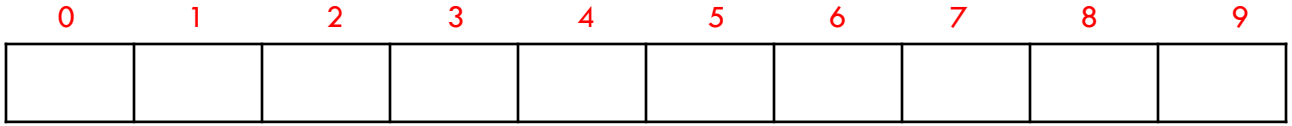
```
int[][] M1 = new int[2][3];
M1[0][0] = 11;
M1[0][1] = 12;
M1[0][2] = 13;

M1[1][0] = 21;
M1[1][1] = 22;
M1[1][2] = 23;
```

Declaração de Vetor

```
int[] array = new int[10];  //declaração do vetor

for(int coluna=0; coluna<array.length; coluna++){
   array[coluna] = coluna;
   System.out.println("array[" + coluna + "] = " + array[coluna]);
}
```



Carga de dados num Vetor

A entrada de dados poderá ser feita de várias formas: digitação, atribuição (cálculos, sorteio randômico), etc. Em qualquer um dos casos será necessária a construção de um laço `for` (ou outro), que faça a variação do índice de 0 até o final do *array* (`array.length`).

Veja este exemplo:

```
public class Teste1 {

     public static void main (String[] args) {

          int[] nros;
          nros = new int[10];

          System.out.println("Serão sorteados 10 números e armazenados no vetor");

          for (int i=0;i<nros.length;i++){ nros[i]=(int)(100*Math.random());
          }

          System.out.println("Veja os números sorteados");

          for (int i =0;i<nros.length;i++){ System.out.print(nros[i] + " ");
          }
     }
}
```

Tipos Primitivos e Tipos Abstratos de Dados

Primitivos

- int
- boolean
- double
- float
- char

Abstratos

- Integer
- Boolean
- Double
- Float
- String
- Estudante
- Professor
- Pessoa
- Pizza

Tipos Primitivos de Dados

- Inteiro

```
int total = 0;
```

- Real

```
double total = 10.0;   float salario = 1242.50;
```

- Caracter

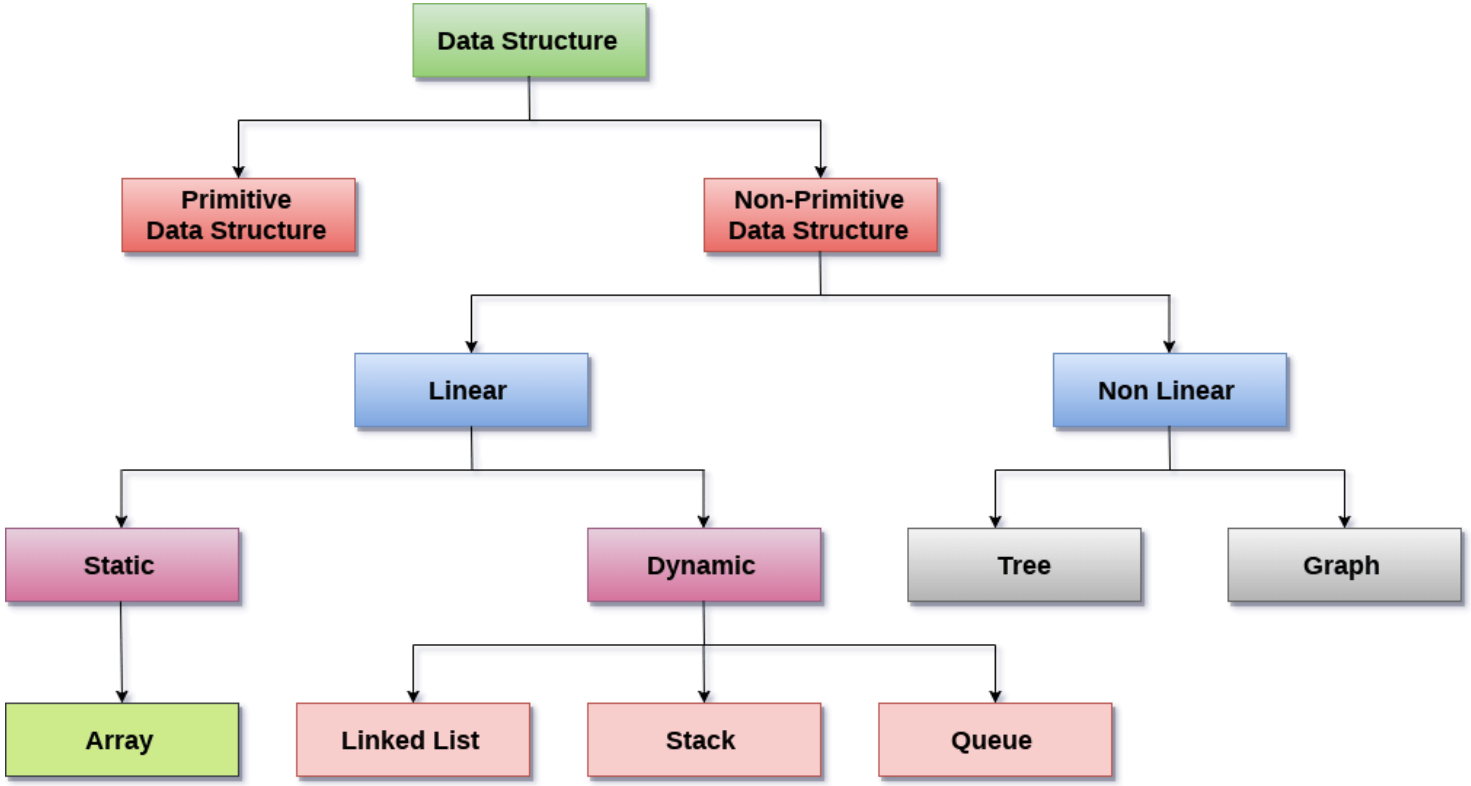
```
char letra = 'A';
```

- Lógico

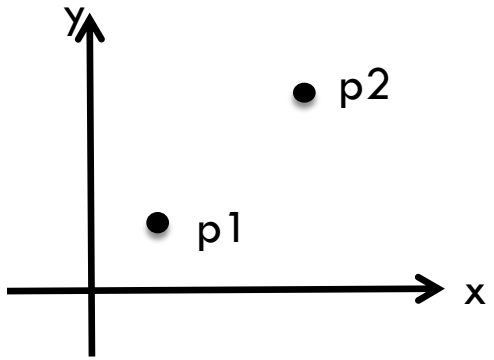
Operadores lógicos

&&	AND
||	OR
!	NOT

Estruturas de Dados Não-Primitivas
Tipos Abstratos de Dados



Uma classe é um TAD
A classe Ponto



Ponto
-x: int -y: int
+Ponto(x: int, y: int) +mover(novo_x: int, novo_y: int): void +obterX(): int +obterY(): int +ehIgual(p: Ponto):boolean

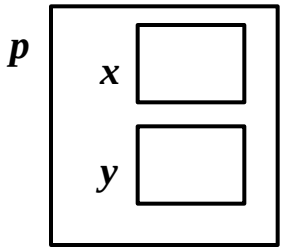
→ nome da classe

→ atributos (campos de dados)

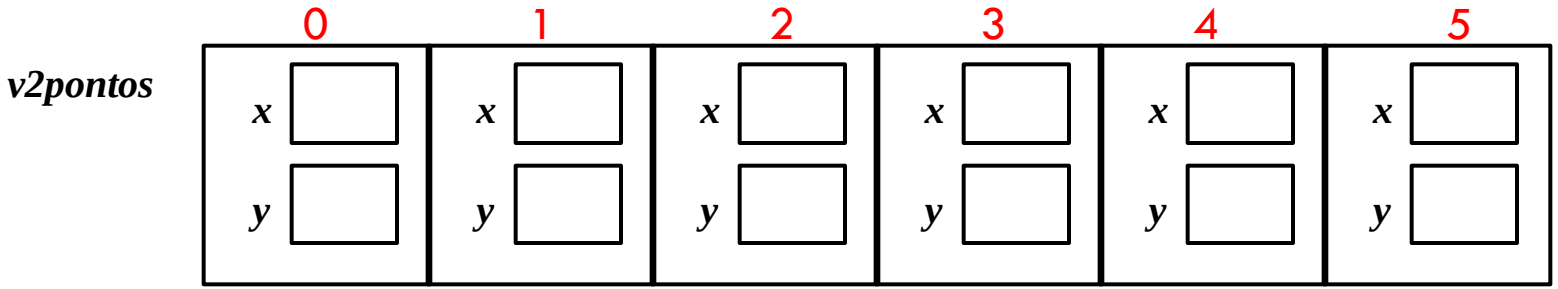
O que os objetos da classe SABEM sobre si.

→ métodos (comportamentos)

O que os objetos da classe FAZEM.

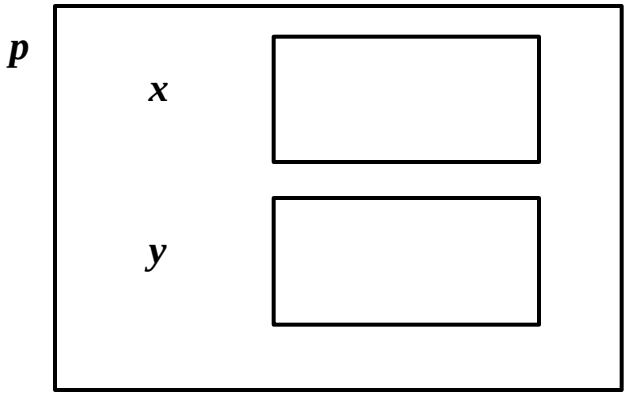


Ponto v2pontos = new Ponto[6];



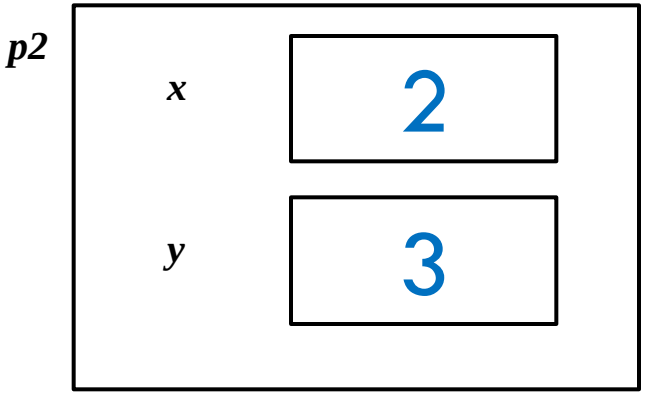
A estrutura do Ponto p

Ponto p;



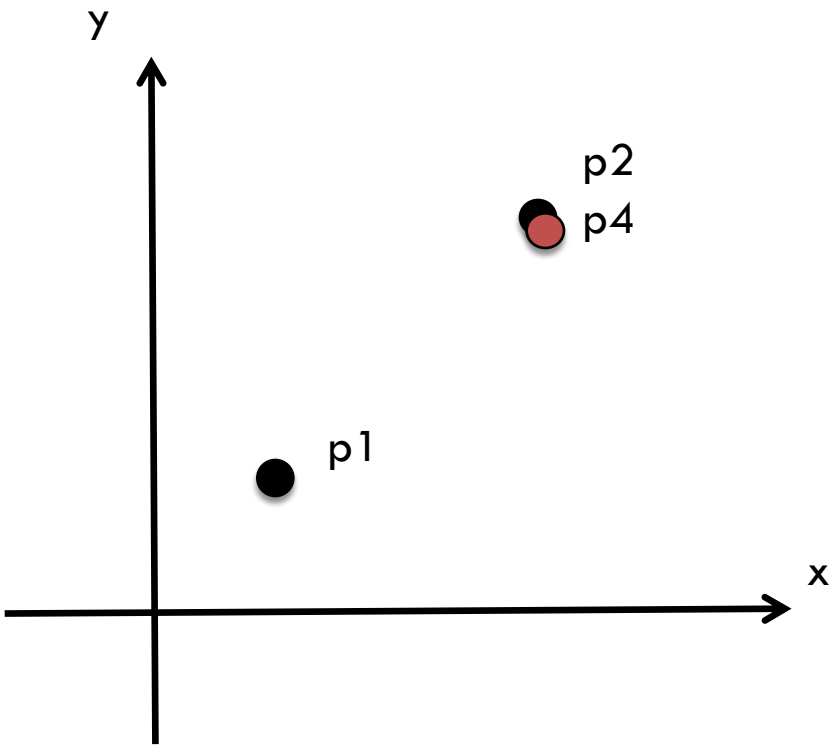
Variável p do tipo Ponto

Ponto p2 = new Ponto(2,3);



Objeto p2 do tipo Ponto

```
Ponto p1 = new Ponto(1,1);
Ponto p2 = new Ponto(2,3);
Ponto p3 = p1;
Ponto p4 = new Ponto(2,3);
```



Análise de Algoritmo

Conceito de análise de algoritmos e objetivos

Em ciência da computação, a **análise de algoritmos** tem como função determinar os recursos necessários para executar um dado **algoritmo**. A maior parte dos **algoritmos** são pensados para trabalhar com entradas (inputs) de tamanho arbitrário.

(Wikipedia)

Objetivos:

- estudar problemas computacionais recorrentes;
- provar que o algoritmo está correto; e
- estimar o tempo que a execução do algoritmo consome.

Perguntas que a análise de algoritmos tenta responder

A análise de algoritmos estuda a correção e o desempenho de (ou eficiência dos) algoritmos.

A análise de algoritmos procura respostas para perguntas do seguinte tipo:

- **Este algoritmo resolve o meu problema?**
- **Quanto tempo o algoritmo consome para processar uma 'entrada' de tamanho n?**

Eficiência do algoritmo: consumo de tempo para gerar a saída

Um algoritmo resolve um problema se, ao receber a descrição de uma instância do problema, devolve uma solução da instância (ou informa que a instância não tem solução).

Para cada instância do problema, o algoritmo consome uma quantidade de tempo diferente.

Digamos que o algoritmo consome $T(I)$ unidades de tempo para processar a instância I. A relação entre $T(I)$ e o tamanho de I dá uma medida da eficiência do algoritmo.

“Pior Caso” e “Melhor Caso”

Um problema tem muitas instâncias diferentes de um mesmo tamanho. Isto exige a introdução dos conceitos de “pior caso” e “melhor caso”.

Vamos criar um algoritmo para pesquisar se o valor 32 existe no vetor de 10 posições e, se encontrar, escreva a posição encontrada.

V	20	15	30	80	50	45	75	32	10	5

Solução não-ótima

```
public int pesquisar(int[] array, int valor){
        int posicao = 0;
        for(int i = o; i < array.length; i++){
                if(array[i] == valor){
                        posicao = i;
                }
        }
        return posicao;
}
```

“Pior Caso” e “Melhor Caso”

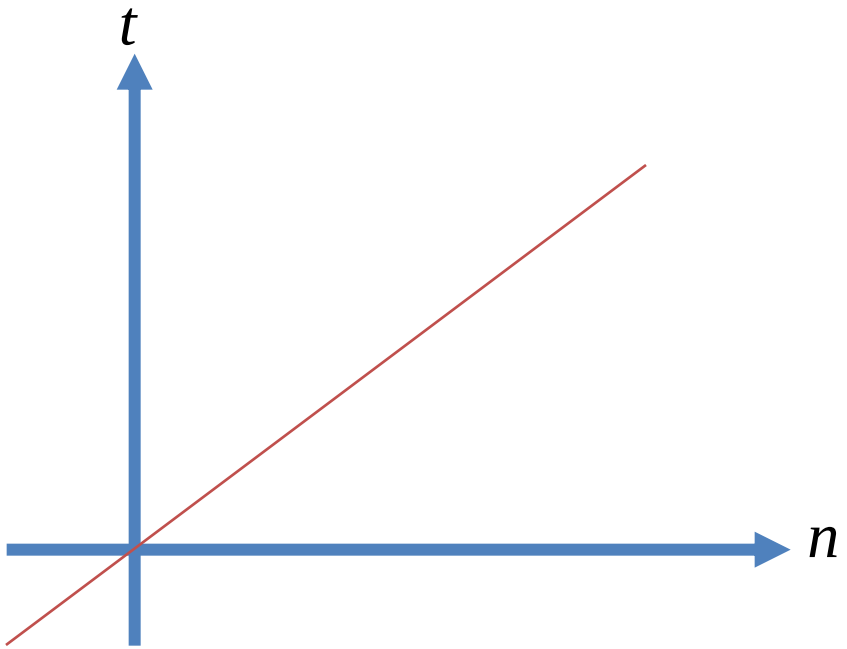
```
public static int pesquisar(int v[], int num){
    int resposta = -1;
    int n = v.length;
    for(int i=0; i<n; i++){
        if(v[i]==num){
          resposta = i;
          n=i;
        }
    }
    return resposta;
}
```

V	20	15	30	80	50	45	75	32	10	5

- $O(1) \Leftrightarrow$ tempo constante
- $O(lg(n)) \Leftrightarrow$ logarítmica
- $O(n) \Leftrightarrow$ tempo linear
- $O(n^2) \Leftrightarrow$ tempo quadrático
- $O(n^3) \Leftrightarrow$ tempo cúbico
- $O(2^n) \Leftrightarrow$ exponencial

Gráfico da complexidade linear

N é o tamanho da entrada



“Pior Caso” e “Melhor Caso”

```
int pesquisar(int v[], int num){
    int resposta = -1;
    int n = v.length;
    for(int i=0; i<n); i++){
        if(v[i]==num){
            resposta = i;
            n=i;
        }
    }
    return resposta;
```

Melhor Caso: quanto o número pesquisado está na primeira posição do vetor.

Pior Caso: quando o número pesquisado está na última posição do vetor.

V	20	15	30	80	50	45	75	32	10	5

Representação linear de matrizes

- int V1[5] = {1, 2, 3, 4, 5};

- int M1[1][5] = {{1, 2, 3, 4, 5}};
- int M2[2][5] = {{1, 2, 3, 4, 5}, {6, 7, 8, 9, 10}};

Exercícios

Dada a classe abaixo, criar as seguintes operações:

```
public class MinhaPrimeiraED {
        private Object[] objetos = new Object[10];
        public int totalDeObjetos = 0;

        public void adiciona(int posicao, Object objeto){..}
        public void adiciona(Object objeto){..}
        private boolean posicaoOcupada(int posicao){..}
        private boolean posicaoValida(int posicao){..}
        public void remove(int posicao){..}
        public boolean contem(Object objeto){..}
        public int tamanho(){..}
}
```

Bibliografias

BÁSICA

- PEREIRA, Silvio do Lago. - Estruturas de Dados Fundamentais. - Ed. Érica, 2000.
- EDELWEISS, Nina; GALANTE, Renata. - Estruturas de Dados - Livros Didáticos Informática Ufrgs, V.18 – Ed. Bookman Companhia, 2009.
- SENAC Estruturas de Dados - 2001 - SENAC RIO.

COMPLEMENTAR

- CORMEN, T. LEISERSON, C.E., RIVEST, R. L. STEIN, C. Algoritmos – teoria e prática. – Ed. Campus, 2001.
- VELOSO, P., e outros. - Estruturas de Dados. - Ed. Campus.
- LAFORE, Robert - Estruturas de Dados e Algoritmos em Java. – Ed. Ciência Moderna, 2005.
- LORENZI, Fabiana; MATTOS, Patricia Noll de. ;
- CARVALHO, Tanisi Pereira de. - Estruturas de Dados. – Ed. Thomson Pioneira, 2006.
- WIRTH, Niklaus - Algoritmos e Estruturas de Dados – Ed. LTC, 1989. MORAES, Celso Roberto - Estruturas de Dados e Algoritmos – Ed. Futura, 2003.
- BACKES, A. Linguagem C Completa e Descomplicada. Campus, 2012.