INE5603 Introdução à POO

Prof. A. G. Silva

16 de outubro de 2017

Processamentos de vetor

- Média e desvio padrão
- Máximo (mínimo)
- Argmax
- Ordenação
  - Bolha
  - Seleção
  - Inserção
- Métodos de busca – busca binária

Algoritmos em vetor – média e desvio padrão (I)

```
import java.lang.Math;

class Vetor {
    protected double v[];
    public Vetor(double vet[]) {
        v = new double[vet.length];
        for (int i=0; i<vet.length; i++)
            v[i] = vet[i];
    }
    public double media() {
        double m = 0.0;
        for (int i=0; i<v.length; i++)
            m = m + v[i];
        return m / v.length;
    }
    public double desvioPadrao() {
        double m = media();
        double s = 0.0;
        for (int i=0; i<v.length; i++)
            s = s + Math.pow(v[i]-m, 2);
        return Math.sqrt(s / v.length);
    }
}
```

Algoritmos em vetor – média e desvio padrão (II)

```java
public class VetorEx {
    public static void main(String[] args) {
        double vet[] = {2.5, 3.0, 2.8, 3.1, 2.4};
        Vetor V;
        V = new Vetor(vet);
        System.out.println("Media: " + V.media());
        System.out.println("Desvio padrao: " + V.desvioPadrao());
    }
}
```

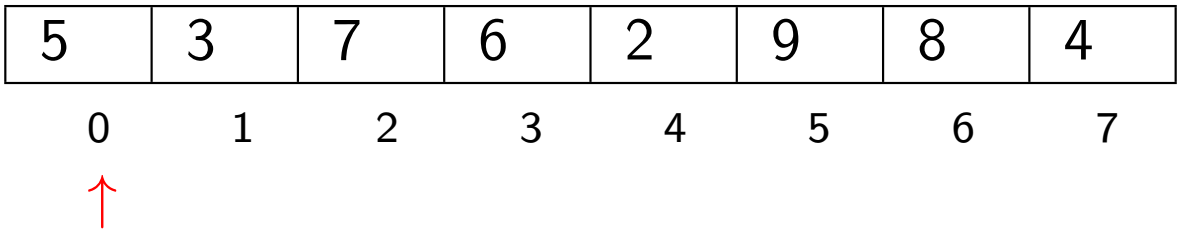
```
$ javac VetorEx.java
$ java VetorEx
Media: 2.7600000000000002
Desvio padrao: 0.27276363393971714
```

- https://www.inf.ufsc.br/~alexandre.silva/courses/16s1/ine5603/codigos/VetorEx.java

Algoritmos em vetor – valor máximo (I)

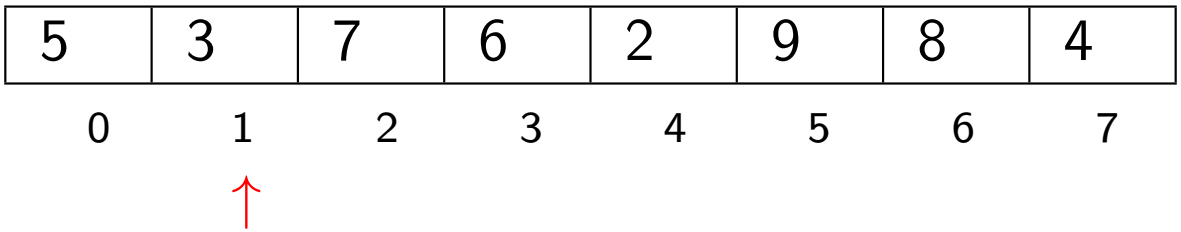
- Tomar o primeiro elemento (posição `i=0`) como o maior, guardando seu valor em uma variável `máximo`
- Verificar se o valor do elemento do vetor da posição seguinte (`i=i+1`) é maior que `máximo`
  - Se for, substituir `máximo` pelo valor do elemento desta posição
  - Se não for, verificar o elemento da posição seguinte (`i=i+1`)
  - Repetir até que não haja mais elemento a verificar (após atingir o comprimento total do vetor)
- A variável `máximo` conterá o maior valor do vetor

Algoritmos em vetor – valor máximo (exemplo)



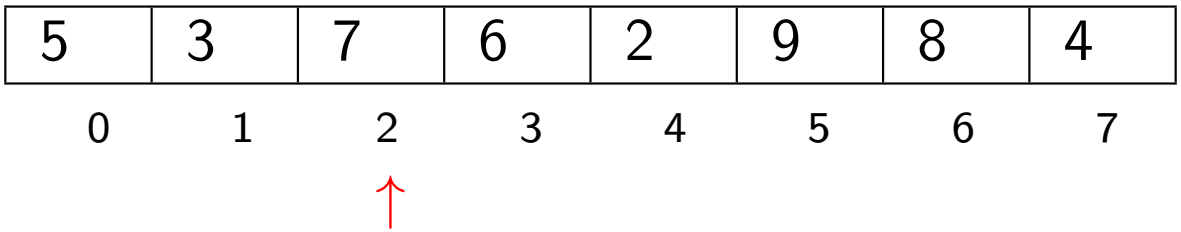
- i = 0
- maximo = v[0] = 5

Algoritmos em vetor – valor máximo (exemplo)



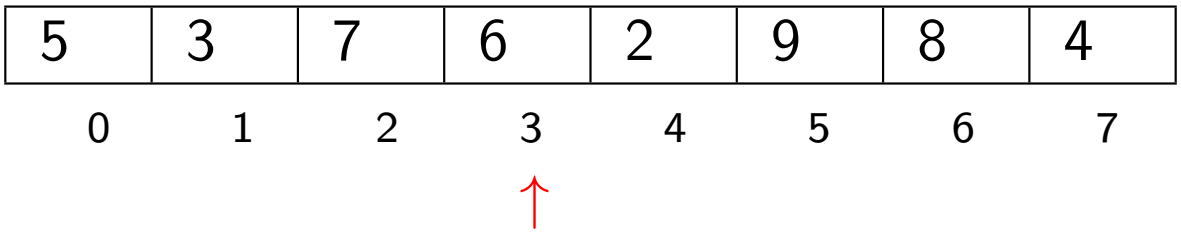
- i $= 1$
- maximo $= 5$
- v[1] $>$ maximo ?
  - *Não (3 $<$ 5): nada a fazer*

Algoritmos em vetor – valor máximo (exemplo)



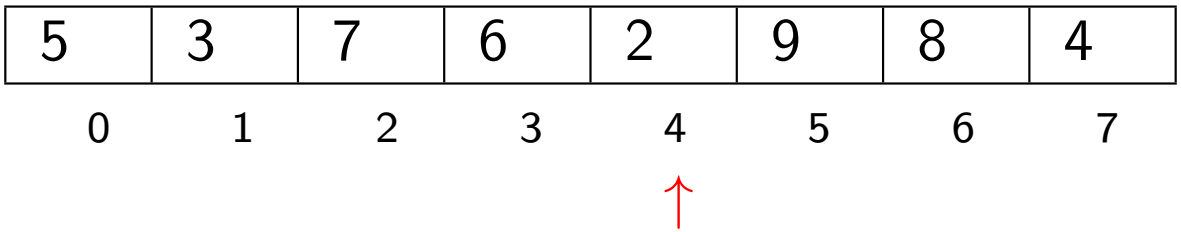
- i = 2
- maximo = 5
- v[2] > maximo ?
  - *Sim (7 > 5):* maximo = 7

Algoritmos em vetor – valor máximo (exemplo)



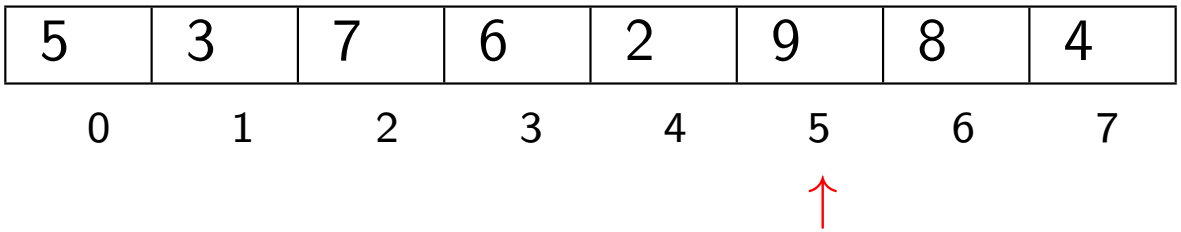
- i = 3
- maximo = 7
- v[3] > maximo ?
  - *Não (6 < 7): nada a fazer*

Algoritmos em vetor – valor máximo (exemplo)



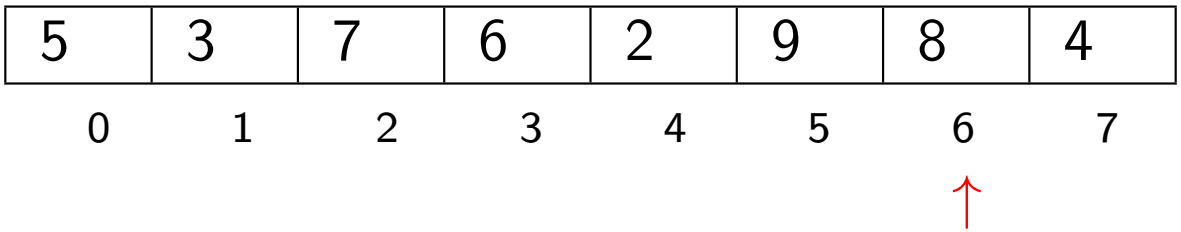
- i = 4
- maximo = 7
- v[4] > maximo ?
  - *Não (2 < 7): nada a fazer*

Algoritmos em vetor – valor máximo (exemplo)



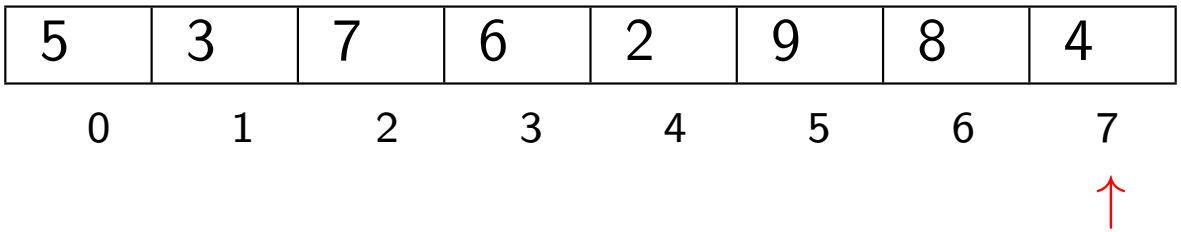
- i $= 5$
- maximo $= 7$
- v[5] $>$ maximo ?
  - *Sim (9 $>$ 7):* maximo $= 9$

Algoritmos em vetor – valor máximo (exemplo)



- i = 6
- maximo = 9
- v[6] > maximo ?
  - *Não (8 < 9): nada a fazer*

Algoritmos em vetor – valor máximo (exemplo)



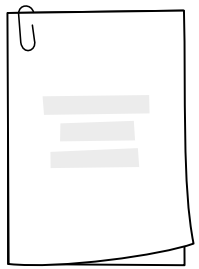
- i = 7
- maximo = 9
- v[7] > maximo ?
  - *Não (4 < 9): nada a fazer*

Algoritmos em vetor – valor máximo (III)

- Implementação

```
import java.lang.Math;

class Vetor {
    protected double v[];

    /*

    ...

    */

    public double valorMaximo() {
        double maximo = v[0];
        for (int i=1; i<v.length; i++)
            if (v[i] > maximo)
                maximo = v[i];
        return maximo;
    }
}
```

Exercício 1



- Considerando a classe `Vetor` dada:
  - https://www.inf.ufsc.br/~alexandre.silva/courses/16s1/ine5603/exercicios/VetorEx.java

Acrescente um método que determine o índice do maior valor no vetor (*arg max*). Exemplo: para o vetor a seguir, o índice do maior valor é 5 (posição na qual se encontra o máximo igual a 9)

5	0	7	6	2	**9**	8	4
0	1	2	3	4	5	6	7

Métodos simples de ordenação

- Tempo de execução proporcional a n^2, ou seja $\mathcal{O}(n^2)$, para n valores
- Implementações:

  https://www.inf.ufsc.br/~alexandre.silva/courses/16s1/ine5603/codigos/OrdenacaoEx.java
- Sugestão de classe:

```
class Ordenacao {
    protected double v[];
    public Ordenacao(double vet[]) {
        /* ... */
    }
    public void bolha() { //bubble sort
        /* ... */
    }
    public void bolha_melhorado() { //bubble sort
        /* ... */
    }
    public void selecao() { //selection sort
        /* ... */
    }
    public void insercao() { //insertion sort
        /* ... */
    }
}
```

Método da bolha – *bubble sort*

```
public void bolha() { //bubble sort
    double aux;
    for (int i=0; i<v.length; i++) {
        for (int j=0; j<v.length-i-1; j++) {
            if (v[j] > v[j+1]) {
                aux = v[j];
                v[j] = v[j+1];
                v[j+1] = aux;
            }
        }
    }
}
```

Método da bolha melhorado – *bubble sort*

```
public void bolha_melhorado() { //bubble sort
    double aux;
    boolean TROCA;
    for (int i=0; i<v.length; i++) {
        TROCA = false;
        for (int j=0; j<v.length-i-1; j++) {
            if (v[j] > v[j+1]) {
                aux = v[j];
                v[j] = v[j+1];
                v[j+1] = aux;
                TROCA = true;
            }
        }
        if ( ! TROCA )
            break;
    }
}
```

Método da bolha – *bubble sort*

- Normal

  20 30 28 31 24 60 45 50
  20 28 30 24 31 45 50 60
  20 28 24 30 31 45 50 60
  20 24 28 30 31 45 50 60
  20 24 28 30 31 45 50 60
  20 24 28 30 31 45 50 60
  20 24 28 30 31 45 50 60
  20 24 28 30 31 45 50 60
  20 24 28 30 31 45 50 60

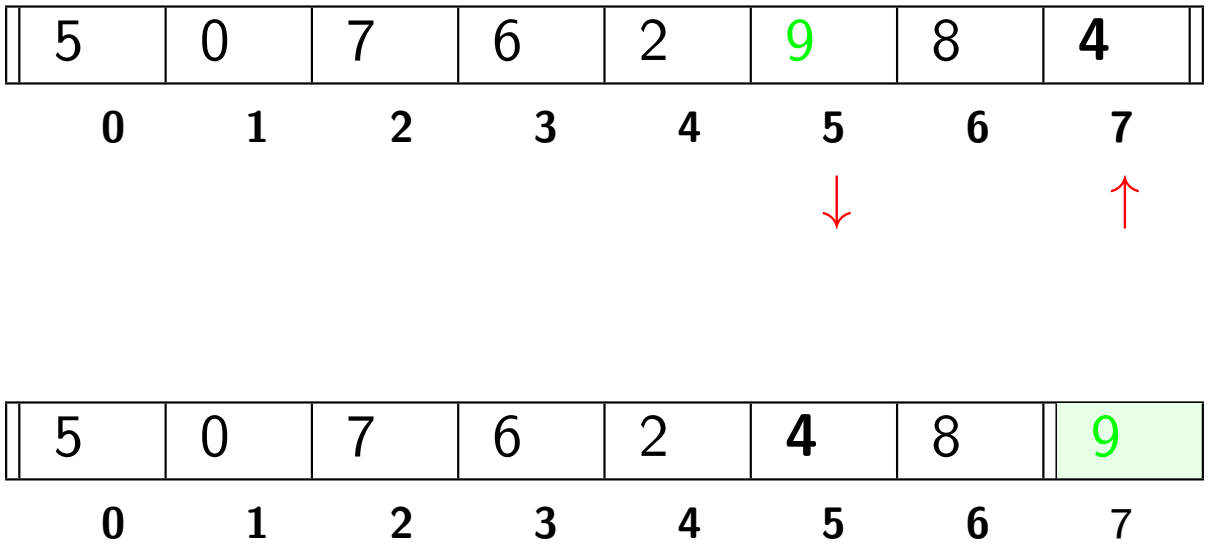
- Melhorado

  20 30 28 31 24 60 45 50
  20 28 30 24 31 45 50 60
  20 28 24 30 31 45 50 60
  20 24 28 30 31 45 50 60
  20 24 28 30 31 45 50 60

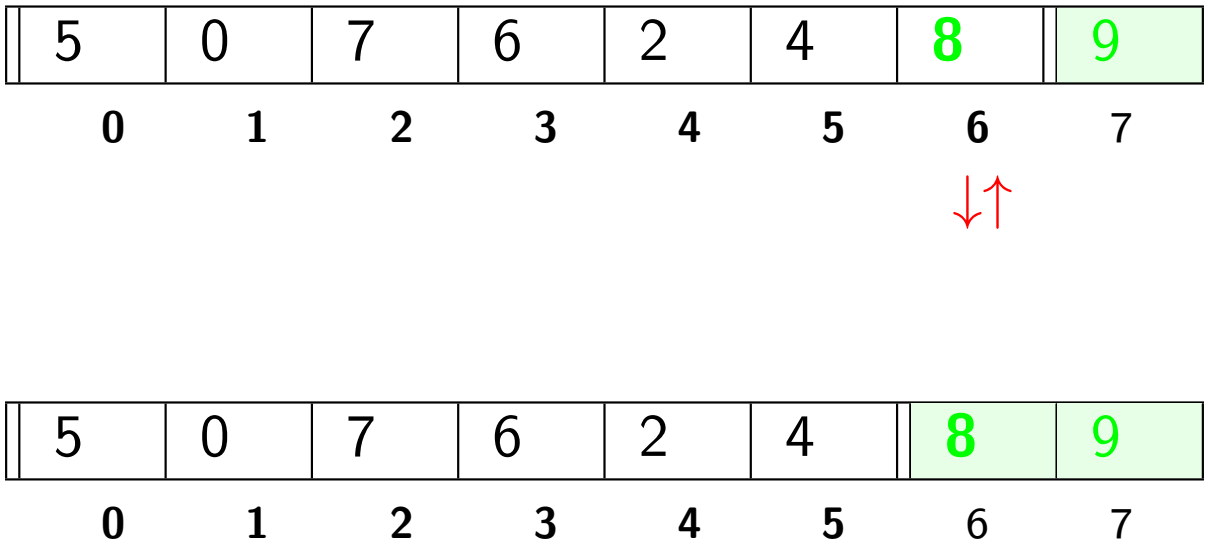
Método da seleção – *selection sort*

1. Partição vai de 0 até k (onde k é o tamanho do vetor menos um)
2. Trocar o maior elemento do vetor (máximo) com a última posição da partição
3. $k = k - 1$, voltar ao passo 1

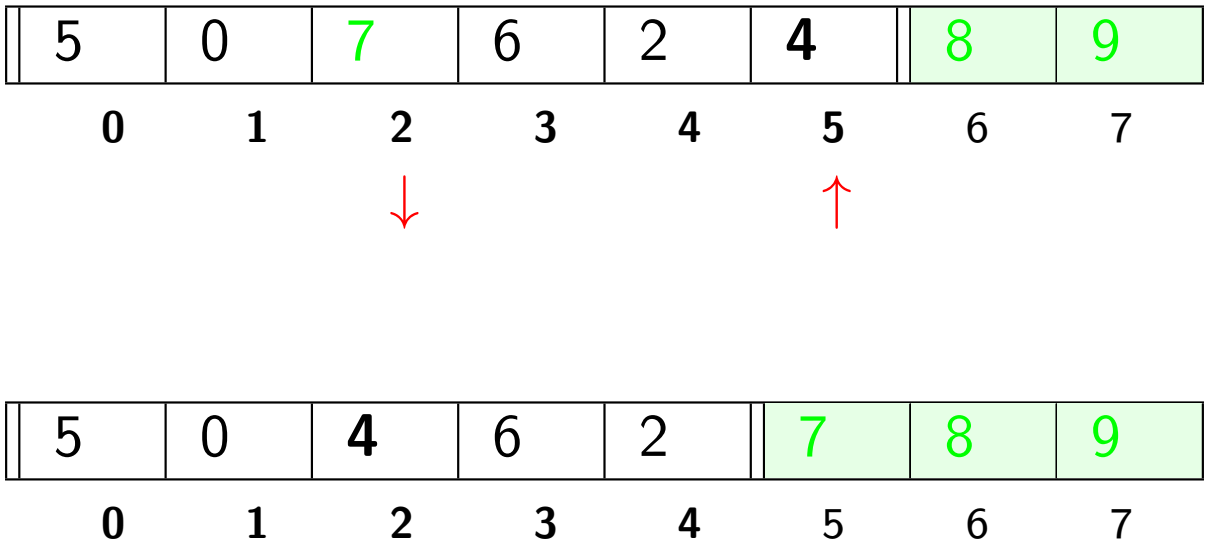
Método da seleção – *selection sort*



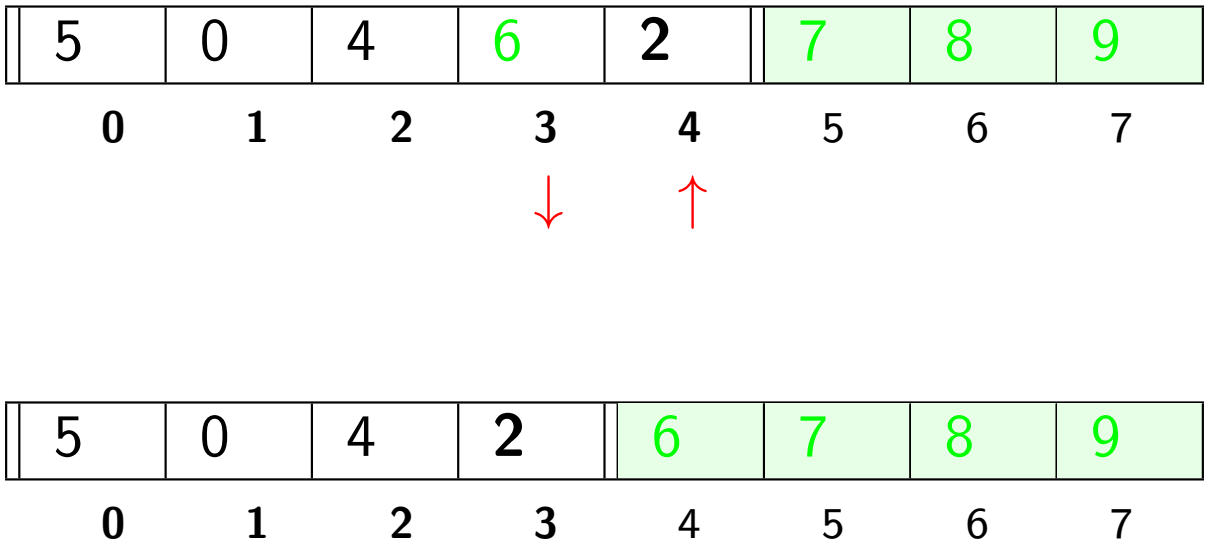
Método da seleção – *selection sort*



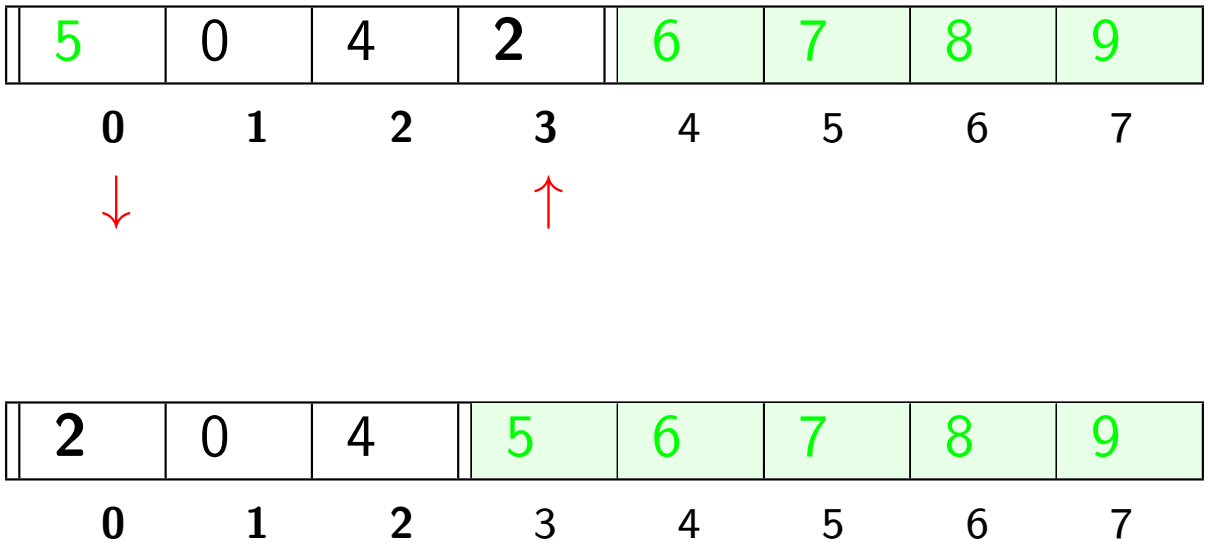
Método da seleção – *selection sort*



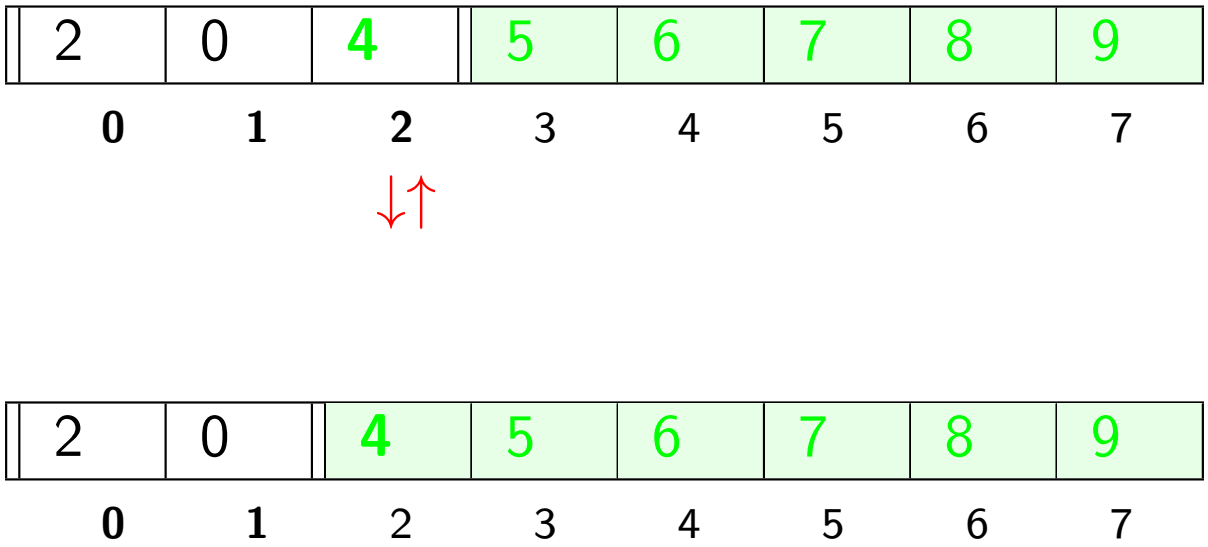
Método da seleção – *selection sort*



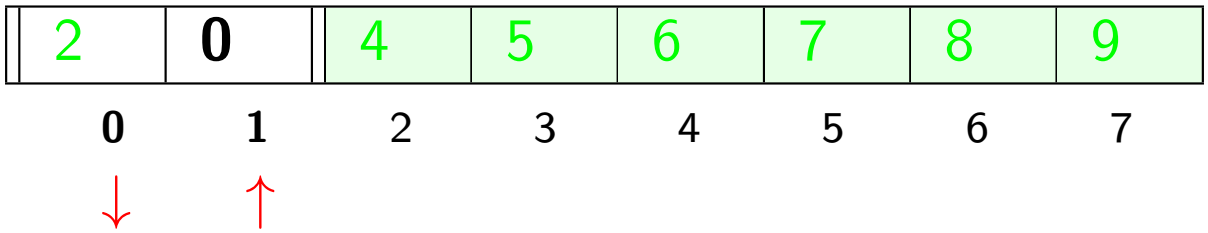
Método da seleção – *selection sort*

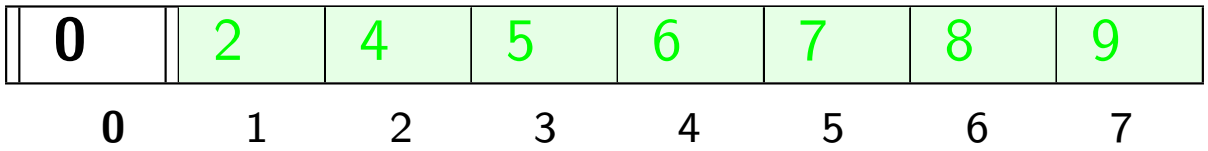


Método da seleção – *selection sort*



Método da seleção – *selection sort*





Método da seleção – *selection sort*

```
public void selecao() { //selection sort
    double aux;
    int posMaior;
    for (int i=0; i<v.length; i++) {
        posMaior = 0;
        for (int j=1; j<v.length-i; j++) {
            if (v[j] > v[posMaior])
                posMaior = j;
        }
        aux = v[posMaior];
        v[posMaior] = v[v.length-i-1];
        v[v.length-i-1] = aux;
    }
}
```

Método da seleção – *selection sort*

```
[20  30  28  31  24  60  45  50]
                     ^       ^

[20  30  28  31  24  50  45] 60
                     ^   ^

[20  30  28  31  24  45] 50  60
                     ^^

[20  30  28  31  24] 45  50  60
                 ^   ^

[20  30  28  24] 31  45  50  60
      ^          ^

[20  24  28] 30  31  45  50  60
         ^^

[20  24] 28  30  31  45  50  60
     ^^

[20] 24  28  30  31  45  50  60
```

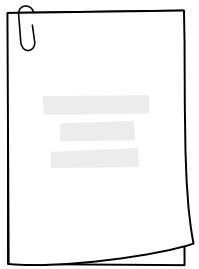
Método da inserção – *insertion sort*

```
public void insercao() { //insertion sort
     double aux;
     int i, j;
     for (i=1; i<v.length; i++) {
          aux = v[i];
          j = i - 1;
          while (j >= 0 && v[j] > aux) {
               v[j+1] = v[j];
               j--;
          }
          v[j+1] = aux;
     }
}
```

Método da inserção – *insertion sort*

```
[20]  30<  28   31   24   60   45   50
   ^

[20   30]  28<  31   24   60   45   50
    ^

[20   28   30]  31<  24   60   45   50
              ^

[20   28   30   31]  24<  60   45   50
    ^

[20   24   28   30   31]  60<  45   50
                    ^

[20   24   28   30   31   60]  45<  50
                         ^

[20   24   28   30   31   45   60]  50<
                              ^

[20   24   28   30   31   45   50   60]
```

Exercício 2



- Para cada método de ordenação, calcule:
  - Número de comparações realizadas
  - Número de trocas efetuadas

Métodos de busca

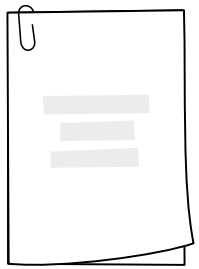
- Normalmente utiliza-se um valor numérico como chave primária (identificador exclusivo) de busca. Por exemplo, matrícula do aluno

- A busca por um determinado valor pode ser eficientemente implementada em um vetor ordenado por meio de:

  - Busca binária: verifica-se o elemento do meio da partição; se valor nesta posição é igual, encontrou (finaliza); se maior, a busca é repetida na primeira metade; se menor, é repetida na segunda metade

  - Busca por interpolação linear: verifica-se o elemento dado pela interpolação linear dos dois valores nas posições extremas da partição; se valor nesta posição é igual, encontrou (finaliza); se maior, repete-se para a primeira parte; caso contrário, repete-se para a segunda parte

Exercício 3



- Dado um vetor qualquer, ordene-o por qualquer método e implemente a busca binária.

- Material:
  - https://www.inf.ufsc.br/~alexandre.silva/courses/16s1/ine5603/codigos/OrdenacaoEx.java