

EDB-I

Estrutura de Dados Básicas I

Aula 13

Algoritmos de Ordenação

Bubble Sort

(material baseado nas notas de aula do Prof. César Rennó Costa e Prof. Eiji Adachi)

Bubble Sort - Idéia geral

ordenação por comparação sucessivas

“Percorra o vetor n vezes, a cada vez ‘flutuando’ o i -ésimo maior elemento para a $(n-i-1)$ -ésima posição do vetor”



Bubble Sort

Compare elemento corrente com seu vizinho:

- Se estão fora de ordem, troquem de posição



Bubble Sort

Compare elemento corrente com seu vizinho:



- Se estão fora de ordem, troquem de posição

1ª Iteração

5	3	1	4	8
3	5	1	4	8

 →

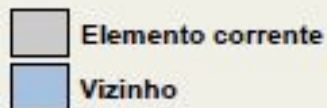
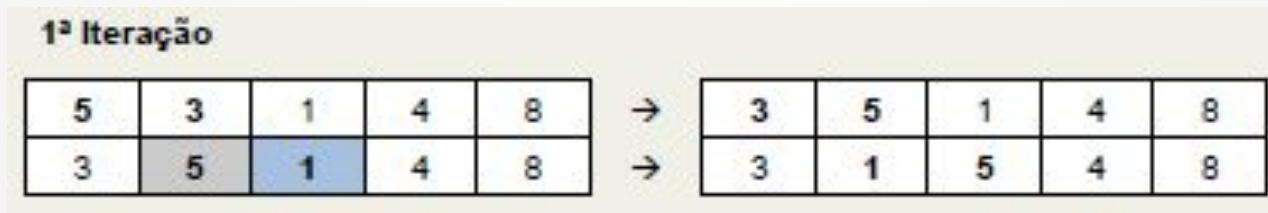
3	5	1	4	8
---	---	---	---	---

	Elemento corrente
	Vizinho

Bubble Sort

Compare elemento corrente com seu vizinho:

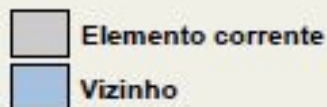
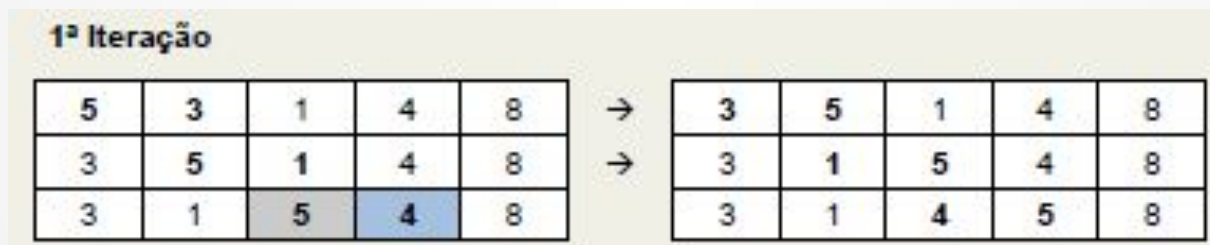
- Se estão fora de ordem, troquem de posição



Bubble Sort

Compare elemento corrente com seu vizinho:

- Se estão fora de ordem, troquem de posição



Bubble Sort

Compare elemento corrente com seu vizinho:

- Se estão fora de ordem, troquem de posição

1ª Iteração

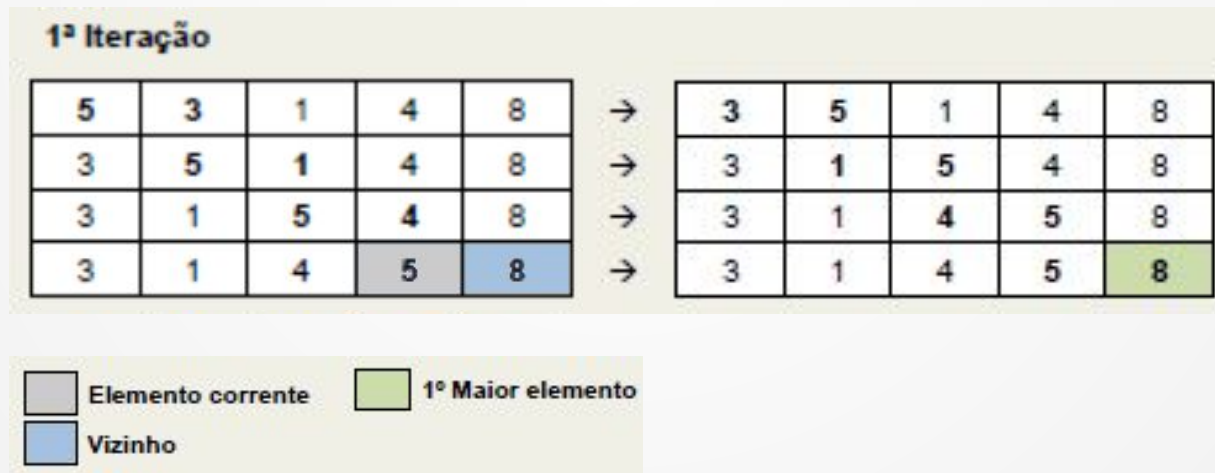
5	3	1	4	8	→	3	5	1	4	8
3	5	1	4	8	→	3	1	5	4	8
3	1	5	4	8	→	3	1	4	5	8
3	1	4	5	8	→	3	1	4	5	8

	Elemento corrente
	Vizinho

Bubble Sort

Compare elemento corrente com seu vizinho:

- Se estão fora de ordem, troquem de posição






Bubble Sort

Compare elemento corrente com seu vizinho:

- Se estão fora de ordem, troquem de posição

2ª Iteração

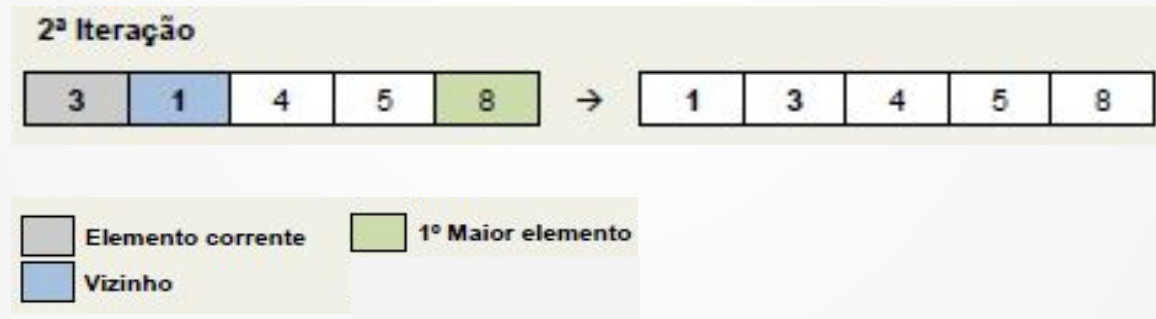
3	1	4	5	8
---	---	---	---	---

	Elemento corrente		1º Maior elemento
	Vizinho		

Bubble Sort

Compare elemento corrente com seu vizinho:

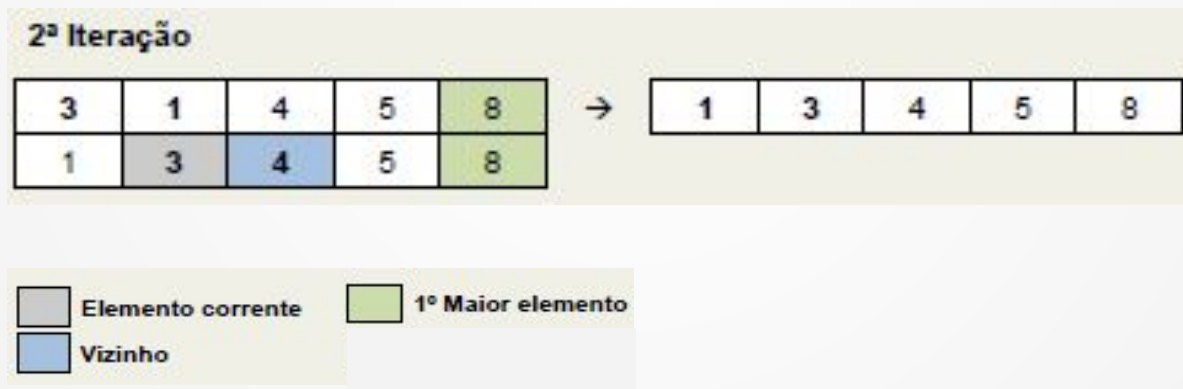
- Se estão fora de ordem, troquem de posição



Bubble Sort

Compare elemento corrente com seu vizinho:

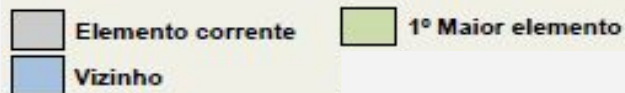
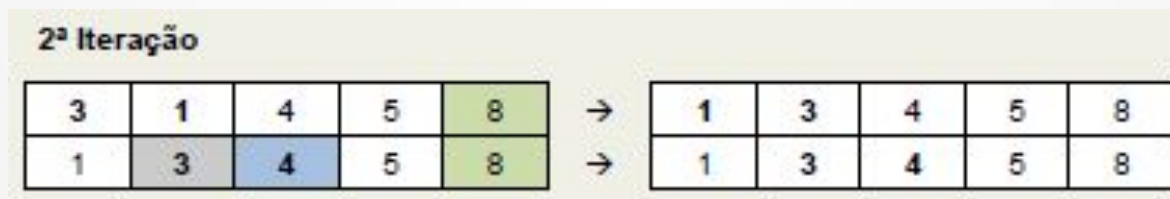
- Se estão fora de ordem, troquem de posição



Bubble Sort

Compare elemento corrente com seu vizinho:

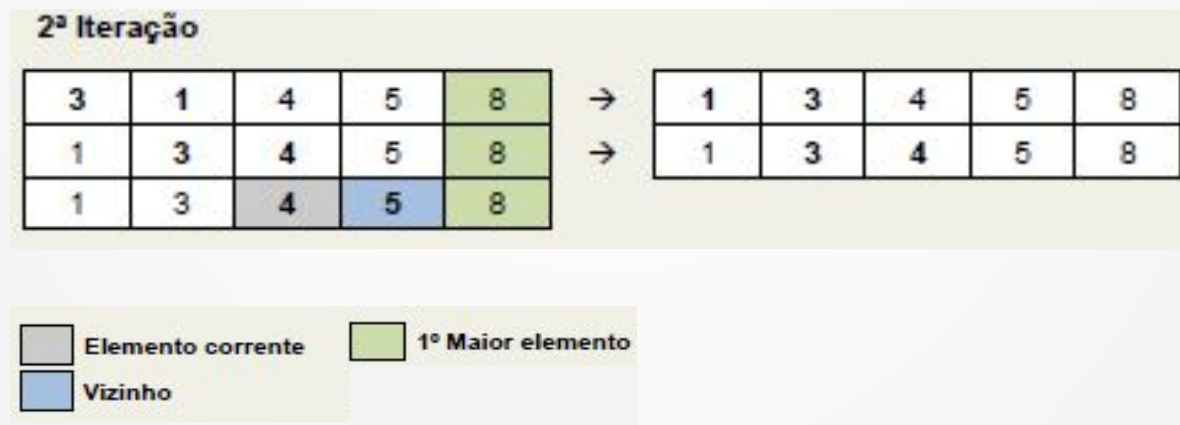
- Se estão fora de ordem, troquem de posição



Bubble Sort

Compare elemento corrente com seu vizinho:

- Se estão fora de ordem, troquem de posição



Bubble Sort

Compare elemento corrente com seu vizinho:

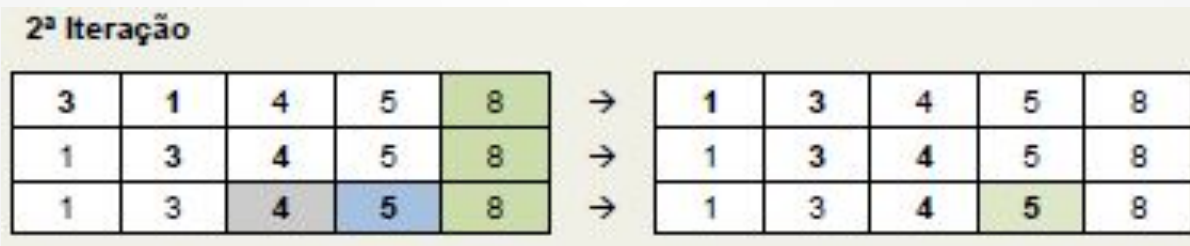
- Se estão fora de ordem, troquem de posição



Bubble Sort

Compare elemento corrente com seu vizinho:

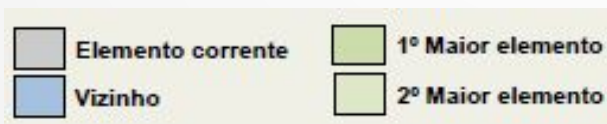
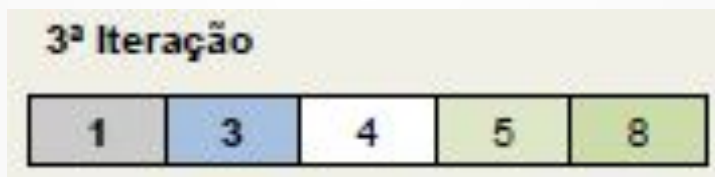
- Se estão fora de ordem, troquem de posição



Bubble Sort

Compare elemento corrente com seu vizinho:

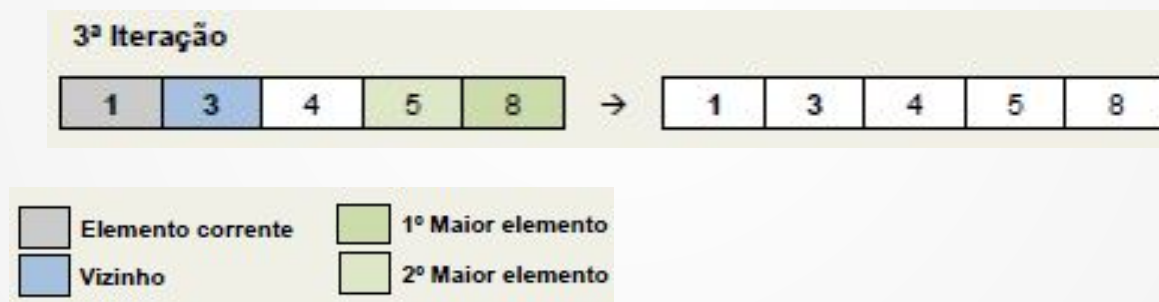
- Se estão fora de ordem, troquem de posição



Bubble Sort

Compare elemento corrente com seu vizinho:

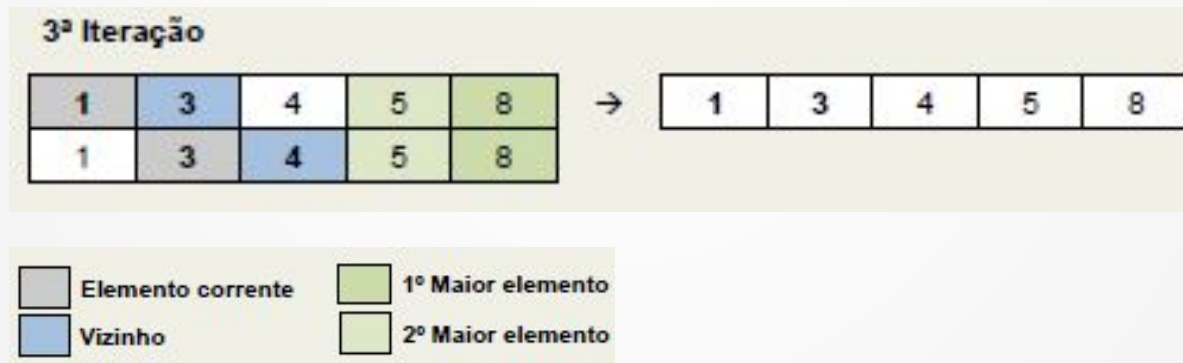
- Se estão fora de ordem, troquem de posição



Bubble Sort

Compare elemento corrente com seu vizinho:

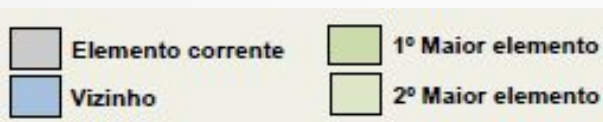
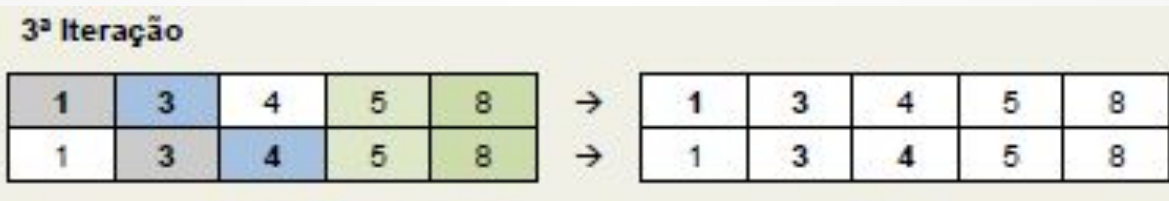
- Se estão fora de ordem, troquem de posição



Bubble Sort

Compare elemento corrente com seu vizinho:

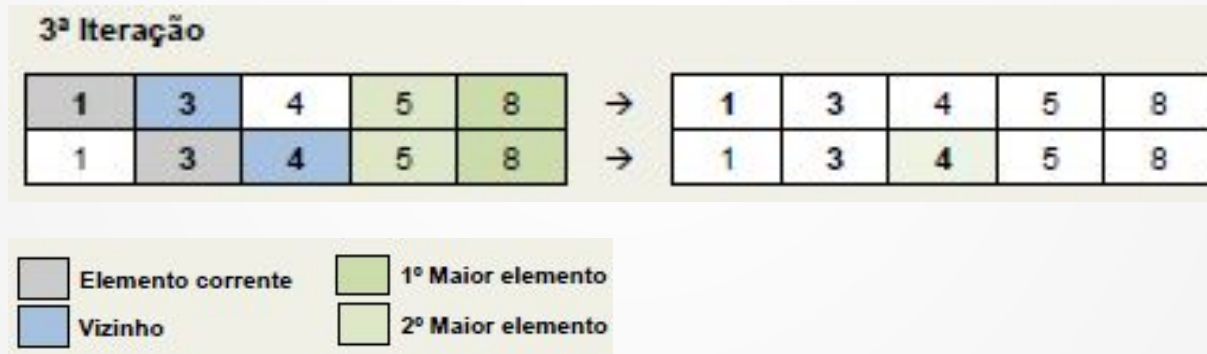
- Se estão fora de ordem, troquem de posição



Bubble Sort

Compare elemento corrente com seu vizinho:

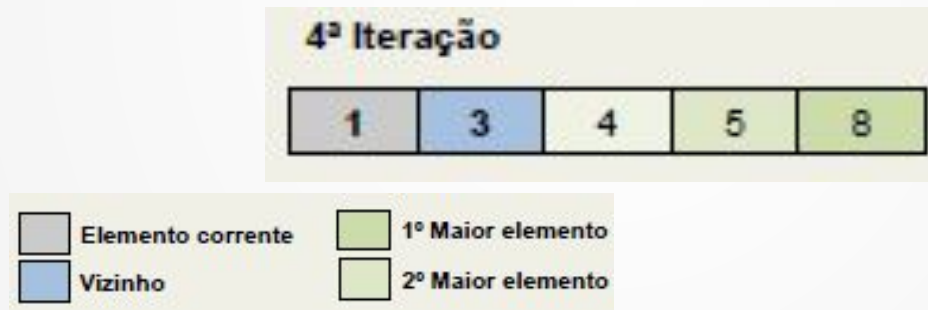
- Se estão fora de ordem, troquem de posição



Bubble Sort

Compare elemento corrente com seu vizinho:

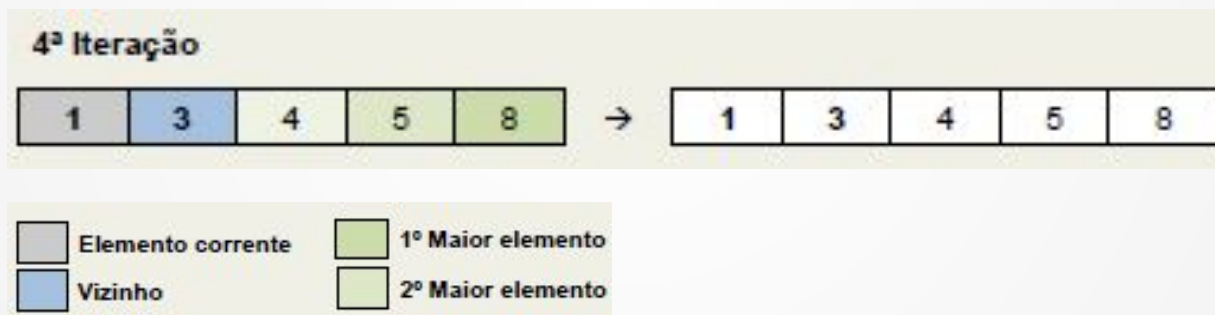
- Se estão fora de ordem, troquem de posição



Bubble Sort

Compare elemento corrente com seu vizinho:

- Se estão fora de ordem, troquem de posição



Bubble Sort

Compare elemento corrente com seu vizinho:

- Se estão fora de ordem, troquem de posição



Bubble Sort

```
// C++ program for implementation
// of Bubble sort
#include <bits/stdc++.h>
using namespace std;

// A function to implement bubble sort
void bubbleSort(int arr[], int n)
{
    int i, j;
    for (i = 0; i < n - 1; i++)

        // Last i elements are already
        // in place
        for (j = 0; j < n - i - 1; j++)
            if (arr[j] > arr[j + 1])
                swap(arr[j], arr[j + 1]);
}
```

```
// Function to print an array
void printArray(int arr[], int size)
{
    int i;
    for (i = 0; i < size; i++)
        cout << arr[i] << " ";
    cout << endl;
}

// Driver code
int main()
{
    int arr[] = { 5, 1, 4, 2, 8};
    int N = sizeof(arr) / sizeof(arr[0]);
    bubbleSort(arr, N);
    cout << "Sorted array: \n";
    printArray(arr, N);
    return 0;
}
// This code is contributed by rathbhupendra
```


Bubble Sort

versão otimizada

```
void swap(int *xp, int *yp)
{
    int temp = *xp;
    *xp = *yp;
    *yp = temp;
}

// An optimized version of Bubble Sort
void bubbleSort(int arr[], int n)
{
    int i, j;
    bool swapped;
    for (i = 0; i < n-1; i++)
    {
        swapped = false;
        for (j = 0; j < n-i-1; j++)
        {
            if (arr[j] > arr[j+1])
            {
                swap(&arr[j], &arr[j+1]);
                swapped = true;
            }
        }
        // IF no two elements were swapped by inner loop, then break
        if (swapped == false)
            break;
    }
}
```

Bubble Sort

Algoritmo simples

- Pior e melhor caso: $\Theta(n^2)$
 - Qualquer vetor

```
Sort(v[n]):  
  FOR i = 0; i < n; i++:  
    FOR j = 0; j < n-i-1; j++:  
      IF v[j] > v[j+1] :  
        Swap v[j], v[j+1]  
      END_IF  
    END_FOR  
  END_FOR  
END
```

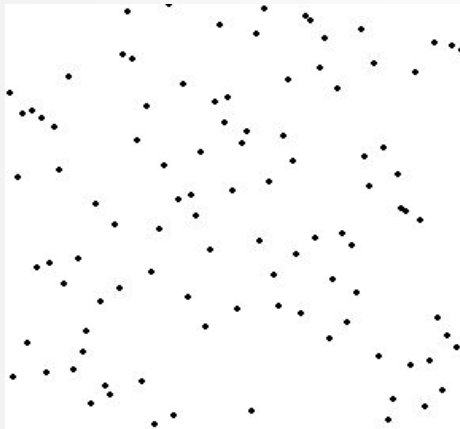
Bubble Sort

Algoritmo melhorado

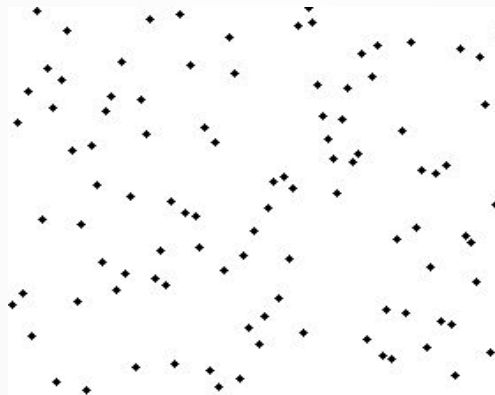
- Pior caso: $\Theta(n^2)$
 - Vetor ordenado em ordem inversa
- Melhor caso: $\Theta(n)$
 - Vetor já ordenado

```
Sort(v[n]):  
    mudou = TRUE  
    fim = n  
    WHILE mudou :  
        mudou = FALSE  
        FOR i = 0; i < fim; i++:  
            IF v[i] > v[i+1] ENTÃO:  
                Swap v[i], v[i+1]  
                mudou = TRUE  
            END_IF  
        END_FOR  
        fim = fim-1  
    END_WHILE  
END
```

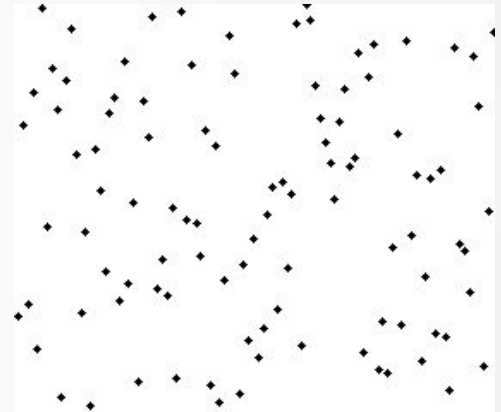
Seleção



Inserção



Bubble sort



Indique para cada um dos métodos de ordenação já vistos qual o estado do vetor após cada passo do algoritmo

[10 99 82 83 84 19 29]

[51 39 33 21 24 27 25 39 56 43 23 40]

SELECTION-SORT($v[N]$)

1. **for** $j \leftarrow 0$ to $(N-2)$
2. $\text{indiceMenor} \leftarrow j$
3. **for** $i \leftarrow (j+1)$ to $(N-1)$
4. **if** $v[i] < v[\text{indiceMenor}]$
5. $\text{indiceMenor} \leftarrow i$
6. **Troque** $v[j] \leftrightarrow v[\text{indiceMenor}]$

INSERTION-SORT ($v[N]$):

1. **for** $i \leftarrow 1$ to $N-1$
2. $\text{novo} = v[i]$
3. $j = i-1$
4. **while** $j \geq 0 \ \&\& \ \text{novo} < v[j]$
5. $v[j+1] = v[j]$
6. $j = j-1$
7. **end**
8. $v[j+1] = \text{novo}$
9. **end**

BUBBLE-SORT ($v[N]$):

1. **for** $i \leftarrow 0$ to $N-1$
2. **for** $j \leftarrow 0$ to $N-i-1$
3. **if** $(v[j] > v[j+1])$
4. $\text{swap}(v[j], v[j+1]);$
5. **end**
6. **end**