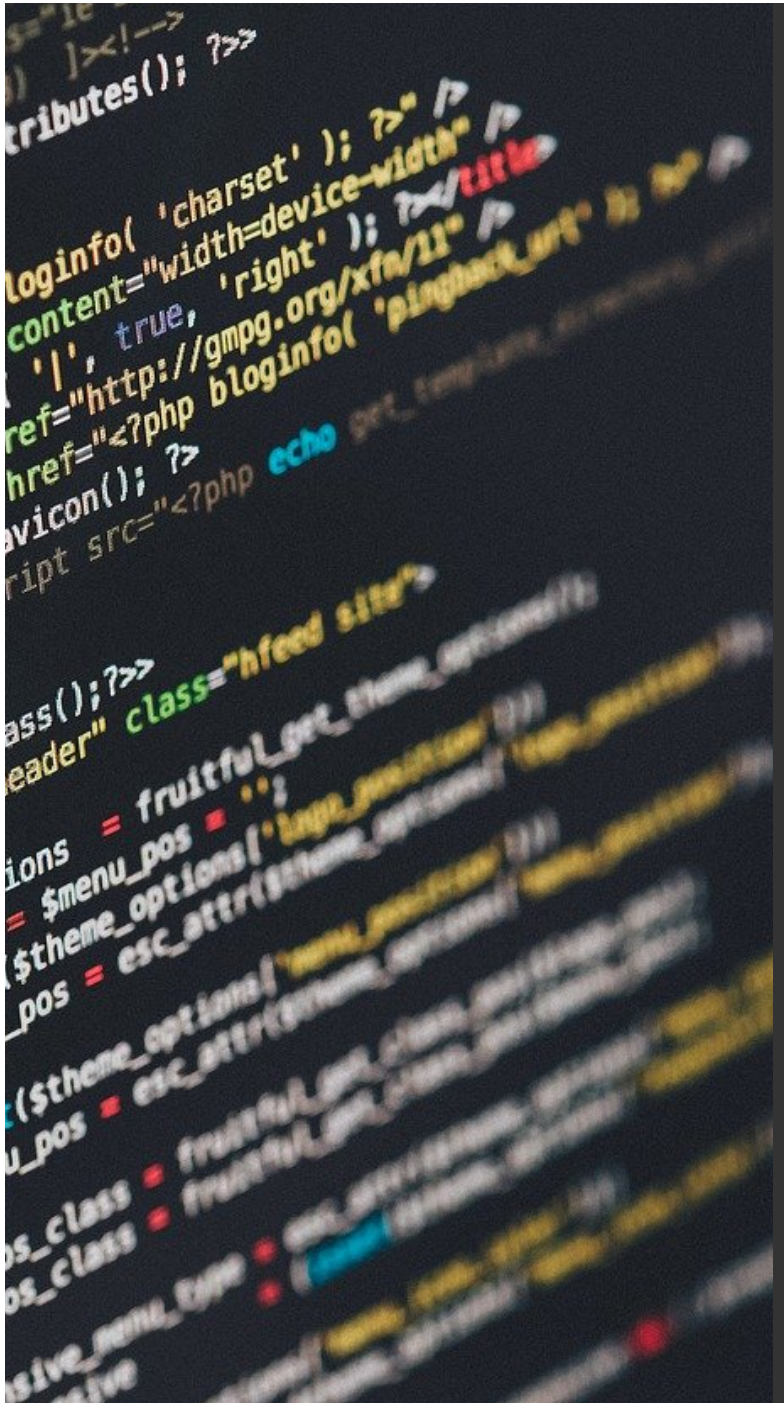


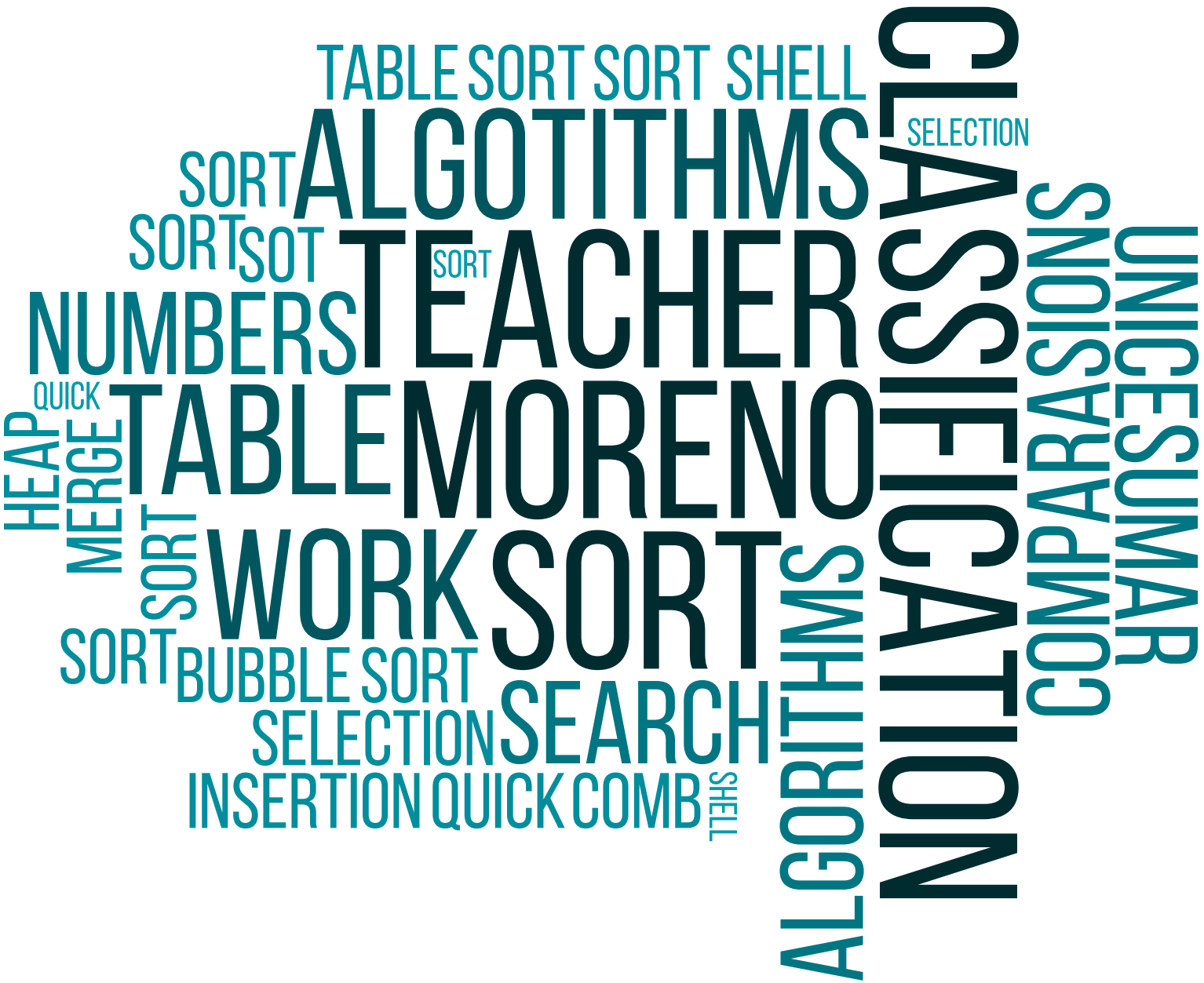
ALGORITMOS E LÓGICA DE PROGRAMAÇÃO II - LINGUAGEM C

CLASSIFICAÇÃO DE VETORES

PROFESSOR ALEXANDRE MORENO

UniCesumar

MORENO
Professor



1	350
2	450
3	275
4	475
5	175
6	25
7	100
8	150
9	400
10	500
11	325
12	250
13	375
14	200
15	50
16	75
17	125
18	225
19	300
20	425

VEJAMOS UMA TABELA CONTENDO 20 ELEMENTOS

1	350
2	450
3	275
4	475
5	175
6	25
7	100
8	150
9	400
10	500
11	325
12	250
13	375
14	200
15	50
16	75
17	125
18	225
19	300
20	425

1	25
2	50
3	75
4	100
5	125
6	150
7	175
8	200
9	200
10	225
11	250
12	275
13	325
14	350
15	375
16	400
17	425
18	450
19	475
20	500

DISORDERED VERSUS SORT

1	25
2	50
3	75
4	100
5	125
6	150
7	175
8	200
9	200
10	225
11	250
12	275
13	325
14	350
15	375
16	400
17	425
18	450
19	475
20	500

MORENO
Professor

MAS QUAL O OBJETIVO DE CLASSIFICARMOS UMA TABELA ?

OBJETIVOS DE CLASSIFICAR UMA TABELA

MORENO
Professor

START

INFORMAÇOES ORGANIZADAS

MENOR E MAIOR ELEMENTOS NAS EXTREMIDADES

BUSCA MAIS EFICIENTE

END

PRINCIPAIS MÉTODOS DE CLASSIFICAÇÃO

MORENO
Professor

1. SELECTION SORT
2. BUBBLE SORT
3. INSERTION SORT
4. SHELL SORT
5. QUICK SORT

SELECTION SORT

A ordenação por seleção é um algoritmo baseado em passar sempre o menor valor do vetor para a primeira posição, depois o de segundo menor valor para a segunda posição, e assim é feito sucessivamente com os (n-1) elementos restantes, até os últimos dois elementos.

SELECTION SORT

O número de comparações é conhecido aplicando a **fórmula** **TC= n * (n-1)/2,** desta forma, uma tabela com **20** elementos executará **190** comparações

O MÉTODO É SIMPLES, MAS NÃO É MUITO EFICIENTE

SELECTION SORT

Uma tabela com 5 elementos:

passo1: 1,2 1,3 1,4 1,5

passo2: 2,3 2,4 2,5

passo3: 3,4 3,5

passo4: 4,5

O MÉTODO É SIMPLES, MAS NÃO É MUITO EFICIENTE

SELECTION SORT

```
for (x=0; x < n-1; x++) {
    for (y=x+1; y < n; y++) {
        if ( tab[y] < tab[x]) {
            tmp = tab[x];
            tab[x] = tab[y];
            tab[y] = tmp ;
        }
    }
}
```

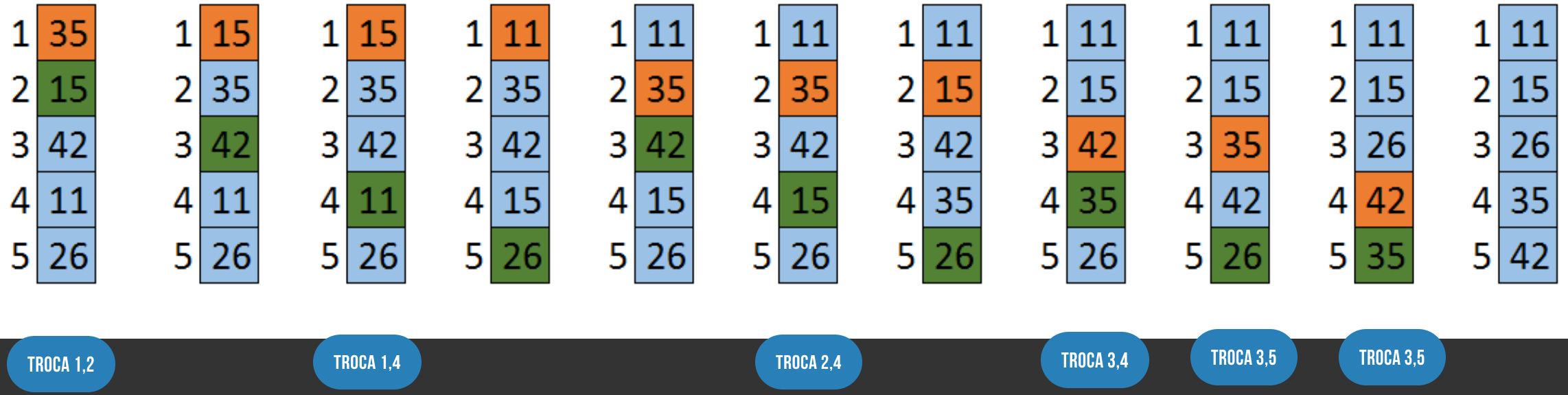
TABELA COM 5 ELEMENTOS

10 comparações com 6 trocas

SELECTION SORT

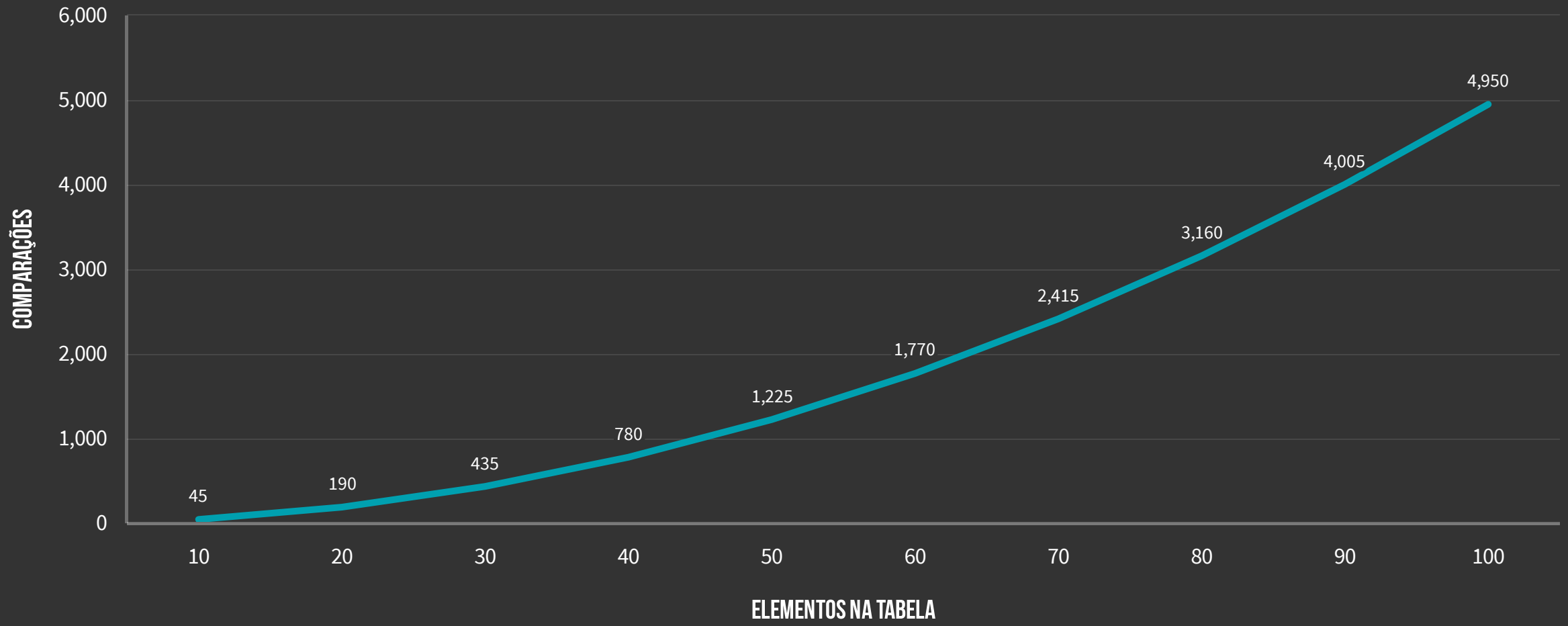
DISORDERED

SORT



ELEMENTOS VERSUS COMPARAÇÕES

SELECTION SORT



fórmula TC= n * (n-1)/2

TC = Total de Comparações **n = número de elementos**

ELEMENTOS	NÚMERO DE COMPARAÇÕES
5	10
10	45
100	4.950
1.000	499.500
10.000	49.995.000
100.000	4.999.950.000

PRINCIPAIS MÉTODOS DE CLASSIFICAÇÃO

1 SELECTION SORT

2 BUBBLE SORT

3 INSERTION SORT

4 SHELL SORT

5 QUICK SORT (REQUIRED RECURSION)

BUBBLE SORT

Conhecido como “**Algoritmo da Bolha**”, tem por objetivo comparar elementos dois a dois empurrando o maior elemento para o fim da tabela.

Uma tabela com **5** elementos:

passo1: 1,2 2,3 3,4 4,5

passo2: 1,2 2,3 3,4

passo3: 1,2 2,3

passo4: 1,2

BUBBLE SORT

O número de comparações efetuadas na classificação é obtido através da fórmula **TC= n * (n-1)/2.** Assim, uma tabela com **20** elementos executará **190** comparações para a classificação.

***n** é o número total de elementos.

O MÉTODO É SIMPLES, MAS NÃO É MUITO EFICIENTE

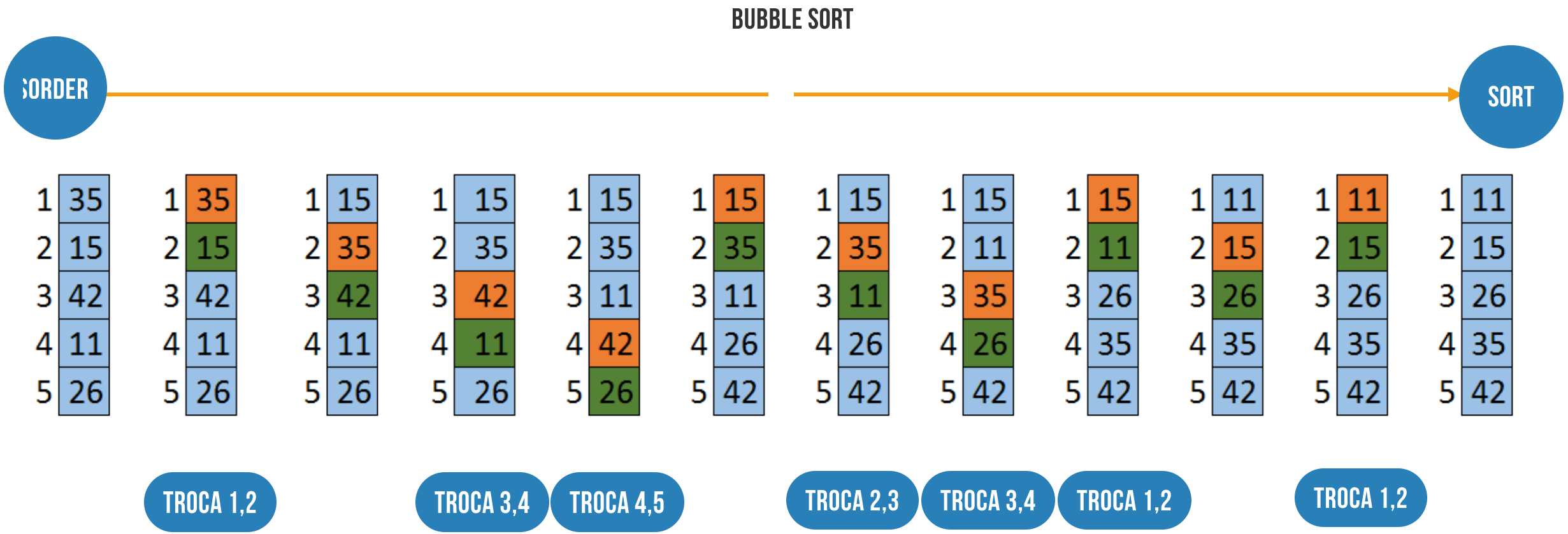
BUBBLE SORT

```
for (x=0; x < n-1; x++) {
    for (y=0; y < n - x-1;  y++){
        if (tab[y] > tab[y+1]) {
            tmp = tab[y];
            tab[y] = tab[y+1];
            tab[y+1] = tmp;
        }
    }
}
```

MORENO
Professor

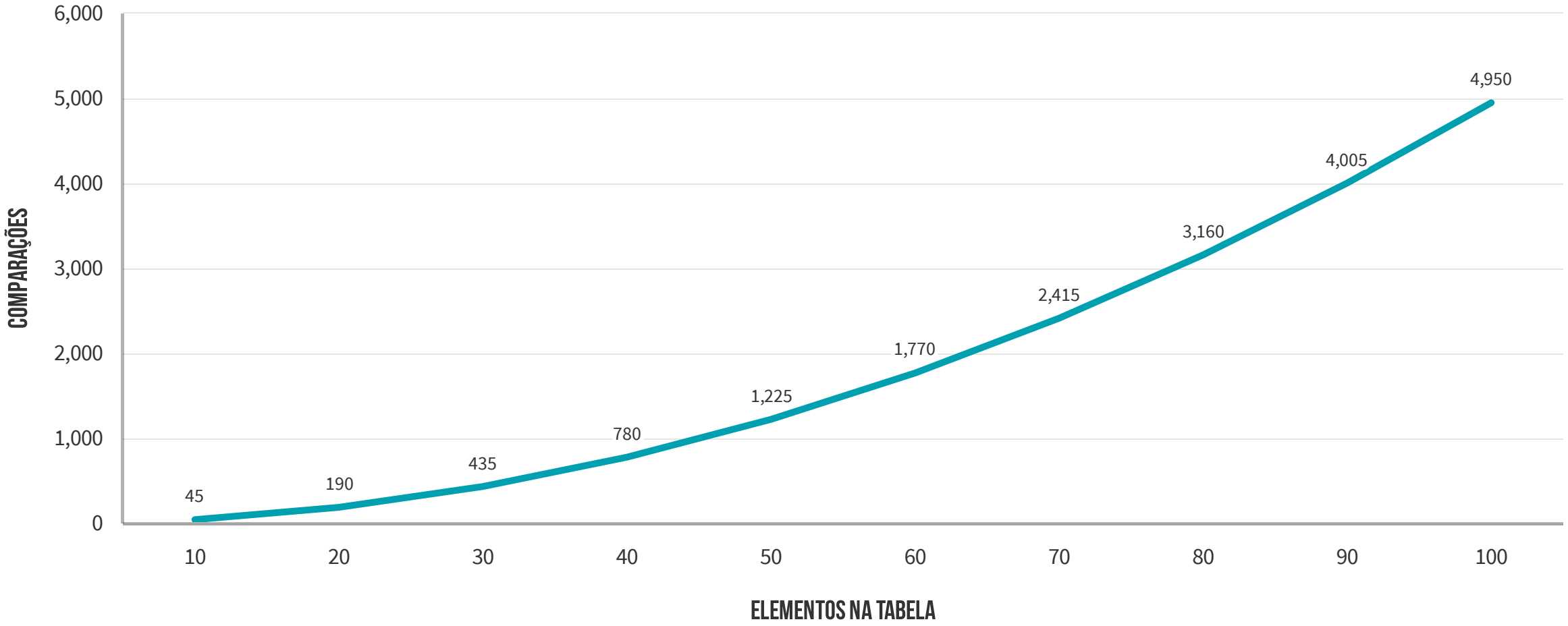
TABELA COM 5 ELEMENTOS

10 comparações com 7 trocas



ELEMENTOS VERSUS COMPARAÇÕES

BUBBLE SORT



PRINCIPAIS MÉTODOS DE CLASSIFICAÇÃO

1 SELECTION SORT

2 BUBBLE SORT

3 INSERTION SORT

4 SHELL SORT

5 QUICK SORT (REQUIRED RECURSION)

INSERTION SORT

Simples e **eficiente** quando aplicado em **pequenas listas**. A lista é percorrida segundo até o último elemento, comparando com o elemento anterior e trocando de lugar quando encontrado um valor menor (objetivo crescente)

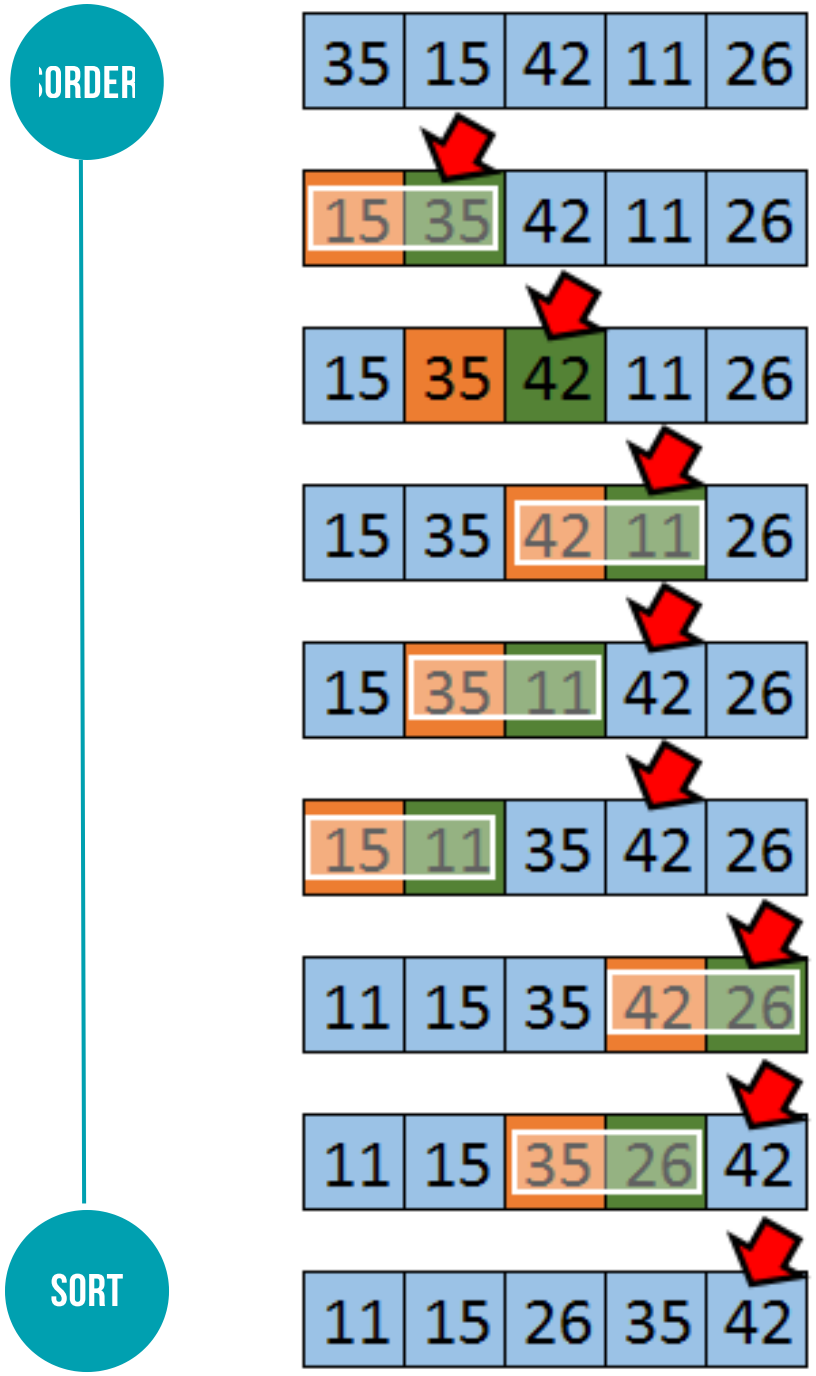
Funciona da mesma forma que as pessoas usam para **ordenar cartas** em um jogo de **baralho** como pôquer

INSERTION SORT

```
for (x=1; x<n; x++) {
    y=x ;
    while (tab[y] < tab[y-1])  {
        tmp = tab[y];
        tab[y] = tab[y-1];
        tab[y-1] = tmp;
        if (y-1 > 1)
            y-- ;
    }
}
```

TABELA COM 5 ELEMENTOS

INSERTION SORT



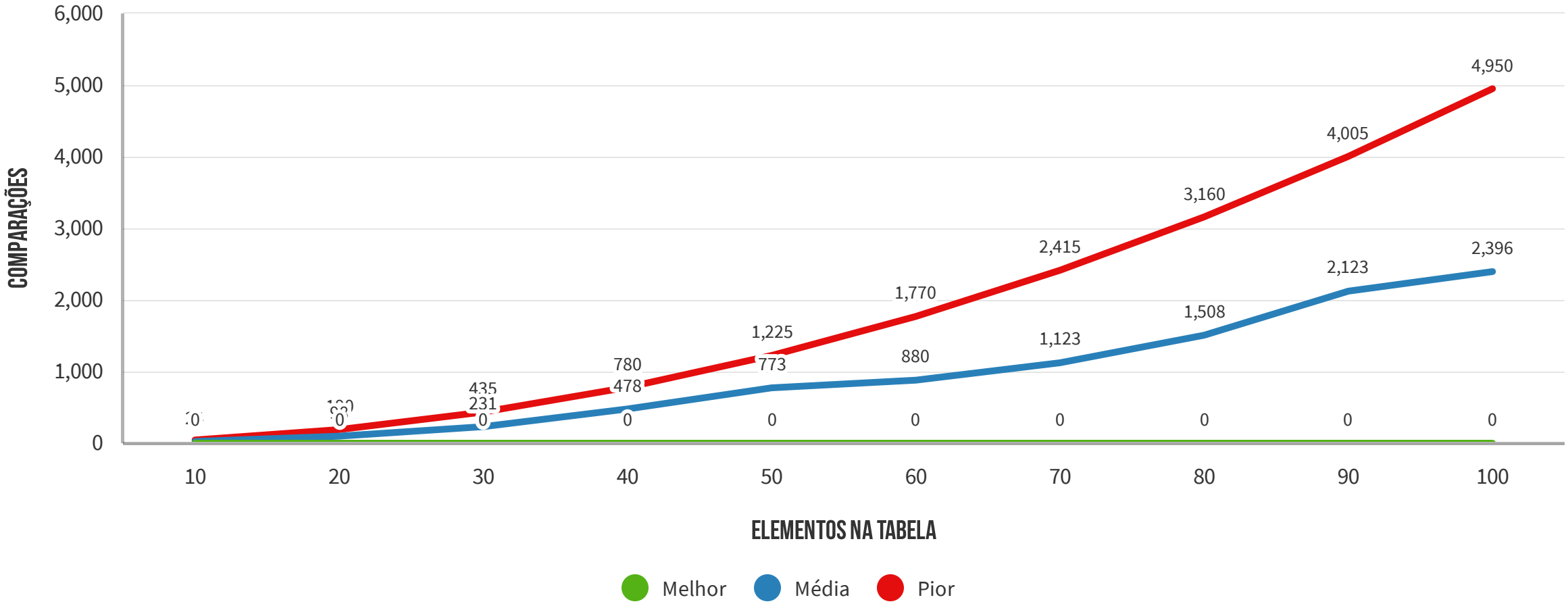
INSERTION SORT

O número de comparações deste algorítmo é **dinâmico**, ou seja, não é possível determinar uma fórmula.
Em uma tabela contendo 20 elementos, o **pior caso** será **190 comparações** (igual ao Selection Sort e também o Bubble Sort), entretanto o **melhor caso** será igual a 0 (**zero**) **comparações**

O MÉTODO É EFICIENTE PARA PEQUENAS LISTAS

ELEMENTOS VERSUS COMPARAÇÕES

INSERTION SORT



PARA COPIAR VARIÁVEIS DO TIPO STRING É NECESSÁRIO UTILIZAR O STRCPY

#INCLUDE STRING.H