

Aula 16: Laboratório

- Métodos simples de ordenação (parte 2)

Prof. Jesús P. Mena-Chalco
jesus.mena@ufabc.edu.br

3Q-2017



Atividade 01: Benchmark

Vetores ordenados

```
1 #include <time.h>
2 #include <stdio.h>
3 #include <stdlib.h>
4
5 void main(int argc, char *argv[])
6 {
7     long int i;
8     long int n = atoi(argv[1]);
9     long int vetor[n];
10
11     for (i=0; i<n; i++) {
12         vetor[i] = i;
13     }
14
15     for (i=0; i<n-1; i++) {
16         printf("%ld\n", vetor[i]);
17     }
18     printf("%ld", vetor[i]);
19 }
```

```
1 #include <time.h>
2 #include <stdio.h>
3 #include <stdlib.h>
4
5 void main(int argc, char *argv[])
6 {
7     long int i;
8     long int n = atoi(argv[1]);
9     long int vetor[n];
10
11     for (i=0; i<n; i++) {
12         vetor[n-i-1] = i;
13     }
14
15     for (i=0; i<n-1; i++) {
16         printf("%ld\n", vetor[i]);
17     }
18     printf("%ld", vetor[i]);
19 }
```

Vetor aleatório

```
1 #include <time.h>
2 #include <stdio.h>
3 #include <stdlib.h>
4
5 void main(int argc, char *argv[])
6 {
7     srand(time(NULL));
8     long int i;
9     long int n = atoi(argv[1]);
10    long int vetor[n];
11
12    for (i=0; i<n; i++) {
13        vetor[i] = rand();
14    }
15
16    for (i=0; i<n-1; i++) {
17        printf("%ld\n", vetor[i]);
18    }
19    printf("%ld", vetor[i]);
20 }
```

Vetor Parcialmente Ordenados

```
1 #include <time.h>
2 #include <stdio.h>
3 #include <stdlib.h>
4
5 void main(int argc, char *argv[])
6 {
7     srand(time(NULL));
8
9     long int i, j, aux;
10    long int n = atoi(argv[1]);
11    long int vetor[n];
12
13    // vetor ordenado
14    for (i=0; i<n; i++) {
15        vetor[i] = i;
16    }
17
18    // vetor quase ordenado
19    int k = 50;
20
21    for (i=0; i<=n-k; i++) {
22        j = i+rand()%k;
23        aux = vetor[i];
24        vetor[i] = vetor[j];
25        vetor[j] = aux;
26    }
27
28    for (i=0; i<n-1; i++) {
29        printf("%ld\n", vetor[i]);
30    }
31    printf("%ld", vetor[i]);
32 }
```

```
1 #include <time.h>
2 #include <stdio.h>
3 #include <stdlib.h>
4
5 void main(int argc, char *argv[])
6 {
7     srand(time(NULL));
8
9     long int i, j, aux;
10    long int n = atoi(argv[1]);
11    long int vetor[n];
12
13    // vetor ordenado
14    for (i=0; i<n; i++) {
15        vetor[n-i-1] = i;
16    }
17
18    // vetor quase ordenado
19    int k = 50;
20
21    for (i=0; i<=n-k; i++) {
22        j = i+rand()%k;
23        aux = vetor[i];
24        vetor[i] = vetor[j];
25        vetor[j] = aux;
26    }
27
28    for (i=0; i<n-1; i++) {
29        printf("%ld\n", vetor[i]);
30    }
31    printf("%ld", vetor[i]);
32 }
```

Benchmark

```
1 # vetores aleatorios
2 gcc gerarVetorAleatorio.c -o gerarVetorAleatorio
3 ./gerarVetorAleatorio 50000 > vetorAleatorio50000.dat
4
5 # vetores crescente
6 gcc gerarVetorCrescente.c -o gerarVetorCrescente
7 ./gerarVetorCrescente 50000 > vetorCrescente50000.dat
8
9 # vetores decrescente
10 gcc gerarVetorDecrescente.c -o gerarVetorDecrescente
11 ./gerarVetorDecrescente 50000 > vetorDecrescente50000.dat
12
13 # vetores parcialmente ordenados de forma crescente
14 gcc gerarVetorParcialmenteOrdenadoCrescente.c -o gerarVetorParcialmenteOrdenadoCrescente
15 ./gerarVetorParcialmenteOrdenadoCrescente 50000 > vetorPCrescente50000.dat
16
17 # vetores parcialmente ordenados de forma decrescente
18 gcc gerarVetorParcialmenteOrdenadoDecrescente.c -o gerarVetorParcialmenteOrdenadoDecrescente
19 ./gerarVetorParcialmenteOrdenadoDecrescente 50000 > vetorPDecrescente50000.dat
```

```
$ sh -v benchmark.sh
```



Sobre ordenação

Ordenação

- Ordenar corresponde ao **processo de re-arranjar um conjunto de objetos** em ordem ascendente ou descendente.
- **Por que ordenar?**
O objetivo principal da ordenação é **facilitar a recuperação posterior** de itens do conjunto ordenado.
Geralmente considerado no primeiro passo para resolver um problema prático.
- As ordens mais utilizadas são a numérica e lexicográfica.

Show results for **Age Range**

- Birth to 24 Months (362,856)
- 2 to 4 Years (1,985,067)
- 5 to 7 Years (362,563)
- 8 to 13 Years (412,082)
- 14 Years & Up (540,479)

- Toys & Games**
- Action Figures & Statues (864,805)
 - Arts & Crafts (136,514)
 - Baby & Toddler Toys (200,977)
 - Building Toys (48,366)
 - Dolls & Accessories (119,524)
 - Dress Up & Pretend Play (274,580)
 - Electronics for Kids (63,017)
 - Games (405,915)
 - Grown-Up Toys (513,571)
 - Hobbies (450,727)
 - Kids' Furniture, Décor & Storage (205,822)
 - Learning & Education (110,632)
 - Novelty & Gag Toys (127,291)
 - Party Supplies (185,666)
 - Puppets (51,512)
 - Puzzles (723,919)
 - Sports & Outdoor Play (123,028)
 - Stuffed Animals & Plush Toys (584,298)
 - Toy Remote Control & Play Vehicles (160,735)
 - Tricycles, Scooters & Wagons (53,253)
 - Video Games (13,034)

Refine by **International Shipping**


- Ship to Brazil

Featured Characters & Brands


- Magic the Gathering (76,831)
- Yu-Gi-Oh! (74,198)
- Pokemon (48,569)
- Star Wars (35,715)
- Barbie (30,257)
- Hello Kitty (16,918)
- Disney Princess (16,171)
- + See more

Interest

- Animals & Nature (682,287)
- Anime & Manga (24,742)
- Comics (8,060)
- Fantasy & Sci-Fi (228,280)
- Fashion (124,567)



Manley Tekno Best Friend Robot, Sakura
More Choices from \$1,000,000,000,000.00
★☆☆☆☆ * 14



A Really Expensive Rock
More Choices from \$500,000.00
★★★★☆ * 95



Disguise Funky Witch Costume Nose
\$99,996.98
★★★★☆ * 8



Rubies Costumes Inflatable Mallet
\$99,996.75
★★★★★ * 1



Ms. Popeye (Popeye) Plus Size Costume
\$99,996.31




Rubie's Costume Co Green Hornet Mask Costume
More Choices from \$99,996.01



Delta Force Kids Costume
\$99,995.77




1795 Flowing Hair Dollar MS62 NGC
\$91,900.00
Only 1 left in stock - order soon.




BARRACUDA NETWORKS Bma1050a-E3 3 Year Energize Updates For Barracuda Message Archiver 1050
\$88,301.54 ~~\$92,049.00~~




1806 Draped Bust Half Dollar MS65 NGC
\$78,800.00
Only 1 left in stock - order soon.




BARRACUDA NETWORKS Bma1050a-P3 3 Year Premium Support For Barracuda Message Archiver 1050
\$71,961.54 ~~\$75,749.00~~




BARRACUDA NETWORKS Byf1010a-H3 3 Year Instant Replacement For Barracuda Web Filter 1010
\$52,344.05 ~~\$55,099.00~~




CTA DIGITAL - IPAD W/RD/3/2 KIDS EASEL
\$50,000.00




Scotch 1427 Multi-Purpose Scissors - 7" Overall Length - Straight-left/right - Stainless Steel - Red, Silver
\$49,999.98




Graco RoadLazer RoadPak Tow-Behind System - Option 4 - 16N438
\$49,240.00



Princess Modular Play Set
\$47,056.92
★★★★★ * 1




Thomas Modular Play Set
★★★★★ * 1




BARRACUDA NETWORKS Byf1010a-P3 3 Year Premium Support For Barracuda Web Filter 1010
\$52,344.05 ~~\$55,099.00~~



FlexCourt Double Tennis Court 108'x120' - Complete kit
More Choices from \$41,455.00



1802 Draped Bust Two and a half Dollar MS61 NGC
\$36,800.00
Only 1 left in stock - order soon.



Aeronavics SkyJib-8 Ti-QR Professional Multirotor Helicopter
More Choices from \$36,500.00

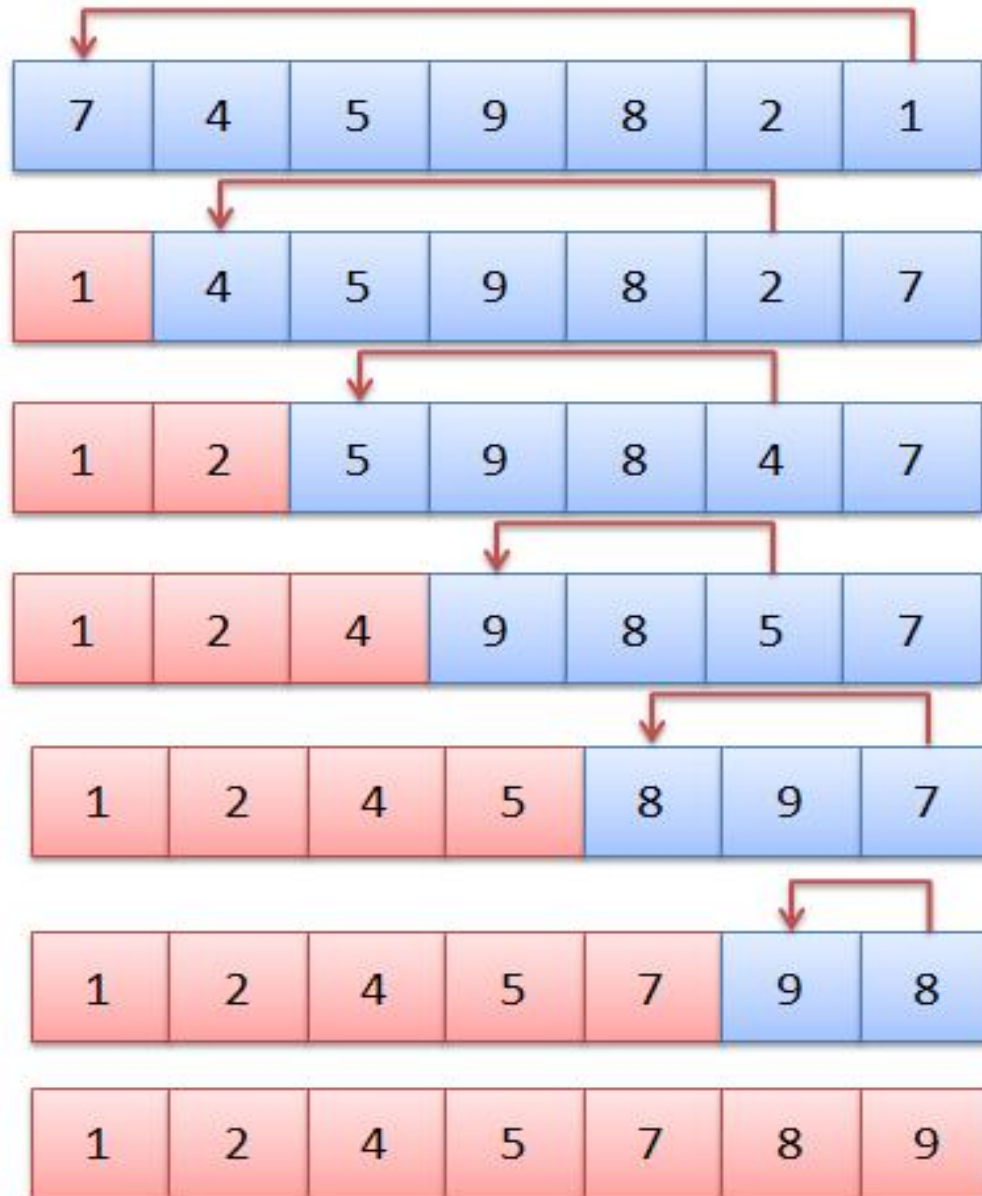
Algoritmos para ordenar elementos

- Baseado em comparações:
 - Bogo sort
 - Selection sort
 - Insertion sort
 - Bubble sort
 - Mergesort
 - Quicksort
 - Heapsort
- Ordenação em tempo linear:
 - Radix sort
- Ordenação de primeiros elementos (seleção parcial):
 - Partial Quicksort



Algoritmos de ordenação

Selection Sort



Selection Sort

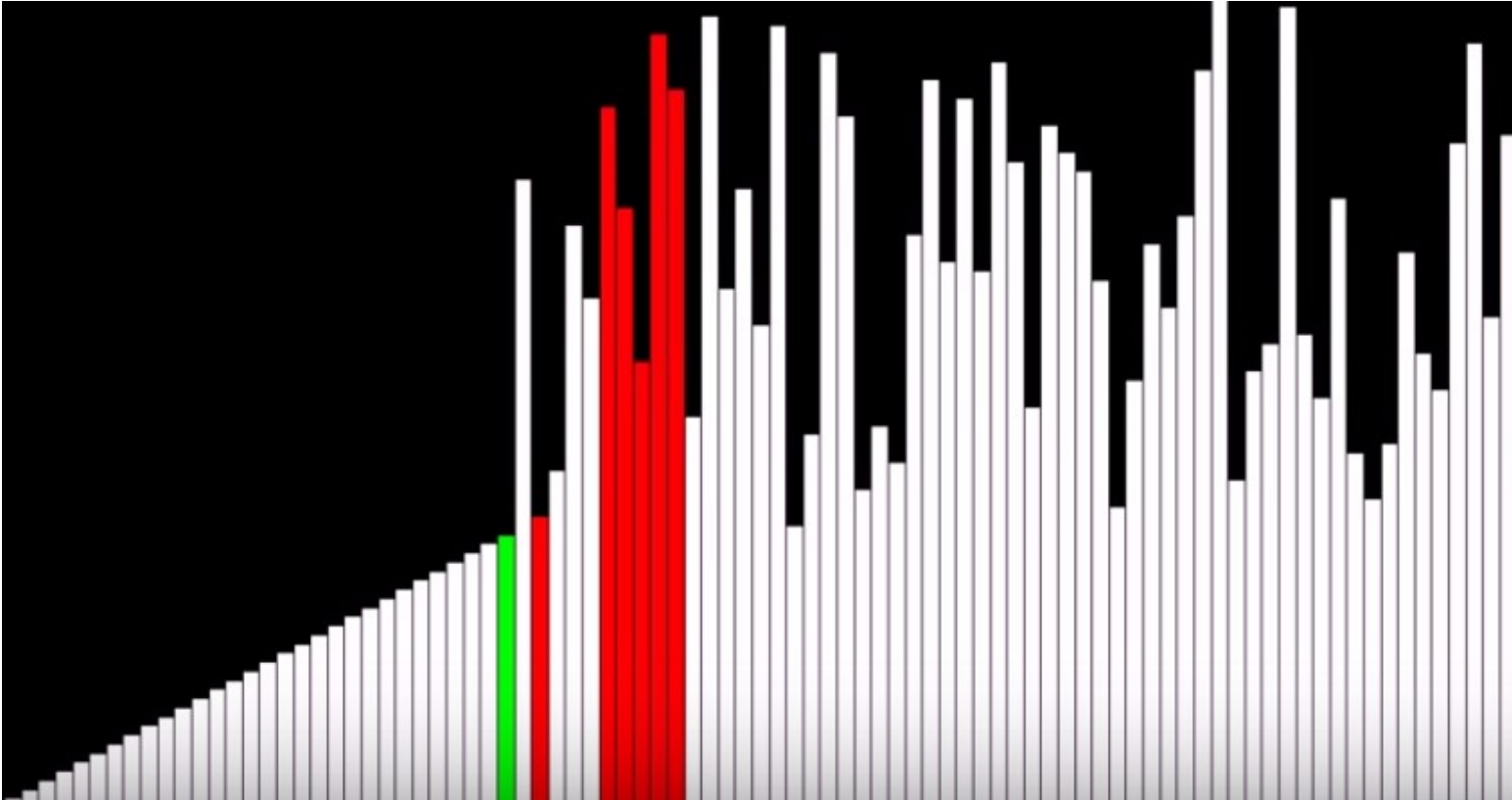
```
void SelectionSort (int v[], int n) {
    int i, j, iMin, aux;

    for (i=0; i<n-1; i=i+1) {
        iMin = i;

        for (j=i+1; j<n; j=j+1) {
            if (v[iMin]>v[j])
                iMin = j;
        }

        if (iMin!=i) {
            aux      = v[iMin];
            v[iMin] = v[i];
            v[i]     = aux;
        }
    }
}
```

Selection sort



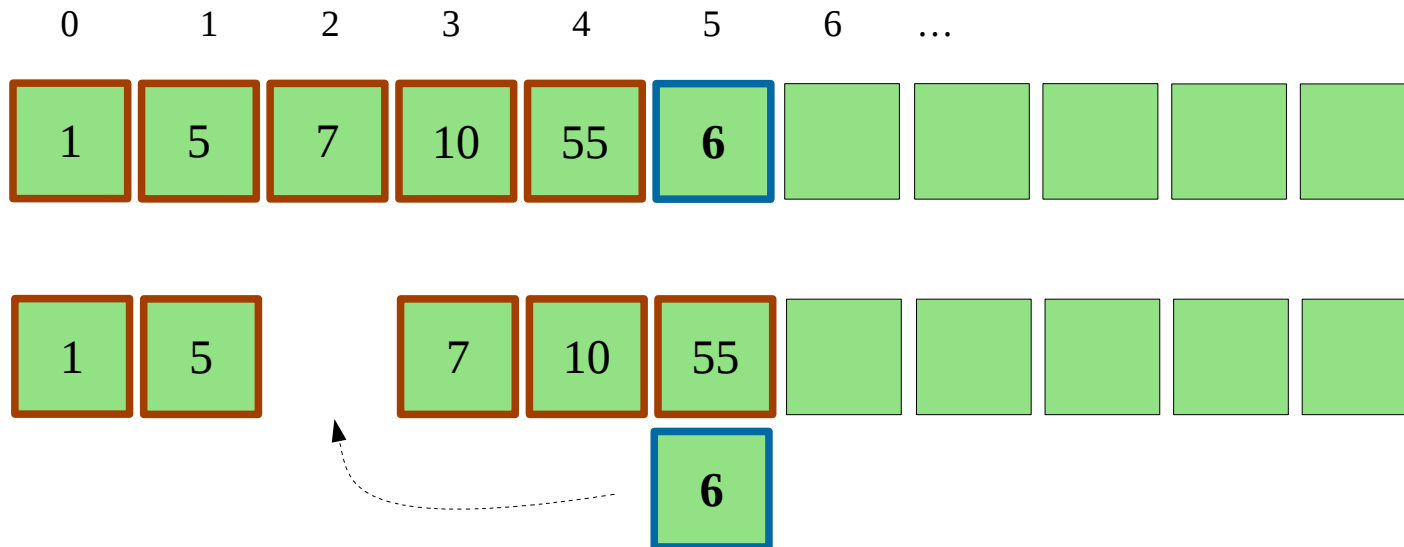
<https://www.youtube.com/watch?v=92BfuxHn2XE>

Insertion Sort



Método preferido dos jogadores de cartas

Em cada passo, a partir do $i=1$, o i -ésimo elemento da sequência fonte é apanhado e transferido para a sequência destino, sendo inserido no seu lugar apropriado.



Insertion Sort

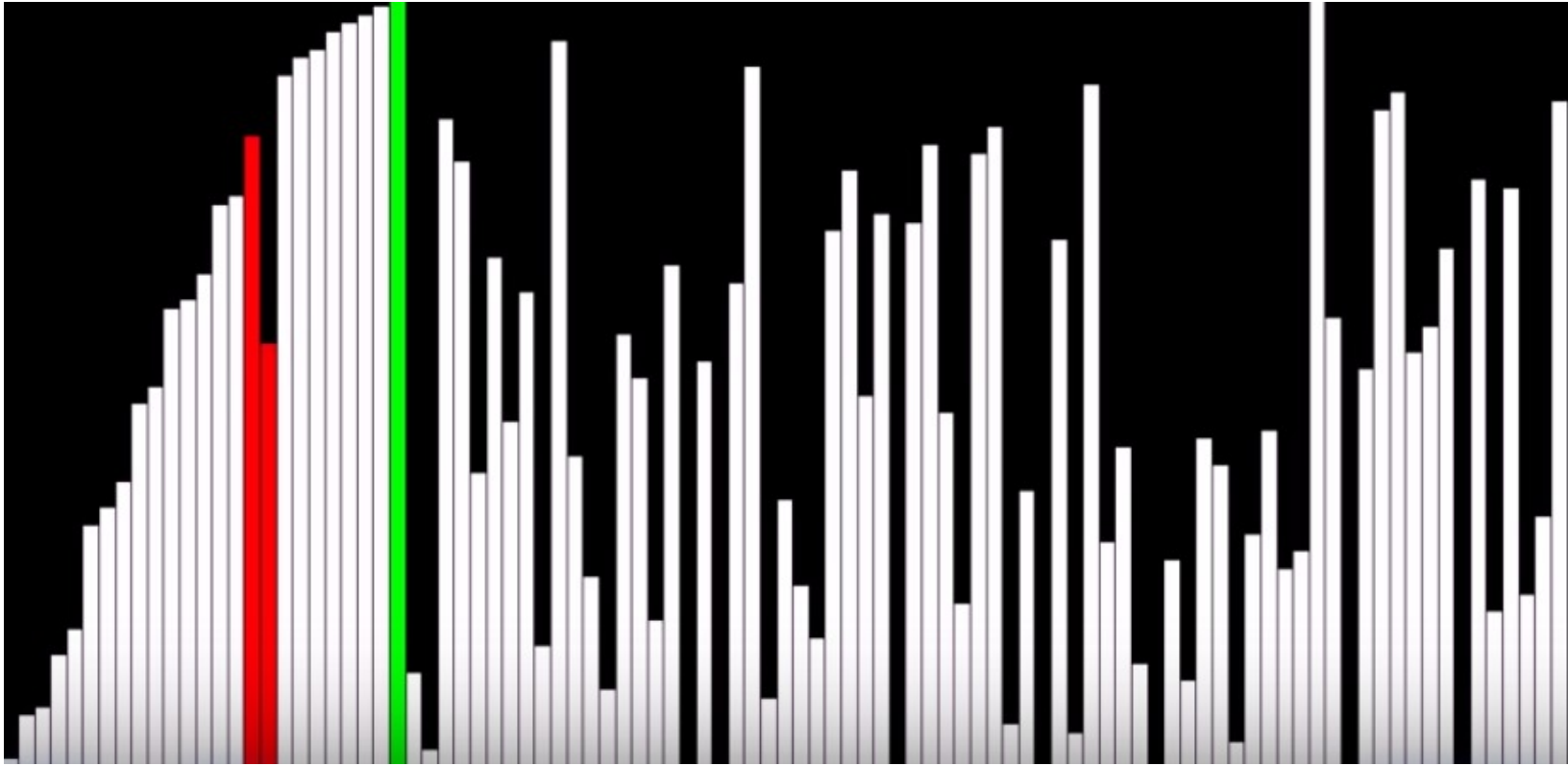
```
void InsertionSort (int v[], int n) {  
    int i, j, aux;  
  
    for (i=1; i<n; i=i+1) {  
        aux = v[i];  
  
        for (j=i-1; j>=0 && v[j]>aux; j=j-1) {  
            v[j+1] = v[j];  
        }  
  
        v[j+1] = aux;  
    }  
}
```

Número de comparações $T(n)$:

- No melhor caso: $T(n) = n-1$

- No pior caso: $T(n) = ?$

Insertion sort



<https://www.youtube.com/watch?v=8oJS1BMKE64>

Bubble Sort

- O algoritmo de ordenação baseado em **troca**, consiste em intercalar pares de elementos que não estão em ordem até que não exista mais pares.
- **O princípio do bolha é a troca de valores entre posições consecutivas fazendo com que os valores mais altos “borbulhem” para o final do vetor.**



Bubble Sort

```
void BubbleSort1 (int v[], int n) {
    int k, i, aux;

    for (k=n-1; k>=1; k=k-1) {
        for (i=0; i<k; i=i+1) {
            if (v[i]>v[i+1]) {
                aux = v[i];
                v[i] = v[i+1];
                v[i+1] = aux;
            }
        }
    }
}
```

Número de comparações $T(n)$:

- No melhor caso: $T(n) = n^2/2 - n/2$
- No pior caso: $T(n) = n^2/2 - n/2$

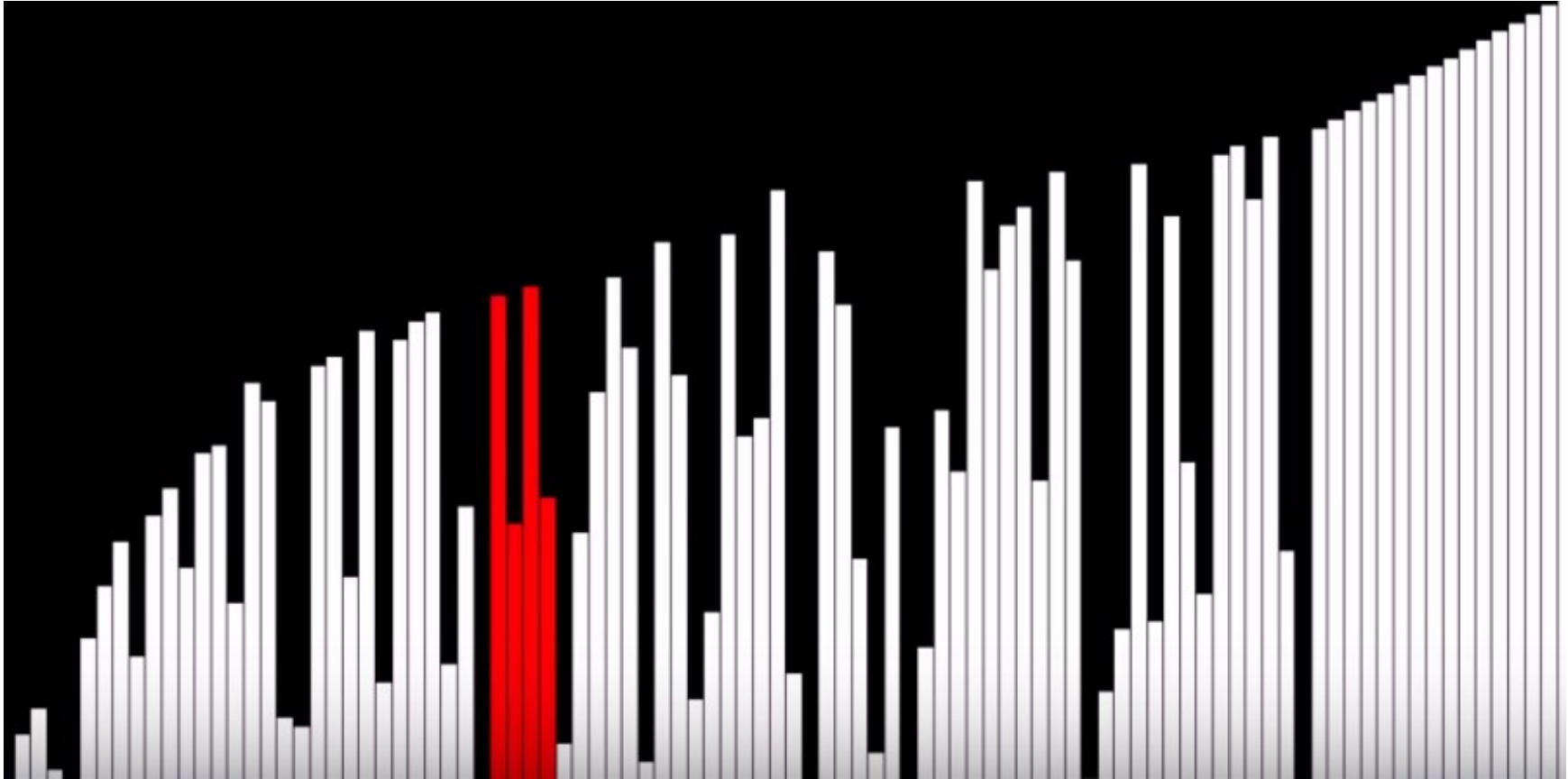
```
void BubbleSort2 (int v[], int n) {
    int i, aux, hasChanged;

    do {
        hasChanged = 0;
        for (i=0; i<n-1; i=i+1) {
            if (v[i]>v[i+1]) {
                aux = v[i];
                v[i] = v[i+1];
                v[i+1] = aux;
                hasChanged = 1;
            }
        }
    }while(hasChanged==1);
}
```

Número de comparações $T(n)$:

- No melhor caso: $T(n) = n-1$
- No pior caso: $T(n) = n^2/2 - n/2$

Bubble sort



<https://www.youtube.com/watch?v=Cq7SMsQBEUw>



Atividade 02

- Tempo de processamento (segundos)**
- Número de comparações**

Atividade 02: Teste empírico

```
1 # Testando os algoritmos de ordenacao
2 gcc insertionSort.c -o insertionSort.exe
3 gcc selectionSort.c -o selectionSort.exe
4 gcc bubbleSort.c -o bubbleSort.exe
5
6 ./selectionSort.exe 50000 < vetorAleatorio50000.dat
7 ./insertionSort.exe 50000 < vetorAleatorio50000.dat
8 ./bubbleSort.exe 50000 < vetorAleatorio50000.dat
9
10 ./selectionSort.exe 50000 < vetorCrescente50000.dat
11 ./insertionSort.exe 50000 < vetorCrescente50000.dat
12 ./bubbleSort.exe 50000 < vetorCrescente50000.dat
13
14 ./selectionSort.exe 50000 < vetorDecrescente50000.dat
15 ./insertionSort.exe 50000 < vetorDecrescente50000.dat
16 ./bubbleSort.exe 50000 < vetorDecrescente50000.dat
17
18 ./selectionSort.exe 50000 < vetorPCrescente50000.dat
19 ./insertionSort.exe 50000 < vetorPCrescente50000.dat
20 ./bubbleSort.exe 50000 < vetorPCrescente50000.dat
21
22 ./selectionSort.exe 50000 < vetorPDecrescente50000.dat
23 ./insertionSort.exe 50000 < vetorPDecrescente50000.dat
24 ./bubbleSort.exe 50000 < vetorPDecrescente50000.dat
```

<http://sortvis.org/>

<http://sorting.at/>