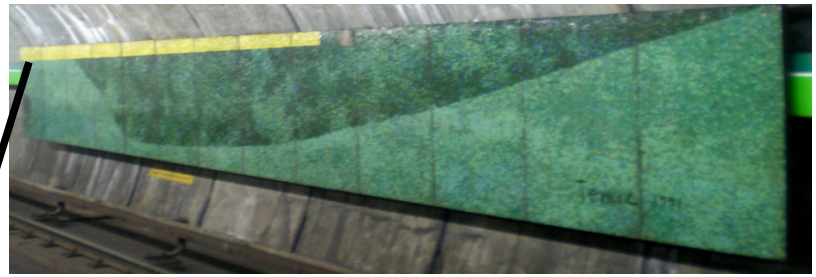
Aula 15:
- Métodos simples de busca
- Métodos simples de ordenação

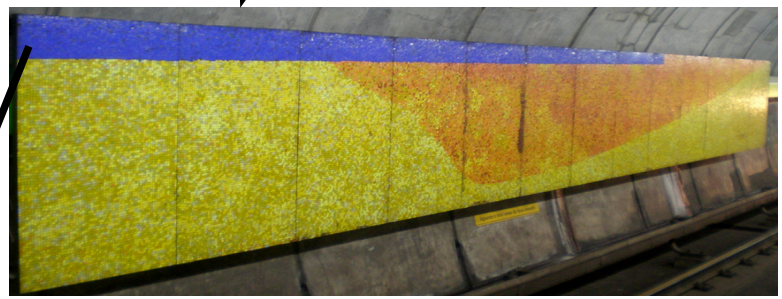
Prof. Jesús P. Mena-Chalco
jesus.mena@ufabc.edu.br

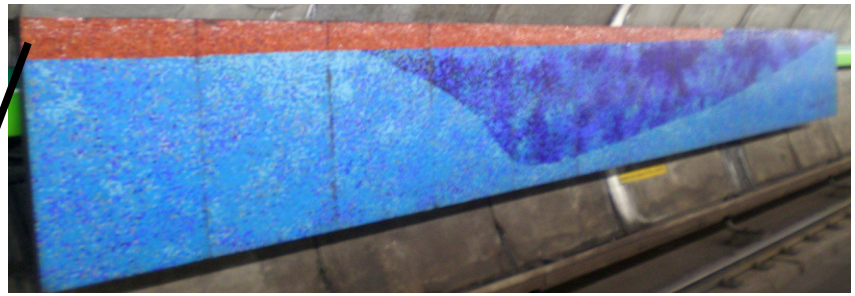
3Q-2017

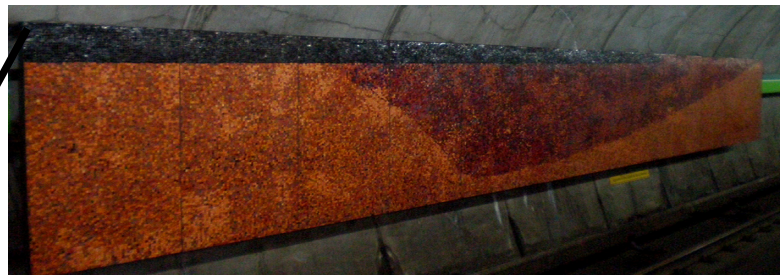
Listas encadeadas

Estação Consolação: “Quatro estações” (1991). Mosaico abstrato feito de pastilhas vitrificadas por Tomie Ohtake





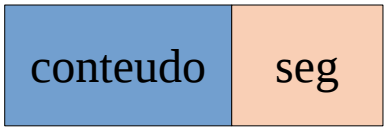


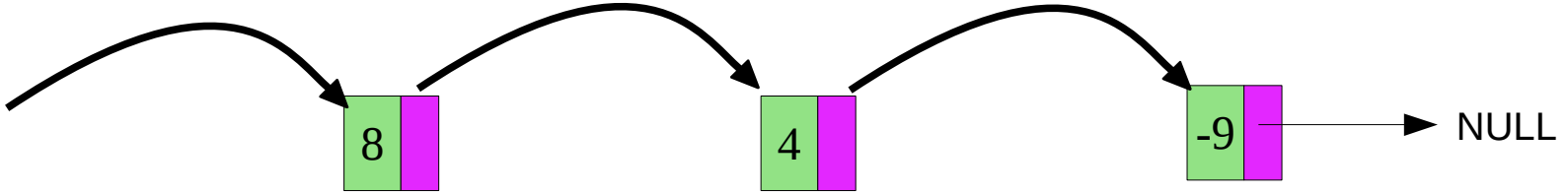


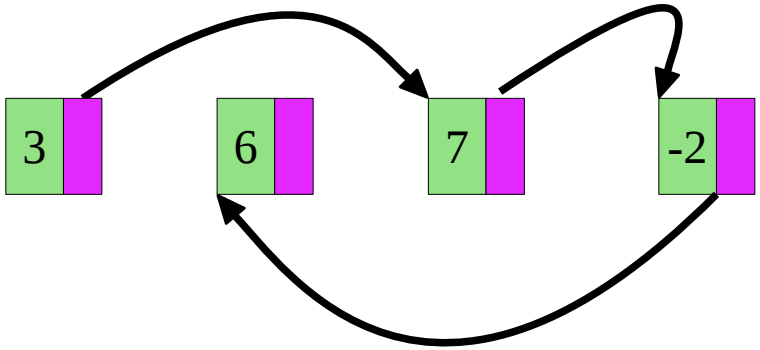
NULL

Exemplos lista encadeada

```
struct cel {
    int conteudo;
    struct cel *seg;
};

typedef struct cel celula;
```



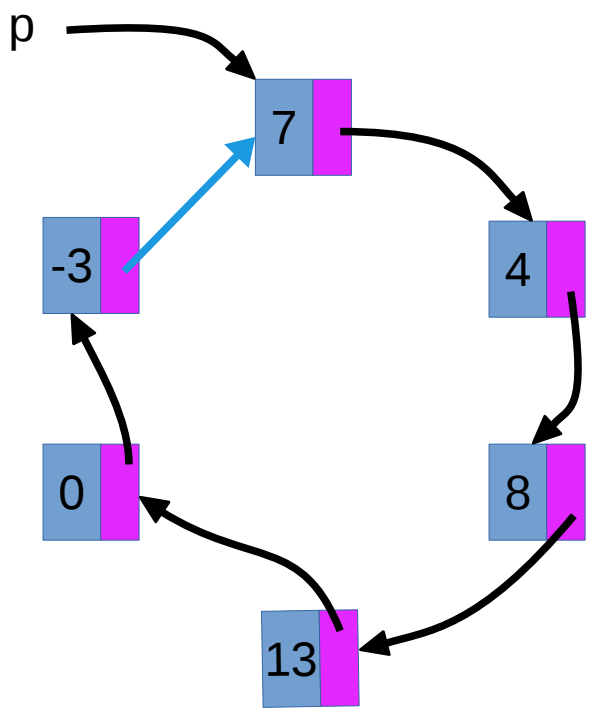




No caso da última célula, o endereço é NULL

Outros tipos de listas encadeadas:

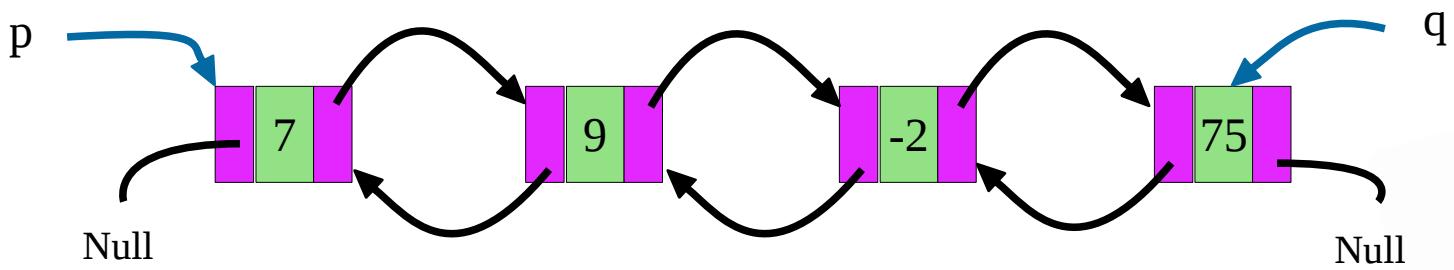
- **Lista circular**



A última célula aponta para a primeira

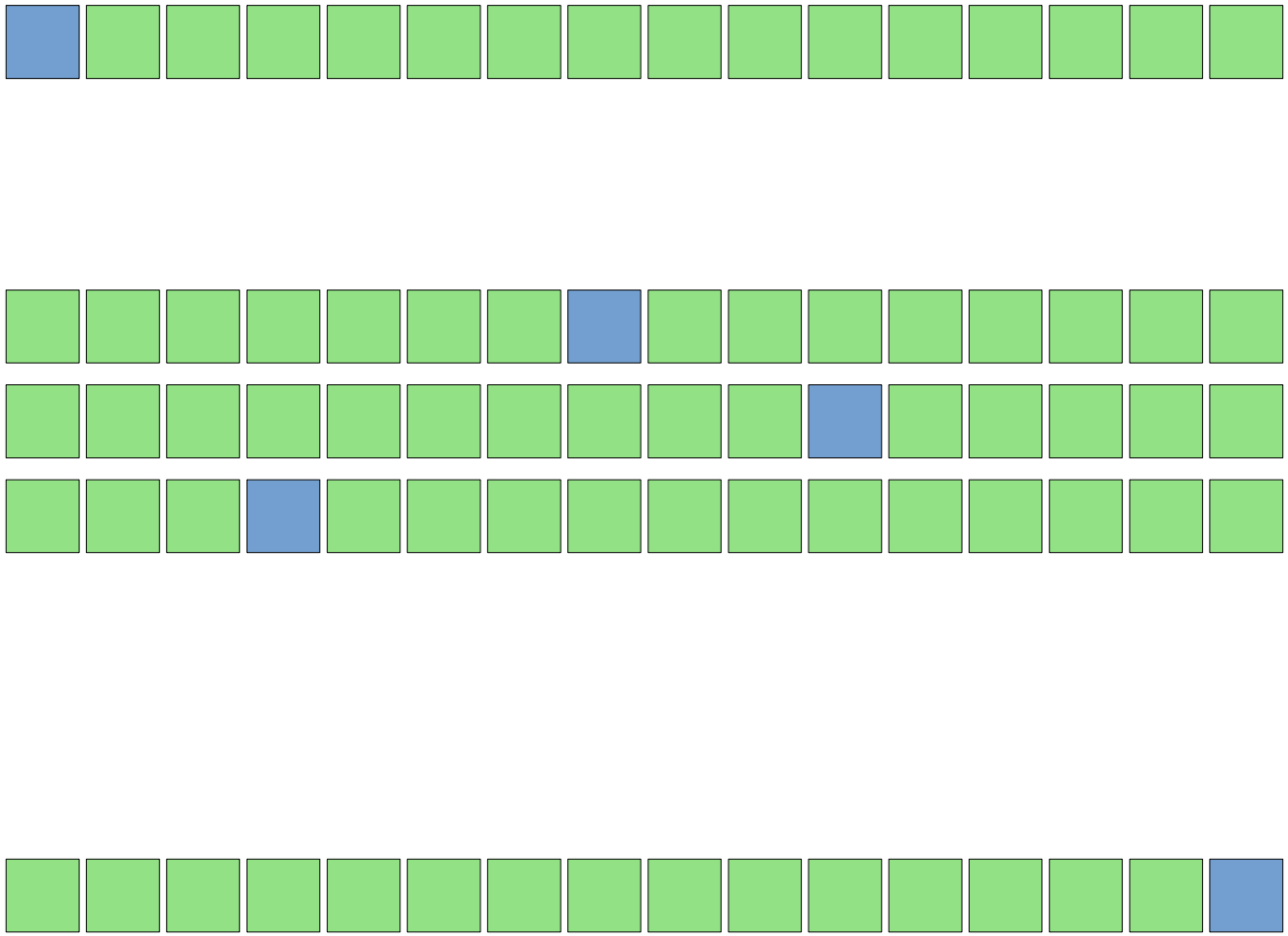
Outros tipos de listas encadeadas:

- **Lista duplamente encadeada**



Cada célula contém o endereço da célula anterior e o da seguinte

Busca de um elemento em um vetor



```c
int buscaChave(int chave , int A[], int n) {
    int i;

    for(i=0; i<n; i++) {
        if (chave == A[i])
            return i;
    }
    return -1;
}
```

(*) Fonte: P. Feofiloff. Algoritmos em Linguagem C. 1ª Edição, Editora Campos, 2008.

Busca Linear

```
int buscaChave2(int chave , int A[], int n) {
    int i;
    i = n-1;

    while(i>=0 && A[i]!=chave) {
        i-=1;
    }
    return i;
}
```

Versão Elegante!

Busca Linear Recursiva

Crie uma função recursiva para identificar um elemento em um vetor.

Assinatura / Prototipo:

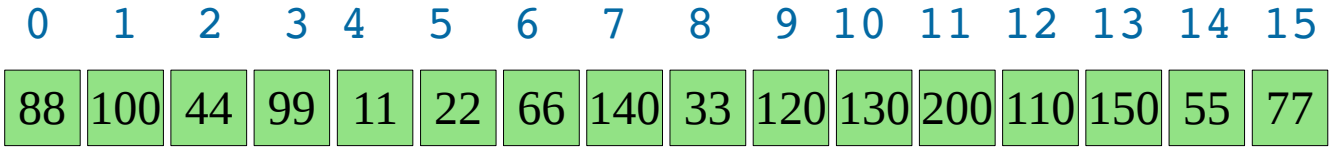
```
int buscaChaveRec (int chave, int A[], int n)
```

Busca Linear Recursiva

```
int buscaChaveRec (int chave, int A[], int n) {
    if (n==0)
        return -1;
    if (chave==A[n-1])
        return n-1;

    return buscaChaveRec(chave, A, n-1);
}


int main(int argc, char *argv[])
```

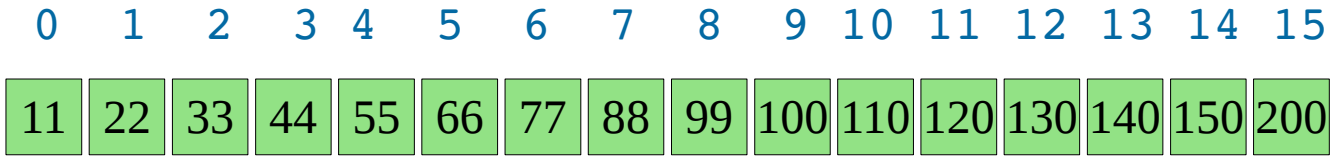
Busca Binária

Vetor sem ordem



Melhor caso: 1
Pior caso: n

Vetor crescente



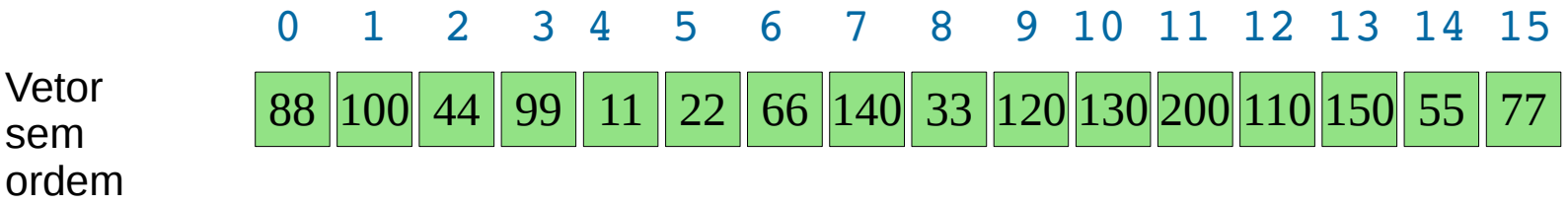
Melhor caso: 1
Pior caso: log(n)

Busca Binaria Recursiva

```
int buscaBinariaRec (int chave, int A[], int inf, int sup) {
    int meio = (inf+sup)/2;

    if (inf>sup)
        return -1;

    if (chave==A[meio])
        return meio;

    if (chave<A[meio])
        return buscaBinariaRec(chave, A, inf, meio-1);
    else
        return buscaBinariaRec(chave, A, meio+1, sup);
}
```

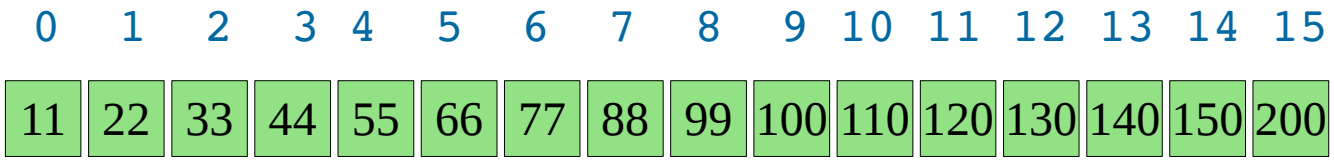
Melhor caso: 1
Pior caso: log(n)

```
buscaBinariaRec(9000000, vetor, 0, n-1) );
```

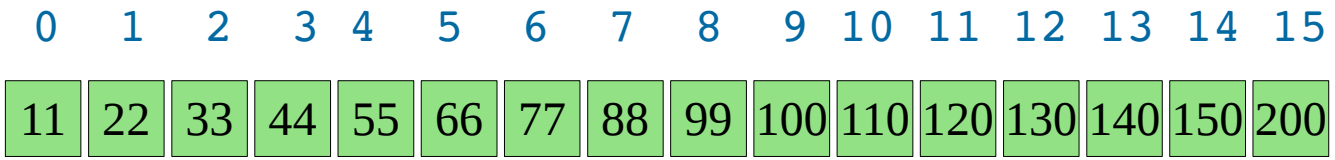


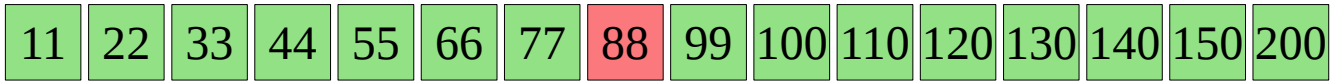
Melhor caso: 1
Pior caso: n

Chave = 101


0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
11 22 33 44 55 66 77 88 99 100 110 120 130 140 150 200

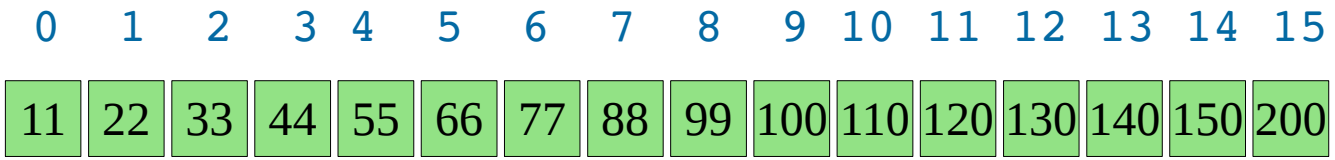
Chave = 101

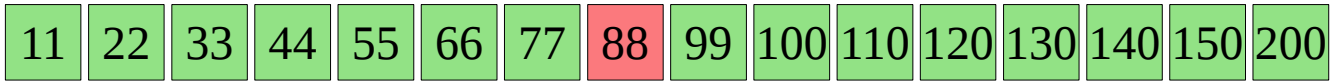




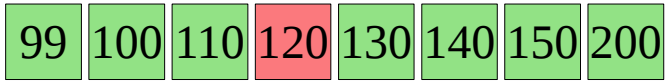
Inf = 0
Sup = 15

Chave = 101



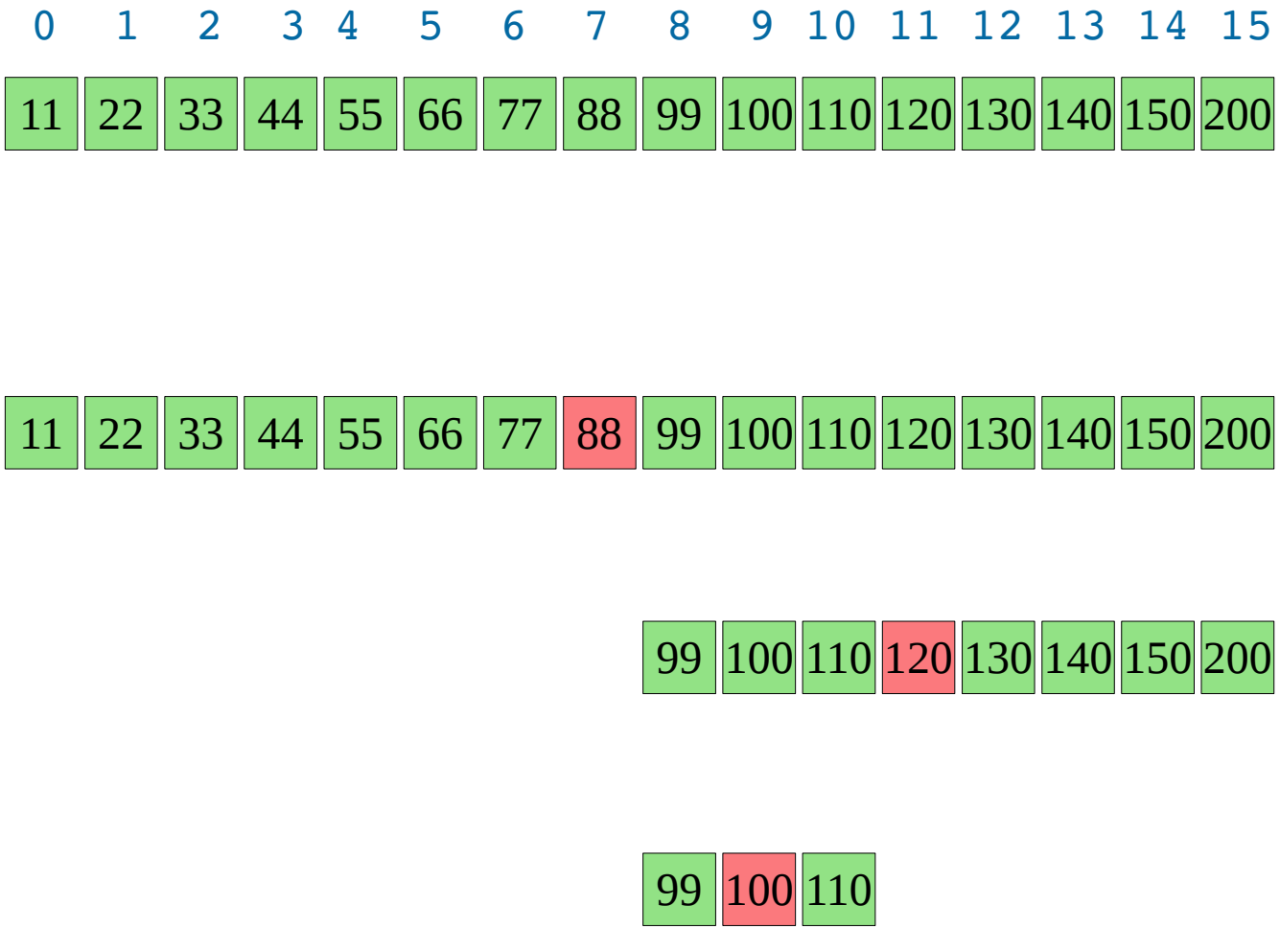


Inf = 0
Sup = 15

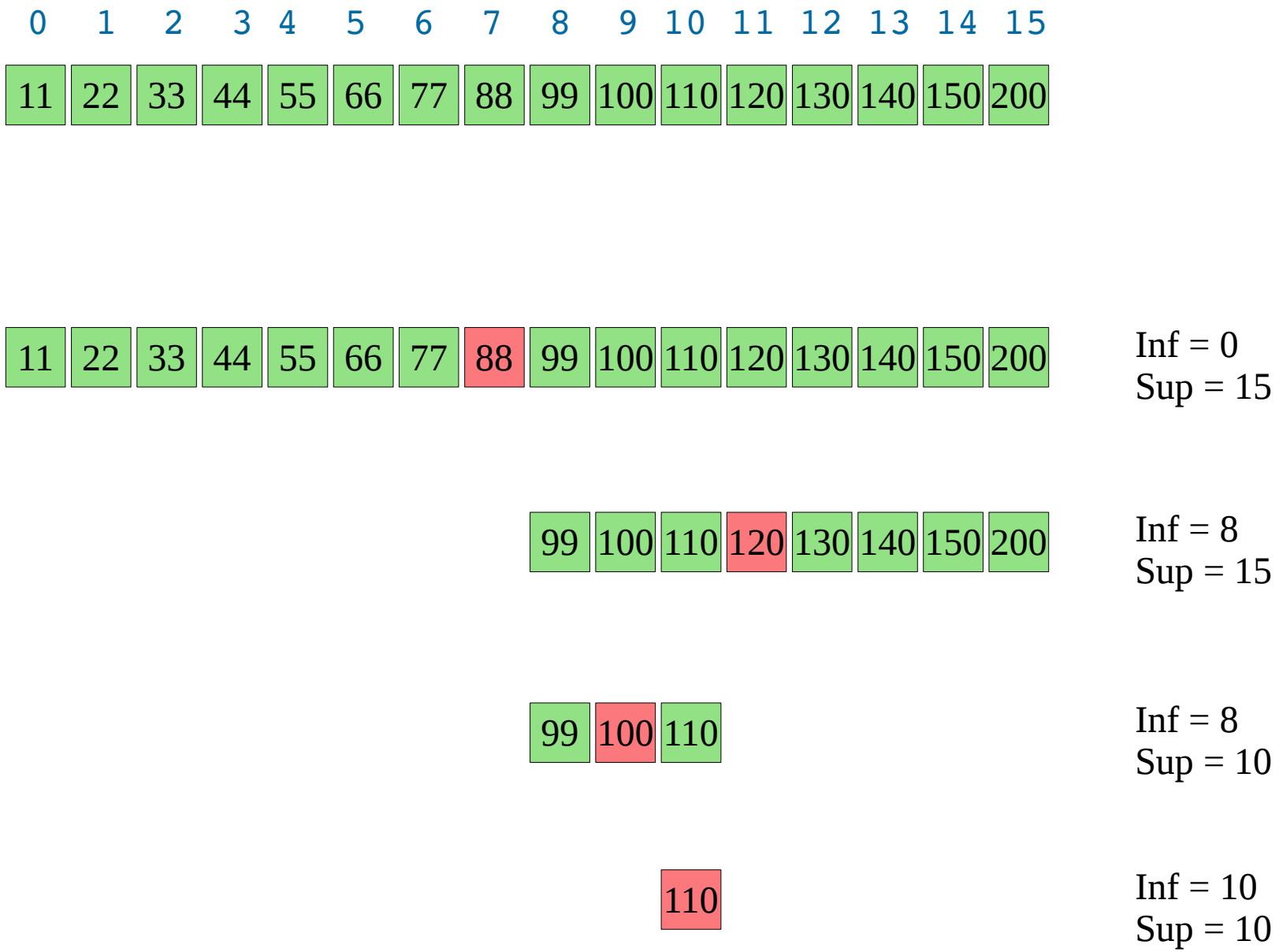


Inf = 8
Sup = 15

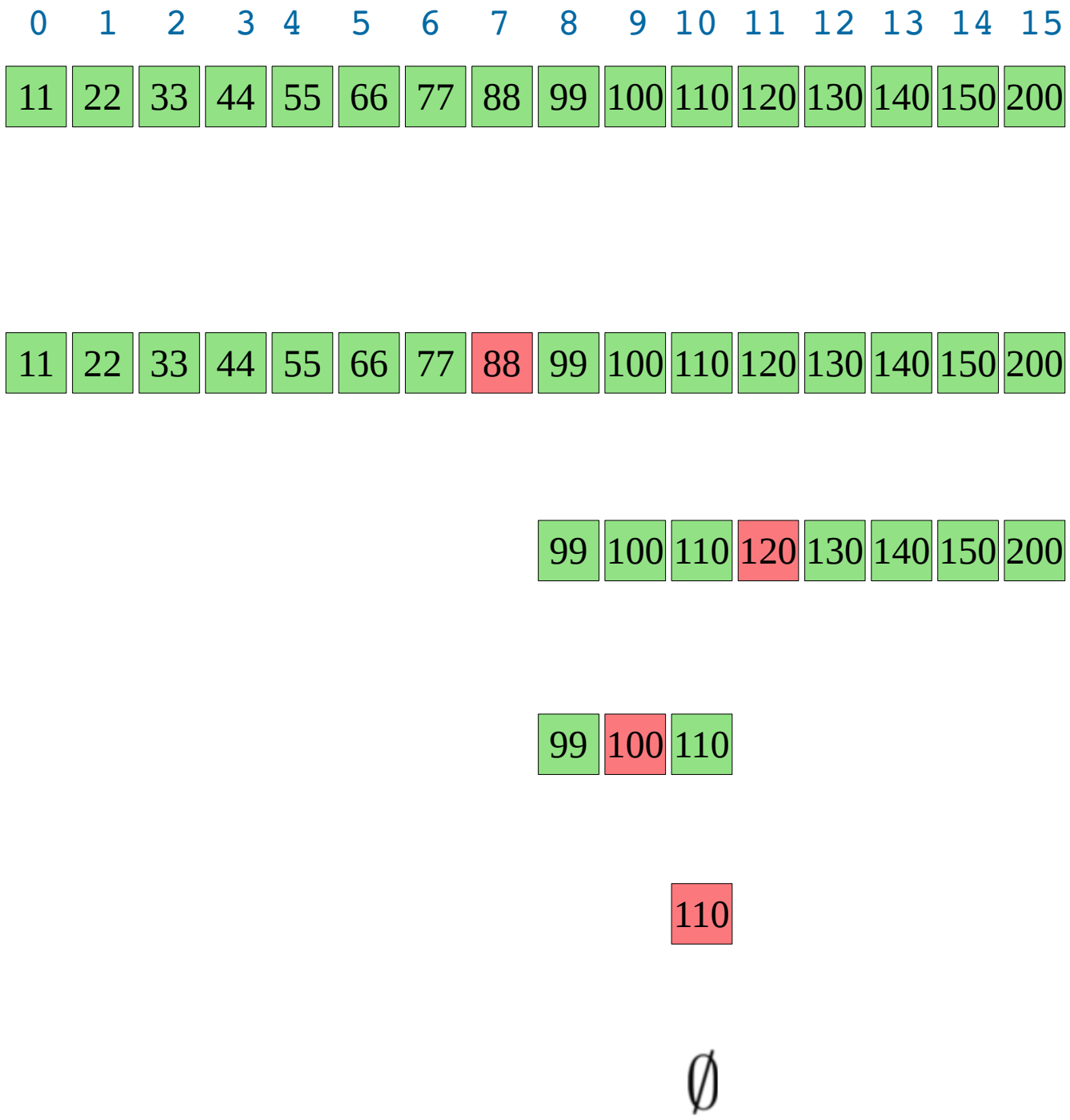
Chave = 101


0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
11 22 33 44 55 66 77 88 99 100 110 120 130 140 150 200
11 22 33 44 55 66 77 88 99 100 110 120 130 140 150 200
Inf = 0
Sup = 15
99 100 110 120 130 140 150 200
Inf = 8
Sup = 15
99 100 110
Inf = 8
Sup = 10

Chave = 101


0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
11 22 33 44 55 66 77 88 99 100 110 120 130 140 150 200
11 22 33 44 55 66 77 88 99 100 110 120 130 140 150 200
Inf = 0
Sup = 15
99 100 110 120 130 140 150 200
Inf = 8
Sup = 15
99 100 110
Inf = 8
Sup = 10
110
Inf = 10
Sup = 10

Chave = 101


0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
11 22 33 44 55 66 77 88 99 100 110 120 130 140 150 200
11 22 33 44 55 66 77 88 99 100 110 120 130 140 150 200
Inf = 0
Sup = 15
99 100 110 120 130 140 150 200
Inf = 8
Sup = 15
99 100 110
Inf = 8
Sup = 10
110
Inf = 10
Sup = 10
∅
Inf = 10
Sup = 9

Para pensar... P2?

A) Se a chave não estiver no vetor, então devolver o índice do elemento mais próximo à chave.

B) Faça uma função iterativa para a busca binaria

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.

21/03/2016

amazon Try Prime | Toys & Games | Easter Shop

Shop by Department | Your Amazon.com | Today's Deals | Gift Cards | Sell | Help | Hello. Sign in Your Account | Try Prime | Your Lists | 0 Cart

Toys & Games | Deals | STEM Toys | Best Sellers | Preschool Toys | Boys' Toys | Girls' Toys | Kids' Birthdays | Games | New Releases | Hobby, Models & Trains | Best Toys of the Month | Outdoor

1-60 of 5,148,058 results for Toys & Games

Sort by Price: High to Low

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 $1,000,000,000,000,000.00

★★½☆☆ ▾ 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 Barracuda Networks Bma1050a-E3 3 Year Energize Updates For Barracuda Message Archiver 1050

$88,301.54 ~~$92,949.00~~

1806 Draped Bust Half Dollar MS65 NGC

$78,800.00

Only 1 left in stock - order soon.

BARRACUDA NETWORKS Barracuda Networks Bma1050a-P3 3 Year Premium Support For Barracuda Message Archiver 1050

$71,961.54 ~~$75,749.00~~

BARRACUDA NETWORKS 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

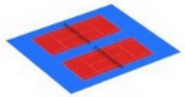
$47,056.92

★★★★★ ▾ 1

Thomas Modular Play Set

★★★★★ ▾ 1

BARRACUDA NETWORKS Barracuda Networks Byf1010a-P3 3 Year Premium Support For Barracuda Web



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

Por que ordenar?

- **Busca de elementos:**
  O usuário geralmente quer os dados ordenados.

- **Deduplicação de elementos:**
  Ordenar é muito útil para eliminar duplicações.

- Projeção de objetos em um espaço 3D.

- Criação de uma árvore geradora mínima.

- Ordenar é o primeiro passo para “carregar” os dados em uma estrutura tipo árvore B+.

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

O problema de ordenar na forma crescente

- Um vetor v[0..n-1] é crescente se v[0] ≤ v[1] ≤ … ≤ [n-1]

- O problema de ordenação de um vetor consiste em **rearranjar (ou seja, permutar) os elementos de um vetor v[0..n-1] de tal modo que ele se torne crescente.**

- **Vetores ordenados na forma crescente:**
  - {1, 1, 1, 1, 1, 1, 1, 1}
  - {0, 1, 1, 1, 2, 3, 4, 4, 4, 4, 4, 4, 100}

Verificar se um vetor v[0..n-1] é crescente

```c
int crescente(int v[], int n) {
    int i;

    for (i=0; i<n-1; i=i+1)
        if (v[i]>v[i+1])
            return 0;

    return 1;
}
```

```c
int crescente2(int v[], int n) {
    int i=0;

    while(i<n-1 && v[i]<=v[i+1]) {
        i = i+1;
    }

    if (i==n-1)
        return 1;
    else
        return 0;
}
```

Número de comparações T(n):
- No melhor caso: T(n) = 1
- No pior caso: T(n) = n-1

T(n) = O(n)

Crie uma versão recursiva da função crescente

Atividade em aula

Vetor crecente (recursiva)

```c
int crescenteRec(int v[], int n) {
    if (n==1)
        return 1;

    if (v[n-2]<=v[n-1])
        return crescenteRec(v, n-1);
    else
        return 0;
}
```

```c
int main()
{
    int v[] = {0, 1, 1, 1, 2, 3, 4, 4, 4, 4, 4, 4, 100};
    int n=sizeof(v)/sizeof(v[0]);

    printf("%d\n", crescenteRec(v, n) );
}
```

Número de comparações T(n):
- No melhor caso: T(n) = 1
- No pior caso: T(n) = n-1

T(n) = O(n)

Embaralhando um vetor

```c
#include <time.h>
#include <stdio.h>
#include <stdlib.h>

void embaralhar(int v[], int n) {
    int i, r, temp;
    srand ( time(NULL) );

    for (i=0; i<n; i=i+1) {
        r = rand()%n;

        temp = v[i];
        v[i] = v[r];
        v[r] = temp;
    }
}
```

Número de trocas de elementos T(n):
- No melhor caso: T(n) = n
- No pior caso: T(n) = n

T(n) = O(n)

(1)
Bogo sort
Miracle sort
Monkey sort
Stupid sort

```c
void imprimir(int v[], int n) {
    int i;
    for (i=0; i<n; i=i+1) {
        printf("%d ", v[i] );
    }
    printf("\n");
}
```

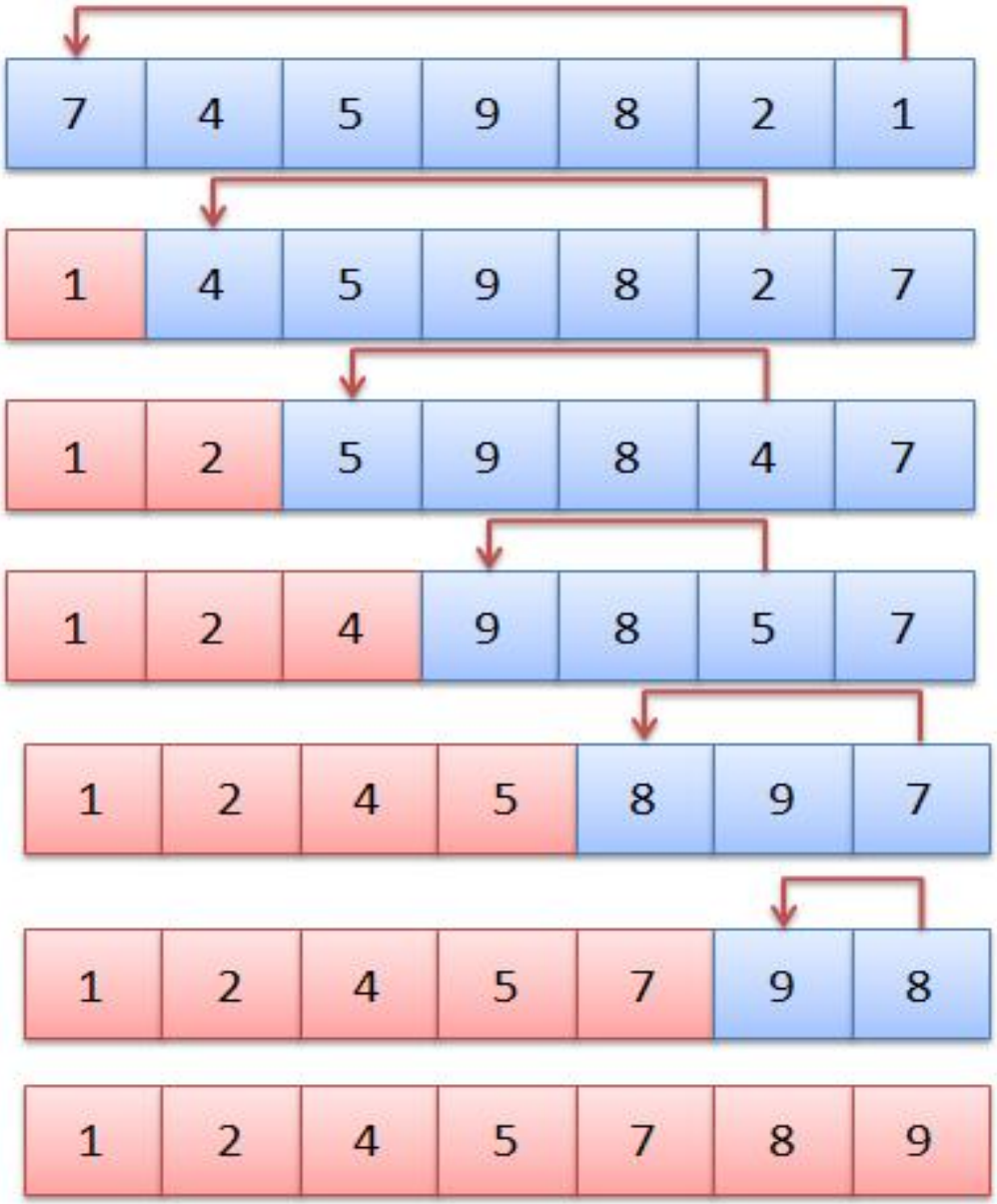
```c
void bogoSort(int v[], int n) {
    while (crescenteRec(v,n)==0) {
        embaralhar(v,n);
    }
}
```

```c
int main()
{
    int v[] = {100,4,999,555,222,3,0,-1};
    int n=sizeof(v)/sizeof(v[0]);

    imprimir(v, n);
    bogoSort(v, n);
    imprimir(v, n);
}
```

Número de comparações T(n):
- No melhor caso: T(n) = n-1
- No pior caso: T(n) = ?

(2)
Selection Sort:
Algoritmo de ordenação por seleção

Selection Sort



Animação: https://www.cs.usfca.edu/~galles/visualization/ComparisonSort.html

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;
        }
    }
}
```

Complexidade computacional: No pior caso?