

# AULA 16

# Busca em vetor ordenado



Fonte: <http://www.php5dp.com/>

PF 7.1 a 7.8

<http://www.ime.usp.br/~pf/algoritmos/aulas/bubi2.html>

# Busca em vetor ordenado

Um vetor  $v[0..n-1]$  é **crescente** se

$$v[0] \leq v[1] \leq v[2] \leq \dots \leq v[n-1].$$

**Problema:** Dado um número  $x$  e um vetor **crescente**  $v[0..n-1]$  encontrar um índice  $m$  tal que  $v[m] == x$ .

Entra:  $x == 50$

|   |    |    |    |    |    |    |    |    |    |       |    |
|---|----|----|----|----|----|----|----|----|----|-------|----|
|   | 0  |    |    |    |    |    | 7  |    |    | $n-1$ |    |
| v | 10 | 20 | 25 | 35 | 38 | 40 | 44 | 50 | 55 | 65    | 99 |

Sai:  $m == 7$

# Busca em vetor ordenado

Entra:  $x == 57$

|     |    |    |    |    |    |    |    |    |    |    |       |
|-----|----|----|----|----|----|----|----|----|----|----|-------|
|     | 0  |    |    |    |    |    |    |    |    |    | $n-1$ |
| $v$ | 10 | 20 | 25 | 35 | 38 | 40 | 44 | 50 | 55 | 65 | 99    |

Sai:  $m == -1$  ( $x$  não está em  $v$ )

## Busca sequencial

```
int buscaSequencial(int x, int n, int v[])  
{  
1   int m = 0;  
2   while (/*1*/ m < n && v[m] < x) m++;  
3   if (m < n && v[m] == x)  
4       return m;  
5   return -1;  
}
```

# Exemplo

x == 55

|   |    |    |    |    |    |    |    |    |    |    |    |
|---|----|----|----|----|----|----|----|----|----|----|----|
|   | 0  |    |    |    |    |    |    |    |    |    | 10 |
| v | 10 | 20 | 25 | 35 | 38 | 40 | 44 | 50 | 55 | 65 | 99 |

|   |    |    |    |    |    |    |    |    |    |    |    |
|---|----|----|----|----|----|----|----|----|----|----|----|
|   | m  |    |    |    |    |    |    |    |    |    | 10 |
| v | 10 | 20 | 25 | 35 | 38 | 40 | 44 | 50 | 55 | 65 | 99 |

|   |    |    |    |    |    |    |    |    |    |    |    |
|---|----|----|----|----|----|----|----|----|----|----|----|
|   | 0  | m  |    |    |    |    |    |    |    |    | 10 |
| v | 10 | 20 | 25 | 35 | 38 | 40 | 44 | 50 | 55 | 65 | 99 |

|   |    |    |    |    |    |    |    |    |    |    |    |
|---|----|----|----|----|----|----|----|----|----|----|----|
|   | 0  |    | m  |    |    |    |    |    |    |    | 10 |
| v | 10 | 20 | 25 | 35 | 38 | 40 | 44 | 50 | 55 | 65 | 99 |

|   |    |    |    |    |    |    |    |    |    |    |    |
|---|----|----|----|----|----|----|----|----|----|----|----|
|   | 0  |    |    | m  |    |    |    |    |    |    | 10 |
| v | 10 | 20 | 25 | 35 | 38 | 40 | 44 | 50 | 55 | 65 | 99 |

# Exemplo

x == 55

|   |    |    |    |    |    |    |    |    |    |    |    |
|---|----|----|----|----|----|----|----|----|----|----|----|
|   | 0  |    |    |    | m  |    |    |    |    |    | 10 |
| v | 10 | 20 | 25 | 35 | 38 | 40 | 44 | 50 | 55 | 65 | 99 |

|   |    |    |    |    |    |    |    |    |    |    |    |
|---|----|----|----|----|----|----|----|----|----|----|----|
|   | 0  |    |    |    |    | m  |    |    |    |    | 10 |
| v | 10 | 20 | 25 | 35 | 38 | 40 | 44 | 50 | 55 | 65 | 99 |

|   |    |    |    |    |    |    |    |    |    |    |    |
|---|----|----|----|----|----|----|----|----|----|----|----|
|   | 0  |    |    |    |    |    | m  |    |    |    | 10 |
| v | 10 | 20 | 25 | 35 | 38 | 40 | 44 | 50 | 55 | 65 | 99 |

|   |    |    |    |    |    |    |    |    |    |    |    |
|---|----|----|----|----|----|----|----|----|----|----|----|
|   | 0  |    |    |    |    |    |    | m  |    |    | 10 |
| v | 10 | 20 | 25 | 35 | 38 | 40 | 44 | 50 | 55 | 65 | 99 |

|   |    |    |    |    |    |    |    |    |    |    |    |
|---|----|----|----|----|----|----|----|----|----|----|----|
|   | 0  |    |    |    |    |    |    |    | m  |    | 10 |
| v | 10 | 20 | 25 | 35 | 38 | 40 | 44 | 50 | 55 | 65 | 99 |

## Correção

Relação **invariante** chave:

$(i0)$  em  $/*1*/$  vale que:  $v[m-1] < x$ . ♥

$x == 55$

|   |    |    |    |    |    |    |    |    |    |    |    |
|---|----|----|----|----|----|----|----|----|----|----|----|
|   | 0  |    |    |    |    |    | m  |    |    | 10 |    |
| v | 10 | 20 | 25 | 35 | 38 | 40 | 44 | 50 | 55 | 65 | 99 |

A relação  $(i0)$  vale no começo da primeira iteração se supusermos que  $v[-1] = -\infty$ .

No início da última iteração  $m \geq n$  ou  $v[m] \geq x$ .

Portanto, se a função devolve  $-1$ ,  
então  $x$  não está em  $v[0..n-1]$

## Consumo de tempo buscaSequencial

Se a execução de cada linha de código consome  
**1 unidade** de tempo o consumo total é:

| linha | todas as execuções da linha |             |
|-------|-----------------------------|-------------|
| 1     | $= 1$                       | $= 1$       |
| 2     | $\leq n + 1$                | $\approx n$ |
| 3     | $= 1$                       | $= 1$       |
| 4     | $\leq 1$                    | $\leq 1$    |
| 5     | $\leq 1$                    | $\leq 1$    |
| total | $\leq n + 3$                | $= O(n)$    |

# Conclusão

O consumo de tempo do algoritmo `buscaSequencial` no pior caso é proporcional a  $n$ .

O consumo de tempo do algoritmo `buscaSequencial` é  $O(n)$ .

# Busca binária

```
int buscaBinaria(int x, int n, int v[]) {  
    int e, m, d;  
1   e = 0; d = n-1;  
2   while (/*1*/ e <= d) {  
3       m = (e + d)/2;  
4       if (v[m] == x) return m;  
5       if (v[m] < x) e = m + 1;  
6       else d = m - 1;  
    }  
7   return -1;  
}
```

# Exemplo

x == 48

0 10

v

|    |    |    |    |    |    |    |    |    |    |    |
|----|----|----|----|----|----|----|----|----|----|----|
| 10 | 20 | 25 | 35 | 38 | 40 | 44 | 50 | 55 | 65 | 99 |
|----|----|----|----|----|----|----|----|----|----|----|

e d

v

|    |    |    |    |    |    |    |    |    |    |    |
|----|----|----|----|----|----|----|----|----|----|----|
| 10 | 20 | 25 | 35 | 38 | 40 | 44 | 50 | 55 | 65 | 99 |
|----|----|----|----|----|----|----|----|----|----|----|

e m d

v

|    |    |    |    |    |    |    |    |    |    |    |
|----|----|----|----|----|----|----|----|----|----|----|
| 10 | 20 | 25 | 35 | 38 | 40 | 44 | 50 | 55 | 65 | 99 |
|----|----|----|----|----|----|----|----|----|----|----|

0 e d

v

|    |    |    |    |    |    |    |    |    |    |    |
|----|----|----|----|----|----|----|----|----|----|----|
| 10 | 20 | 25 | 35 | 38 | 40 | 44 | 50 | 55 | 65 | 99 |
|----|----|----|----|----|----|----|----|----|----|----|

0 e m d

v

|    |    |    |    |    |    |    |    |    |    |    |
|----|----|----|----|----|----|----|----|----|----|----|
| 10 | 20 | 25 | 35 | 38 | 40 | 44 | 50 | 55 | 65 | 99 |
|----|----|----|----|----|----|----|----|----|----|----|

# Exemplo

x == 48

|   |    |    |    |    |    |    |    |    |    |    |    |
|---|----|----|----|----|----|----|----|----|----|----|----|
|   | 0  |    |    |    |    |    | e  | d  |    |    | 10 |
| v | 10 | 20 | 25 | 35 | 38 | 40 | 44 | 50 | 55 | 65 | 99 |

|   |    |    |    |    |    |    |    |    |    |    |    |
|---|----|----|----|----|----|----|----|----|----|----|----|
|   | 0  |    |    |    |    |    | m  |    |    |    |    |
|   |    |    |    |    |    |    | e  | d  |    |    | 10 |
| v | 10 | 20 | 25 | 35 | 38 | 40 | 44 | 50 | 55 | 65 | 99 |

|   |    |    |    |    |    |    |    |    |    |    |    |
|---|----|----|----|----|----|----|----|----|----|----|----|
|   | 0  |    |    |    |    |    |    | e  |    |    |    |
|   |    |    |    |    |    |    |    | d  |    |    | 10 |
| v | 10 | 20 | 25 | 35 | 38 | 40 | 44 | 50 | 55 | 65 | 99 |

# Exemplo

x == 48

|   |    |    |    |    |    |    |    |    |    |    |    |
|---|----|----|----|----|----|----|----|----|----|----|----|
|   |    |    |    |    |    |    | m  |    |    |    |    |
|   |    |    |    |    |    |    | e  |    |    |    |    |
|   | 0  |    |    |    |    |    | d  |    |    | 10 |    |
| v | 10 | 20 | 25 | 35 | 38 | 40 | 44 | 50 | 55 | 65 | 99 |

|   |    |    |    |    |    |    |    |    |    |    |    |
|---|----|----|----|----|----|----|----|----|----|----|----|
|   |    |    |    |    |    |    | d  | e  |    |    |    |
|   | 0  |    |    |    |    |    |    |    |    | 10 |    |
| v | 10 | 20 | 25 | 35 | 38 | 40 | 44 | 50 | 55 | 65 | 99 |

# Correção

Relação **invariante** chave:

(i0) em /\*1\*/ vale que:  $v[e-1] < x < v[d+1]$ . ♥

$x == 48$

|   |    |    |    |    |    |    |    |    |    |    |     |
|---|----|----|----|----|----|----|----|----|----|----|-----|
|   | 0  |    |    |    | e  |    | d  |    |    |    | n-1 |
| v | 10 | 20 | 25 | 35 | 38 | 40 | 44 | 50 | 55 | 65 | 99  |

A relação (i0) vale no começo da primeira iteração se supusermos que  $v[-1] = -\infty$  e  $v[n] = +\infty$ .

## Correção

Relação **invariante** chave:

(i0) em **/\*1\*/** vale que:  $v[e-1] < x < v[d+1]$ . ♥

$x == 48$

|   |    |    |    |    |    |    |    |    |    |    |     |
|---|----|----|----|----|----|----|----|----|----|----|-----|
|   | 0  |    |    |    | e  |    | d  |    |    |    | n-1 |
| v | 10 | 20 | 25 | 35 | 38 | 40 | 44 | 50 | 55 | 65 | 99  |

No início da última iteração, quando  $e > d$ , nenhum elemento é “ $> v[e-1]$ ” e “ $< v[d+1]$ ”, pois o vetor é **crescente** (!). Logo,  $x$  não está em  $v[0..n-1]$  e função devolve  $-1$

## Correção

Relação **invariante** chave:

(i0) em **/\*1\*/** vale que:  $v[e-1] < x < v[d+1]$ . ♥

$x == 48$

|   |    |    |    |    |    |    |    |    |    |    |     |
|---|----|----|----|----|----|----|----|----|----|----|-----|
|   | 0  |    |    |    |    | e  |    | d  |    |    | n-1 |
| v | 10 | 20 | 25 | 35 | 38 | 40 | 44 | 50 | 55 | 65 | 99  |

O valor de  $d - e$  diminui a cada iteração. Portanto, se a função não encontra **m** tal que  $v[m] == x$ , então a função para quando  $d - e < 0$ .

## Consumo de tempo buscaBinaria

O consumo de tempo da função `buscaBinaria` é proporcional ao número  $k$  de iterações do `while`.

No início da 1ª. iteração tem-se que

$$d - e = n - 1 \approx n.$$

Sejam

$$(e_0, d_0), (e_1, d_1), \dots, (e_k, d_k),$$

os valores das variáveis  $e$  e  $d$  no início de cada uma das iterações. No pior caso  $x$  não está em  $v$ .

Assim,  $d_{k-1} - e_{k-1} \geq 0$  e  $d_k - e_k < 0$

## Número iterações

Estimaremos o valor de  $k$  em função de  $d - e$ .

Note que  $d_{i+1} - e_{i+1} \leq (d_i - e_i)/2$   
para  $i=1, 2, \dots, k-1$ .

Desta forma tem-se que

$$\begin{array}{ccccccc} d_0 - e_0 & = & n - 1 & < & n \\ d_1 - e_1 & \leq & (d_0 - e_0)/2 & < & n/2 \\ d_2 - e_2 & \leq & (d_1 - e_1)/2 & < & (n/2)/2 = n/2^2 \\ d_3 - e_3 & \leq & (d_2 - e_2)/2 & < & (n/2^2)/2 = n/2^3 \\ d_4 - e_4 & \leq & (d_3 - e_3)/2 & < & (n/2^3)/2 = n/2^4 \\ \vdots & & \vdots & & \vdots & & \vdots \end{array}$$

## Número iterações

Percebe-se que depois de cada iteração o valor de  $d - e$  é reduzido pela metade.

Seja  $t$  o número inteiro tal que

$$2^t \leq n < 2^{t+1}.$$

Da primeira desigualdade temos que

$$t \leq \lg n,$$

onde  $\lg n$  denota o logaritmo de  $n$  na base 2.

## Número iterações

Da desigualdade estrita, concluímos que

$$0 \leq (d_{k-1} - e_{k-1})/2^{k-1} < \underline{n}/2^{k-1} < \underline{2^{t+1}}/2^{k-1}.$$

Assim, em particular temos que

$$1 \leq 2^{t+1}/2^{k-1}$$

ou, em outras palavras

$$k \leq t + 2.$$

Portanto, o número  $k$  de iterações é não superior a

$$t + 2 \leq \lg n + 2.$$

# Conclusão

O consumo de tempo do algoritmo `buscaBinaria` no pior caso é proporcional a  $\lg n$ .

O consumo de tempo do algoritmo `buscaBinaria` é  $O(\lg n)$ .

## Número de iterações

| buscaSequencial | buscaBinaria |
|-----------------|--------------|
| $n$             | $\lg n$      |
| 256             | 8            |
| 512             | 9            |
| 1024            | 10           |
| 2048            | 11           |
| 4096            | 12           |
| 8192            | 13           |
| 16384           | 14           |
| 32768           | 15           |
| 65536           | 16           |
| 262144          | 18           |
| 1048576         | 20           |
| $\vdots$        | $\vdots$     |
| 4294967296      | 32           |

## Versão recursiva da busca binária

Para formular uma versão recursiva é necessário generalizar um pouco o problema trocando  $v[0 \dots n-1]$  por  $v[e \dots d]$ .

```
int buscaBinaria(int x, int n, int v[])  
{  
1  return buscaBinariaR(x, 0, n-1, v);  
}
```

## Versão recursiva da busca binária

Recebe um vetor crescente  $v[e..d]$  e devolve um índice  $m$  tal que  $v[m] == x$ . Se tal  $m$  não existe, devolve  $-1$ .

```
int
buscaBinariaR(int x, int e, int d, int v[]){
    int m;
1   if (d < e) return -1;
2   m = (e + d)/2;
3   if (v[m] == x) return m;
4   if (v[m] < x)
5       return buscaBinariaR(x, m+1, d, v);
6   return buscaBinariaR(x, e, m-1, v);
}
```

## Outra versão recursiva

### Observações:

- ▶ As declarações `int v[]` e `int *v` no protótipo de funções são **equivalentes**. Abaixo escolhemos `int *v` apenas para deixar **mais explícito** que em ambos os casos o que está sendo passado como parâmetro é um **endereço(!)**.
- ▶ As expressões “`&v[m+1]`” e “`v+m+1`” são equivalentes (=têm o mesmo valor =representam o mesmo endereço).
- ▶ Tem um problema . . .

## Outra versão recursiva

**A função abaixo não resolve o problema...**

Por quê? Como consertar?

```
int  
buscaBinariaR(int x, int n, int *v) {  
    int m;  
    if (n == 0) return -1;  
    m = n/2;  
    if (v[m] == x) return m;  
    if (v[m] < x)  
        return buscaBinariaR(x, n-m-1, &v[m+1]);  
    return buscaBinariaR(x, m, v);  
}
```

# Ordenação

$v[0 \dots n-1]$  é **crescente** se  $v[0] \leq \dots \leq v[n-1]$ .

**Problema:** Rearranjar um vetor  $v[0 \dots n-1]$  de modo que ele fique **crescente**.

Entra:

|    |    |    |    |    |    |    |    |    |    |            |
|----|----|----|----|----|----|----|----|----|----|------------|
| 1  |    |    |    |    |    |    |    |    |    | <b>n-1</b> |
| 33 | 55 | 33 | 44 | 33 | 22 | 11 | 99 | 22 | 55 | 77         |

# Ordenação

$v[0 \dots n-1]$  é **crescente** se  $v[0] \leq \dots \leq v[n-1]$ .

**Problema:** Rearranjar um vetor  $v[0 \dots n-1]$  de modo que ele fique **crescente**.

Entra:

|    |    |    |    |    |    |    |    |    |    |                         |
|----|----|----|----|----|----|----|----|----|----|-------------------------|
| 1  |    |    |    |    |    |    |    |    |    | <b><math>n-1</math></b> |
| 33 | 55 | 33 | 44 | 33 | 22 | 11 | 99 | 22 | 55 | 77                      |

Sai:

|    |    |    |    |    |    |    |    |    |    |                         |
|----|----|----|----|----|----|----|----|----|----|-------------------------|
| 0  |    |    |    |    |    |    |    |    |    | <b><math>n-1</math></b> |
| 11 | 22 | 22 | 33 | 33 | 33 | 44 | 55 | 55 | 77 | 99                      |

# Ordenação por inserção (iteração)

$x = 38$

|    |    |     |    |    |    |    |       |    |    |    |
|----|----|-----|----|----|----|----|-------|----|----|----|
| 0  |    | $i$ |    |    |    |    | $n-1$ |    |    |    |
| 20 | 25 | 35  | 40 | 44 | 55 | 38 | 99    | 10 | 65 | 50 |

# Ordenação por inserção (iteração)

$x = 38$

|    |    |    |    |    |    |    |    |    |    |    |     |
|----|----|----|----|----|----|----|----|----|----|----|-----|
| 0  |    |    |    |    | j  | i  |    |    |    |    | n-1 |
| 20 | 25 | 35 | 40 | 44 | 55 | 38 | 99 | 10 | 65 | 50 |     |

# $x = 38$ Ordenação por inserção (iteração)

|    |    |    |    |    |    |    |    |    |    |    |     |
|----|----|----|----|----|----|----|----|----|----|----|-----|
| 0  |    |    |    |    | j  | i  |    |    |    |    | n-1 |
| 20 | 25 | 35 | 40 | 44 | 55 | 38 | 99 | 10 | 65 | 50 |     |

|    |    |    |    |    |  |    |    |    |    |    |     |
|----|----|----|----|----|--|----|----|----|----|----|-----|
| 0  |    |    |    | j  |  | i  |    |    |    |    | n-1 |
| 20 | 25 | 35 | 40 | 44 |  | 55 | 99 | 10 | 65 | 50 |     |

## $x = 38$ Ordenação por inserção (iteração)

|    |    |    |    |    |    |    |    |    |    |    |     |
|----|----|----|----|----|----|----|----|----|----|----|-----|
| 0  |    |    |    |    | j  | i  |    |    |    |    | n-1 |
| 20 | 25 | 35 | 40 | 44 | 55 | 38 | 99 | 10 | 65 | 50 |     |

|    |    |    |    |    |  |    |    |    |    |    |     |
|----|----|----|----|----|--|----|----|----|----|----|-----|
| 0  |    |    |    | j  |  | i  |    |    |    |    | n-1 |
| 20 | 25 | 35 | 40 | 44 |  | 55 | 99 | 10 | 65 | 50 |     |

|    |    |    |    |  |    |    |    |    |    |    |     |
|----|----|----|----|--|----|----|----|----|----|----|-----|
| 0  |    |    | j  |  |    | i  |    |    |    |    | n-1 |
| 20 | 25 | 35 | 40 |  | 44 | 55 | 99 | 10 | 65 | 50 |     |

# $x = 38$ Ordenação por inserção (iteração)

|    |    |    |    |    |    |    |    |    |    |     |
|----|----|----|----|----|----|----|----|----|----|-----|
| 0  |    |    |    | j  | i  |    |    |    |    | n-1 |
| 20 | 25 | 35 | 40 | 44 | 55 | 38 | 99 | 10 | 65 | 50  |

|    |    |    |    |    |  |    |    |    |    |     |
|----|----|----|----|----|--|----|----|----|----|-----|
| 0  |    |    |    | j  |  | i  |    |    |    | n-1 |
| 20 | 25 | 35 | 40 | 44 |  | 55 | 99 | 10 | 65 | 50  |

|    |    |    |    |  |    |    |    |    |    |     |
|----|----|----|----|--|----|----|----|----|----|-----|
| 0  |    |    | j  |  |    | i  |    |    |    | n-1 |
| 20 | 25 | 35 | 40 |  | 44 | 55 | 99 | 10 | 65 | 50  |

|    |    |    |  |    |    |    |    |    |    |     |
|----|----|----|--|----|----|----|----|----|----|-----|
| 0  |    | j  |  |    |    | i  |    |    |    | n-1 |
| 20 | 25 | 35 |  | 40 | 44 | 55 | 99 | 10 | 65 | 50  |

## $x = 38$ Ordenação por inserção (iteração)

|    |    |    |    |    |    |    |    |    |    |     |
|----|----|----|----|----|----|----|----|----|----|-----|
| 0  |    |    |    | j  | i  |    |    |    |    | n-1 |
| 20 | 25 | 35 | 40 | 44 | 55 | 38 | 99 | 10 | 65 | 50  |

|    |    |    |    |    |  |    |    |    |    |     |
|----|----|----|----|----|--|----|----|----|----|-----|
| 0  |    |    |    | j  |  | i  |    |    |    | n-1 |
| 20 | 25 | 35 | 40 | 44 |  | 55 | 99 | 10 | 65 | 50  |

|    |    |    |    |  |    |    |    |    |    |     |
|----|----|----|----|--|----|----|----|----|----|-----|
| 0  |    |    | j  |  |    | i  |    |    |    | n-1 |
| 20 | 25 | 35 | 40 |  | 44 | 55 | 99 | 10 | 65 | 50  |

|    |    |    |  |    |    |    |    |    |    |     |
|----|----|----|--|----|----|----|----|----|----|-----|
| 0  |    | j  |  |    |    | i  |    |    |    | n-1 |
| 20 | 25 | 35 |  | 40 | 44 | 55 | 99 | 10 | 65 | 50  |

|    |    |    |    |    |    |    |    |    |    |     |
|----|----|----|----|----|----|----|----|----|----|-----|
| 0  |    | j  |    |    |    | i  |    |    |    | n-1 |
| 20 | 25 | 35 | 38 | 40 | 44 | 55 | 99 | 10 | 65 | 50  |

# Ordenação por inserção

|          |    |    |    |    |    |    |    |          |     |    |    |
|----------|----|----|----|----|----|----|----|----------|-----|----|----|
| <b>x</b> | 0  |    |    |    |    |    |    | <b>i</b> | n-1 |    |    |
| 99       | 20 | 25 | 35 | 38 | 40 | 44 | 55 | 99       | 10  | 65 | 50 |

# Ordenação por inserção

|     |    |    |    |    |    |    |    |     |    |    |       |
|-----|----|----|----|----|----|----|----|-----|----|----|-------|
| $x$ | 0  |    |    |    |    |    |    | $i$ |    |    | $n-1$ |
| 99  | 20 | 25 | 35 | 38 | 40 | 44 | 55 | 99  | 10 | 65 | 50    |

|     |    |    |    |    |    |    |    |     |    |    |       |
|-----|----|----|----|----|----|----|----|-----|----|----|-------|
| $x$ | 0  |    |    |    |    |    |    | $i$ |    |    | $n-1$ |
| 10  | 20 | 25 | 35 | 38 | 40 | 44 | 55 | 99  | 10 | 65 | 50    |

# Ordenação por inserção

|     |    |    |    |    |    |    |    |     |    |    |       |
|-----|----|----|----|----|----|----|----|-----|----|----|-------|
| $x$ | 0  |    |    |    |    |    |    | $i$ |    |    | $n-1$ |
| 99  | 20 | 25 | 35 | 38 | 40 | 44 | 55 | 99  | 10 | 65 | 50    |

|     |    |    |    |    |    |    |    |     |    |    |       |
|-----|----|----|----|----|----|----|----|-----|----|----|-------|
| $x$ | 0  |    |    |    |    |    |    | $i$ |    |    | $n-1$ |
| 10  | 10 | 20 | 25 | 35 | 38 | 40 | 44 | 55  | 99 | 65 | 50    |

# Ordenação por inserção

|     |    |    |    |    |    |    |    |     |    |    |       |
|-----|----|----|----|----|----|----|----|-----|----|----|-------|
| $x$ | 0  |    |    |    |    |    |    | $i$ |    |    | $n-1$ |
| 99  | 20 | 25 | 35 | 38 | 40 | 44 | 55 | 99  | 10 | 65 | 50    |

|     |    |    |    |    |    |    |    |     |    |    |       |
|-----|----|----|----|----|----|----|----|-----|----|----|-------|
| $x$ | 0  |    |    |    |    |    |    | $i$ |    |    | $n-1$ |
| 10  | 10 | 20 | 25 | 35 | 38 | 40 | 44 | 55  | 99 | 65 | 50    |

|     |    |    |    |    |    |    |    |    |     |       |    |
|-----|----|----|----|----|----|----|----|----|-----|-------|----|
| $x$ | 0  |    |    |    |    |    |    |    | $i$ | $n-1$ |    |
| 65  | 10 | 20 | 25 | 35 | 38 | 40 | 44 | 55 | 99  | 65    | 50 |

# Ordenação por inserção

|     |    |    |    |    |    |    |    |     |    |    |       |
|-----|----|----|----|----|----|----|----|-----|----|----|-------|
| $x$ | 0  |    |    |    |    |    |    | $i$ |    |    | $n-1$ |
| 99  | 20 | 25 | 35 | 38 | 40 | 44 | 55 | 99  | 10 | 65 | 50    |

|     |    |    |    |    |    |    |    |    |     |    |       |
|-----|----|----|----|----|----|----|----|----|-----|----|-------|
| $x$ | 0  |    |    |    |    |    |    |    | $i$ |    | $n-1$ |
| 10  | 10 | 20 | 25 | 35 | 38 | 40 | 44 | 55 | 99  | 65 | 50    |

|     |    |    |    |    |    |    |    |    |    |     |       |
|-----|----|----|----|----|----|----|----|----|----|-----|-------|
| $x$ | 0  |    |    |    |    |    |    |    |    | $i$ | $n-1$ |
| 65  | 10 | 20 | 25 | 35 | 38 | 40 | 44 | 55 | 65 | 99  | 50    |

# Ordenação por inserção

|    |    |    |    |    |    |    |    |    |    |    |     |
|----|----|----|----|----|----|----|----|----|----|----|-----|
| x  | 0  |    |    |    |    |    |    | i  |    |    | n-1 |
| 99 | 20 | 25 | 35 | 38 | 40 | 44 | 55 | 99 | 10 | 65 | 50  |

|    |    |    |    |    |    |    |    |    |    |    |     |
|----|----|----|----|----|----|----|----|----|----|----|-----|
| x  | 0  |    |    |    |    |    |    | i  |    |    | n-1 |
| 10 | 10 | 20 | 25 | 35 | 38 | 40 | 44 | 55 | 99 | 65 | 50  |

|    |    |    |    |    |    |    |    |    |    |    |     |
|----|----|----|----|----|----|----|----|----|----|----|-----|
| x  | 0  |    |    |    |    |    |    |    | i  |    | n-1 |
| 65 | 10 | 20 | 25 | 35 | 38 | 40 | 44 | 55 | 65 | 99 | 50  |

|    |    |    |    |    |    |    |    |    |    |    |    |
|----|----|----|----|----|----|----|----|----|----|----|----|
| x  | 0  |    |    |    |    |    |    |    |    | i  |    |
| 50 | 10 | 20 | 25 | 35 | 38 | 40 | 44 | 55 | 65 | 99 | 50 |

# Ordenação por inserção

|          |    |    |    |    |    |    |          |    |            |    |    |
|----------|----|----|----|----|----|----|----------|----|------------|----|----|
| <b>x</b> | 0  |    |    |    |    |    | <b>i</b> |    | <b>n-1</b> |    |    |
| 99       | 20 | 25 | 35 | 38 | 40 | 44 | 55       | 99 | 10         | 65 | 50 |

|          |    |    |    |    |    |    |    |    |          |            |    |
|----------|----|----|----|----|----|----|----|----|----------|------------|----|
| <b>x</b> | 0  |    |    |    |    |    |    |    | <b>i</b> | <b>n-1</b> |    |
| 10       | 10 | 20 | 25 | 35 | 38 | 40 | 44 | 55 | 99       | 65         | 50 |

|          |    |    |    |    |    |    |    |    |          |    |    |            |
|----------|----|----|----|----|----|----|----|----|----------|----|----|------------|
| <b>x</b> | 0  |    |    |    |    |    |    |    | <b>i</b> |    |    | <b>n-1</b> |
| 65       | 10 | 20 | 25 | 35 | 38 | 40 | 44 | 55 | 65       | 99 | 50 |            |

|          |    |    |    |    |    |    |    |    |    |    |          |
|----------|----|----|----|----|----|----|----|----|----|----|----------|
| <b>x</b> | 0  |    |    |    |    |    |    |    |    |    | <b>i</b> |
| 50       | 10 | 20 | 25 | 35 | 38 | 40 | 44 | 50 | 55 | 65 | 99       |

## insercao

Função rearranja  $v[0..n-1]$  em ordem crescente.

```
void insercao (int n, int v[])
{
    int i, j, x;
1   for (i = 1; /*A*/ i < n; i++){
2       x = v[i];
3       for (j=i-1; j >= 0 && v[j] > x; j--)
4           v[j+1] = v[j];
5       v[j+1] = x;
    }
}
```

# O algoritmo faz o que promete?

Relação **invariante** chave:

♥ (i0) Em */\*A\*/* vale que:  $v[0 \dots i-1]$  é crescente.

|    |    |    |    |    |    |    |    |    |    |     |
|----|----|----|----|----|----|----|----|----|----|-----|
| 0  |    |    |    |    |    | i  |    |    |    | n-1 |
| 20 | 25 | 35 | 40 | 44 | 55 | 38 | 99 | 10 | 65 | 50  |

## O algoritmo faz o que promete?

Relação **invariante** chave:

♥ (i0) Em */\*A\*/* vale que:  $v[0 \dots i-1]$  é crescente.

|    |    |    |    |    |    |    |    |    |    |     |
|----|----|----|----|----|----|----|----|----|----|-----|
| 0  |    |    |    |    |    | i  |    |    |    | n-1 |
| 20 | 25 | 35 | 40 | 44 | 55 | 38 | 99 | 10 | 65 | 50  |

Supondo que a invariante vale.

Correção do algoritmo é **evidente**.

No início da última iteração tem-se que  $i = n$ .

Da invariante conclui-se que  $v[0 \dots n-1]$  é **crescente**.

## Mais invariantes

Na linha 3, antes de “ $j \geq 0 \dots$ ”, vale que:

(i1)  $v[0 \dots j]$  e  $v[j+2 \dots i]$  são crescentes

(i2)  $v[0 \dots j] \leq v[j+2 \dots i]$

(i3)  $v[j+2 \dots i] > x$

|          |    |    |          |  |    |    |          |    |    |    |            |
|----------|----|----|----------|--|----|----|----------|----|----|----|------------|
| <b>x</b> | 0  |    | <b>j</b> |  |    |    | <b>i</b> |    |    |    | <b>n-1</b> |
| 38       | 20 | 25 | 35       |  | 40 | 44 | 55       | 99 | 10 | 65 | 50         |

## Mais invariantes

Na linha 3, antes de “ $j \geq 0 \dots$ ”, vale que:

(i1)  $v[0 \dots j]$  e  $v[j+2 \dots i]$  são crescentes

(i2)  $v[0 \dots j] \leq v[j+2 \dots i]$

(i3)  $v[j+2 \dots i] > x$

|     |    |    |     |  |    |    |     |    |    |    |       |
|-----|----|----|-----|--|----|----|-----|----|----|----|-------|
| $x$ | 0  |    | $j$ |  |    |    | $i$ |    |    |    | $n-1$ |
| 38  | 20 | 25 | 35  |  | 40 | 44 | 55  | 99 | 10 | 65 | 50    |

invariantes (i1), (i2) e (i3)

+ condição de parada do for da linha 3

+ atribuição da linha 5  $\Rightarrow$  validade (i0)

Verifique!

## Correção de algoritmos iterativos

Estrutura “**típica**” de demonstrações da correção de algoritmos iterativos através de suas relações invariantes consiste em:

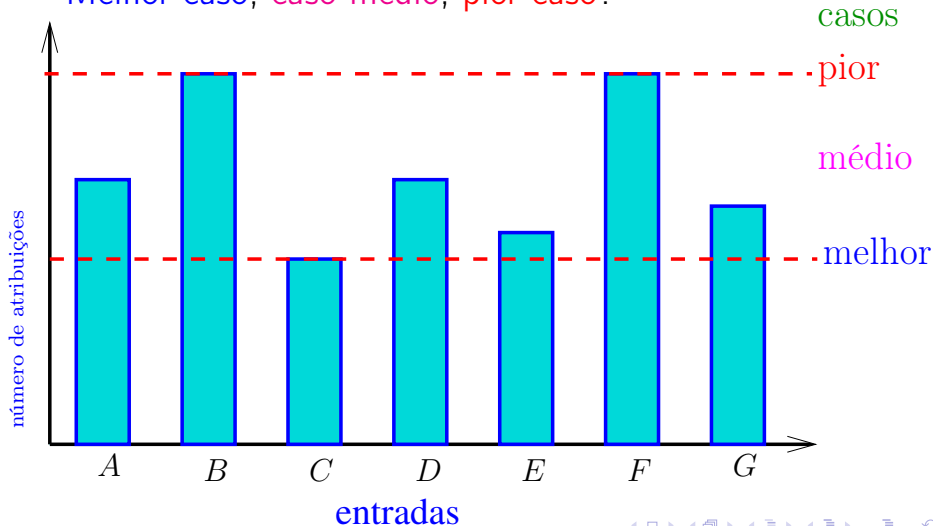
1. verificar que a relação **vale no início** da primeira iteração;
2. demonstrar que  
*se a relação **vale no início** da iteração, **então** ela **vale no final** da iteração (com os papéis de alguns atores **possivelmente trocados**);*
3. concluir que, se **relação vale** no início da **última iteração**, **então** a **relação junto com a condição de parada implicam na correção** do algoritmo.

Quantas atribuições faz a função?

# Quantas atribuições faz a função?

Número mínimo, médio ou máximo?

Melhor caso, caso médio, pior caso?



## Quantas atribuições faz a função?

LINHAS 2-4 (v, i, x)

```
2      x = v[i];  
3      for (j=i-1; j >= 0 && v[j] > x; j--)  
4          v[j+1] = v[j];
```

## Quantas atribuições faz a função?

LINHAS 2-4 (v, i, x)

```
2      x = v[i];  
3      for (j=i-1; j >= 0 && v[j] > x; j--)  
4          v[j+1] = v[j];
```

| linha | atribuições (número máximo) |
|-------|-----------------------------|
| 2     | ?                           |
| 3     | ?                           |
| 4     | ?                           |
| <hr/> |                             |
| total | ?                           |

## Quantas atribuições faz a função?

LINHAS 2-4 ( $v, i, x$ )

```
2       $x = v[i]$  ;  
3      for ( $j=i-1$ ;  $j \geq 0 \ \&\& \ v[j] > x$ ;  $j--$ )  
4           $v[j+1] = v[j]$  ;
```

| linha | atribuições (número máximo) |
|-------|-----------------------------|
| 2     | $= 1$                       |
| 3     | $\leq 1 + i$                |
| 4     | $?$                         |
| <hr/> |                             |
| total | $?$                         |

## Quantas atribuições faz a função?

LINHAS 2-4 ( $v, i, x$ )

```
2       $x = v[i]$ ;  
3      for ( $j=i-1$ ;  $j \geq 0 \ \&\& \ v[j] > x$ ;  $j--$ )  
4           $v[j+1] = v[j]$ ;
```

| linha | atribuições (número máximo) |
|-------|-----------------------------|
| 2     | $= 1$                       |
| 3     | $\leq 1 + i$                |
| 4     | $\leq i$                    |

$$\text{total} \leq 2i + 1 \leq 2n$$

## Quantas atribuições faz a função?

```
void insercao (int n, int v[]) {  
    int i, j, x;  
1   for (i = 1; /*A*/ i < n; i++){  
2       LINHAS 2-4 (v, i, x)  
5       v[j+1] = x;  
    }  
}
```

| linha | atribuições (número máximo) |
|-------|-----------------------------|
| 1     | ?                           |
| 2-4   | ?                           |
| 5     | ?                           |

total ?

## Quantas atribuições faz a função?

```
void insercao (int n, int v[]) {  
    int i, j, x;  
1   for (i = 1; /*A*/ i < n; i++){  
2       LINHAS 2-4 (v, i, x)  
5       v[j+1] = x;  
    }  
}
```

| linha | atribuições (número máximo) |
|-------|-----------------------------|
| 1     | = $n$                       |
| 2-4   | $\leq (n - 1)2n$            |
| 5     | = $n - 1$                   |

$$\text{total} \leq 2n^2 - 1$$

## Análise mais fina

| linha | atribuições (número máximo) |
|-------|-----------------------------|
| 1     | ?                           |
| 2     | ?                           |
| 3     | ?                           |
| 4     | ?                           |
| 5     | ?                           |
| total | ?                           |

## Análise mais fina

| linha | atribuições (número máximo)                 |
|-------|---------------------------------------------|
| 1     | $= n$                                       |
| 2     | $= n - 1$                                   |
| 3     | $\leq 1 + 2 + \dots + n = n(n + 1)/2$       |
| 4     | $\leq 1 + 2 + \dots + (n - 1) = (n - 1)n/2$ |
| 5     | $= n - 1$                                   |

$$\text{total} \leq n^2 + 3n - 2$$

$n^2 + 3n - 2$  versus  $n^2$

| $n$ | $n^2 + 3n - 2$ | $n^2$ |
|-----|----------------|-------|
|-----|----------------|-------|

---

|   |   |   |
|---|---|---|
| 1 | 2 | 1 |
|---|---|---|

|   |   |   |
|---|---|---|
| 2 | 8 | 4 |
|---|---|---|

$n^2 + 3n - 2$  versus  $n^2$

| $n$ | $n^2 + 3n - 2$ | $n^2$ |
|-----|----------------|-------|
| 1   | 2              | 1     |
| 2   | 8              | 4     |
| 3   | 16             | 9     |
| 10  | 128            | 100   |

$n^2 + 3n - 2$  versus  $n^2$

| $n$  | $n^2 + 3n - 2$ | $n^2$   |
|------|----------------|---------|
| 1    | 2              | 1       |
| 2    | 8              | 4       |
| 3    | 16             | 9       |
| 10   | 128            | 100     |
| 100  | 10298          | 10000   |
| 1000 | 1002998        | 1000000 |

$n^2 + 3n - 2$  versus  $n^2$

| $n$    | $n^2 + 3n - 2$ | $n^2$       |
|--------|----------------|-------------|
| 1      | 2              | 1           |
| 2      | 8              | 4           |
| 3      | 16             | 9           |
| 10     | 128            | 100         |
| 100    | 10298          | 10000       |
| 1000   | 1002998        | 1000000     |
| 10000  | 100029998      | 100000000   |
| 100000 | 10000299998    | 10000000000 |

$n^2$  domina os outros termos

## Consumo de tempo

Se a execução de cada linha de código consome 1 unidade de tempo, qual o consumo total?

```
void insercao (int n, int v[])  
{  
    int i, j, x;  
1   for (i = 1; /*A*/ i < n; i++){  
2       x = v[i];  
3       for (j = i-1; j >= 0 && v[j] > x; j--)  
4           v[j+1] = v[j];  
5       v[j+1] = x;  
    }  
}
```

## Consumo de tempo no pior caso

| linha | todas as execuções da linha                 |
|-------|---------------------------------------------|
| 1     | $= n$                                       |
| 2     | $= n - 1$                                   |
| 3     | $\leq 2 + 3 + \dots + n = (n - 1)(n + 2)/2$ |
| 4     | $\leq 1 + 2 + \dots + (n - 1) = n(n - 1)/2$ |
| 5     | $= n - 1$                                   |
| total | $\leq (3/2)n^2 + (7/2)n - 4 = O(n^2)$       |

## Consumo de tempo no melhor caso

| linha | todas as execuções da linha |
|-------|-----------------------------|
| 1     | = $n$                       |
| 2     | = $n - 1$                   |
| 3     | = $n - 1$                   |
| 4     | = $0$                       |
| 5     | = $n - 1$                   |
| <hr/> |                             |
| total | $\leq 4n - 3 = O(n)$        |

## Pior e melhor casos

O maior consumo de tempo da função `insercao` ocorre quando o vetor `v[0 . . n-1]` dado é *decrecente*. Este é o *pior caso* para a função `insercao`.

O menor consumo de tempo da função `insercao` ocorre quando o vetor `v[0 . . n-1]` dado é já *crescente*. Este é o *melhor caso* para a função `insercao`.

## Conclusão

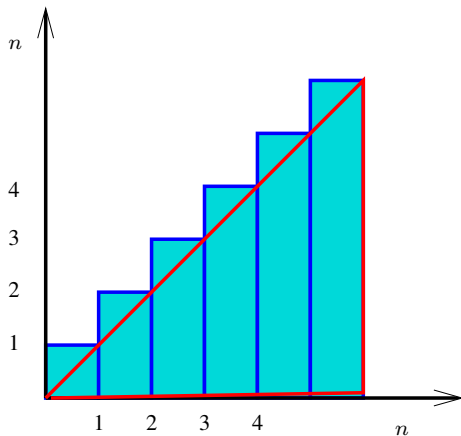
O consumo de tempo da função `insercao` no pior caso é proporcional a  $n^2$ .

O consumo de tempo da função `insercao` melhor caso é proporcional a  $n$ .

O consumo de tempo da função `insercao` é  $O(n^2)$ .

$$1 + 2 + \cdots + (n - 1) + n = ?$$

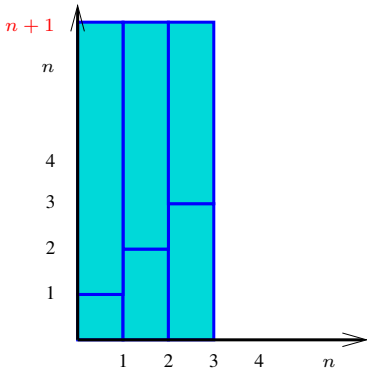
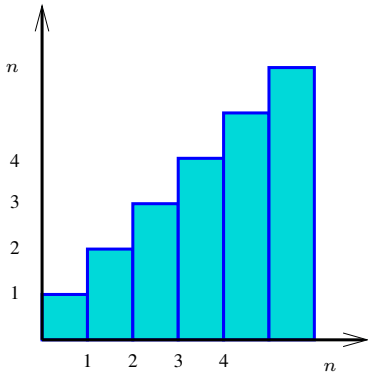
Carl Friedrich Gauss, 1787



$$\frac{n^2}{2} + \frac{n}{2} = \frac{n(n+1)}{2}$$

$$1 + 2 + \cdots + (n - 1) + n = ?$$

Carl Friedrich Gauss, 1787



$$(n + 1) \times \frac{n}{2} = \frac{n(n + 1)}{2}$$

# Ordenação por inserção binária



**Binary Search**

Fonte: <http://www.php5dp.com/>

PF 7.3, 8.1 e 8.2

<http://www.ime.usp.br/~pf/algoritmos/aulas/ordena.html>

## Busca binária

Esta função recebe um vetor crescente  $v[0 \dots n-1]$  com  $n \geq 1$  e um inteiro  $x$  e devolve um índice  $j$  em  $0 \dots n-1$  tal que  $v[j] \leq x < v[j+1]$

```
int buscaBinaria (int x, int n, int v[]) {  
    int e, m, d;  
1   e = -1; d = n;  
2   while (/*A*/e < d-1) {  
3       m = (e + d)/2;  
4       if (v[m] <= x) e = m;  
5       else d = m;  
    }  
6   return e;  
}
```

## Relações invariantes

A relação invariante **chave** da função **buscaBinaria** é

(i0) Em **/\*A\*/** vale que  $v[e] \leq x < v[d]$

A correção da função segue facilmente dessa relação e da condição de parada do **while**.

## Busca binária: recordação

O consumo de tempo da função `buscaBinaria` é proporcional a  $\lg n$ .

O consumo de tempo da função `buscaBinaria` é  $O(\lg n)$ .

## insercao

Função rearranja  $v[0..n-1]$  em ordem crescente.

```
void insercao (int n, int v[])
{
    int i, j, x;
1   for (i = 1; /*A*/ i < n; i++){
2       x = v[i];
3       for (j=i-1; j >= 0 && v[j] > x; j--)
4           v[j+1] = v[j];
5       v[j+1] = x;
    }
}
```

## insercaoBinaria

Função rearranja  $v[0..n-1]$  em ordem crescente.

```
void insercaoBinaria (int n, int v[])
{
    int i, j, k, x;
1   for (i = 1; /*A*/ i < n; i++){
2       x = v[i];
3       j = buscaBinaria(x,i,v);
4       for (k = i; k > j+1; k--)
5           v[k] = v[k-1];
6       v[j+1] = x;
    }
}
```

## Pior e melhor casos

O maior consumo de tempo da função `insercaoBinaria` ocorre quando o vetor  $v[0 \dots n-1]$  dado é decrescente. Este é o pior caso para a função `insercaoBinaria`.

O menor consumo de tempo da função `insercaoBinaria` ocorre quando o vetor  $v[0 \dots n-1]$  dado é já é crescente. Este é o melhor caso para a função `insercaoBinaria`.

## Consumo de tempo no pior caso

| linha | consumo de tempo (proporcional a)               |
|-------|-------------------------------------------------|
| 1     | $= n$                                           |
| 2     | $= n$                                           |
| 3     | $= \lg 1 + \lg 2 + \cdots + \lg n \leq n \lg n$ |
| 4     | $\leq 2 + 2 + \cdots + n = (n-1)(n+2)/2$        |
| 5     | $\leq 1 + 2 + \cdots + (n-1) = n(n-1)/2$        |
| 6     | $= n$                                           |
| <hr/> |                                                 |
| total | $\leq n^2 + n \lg n + 3n - 1 = O(n^2)$          |

## Consumo de tempo no melhor caso

| linha | consumo de tempo (proporcional a)              |
|-------|------------------------------------------------|
| 1     | $= n$                                          |
| 2     | $= n$                                          |
| 3     | $= \lg 1 + \lg 2 + \dots + \lg n \leq n \lg n$ |
| 4     | $= 1 + 1 + \dots + 1 = n$                      |
| 5     | $= 0$                                          |
| 6     | $= n$                                          |

$$\text{total} = n \lg n + 4n = O(n \lg n)$$

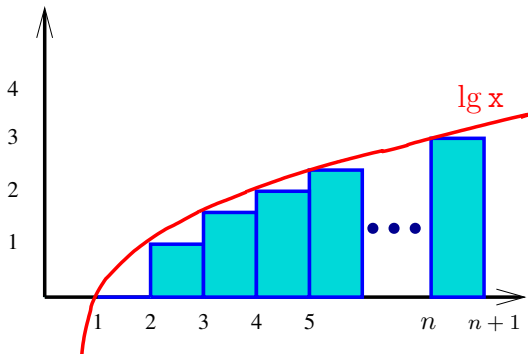
# Conclusões

O consumo de tempo da função `insercaoBinaria` no pior caso é proporcional a  $n^2$ .

O consumo de tempo da função `insercaoBinaria` no melhor caso é proporcional a  $n \lg n$ .

O consumo de tempo da função `insercaoBinaria` é  $O(n^2)$ .

$$\lg 1 + \lg 2 + \cdots + \lg n = O(n \lg n)$$



$$\lg 1 + \lg 2 + \cdots + \lg n \leq \int_1^{n+1} \lg x \, dx$$

$$\lg 1 + \lg 2 + \cdots + \lg n = O(n \lg n)$$

$$\begin{aligned}
 \int_1^{n+1} \lg x \, dx &= \left( \int_1^{n+1} \ln x \, dx \right) / \ln 2 \\
 &= (x \ln x - x) \Big|_0^{n+1} / \ln 2 \\
 &= ((n+1) \ln(n+1) - (n+1)) / \ln 2 \\
 &= (n+1) \lg(n+1) - (n+1) / \ln 2 \\
 &< (n+1) \lg(n+1) \\
 &= O(n \lg n)
 \end{aligned}$$

# Ordenação por seleção



Fonte: <http://www.exacttarget.com/>  
PF 8.3

<http://www.ime.usp.br/~pf/algoritmos/aulas/ordena.html>

# Ordenação

$v[0 \dots n-1]$  é **crescente** se  $v[0] \leq \dots \leq v[n-1]$ .

**Problema:** Rearranjar um vetor  $v[0 \dots n-1]$  de modo que ele fique **crescente**.

Entra:

|    |    |    |    |    |    |    |    |    |    |                         |
|----|----|----|----|----|----|----|----|----|----|-------------------------|
| 0  |    |    |    |    |    |    |    |    |    | <b><math>n-1</math></b> |
| 33 | 55 | 33 | 44 | 33 | 22 | 11 | 99 | 22 | 55 | 77                      |

Sai:

|    |    |    |    |    |    |    |    |    |    |                         |
|----|----|----|----|----|----|----|----|----|----|-------------------------|
| 0  |    |    |    |    |    |    |    |    |    | <b><math>n-1</math></b> |
| 11 | 22 | 22 | 33 | 33 | 33 | 44 | 55 | 55 | 77 | 99                      |

# Ordenação por seleção (iteração)

$i = 5$

|    |    |     |    |    |    |       |    |    |    |    |
|----|----|-----|----|----|----|-------|----|----|----|----|
| 0  |    | max |    |    |    | $n-1$ |    |    |    |    |
| 38 | 50 | 20  | 44 | 10 | 50 | 55    | 60 | 75 | 85 | 99 |

## Ordenação por seleção (iteração)

$i = 5$

0

$j$

max

$n-1$

|    |    |    |    |    |    |    |    |    |    |    |
|----|----|----|----|----|----|----|----|----|----|----|
| 38 | 50 | 20 | 44 | 10 | 50 | 55 | 60 | 75 | 85 | 99 |
|----|----|----|----|----|----|----|----|----|----|----|

## Ordenação por seleção (iteração)

$i = 5$

|    |    |    |     |     |    |    |    |    |    |       |
|----|----|----|-----|-----|----|----|----|----|----|-------|
| 0  |    |    | $j$ | max |    |    |    |    |    | $n-1$ |
| 38 | 50 | 20 | 44  | 10  | 50 | 55 | 60 | 75 | 85 | 99    |

|    |    |     |     |    |    |    |    |    |    |       |
|----|----|-----|-----|----|----|----|----|----|----|-------|
| 0  |    | $j$ | max |    |    |    |    |    |    | $n-1$ |
| 38 | 50 | 20  | 44  | 10 | 50 | 55 | 60 | 75 | 85 | 99    |

# Ordenação por seleção (iteração)

$i = 5$

|    |    |    |     |     |    |    |    |    |    |       |
|----|----|----|-----|-----|----|----|----|----|----|-------|
| 0  |    |    | $j$ | max |    |    |    |    |    | $n-1$ |
| 38 | 50 | 20 | 44  | 10  | 50 | 55 | 60 | 75 | 85 | 99    |

|    |    |     |     |    |    |    |    |    |    |       |
|----|----|-----|-----|----|----|----|----|----|----|-------|
| 0  |    | $j$ | max |    |    |    |    |    |    | $n-1$ |
| 38 | 50 | 20  | 44  | 10 | 50 | 55 | 60 | 75 | 85 | 99    |

|    |     |    |     |    |    |    |    |    |    |       |
|----|-----|----|-----|----|----|----|----|----|----|-------|
| 0  | $j$ |    | max |    |    |    |    |    |    | $n-1$ |
| 38 | 50  | 20 | 44  | 10 | 50 | 55 | 60 | 75 | 85 | 99    |

## Ordenação por seleção (iteração)

$i = 5$

|    |    |    |     |     |    |    |    |    |    |       |
|----|----|----|-----|-----|----|----|----|----|----|-------|
| 0  |    |    | $j$ | max |    |    |    |    |    | $n-1$ |
| 38 | 50 | 20 | 44  | 10  | 50 | 55 | 60 | 75 | 85 | 99    |

|    |    |     |     |    |    |    |    |    |    |       |
|----|----|-----|-----|----|----|----|----|----|----|-------|
| 0  |    | $j$ | max |    |    |    |    |    |    | $n-1$ |
| 38 | 50 | 20  | 44  | 10 | 50 | 55 | 60 | 75 | 85 | 99    |

|    |     |    |     |    |    |    |    |    |    |       |
|----|-----|----|-----|----|----|----|----|----|----|-------|
| 0  | $j$ |    | max |    |    |    |    |    |    | $n-1$ |
| 38 | 50  | 20 | 44  | 10 | 50 | 55 | 60 | 75 | 85 | 99    |

|     |     |    |    |    |    |    |    |    |    |       |
|-----|-----|----|----|----|----|----|----|----|----|-------|
| $j$ | max |    |    |    |    |    |    |    |    | $n-1$ |
| 38  | 50  | 20 | 44 | 10 | 50 | 55 | 60 | 75 | 85 | 99    |

# Ordenação por seleção (iteração)

$i = 5$

|    |    |    |     |     |    |    |    |    |    |       |
|----|----|----|-----|-----|----|----|----|----|----|-------|
| 0  |    |    | $j$ | max |    |    |    |    |    | $n-1$ |
| 38 | 50 | 20 | 44  | 10  | 50 | 55 | 60 | 75 | 85 | 99    |

|    |    |     |     |    |    |    |    |    |    |       |
|----|----|-----|-----|----|----|----|----|----|----|-------|
| 0  |    | $j$ | max |    |    |    |    |    |    | $n-1$ |
| 38 | 50 | 20  | 44  | 10 | 50 | 55 | 60 | 75 | 85 | 99    |

|    |     |    |     |    |    |    |    |    |    |       |
|----|-----|----|-----|----|----|----|----|----|----|-------|
| 0  | $j$ |    | max |    |    |    |    |    |    | $n-1$ |
| 38 | 50  | 20 | 44  | 10 | 50 | 55 | 60 | 75 | 85 | 99    |

|     |     |    |    |    |    |    |    |    |    |       |
|-----|-----|----|----|----|----|----|----|----|----|-------|
| $j$ | max |    |    |    |    |    |    |    |    | $n-1$ |
| 38  | 50  | 20 | 44 | 10 | 50 | 55 | 60 | 75 | 85 | 99    |

|    |     |    |    |    |    |    |    |    |    |       |
|----|-----|----|----|----|----|----|----|----|----|-------|
| 0  | max |    |    |    |    |    |    |    |    | $n-1$ |
| 38 | 50  | 20 | 44 | 10 | 50 | 55 | 60 | 75 | 85 | 99    |

# Ordenação por seleção

|    |    |    |          |    |    |    |    |    |    |            |
|----|----|----|----------|----|----|----|----|----|----|------------|
| 0  |    |    | <i>i</i> |    |    |    |    |    |    | <i>n-1</i> |
| 38 | 10 | 20 | 44       | 50 | 50 | 55 | 60 | 75 | 85 | 99         |

# Ordenação por seleção

|    |    |    |     |    |    |    |    |    |    |       |
|----|----|----|-----|----|----|----|----|----|----|-------|
| 0  |    |    | $i$ |    |    |    |    |    |    | $n-1$ |
| 38 | 10 | 20 | 44  | 50 | 50 | 55 | 60 | 75 | 85 | 99    |

|    |    |    |     |    |    |    |    |    |    |       |
|----|----|----|-----|----|----|----|----|----|----|-------|
| 0  |    |    | $i$ |    |    |    |    |    |    | $n-1$ |
| 38 | 10 | 20 | 44  | 50 | 50 | 55 | 60 | 75 | 85 | 99    |

# Ordenação por seleção

|    |    |    |     |    |    |    |    |    |    |       |
|----|----|----|-----|----|----|----|----|----|----|-------|
| 0  |    |    | $i$ |    |    |    |    |    |    | $n-1$ |
| 38 | 10 | 20 | 44  | 50 | 50 | 55 | 60 | 75 | 85 | 99    |

|    |    |    |     |    |    |    |    |    |    |       |
|----|----|----|-----|----|----|----|----|----|----|-------|
| 0  |    |    | $i$ |    |    |    |    |    |    | $n-1$ |
| 20 | 10 | 38 | 44  | 50 | 50 | 55 | 60 | 75 | 85 | 99    |

|    |     |    |    |    |    |    |    |    |    |       |
|----|-----|----|----|----|----|----|----|----|----|-------|
| 0  | $i$ |    |    |    |    |    |    |    |    | $n-1$ |
| 20 | 10  | 38 | 44 | 50 | 50 | 55 | 60 | 75 | 85 | 99    |

# Ordenação por seleção

|    |    |    |     |    |    |    |    |    |    |       |
|----|----|----|-----|----|----|----|----|----|----|-------|
| 0  |    |    | $i$ |    |    |    |    |    |    | $n-1$ |
| 38 | 10 | 20 | 44  | 50 | 50 | 55 | 60 | 75 | 85 | 99    |

|    |    |    |     |    |    |    |    |    |    |       |
|----|----|----|-----|----|----|----|----|----|----|-------|
| 0  |    |    | $i$ |    |    |    |    |    |    | $n-1$ |
| 20 | 10 | 38 | 44  | 50 | 50 | 55 | 60 | 75 | 85 | 99    |

|    |     |    |    |    |    |    |    |    |    |       |
|----|-----|----|----|----|----|----|----|----|----|-------|
| 0  | $i$ |    |    |    |    |    |    |    |    | $n-1$ |
| 10 | 20  | 38 | 44 | 50 | 50 | 55 | 60 | 75 | 85 | 99    |

|    |    |    |    |    |    |    |    |    |    |       |
|----|----|----|----|----|----|----|----|----|----|-------|
| 0  |    |    |    |    |    |    |    |    |    | $n-1$ |
| 10 | 20 | 38 | 44 | 50 | 50 | 55 | 60 | 75 | 85 | 99    |

## Função selecao

Algoritmo rearranja  $v[0..n-1]$  em ordem crescente

```
void selecao (int n, int v[])
{
    int i, j, max, x;
1   for (i = n-1; /*A*/ i > 0; i--) {
2       max = i;
3       for (j = i-1; j >= 0; j--)
4           if (v[j] > v[max]) max = j;
5       x=v[i]; v[i]=v[max]; v[max]=x;
    }
}
```

## Invariantes

Relações **invariantes** chaves dizem que em */\*A\*/* vale que:

♥ (i0)  $v[i+1 \dots n-1]$  é crescente e  
 $v[0 \dots i] \leq v[i+1 \dots n-1]$

|    |    |    |    |    |    |     |    |    |    |    |
|----|----|----|----|----|----|-----|----|----|----|----|
| 0  |    | i  |    |    |    | n-1 |    |    |    |    |
| 38 | 50 | 20 | 44 | 10 | 50 | 55  | 60 | 75 | 85 | 99 |

## Invariantes

Relações **invariantes** dizem que em */\*A\*/* vale que:

♥ (i0)  $v[i+1 \dots n-1]$  é **crescente** e  
 $v[0 \dots i] \leq v[i+1 \dots n-1]$

|    |    |    |    |    |    |     |    |    |    |    |
|----|----|----|----|----|----|-----|----|----|----|----|
| 0  |    | i  |    |    |    | n-1 |    |    |    |    |
| 38 | 50 | 20 | 44 | 10 | 50 | 55  | 60 | 75 | 85 | 99 |

Supondo que a **invariantes** valem.

Correção do algoritmo é **evidente**.

No início da **última iteração** das linhas 1-5 tem-se que  $i = 0$ .

Da invariante conclui-se que  $v[1 \dots n-1]$  é **crescente**.  
e que  $v[0] \leq v[1 \dots n-1]$ .

## Mais invariantes

Na linha 1 vale que: (i1)  $v[0 \dots i] \leq v[i+1]$ ;

Na linha 3 vale que: (i2)  $v[j+1 \dots i] \leq v[\text{max}]$

|    |    |    |     |    |    |    |    |    |    |     |
|----|----|----|-----|----|----|----|----|----|----|-----|
| 0  | j  |    | max |    | i  |    |    |    |    | n-1 |
| 38 | 50 | 20 | 44  | 10 | 25 | 55 | 60 | 75 | 85 | 99  |

## Mais invariantes

Na linha 1 vale que:  $(i1) \ v[0 \dots i] \leq v[i+1];$

Na linha 3 vale que:  $(i2) \ v[j+1 \dots i] \leq v[\text{max}]$

|    |    |    |     |    |    |    |    |    |    |     |
|----|----|----|-----|----|----|----|----|----|----|-----|
| 0  | j  |    | max |    | i  |    |    |    |    | n-1 |
| 38 | 50 | 20 | 44  | 10 | 25 | 55 | 60 | 75 | 85 | 99  |

invariantes  $(i1), (i2)$

+ condição de parada do for da linha 3

+ troca linha 5  $\Rightarrow$  validade  $(i0)$

Verifique!

## Consumo de tempo

Se a execução de cada linha de código consome 1 unidade de tempo o consumo total é:

| linha | todas as execuções da linha                    |
|-------|------------------------------------------------|
| 1     | $= n$                                          |
| 2     | $= n - 1$                                      |
| 3     | $= n + (n - 1) + \dots + 1 = n(n + 1)/2$       |
| 4     | $= (n - 1) + (n - 2) + \dots + 1 = (n - 1)n/2$ |
| 5     | $= n - 1$                                      |
| total | $= n^2 + 3n - 2$                               |

## Conclusão

O consumo de tempo do algoritmo **selecao** no **pior caso** e no **no melhor caso** é proporcional a  **$n^2$** .

O consumo de tempo do algoritmo **selecao** é  **$O(n^2)$** .

## Função selecao (versão min)

Algoritmo rearranja  $v[0..n-1]$  em ordem crescente

```
void selecao (int n, intv[])  
{  
    int i, j, min, x;  
1   for (i = 0; i < n-1; i++) {  
2       min = i;  
3       for (j = i+1; j < n; j++)  
4           if (v[j] < v[min]) min = j;  
5       x=v[i]; v[i]=v[min]; v[min]=x;  
    }  
}
```