

## Ordenação por seleção



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

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

Navigation icons: back, forward, search, etc.

## Ordenação por seleção (iteração)

$i = 5$

0					max						n
38	50	20	44	10	50	55	60	75	85	99	

Navigation icons: back, forward, search, etc.

## Ordenação por seleção (iteração)

$i = 5$

0				j	max						n
38	50	20	44	10	50	55	60	75	85	99	

0		j	max								n
38	50	20	44	10	50	55	60	75	85	99	

Navigation icons: back, forward, search, etc.

## Ordenação

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

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

Entra:

0											n
33	55	33	44	33	22	11	99	22	55	77	

Sai:

0											n
11	22	22	33	33	33	44	55	55	77	99	

Navigation icons: back, forward, search, etc.

## Ordenação por seleção (iteração)

$i = 5$

0				j	max						n
38	50	20	44	10	50	55	60	75	85	99	

Navigation icons: back, forward, search, etc.

## Ordenação por seleção (iteração)

$i = 5$

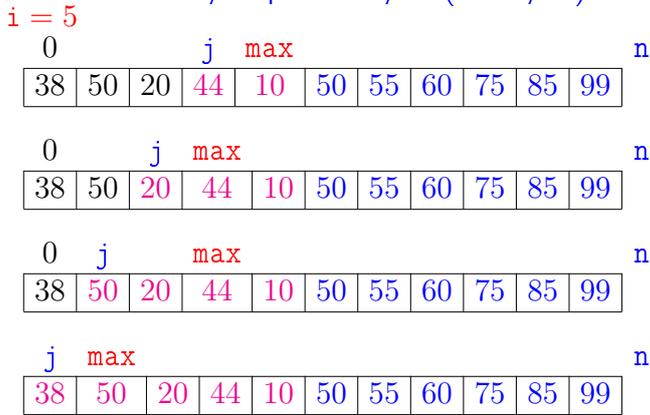
0				j	max						n
38	50	20	44	10	50	55	60	75	85	99	

0		j	max								n
38	50	20	44	10	50	55	60	75	85	99	

0	j	max									n
38	50	20	44	10	50	55	60	75	85	99	

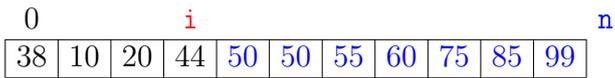
Navigation icons: back, forward, search, etc.

### Ordenação por seleção (iteração)



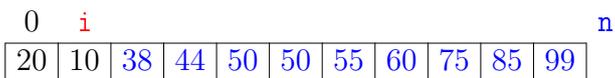
Navigation icons

### Ordenação por seleção



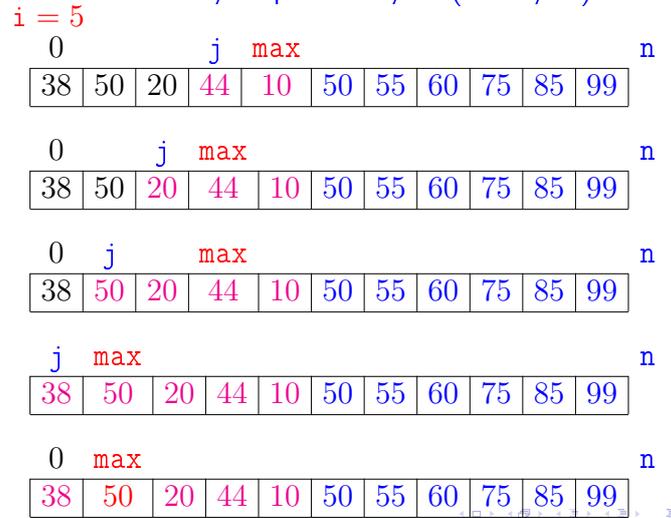
Navigation icons

### Ordenação por seleção



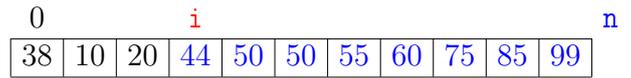
Navigation icons

### Ordenação por seleção (iteração)



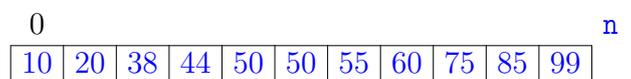
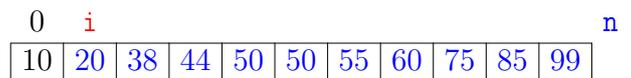
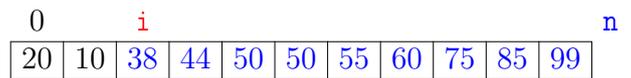
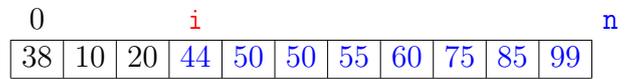
Navigation icons

### Ordenação por seleção



Navigation icons

### Ordenação por seleção



Navigation icons

## Função selecao

Algoritmo rearranja  $v[0 : n]$  em ordem **creciente**

```
def selecao (v):
0  n = len(v)
1  for i in range(n-1,0,-1): # /*A*/
2      max = i
3      for j in range(i-1,-1,-1):
4          if v[j] > v[max]: max = j
5          v[max], v[i] = v[max], v[i]
```

Navigation icons

### Invariantes

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

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

0					i															n	
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  $v[1 : n-1]$  é **creciente**. e que  $v[0] \leq v[1 : n-1]$ .

Navigation icons

### Mais invariantes

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

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

0	j		max		i																n	
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!

Navigation icons

## Invariantes

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

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

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

Navigation icons

### Mais invariantes

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

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

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

Navigation icons

### 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
0	= 1
1	= n
2	= n - 1
3	= n + (n - 1) + ... + 1 = $\frac{n(n+1)}{2}$
4	= (n - 1) + (n - 2) + ... + 1 = $(n-1)\frac{n}{2}$
5	= n - 1
<b>total</b>	= $n^2 + 3n - 1$

Navigation icons

## 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  $O(n^2)$ .

## Resultados experimentais

selecao		
n	tempo (s)	comentário
256	0.00	
512	0.01	
1024	0.05	
2048	0.19	
4096	0.78	
8192	3.10	
16384	15.45	
32768	59.19	≈ 1 min
65536	266.31	> 4 min
131072	1144.31	≈ 19 min

Emquanto isso... em outro computador...

selecao		
n	tempo (s)	comentário
256	0.00	
512	0.01	
1024	0.03	
2048	0.13	
4096	0.51	
8192	2.05	
16384	8.19	
32768	33.33	≈ 0.5 min
65536	132.31	> 2 min

Função `selecao` (versão `min`)

Algoritmo rearranja `v[0 : n-1]` em ordem crescente

```
def selecao (v):
0   n = len(v)
1   for i in range(n-1): # /*A*/
2       min = i
3       for j in range(i+1,n):
4           if v[j] < v[min]: min = j
5       v[max], v[i] = v[max], v[i]
```