Ordenação por seleção

Fonte: `http://www.exacttarget.com/`

PF 8.3

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

Ordenação

$\texttt{v}[0 : \texttt{n}]$ é crescente se $\texttt{v}[0] \leq \cdots \leq \texttt{v}[\texttt{n}-1]$.

Problema: Rearranjar um vetor $\texttt{v}[0 : \texttt{n}]$ de modo que ele fique crescente.

Entra:

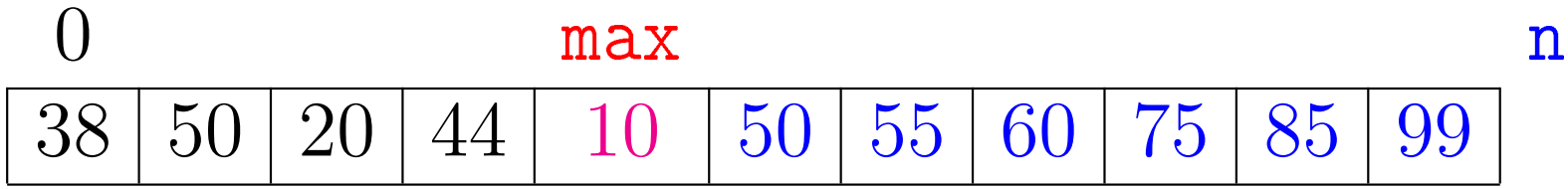
0										n
33	55	33	44	33	22	11	99	22	55	77

Sai:

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

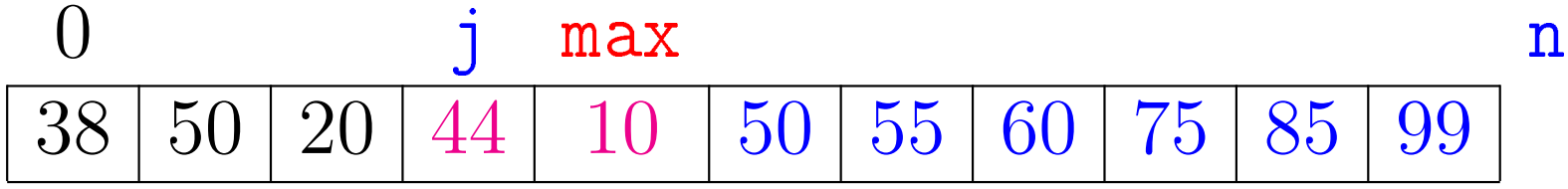
Ordenação por seleção (iteração)

$\texttt{i} = 5$



Ordenação por seleção (iteração)

$\texttt{i} = 5$



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

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

Ordenação por seleção (iteração)

i = 5

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

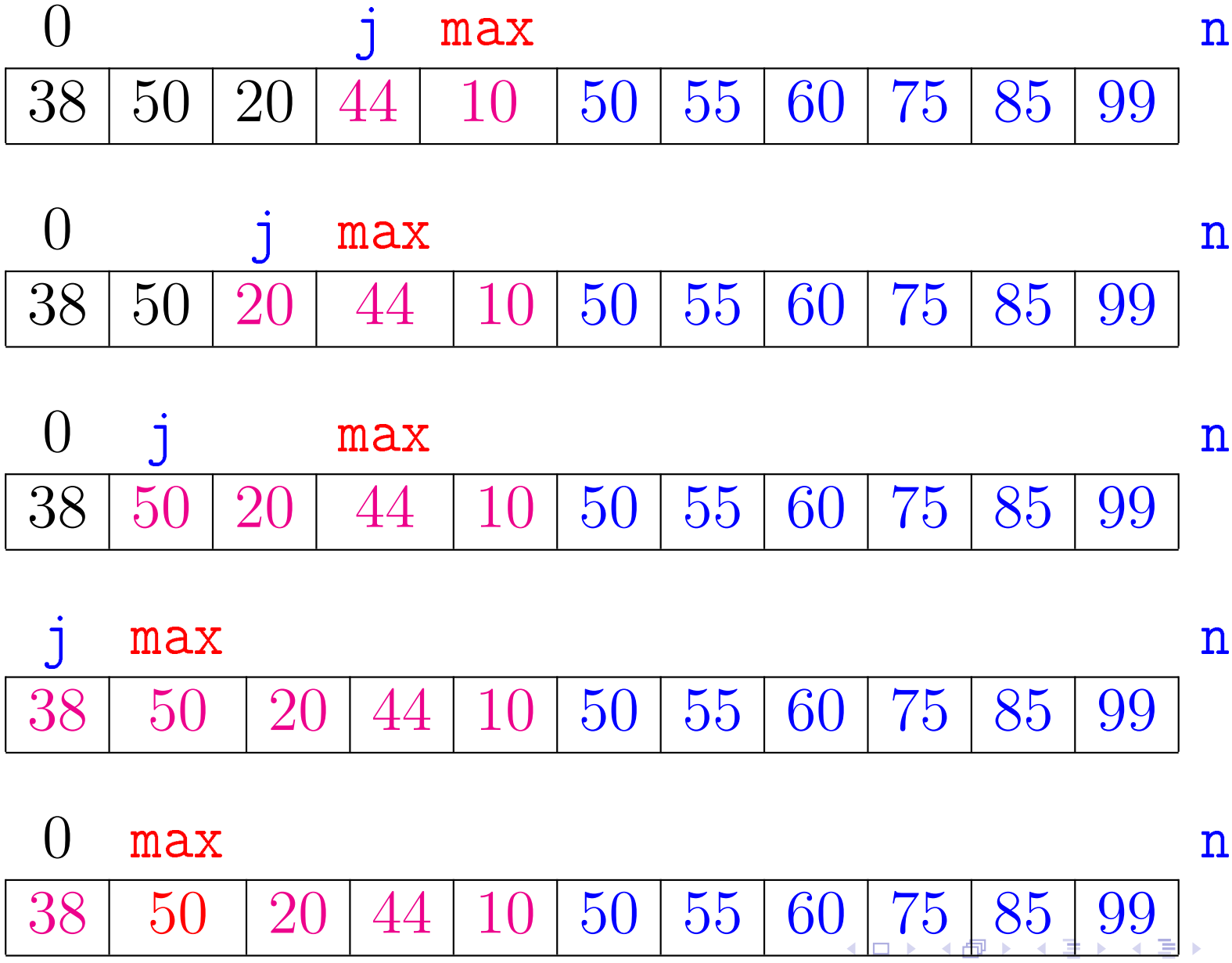
0		j	max							n
38	50	20	44	10	50	55	60	75	85	99

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

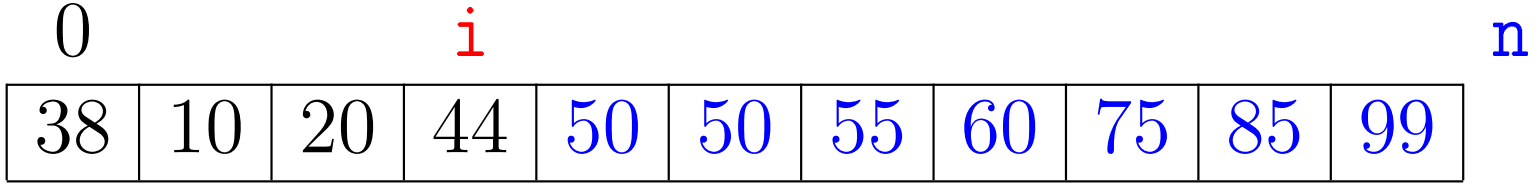
j	max									n
38	50	20	44	10	50	55	60	75	85	99

Ordenação por seleção (iteração)

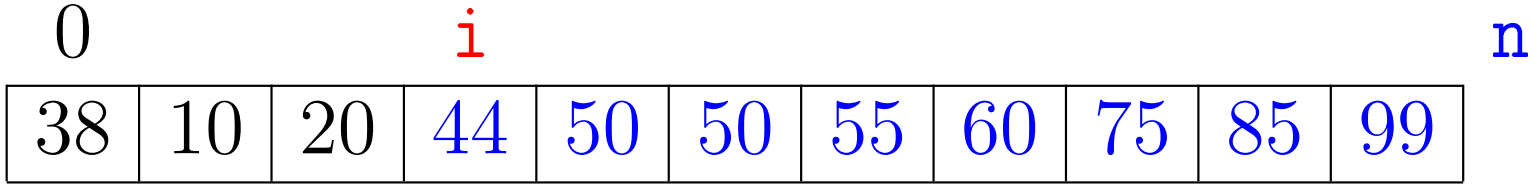
i = 5

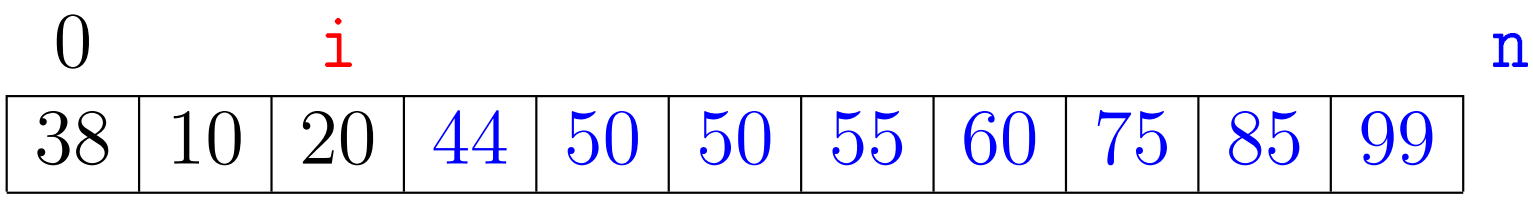


Ordenação por seleção

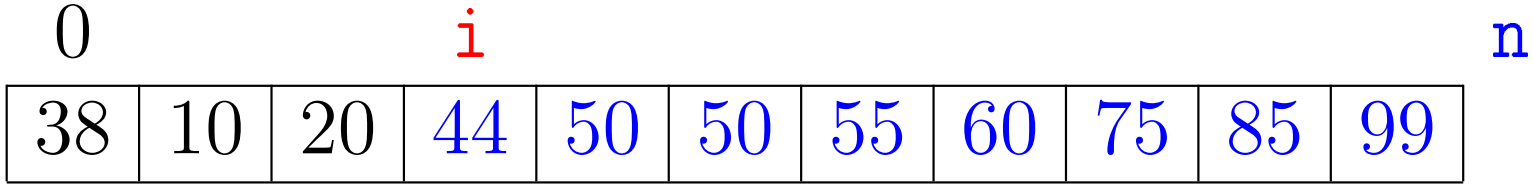


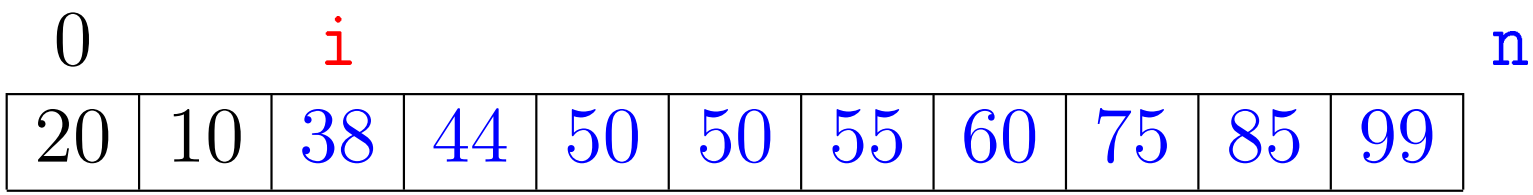
Ordenação por seleção

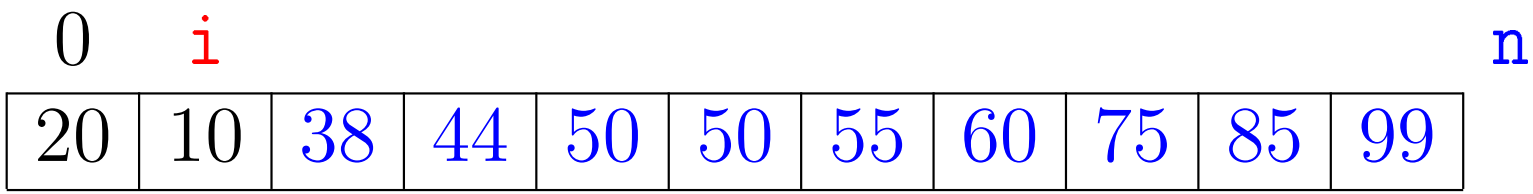




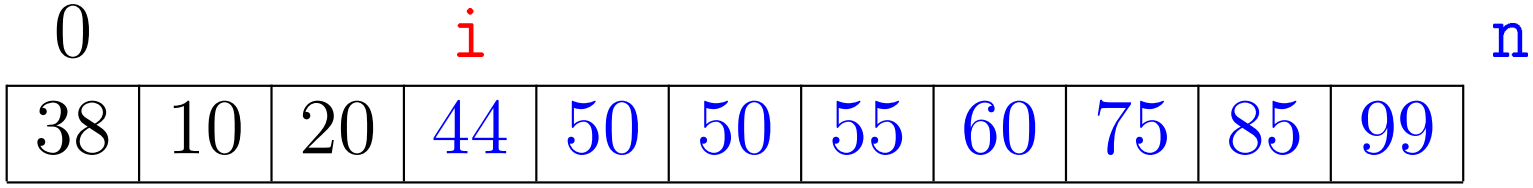
Ordenação por seleção

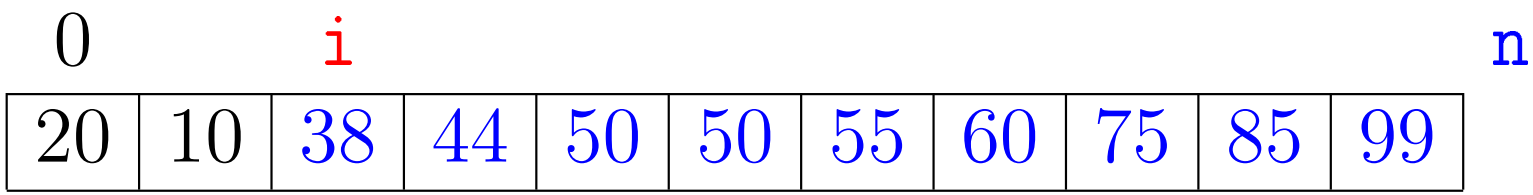


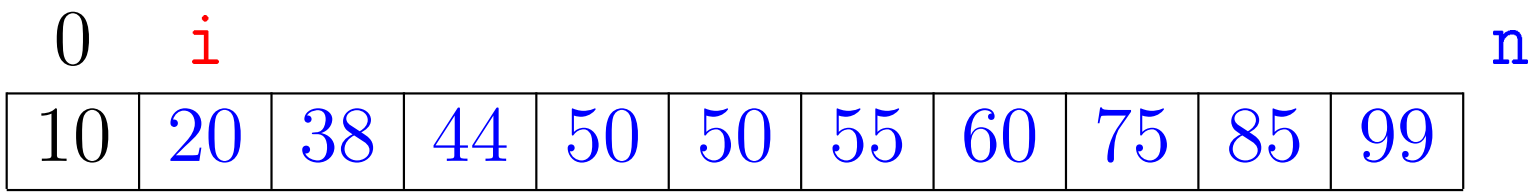


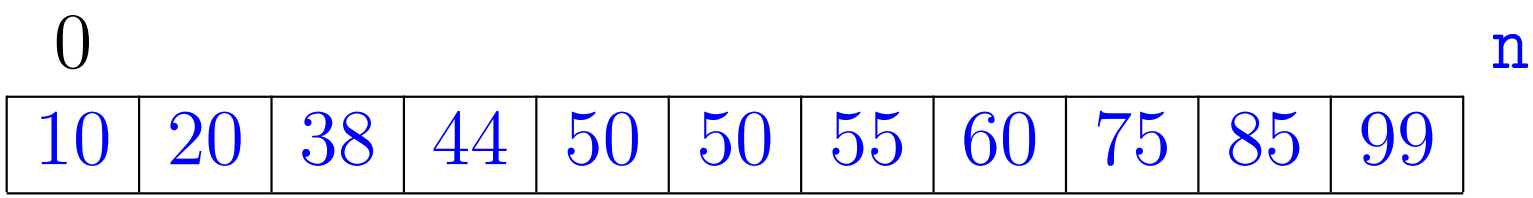


Ordenação por seleção









Função selecao

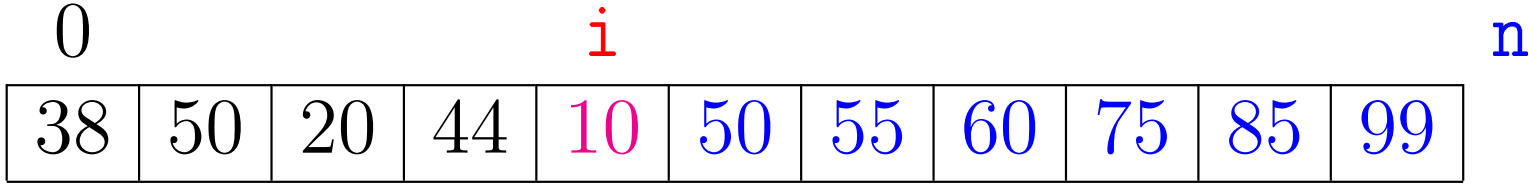
Algoritmo rearranja $v[0:n]$ em ordem crescente

```
 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]
```

Invariantes

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

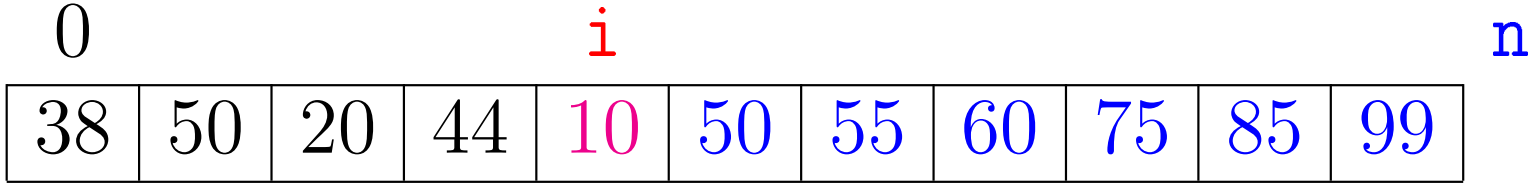
♡ (i0) $\mathtt{v}[\mathtt{i}+1 : \mathtt{n}-1]$ é crescente e
$\mathtt{v}[0 : \mathtt{i}] \leq \mathtt{v}[\mathtt{i}+1 : \mathtt{n}-1]$



Invariantes

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

♡ (i0) $\mathtt{v}[\mathtt{i}+1 : \mathtt{n}-1]$ é crescente e
$\mathtt{v}[0 : \mathtt{i}] \leq \mathtt{v}[\mathtt{i}+1 : \mathtt{n}-1]$

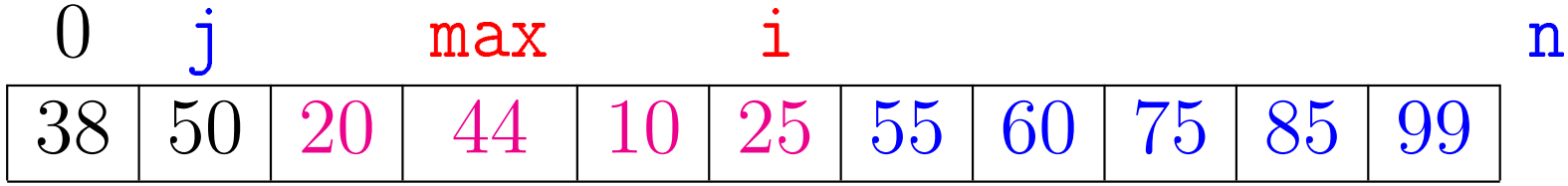


Supondo que a invariantes valem.
Correção do algoritmo é evidente.

No início da última iteração das linhas 1–5 tem-se que $\mathtt{i} = 0$.
Da invariante conclui-se que $\mathtt{v}[1 : \mathtt{n}-1]$ é crescente.
e que $\mathtt{v}[0] \leq \mathtt{v}[1 : \mathtt{n}-1]$.

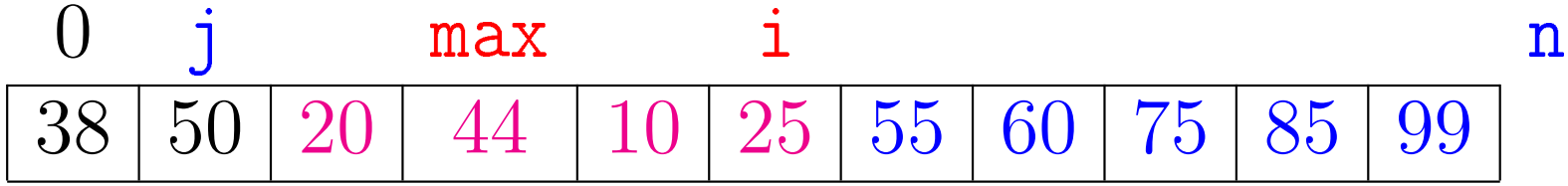
Mais invariantes

Na linha 1 vale que: (i1) $\mathtt{v}[0 : \mathtt{i}] \leq \mathtt{v}[\mathtt{i}+1]$;
Na linha 3 vale que: (i2) $\mathtt{v}[\mathtt{j}+1 : \mathtt{i}] \leq \mathtt{v}[\mathtt{max}]$



Mais invariantes

Na linha 1 vale que: (i1) $\mathtt{v}[0 : \mathtt{i}] \leq \mathtt{v}[\mathtt{i}+1]$;
Na linha 3 vale que: (i2) $\mathtt{v}[\mathtt{j}+1 : \mathtt{i}] \leq \mathtt{v}[\mathtt{max}]$



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
0	$= 1$
1	$= n$
2	$= n - 1$
3	$= n + (n-1) + \cdots + 1 = n(n+1)/2$
4	$= (n-1) + (n-2) + \cdots + 1 = (n-1)n/2$
5	$= n - 1$
total	$= n^2 + 3n - 1$

Conclusão

O consumo de tempo do algoritmo `selecao` no pior caso e no no melhor caso é proporcional a $\mathtt{n}^2$.

O consumo de tempo do algoritmo `selecao` é $\mathrm{O}(\mathtt{n}^2)$.

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	$\approx$ 1 min
65536	266.31	$>$ 4 min
131072	1144.31	$\approx$ 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	$\approx$ 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]
```