"Um computador é como um ar condicionado - se torna inútil se você abrir as *Janelas*" (Linus Torvalds).

Insertion Sort

Paulo Ricardo Lisboa de Almeida

Ordenação de vetores

Considere o seguinte problema

função ordenar (v,a,b)
entrada: vetor v, indexado por [a..b]
saída: o vetor v modificado de forma que $v[i] \leq v[j]$, $\forall\ i,j \in [a..b]$, e $i < j$.

Ordenação de vetores

Considere o seguinte problema

```
função ordenar (v,a,b)
entrada: vetor v, indexado por [a..b]
saída: o vetor v modificado de forma que v[i] ≤ v[j], ∀ i,j ∈ [a..b], e i < j.
```

Existe uma pletora de algoritmos que podemos usar para resolver esse problema:

- Insertion Sort
- Bubble Sort
- Selection Sort
- Merge-Sort
- Quick-Sort
- Estou com Sort (Bogosort)
- ...

Ordenação de vetores

Considere o seguinte problema

função ordenar (v,a,b)
entrada: vetor v, indexado por [a..b]
saída: o vetor v modificado de forma que v[i] ≤ v[j], ∀ i,j ∈ [a..b], e i < j.

Existe uma pletora de algoritmos que podemos usar para resolver esse problema:

Insertion Sort
Bubble Sort
Selection Sort
Merge-Sort
Quick-Sort
Estou com Sort (Bogosort) <- Exemplo de um algoritmo propositalmente ineficiente.
...

Insertion Sort

Ordenação por inserção

Considere o vetor

Começamos pelo vetor de tamanho 1 (em negrito)

O vetor está ordenado

i	**1**	2	3	4	5	6
v[i]	**42**	15	23	8	4	16

Insertion Sort

Adiciona-se o item a direita do vetor ordenado (em azul)

O item adicionado é maior que o item no valor à sua esquerda, então eles devem **trocar** de posição

i	**1**	2	3	4	5	6
v[i]	**42**	15	23	8	4	16

Insertion Sort

Adiciona-se o item a direita do vetor ordenado (em azul)

O item adicionado é maior que o item no valor à sua esquerda, então eles devem **trocar** de posição

Agora o vetor de tamanho 2 em **negrito está ordenado**

Repita o processo

i	1	**2**	3	4	5	6
v[i]	15	**42**	23	8	4	16

Insertion Sort

i	**1**	**2**	3	4	5	6
v[i]	**15**	**42**	23	8	4	16

Insertion Sort

i	**1**	2	**3**	4	5	6
v[i]	**15**	23	**42**	8	4	16

Insertion Sort

i	**1**	**2**	**3**	4	5	6
v[i]	**15**	**23**	**42**	8	4	16

Insertion Sort

i	**1**	**2**	3	**4**	5	6
v[i]	**15**	**23**	8	**42**	4	16

Insertion Sort

Note que continuamos a mover o item para a esquerda, até ele chegar em sua posição correta

i	**1**	2	**3**	**4**	5	6
v[i]	**15**	8	**23**	**42**	4	16

Insertion Sort

Note que continuamos a mover o item para a esquerda, até ele chegar em sua posição correta

i	1	**2**	**3**	**4**	5	6
v[i]	8	**15**	**23**	**42**	4	16

Insertion Sort

i	**1**	**2**	**3**	**4**	5	6
v[i]	**8**	**15**	**23**	**42**	4	16

Insertion Sort

i	**1**	**2**	**3**	4	**5**	6
v[i]	**8**	**15**	**23**	4	**42**	16

Insertion Sort

i	**1**	**2**	3	**4**	**5**	6
v[i]	**8**	**15**	4	**23**	**42**	16

Insertion Sort

i	**1**	2	**3**	**4**	**5**	6
v[i]	**8**	4	**15**	**23**	**42**	16

Insertion Sort

i	1	**2**	**3**	**4**	**5**	6
v[i]	4	**8**	**15**	**23**	**42**	16

Insertion Sort

i	1	2	3	4	5	6
v[i]	4	8	15	23	42	16

Insertion Sort

i	1	2	3	4	5	6
v[i]	4	8	15	23	16	42

Insertion Sort

i	1	2	3	4	5	6
v[i]	4	8	15	16	23	42

Insertion Sort

Ordenado por inserção!

i	1	2	3	4	5	6
v[i]	4	8	15	16	23	42

Faça você mesmo

Tente criar uma versão do algoritmo você mesmo

- Em portugol ou C - iterativo ou recursivo (a escolha é sua)
- 40 minutos

Insertion Sort

```
função insertionSort (v,a,b)
entrada: vetor v, indexado por [a..b]
saída: o vetor v modificado de forma que v[i] ≤ v[j], ∀ i,j ∈ [a..b], e i < j.
se a ≥ b
      retorne v
insertionSort(v,a,b-1)
inserir(v,a,b)
retorne v
```

Insertion Sort

```
função insertionSort (v,a,b)
entrada: vetor v, indexado por [a..b]
saída: o vetor v modificado de forma que v[i] ≤ v[j], ∀ i,j ∈ [a..b], e i < j.
se a ≥ b
      retorne v
insertionSort(v,a,b-1)
inserir(v,a,b)
retorne v
```

```
função inserir (v,a,b)
entrada: vetor v, indexado por [a..b], em que (v,a,b-1) é um vetor ordenado
saída: o vetor v modificado de forma que v[i] ≤ v[j], ∀ i,j ∈ [a..b], e i < j.
p ← buscar(v[b],v,a,b-1)
i ← b
enquanto i > p + 1
      trocar(v, i, i - 1)
      i ← i - 1
retorne v
```

Insertion Sort

```
função insertionSort (v,a,b)
entrada: vetor v, indexado por [a..b]
saída: o vetor v modificado de forma que v[i] ≤ v[j], ∀ i,j ∈ [a..b], e i < j.
se a ≥ b
      retorne v
insertionSort(v,a,b-1)
inserir(v,a,b)
retorne v
```

Obs.: a função buscar é a mesma vista em aulas passadas

```
função inserir (v,a,b)
entrada: vetor v, indexado por [a..b], em que (v,a,b-1) é um vetor ordenado
saída: o vetor v modificado de forma que v[i] ≤ v[j], ∀ i,j ∈ [a..b], e i < j.
p ← buscar(v[b],v,a,b-1)
i ← b
enquanto i > p + 1
      trocar(v, i, i - 1)
      i ← i - 1
retorne v
```

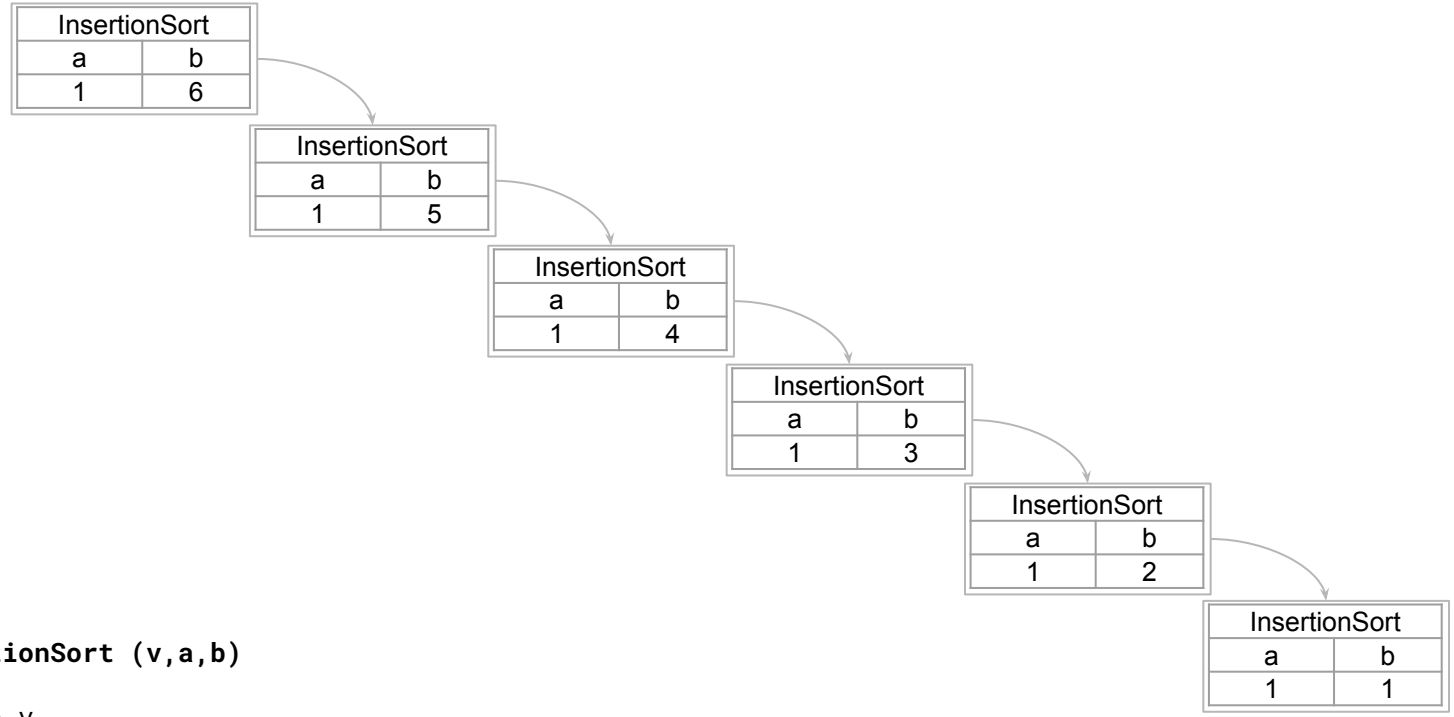
```
função trocar (v,a,b)
entrada: vetor v, indexado por [a..b]
saída: o vetor v modificado de forma que v[a] é trocado com v[b]
x ← v[a]
v[a] ← v[b]
v[b] ← x
```

InsertionSort	
a	b
1	6

```
função insertionSort (v,a,b)
se a ≥ b
      retorne v
insertionSort(v,a,b-1)
inserir(v,a,b)
retorne v
```

```
função inserir (v,a,b)
p ← buscar(v[b],v,a,b-1)
i ← b
enquanto i > p + 1
      trocar(v, i, i - 1)
      i ← i - 1
retorne v
```

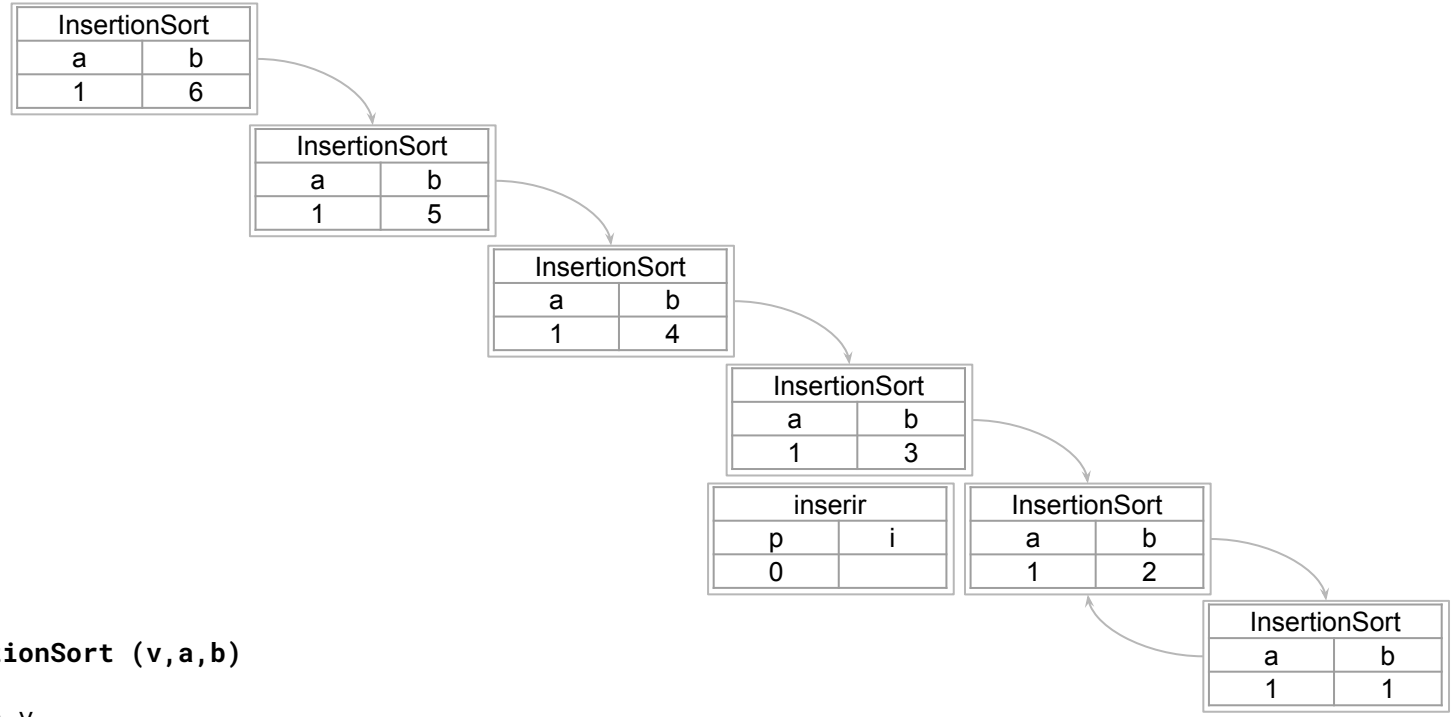
i	1	2	3	4	5	6
v[i]	42	15	23	8	4	16




```
função insertionSort (v,a,b)
se a ≥ b
      retorne v
insertionSort(v,a,b-1)
inserir(v,a,b)
retorne v
```

```
função inserir (v,a,b)
p ← buscar(v[b],v,a,b-1)
i ← b
enquanto i > p + 1
      trocar(v, i, i - 1)
      i ← i - 1
retorne v
```

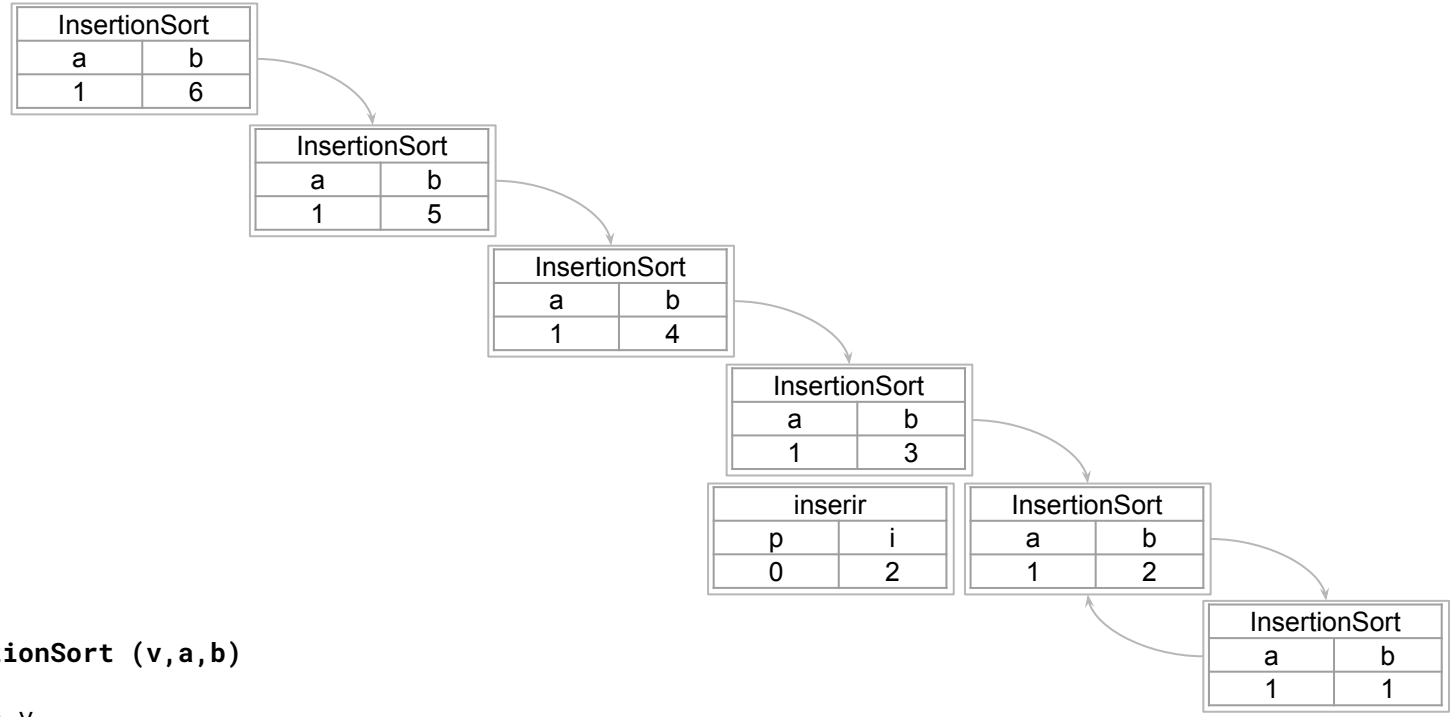
i	1	2	3	4	5	6
v[i]	42	15	23	8	4	16




```
função insertionSort (v,a,b)
se a ≥ b
      retorne v
insertionSort(v,a,b-1)
inserir(v,a,b)
retorne v
```

```
função inserir (v,a,b)
p ← buscar(v[b],v,a,b-1)
i ← b
enquanto i > p + 1
      trocar(v, i, i - 1)
      i ← i - 1
retorne v
```

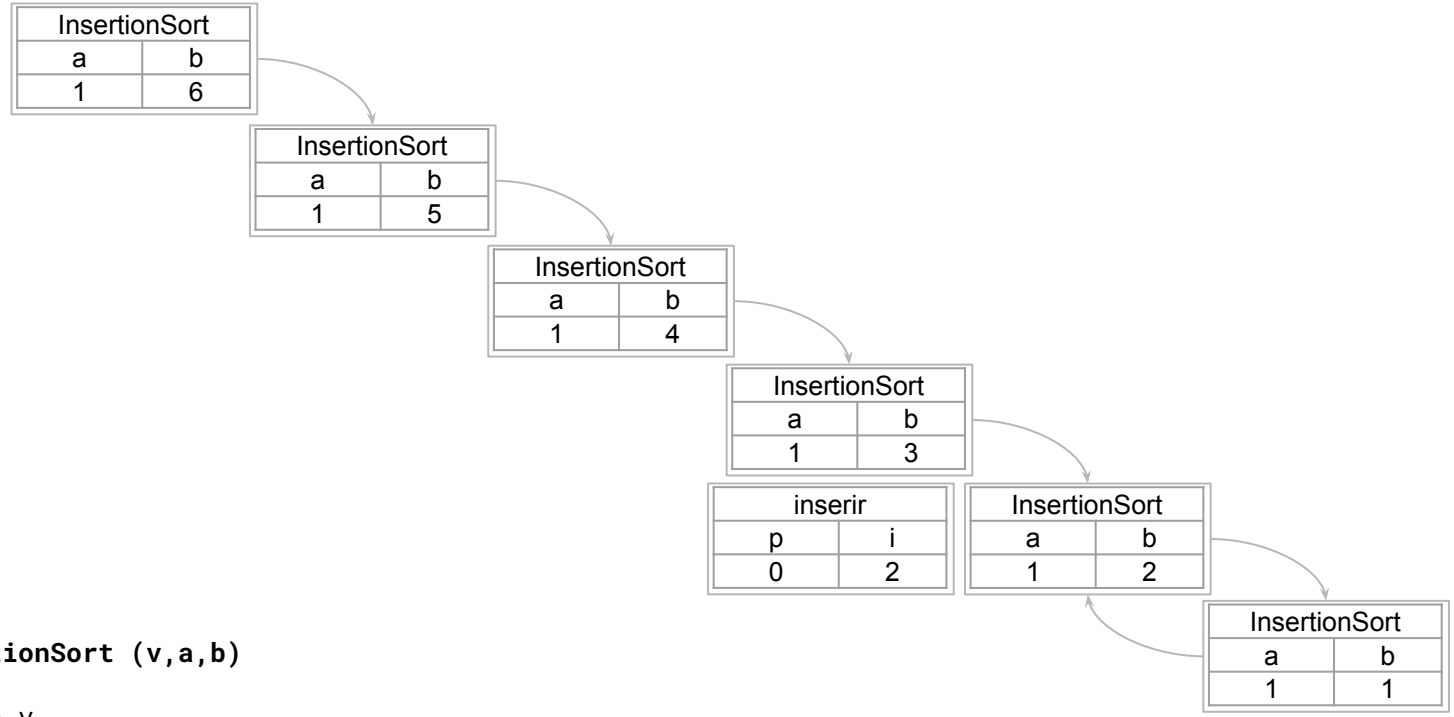
i	**1**	2	3	4	5	6
v[i]	**42**	15	23	8	4	16




```
função insertionSort (v,a,b)
se a ≥ b
      retorne v
insertionSort(v,a,b-1)
inserir(v,a,b)
retorne v
```

```
função inserir (v,a,b)
p ← buscar(v[b],v,a,b-1)
i ← b
enquanto i > p + 1
      trocar(v, i, i - 1)
      i ← i - 1
retorne v
```

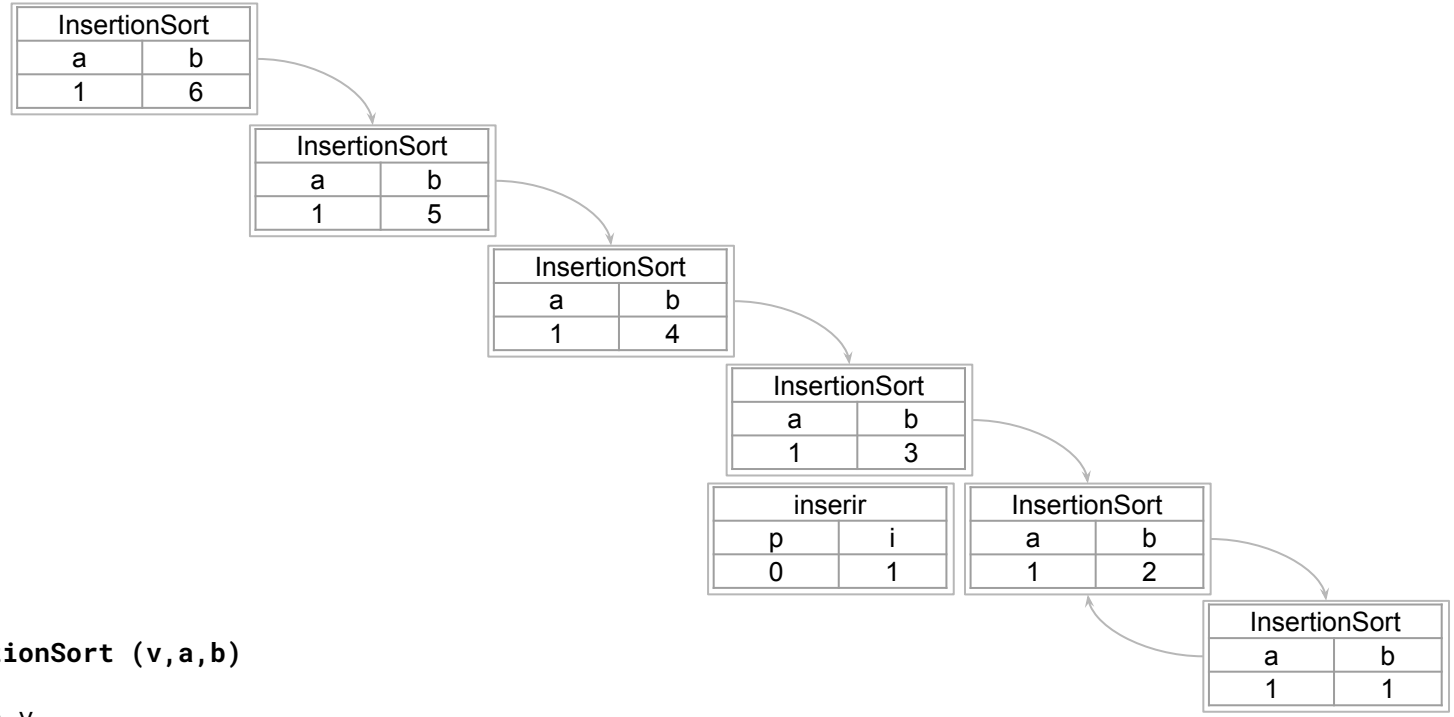
i	**1**	2	3	4	5	6
v[i]	**42**	15	23	8	4	16




```
função insertionSort (v,a,b)
se a ≥ b
      retorne v
insertionSort(v,a,b-1)
inserir(v,a,b)
retorne v
```

```
função inserir (v,a,b)
p ← buscar(v[b],v,a,b-1)
i ← b
enquanto i > p + 1
      trocar(v, i, i - 1)
      i ← i - 1
retorne v
```

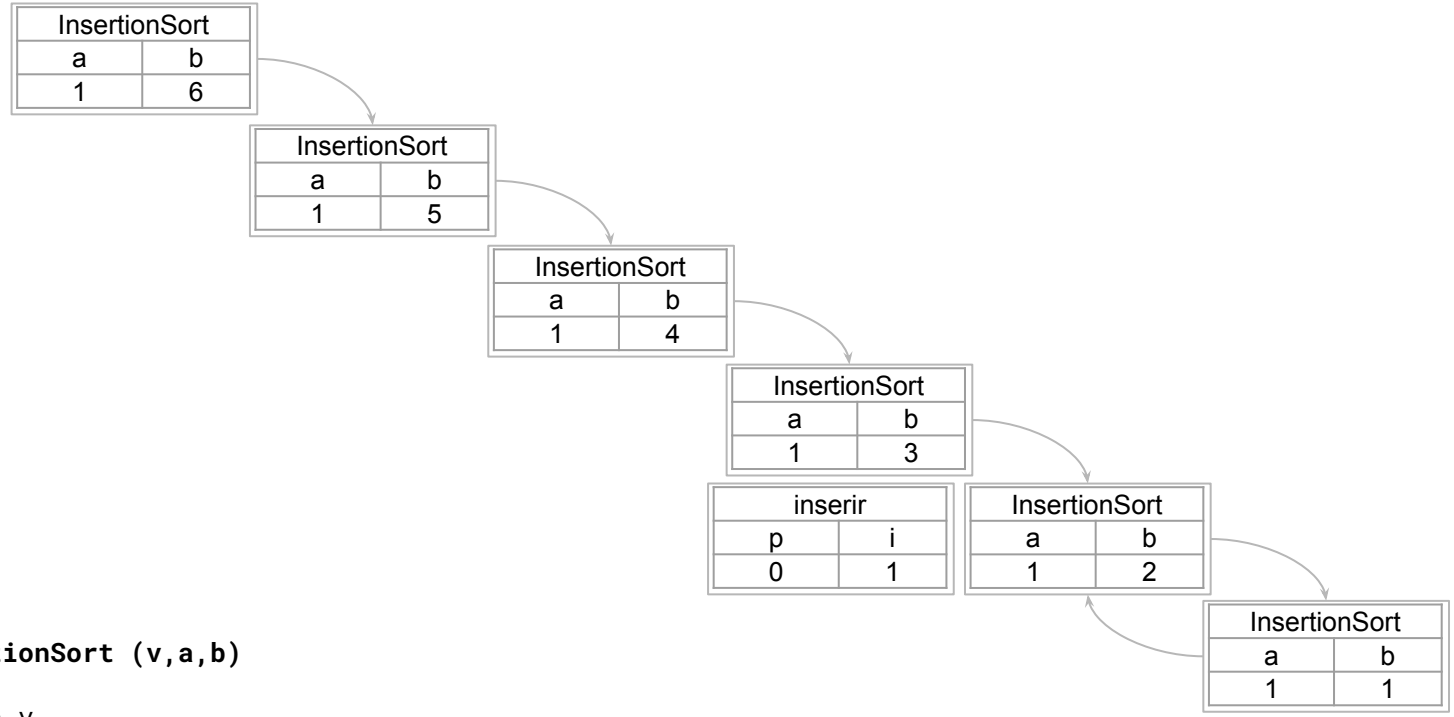
i	1	**2**	3	4	5	6
v[i]	15	**42**	23	8	4	16




```
função insertionSort (v,a,b)
se a ≥ b
      retorne v
insertionSort(v,a,b-1)
inserir(v,a,b)
retorne v
```

```
função inserir (v,a,b)
p ← buscar(v[b],v,a,b-1)
i ← b
enquanto i > p + 1
      trocar(v, i, i - 1)
      i ← i - 1
retorne v
```

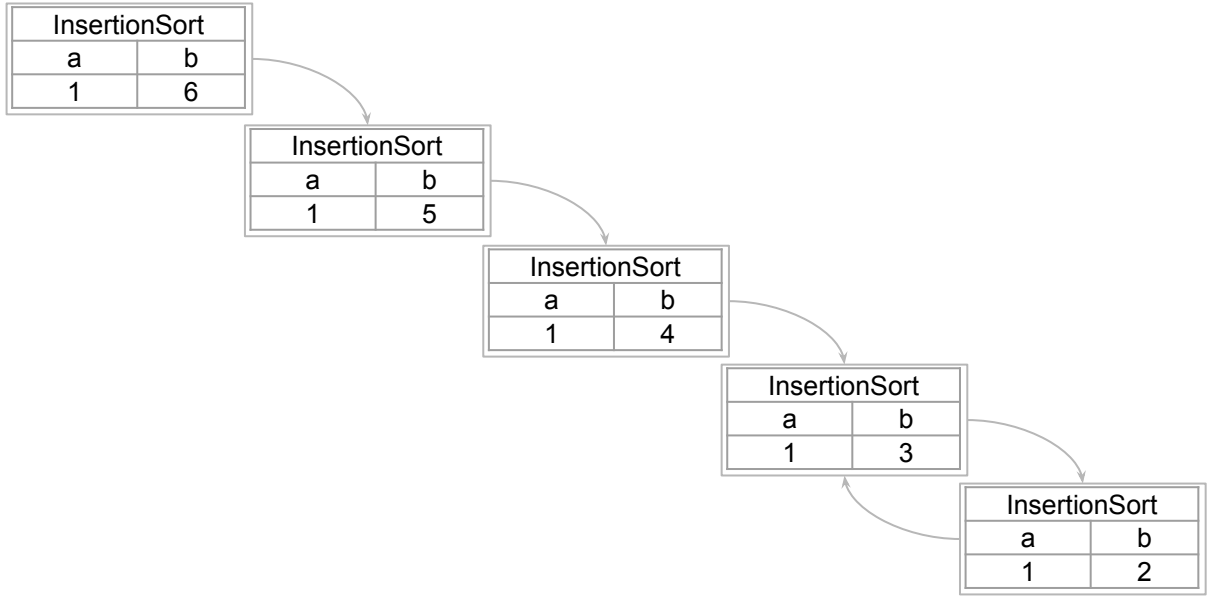
i	1	**2**	3	4	5	6
v[i]	15	**42**	23	8	4	16




```
função insertionSort (v,a,b)
se a ≥ b
      retorne v
insertionSort(v,a,b-1)
inserir(v,a,b)
retorne v
```

```
função inserir (v,a,b)
p ← buscar(v[b],v,a,b-1)
i ← b
enquanto i > p + 1
      trocar(v, i, i - 1)
      i ← i - 1
retorne v
```

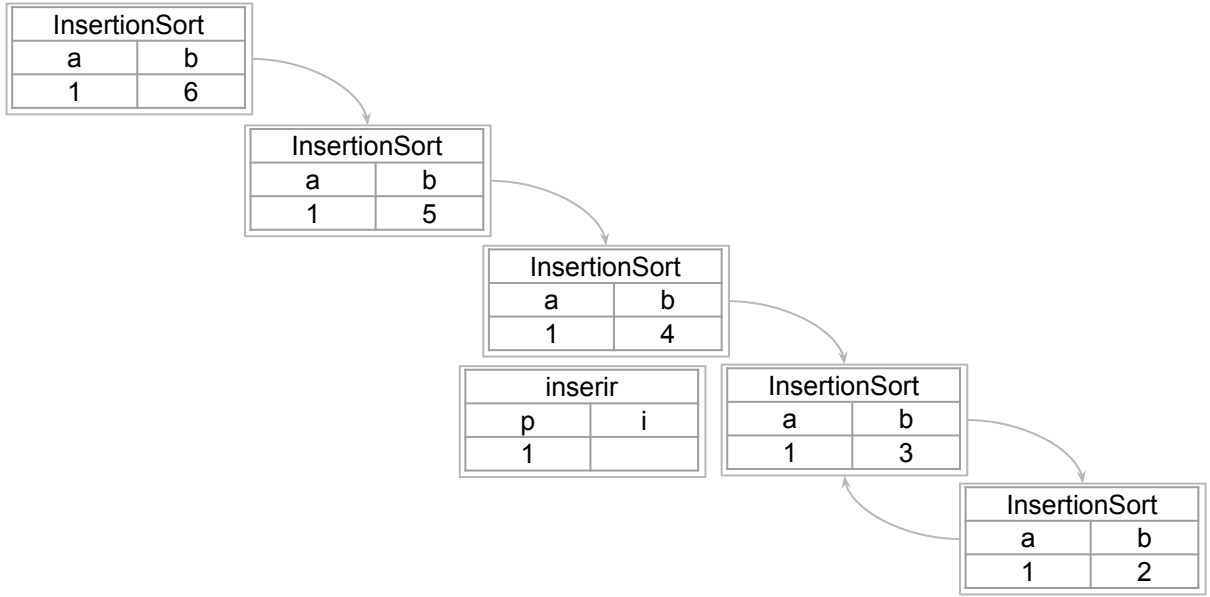
i	1	**2**	3	4	5	6
v[i]	15	**42**	23	8	4	16




```
função insertionSort (v,a,b)
se a ≥ b
      retorne v
insertionSort(v,a,b-1)
inserir(v,a,b)
retorne v
```

```
função inserir (v,a,b)
p ← buscar(v[b],v,a,b-1)
i ← b
enquanto i > p + 1
      trocar(v, i, i - 1)
      i ← i - 1
retorne v
```

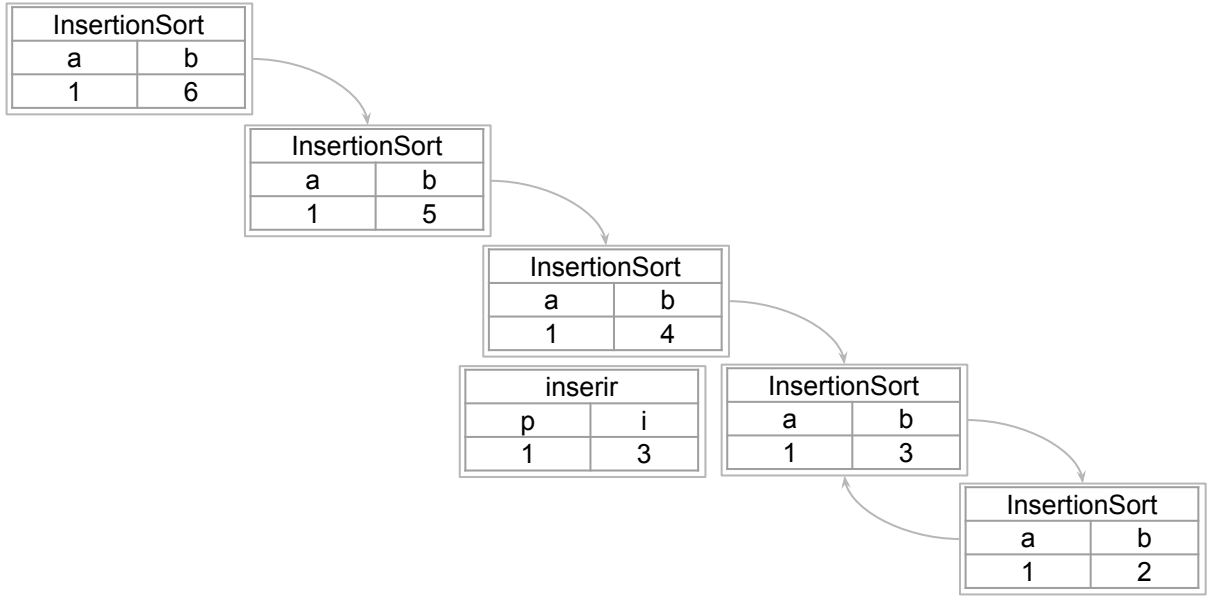
i	**1**	**2**	3	4	5	6
v[i]	**15**	**42**	23	8	4	16




```
função insertionSort (v,a,b)
se a ≥ b
      retorne v
insertionSort(v,a,b-1)
inserir(v,a,b)
retorne v
```

```
função inserir (v,a,b)
p ← buscar(v[b],v,a,b-1)
i ← b
enquanto i > p + 1
      trocar(v, i, i - 1)
      i ← i - 1
retorne v
```

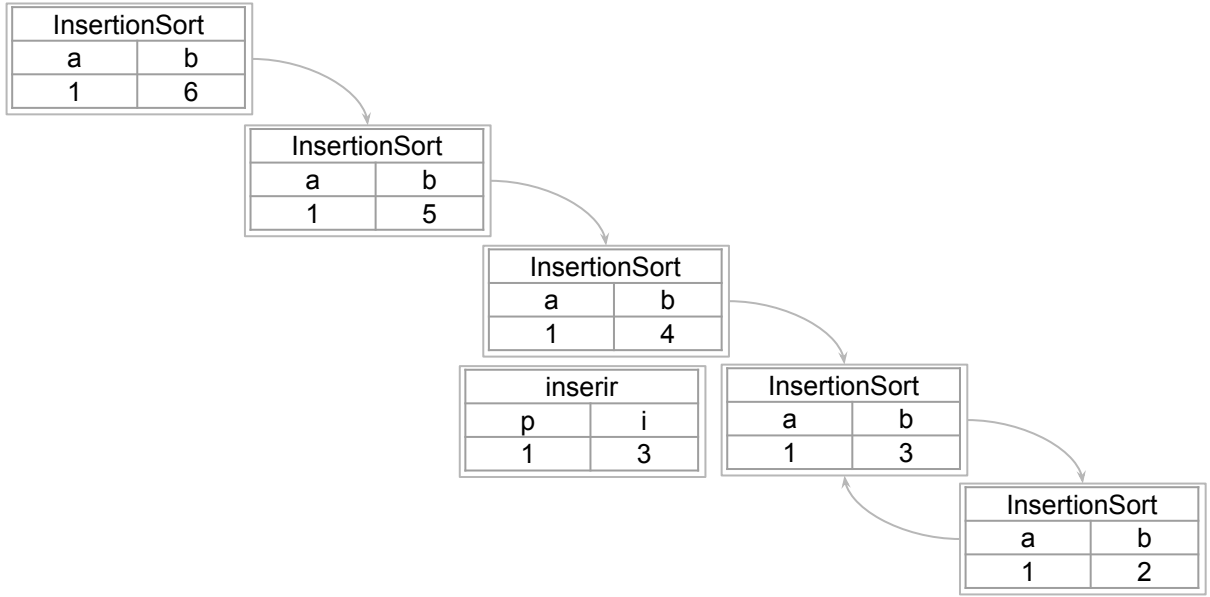
i	**1**	**2**	3	4	5	6
v[i]	**15**	**42**	23	8	4	16




```
função insertionSort (v,a,b)
se a ≥ b
      retorne v
insertionSort(v,a,b-1)
inserir(v,a,b)
retorne v
```

```
função inserir (v,a,b)
p ← buscar(v[b],v,a,b-1)
i ← b
enquanto i > p + 1
      trocar(v, i, i - 1)
      i ← i - 1
retorne v
```

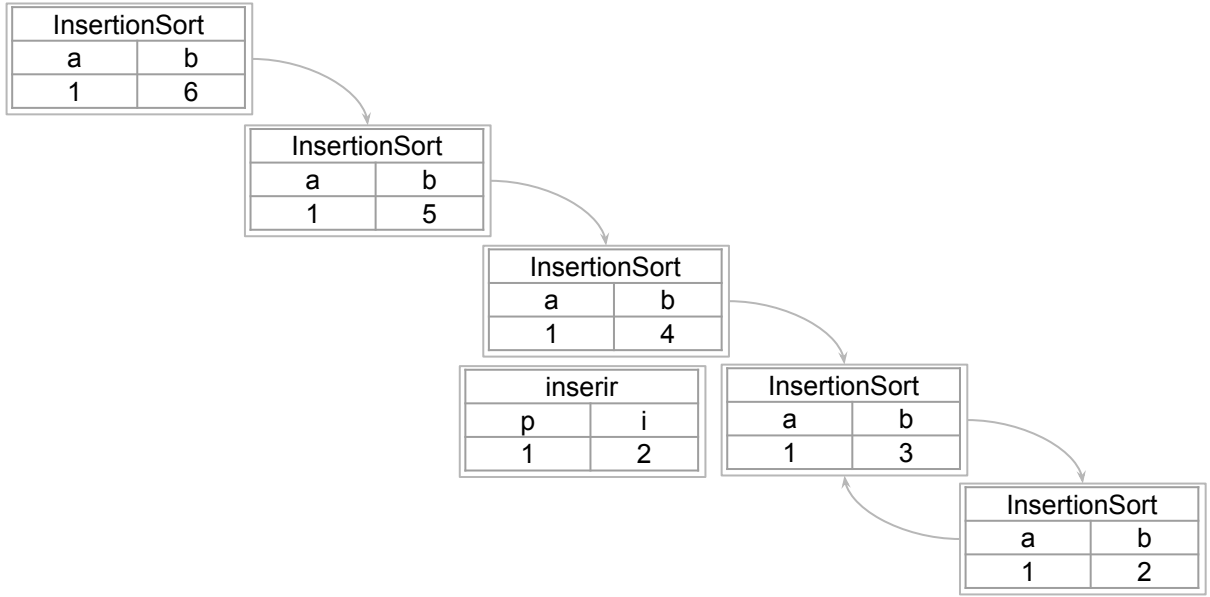
i	**1**	**2**	3	4	5	6
v[i]	**15**	**42**	23	8	4	16




```
função insertionSort (v,a,b)
se a ≥ b
      retorne v
insertionSort(v,a,b-1)
inserir(v,a,b)
retorne v
```

```
função inserir (v,a,b)
p ← buscar(v[b],v,a,b-1)
i ← b
enquanto i > p + 1
      trocar(v, i, i - 1)
      i ← i - 1
retorne v
```

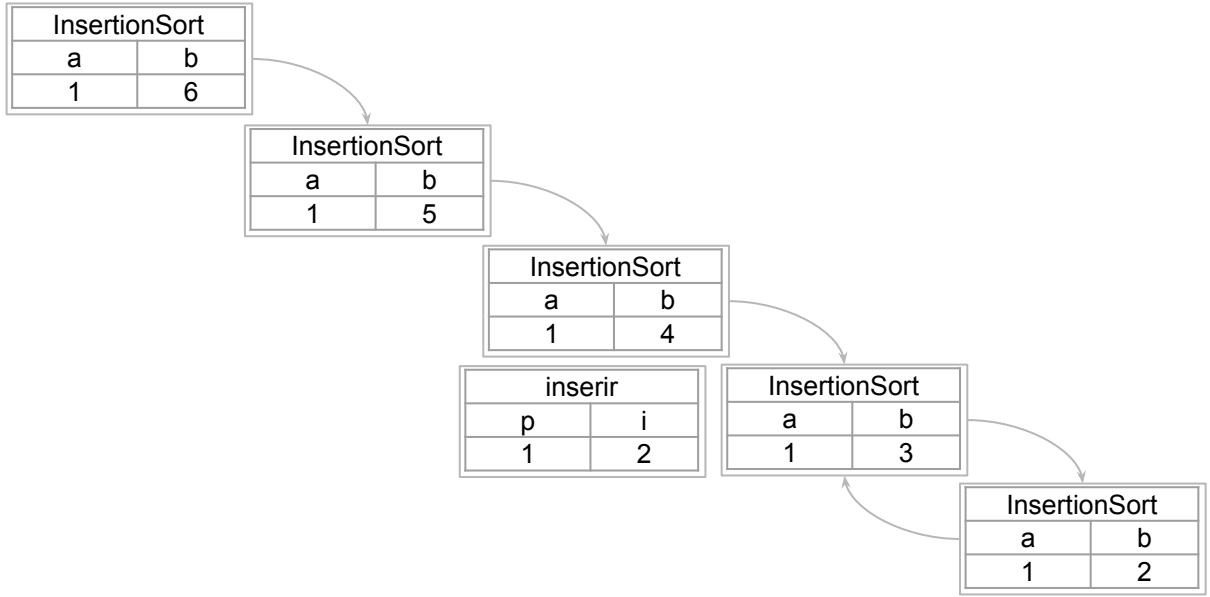
i	**1**	2	**3**	4	5	6
v[i]	**15**	23	**42**	8	4	16




```
função insertionSort (v,a,b)
se a ≥ b
      retorne v
insertionSort(v,a,b-1)
inserir(v,a,b)
retorne v
```

```
função inserir (v,a,b)
p ← buscar(v[b],v,a,b-1)
i ← b
enquanto i > p + 1
      trocar(v, i, i - 1)
      i ← i - 1
retorne v
```

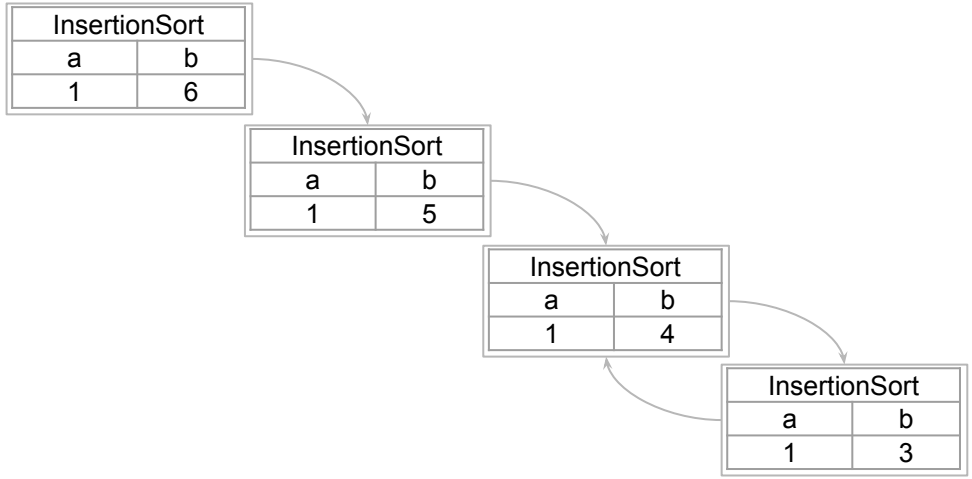
i	**1**	2	**3**	4	5	6
v[i]	**15**	23	**42**	8	4	16




```
função insertionSort (v,a,b)
se a ≥ b
      retorne v
insertionSort(v,a,b-1)
inserir(v,a,b)
retorne v
```

```
função inserir (v,a,b)
p ← buscar(v[b],v,a,b-1)
i ← b
enquanto i > p + 1
      trocar(v, i, i - 1)
      i ← i - 1
retorne v
```

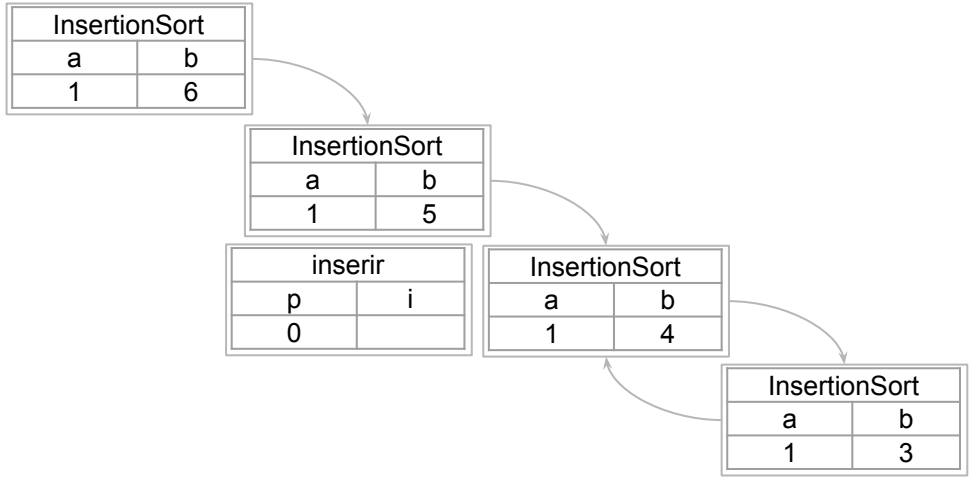
i	**1**	**2**	**3**	4	5	6
v[i]	**15**	**23**	**42**	8	4	16




```
função insertionSort (v,a,b)
se a ≥ b
      retorne v
insertionSort(v,a,b-1)
inserir(v,a,b)
retorne v
```

```
função inserir (v,a,b)
p ← buscar(v[b],v,a,b-1)
i ← b
enquanto i > p + 1
      trocar(v, i, i - 1)
      i ← i - 1
retorne v
```

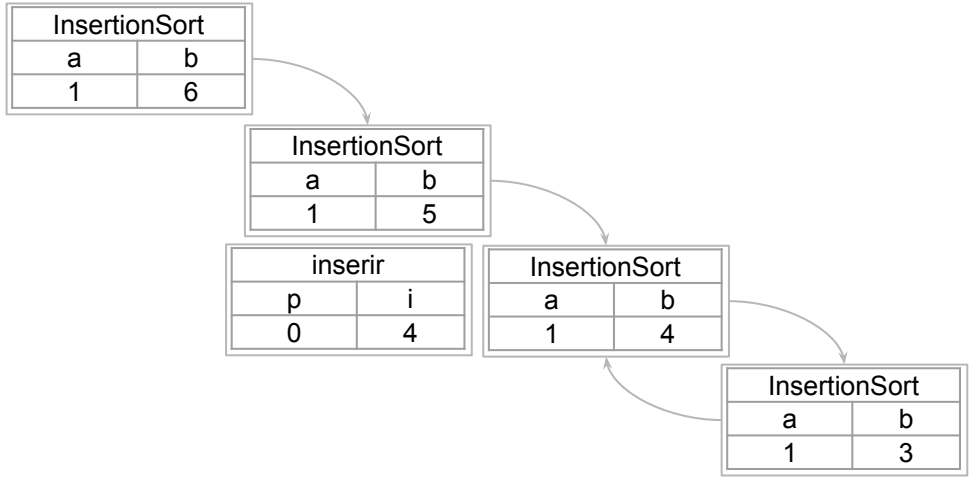
i	**1**	**2**	**3**	4	5	6
v[i]	**15**	**23**	**42**	8	4	16




```
função insertionSort (v,a,b)
se a ≥ b
      retorne v
insertionSort(v,a,b-1)
inserir(v,a,b)
retorne v
```

```
função inserir (v,a,b)
p ← buscar(v[b],v,a,b-1)
i ← b
enquanto i > p + 1
      trocar(v, i, i - 1)
      i ← i - 1
retorne v
```

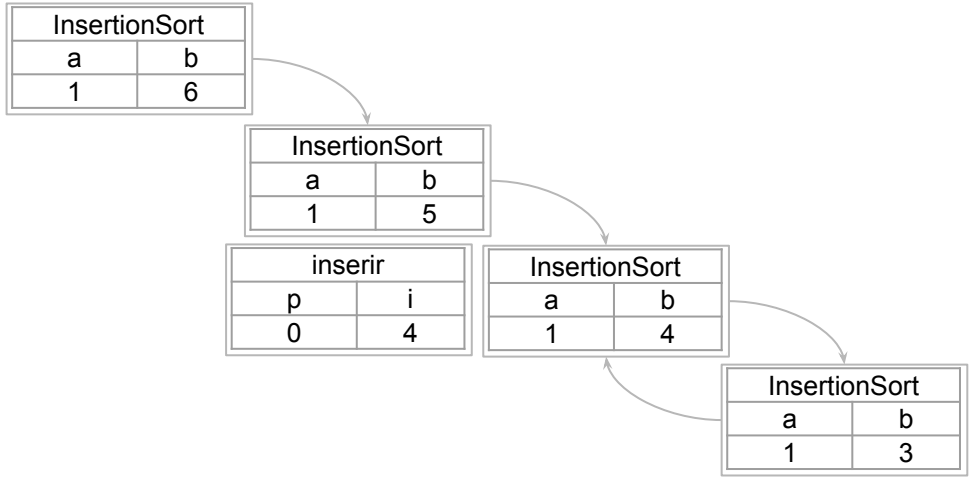
i	**1**	**2**	**3**	4	5	6
v[i]	**15**	**23**	**42**	8	4	16




```
função insertionSort (v,a,b)
se a ≥ b
      retorne v
insertionSort(v,a,b-1)
inserir(v,a,b)
retorne v
```

```
função inserir (v,a,b)
p ← buscar(v[b],v,a,b-1)
i ← b
enquanto i > p + 1
      trocar(v, i, i - 1)
      i ← i - 1
retorne v
```

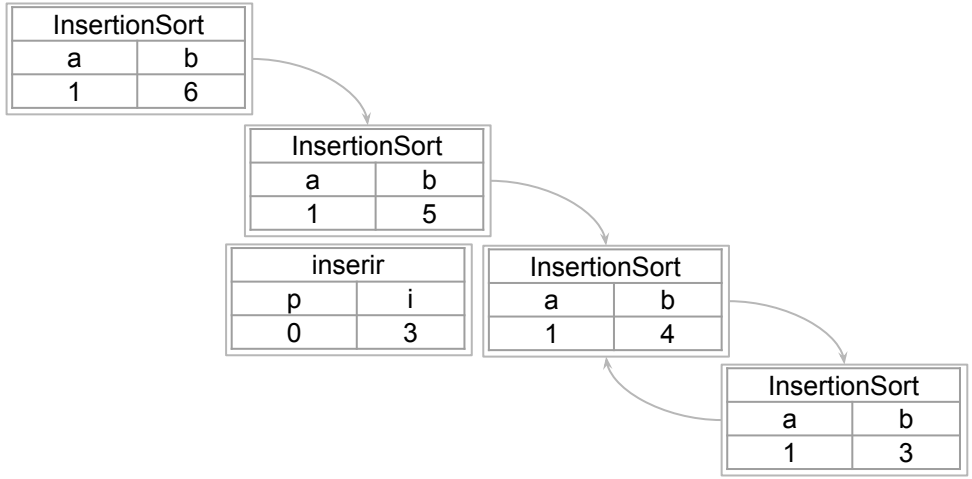
i	**1**	**2**	**3**	4	5	6
v[i]	**15**	**23**	**42**	8	4	16




```
função insertionSort (v,a,b)
se a ≥ b
      retorne v
insertionSort(v,a,b-1)
inserir(v,a,b)
retorne v
```

```
função inserir (v,a,b)
p ← buscar(v[b],v,a,b-1)
i ← b
enquanto i > p + 1
      trocar(v, i, i - 1)
      i ← i - 1
retorne v
```

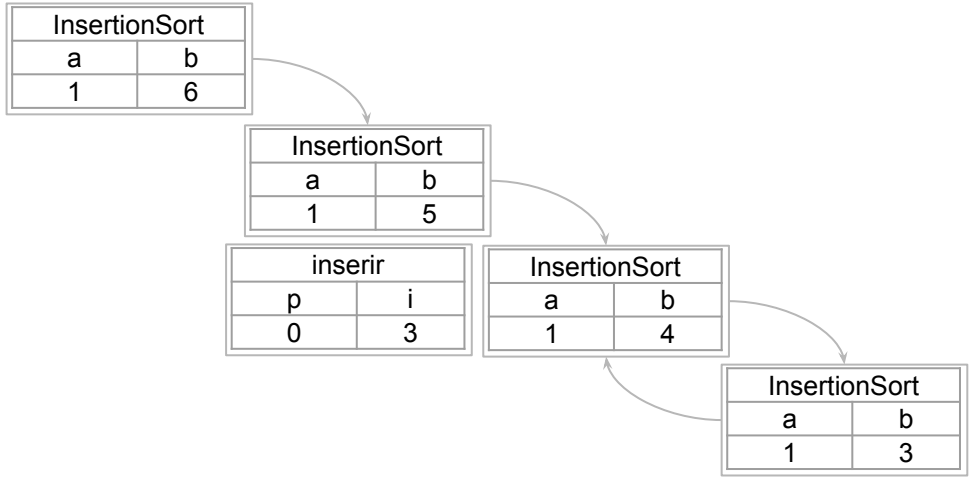
i	**1**	**2**	3	**4**	5	6
v[i]	**15**	**23**	8	**42**	4	16




```
função insertionSort (v,a,b)
se a ≥ b
      retorne v
insertionSort(v,a,b-1)
inserir(v,a,b)
retorne v
```

```
função inserir (v,a,b)
p ← buscar(v[b],v,a,b-1)
i ← b
enquanto i > p + 1
      trocar(v, i, i - 1)
      i ← i - 1
retorne v
```

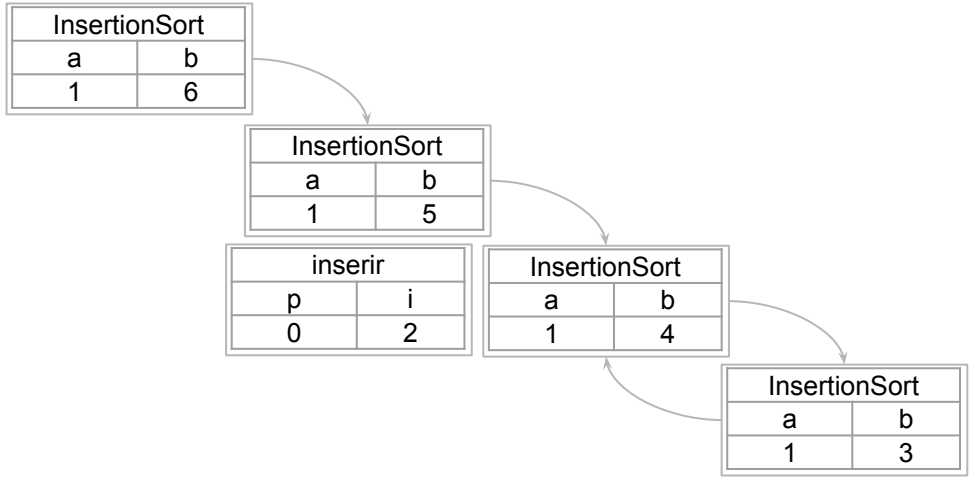
i	**1**	**2**	3	**4**	5	6
v[i]	**15**	**23**	8	**42**	4	16




```
função insertionSort (v,a,b)
se a ≥ b
      retorne v
insertionSort(v,a,b-1)
inserir(v,a,b)
retorne v
```

```
função inserir (v,a,b)
p ← buscar(v[b],v,a,b-1)
i ← b
enquanto i > p + 1
      trocar(v, i, i - 1)
      i ← i - 1
retorne v
```

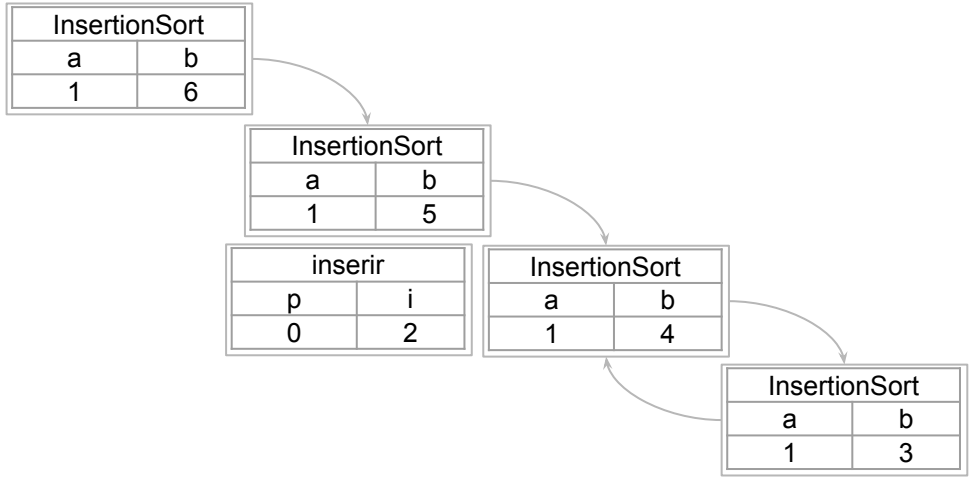
i	**1**	2	**3**	**4**	5	6
v[i]	**15**	8	**23**	**42**	4	16




```
função insertionSort (v,a,b)
se a ≥ b
      retorne v
insertionSort(v,a,b-1)
inserir(v,a,b)
retorne v
```

```
função inserir (v,a,b)
p ← buscar(v[b],v,a,b-1)
i ← b
enquanto i > p + 1
      trocar(v, i, i - 1)
      i ← i - 1
retorne v
```

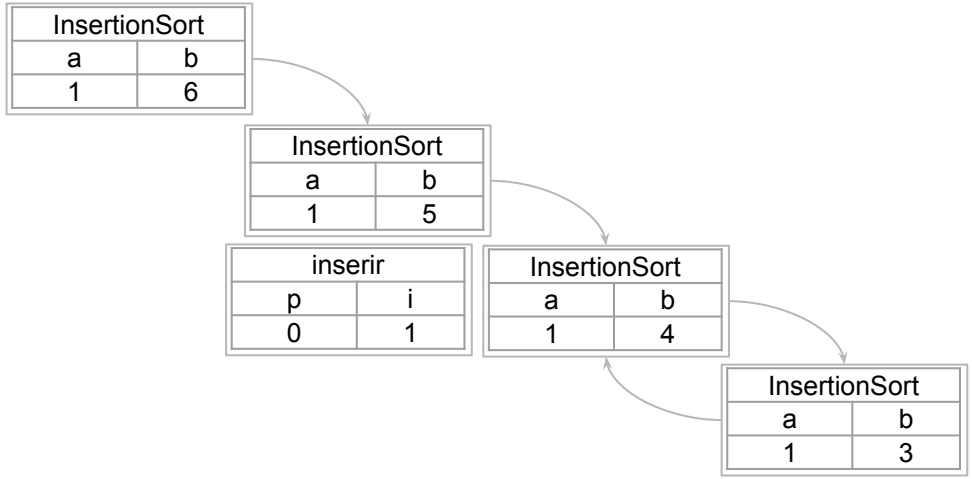
i	**1**	2	**3**	**4**	5	6
v[i]	**15**	8	**23**	**42**	4	16




```
função insertionSort (v,a,b)
se a ≥ b
      retorne v
insertionSort(v,a,b-1)
inserir(v,a,b)
retorne v
```

```
função inserir (v,a,b)
p ← buscar(v[b],v,a,b-1)
i ← b
enquanto i > p + 1
      trocar(v, i, i - 1)
      i ← i - 1
retorne v
```

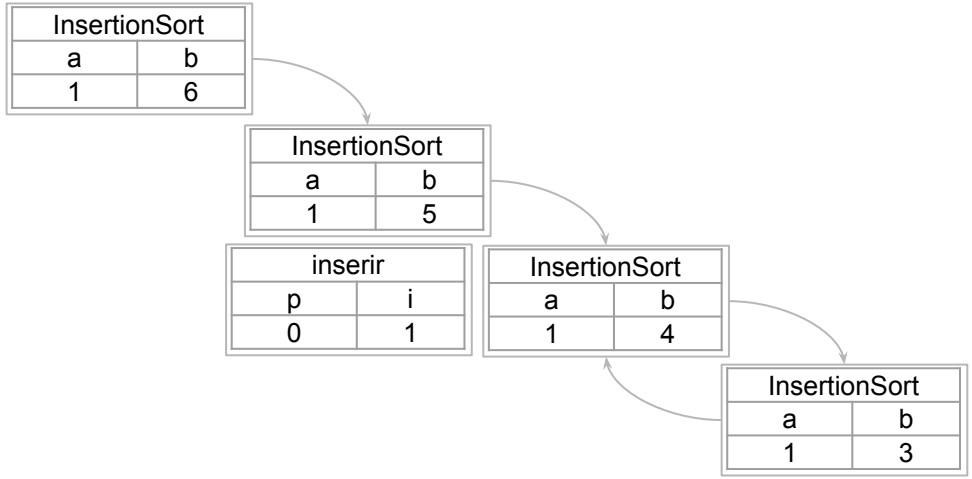
i	1	**2**	**3**	**4**	5	6
v[i]	8	**15**	**23**	**42**	4	16




```
função insertionSort (v,a,b)
se a ≥ b
      retorne v
insertionSort(v,a,b-1)
inserir(v,a,b)
retorne v
```

```
função inserir (v,a,b)
p ← buscar(v[b],v,a,b-1)
i ← b
enquanto i > p + 1
      trocar(v, i, i - 1)
      i ← i - 1
retorne v
```

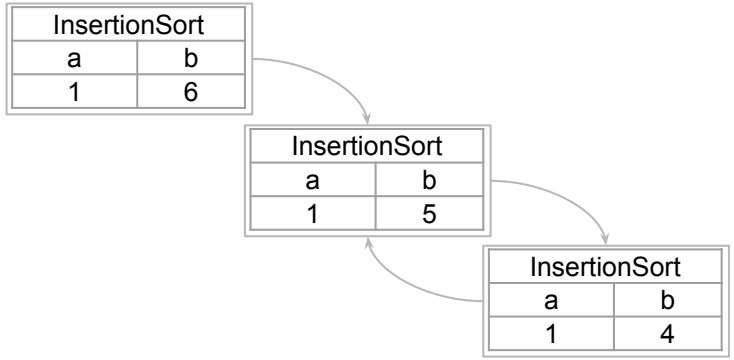
i	1	**2**	**3**	**4**	5	6
v[i]	8	**15**	**23**	**42**	4	16




```
função insertionSort (v,a,b)
se a ≥ b
      retorne v
insertionSort(v,a,b-1)
inserir(v,a,b)
retorne v
```

```
função inserir (v,a,b)
p ← buscar(v[b],v,a,b-1)
i ← b
enquanto i > p + 1
      trocar(v, i, i - 1)
      i ← i - 1
retorne v
```

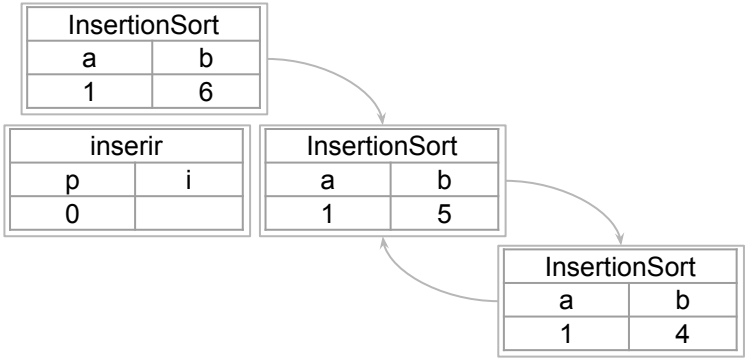
i	**1**	**2**	**3**	**4**	5	6
v[i]	**8**	**15**	**23**	**42**	4	16




```
função insertionSort (v,a,b)
se a ≥ b
      retorne v
insertionSort(v,a,b-1)
inserir(v,a,b)
retorne v
```

```
função inserir (v,a,b)
p ← buscar(v[b],v,a,b-1)
i ← b
enquanto i > p + 1
      trocar(v, i, i - 1)
      i ← i - 1
retorne v
```

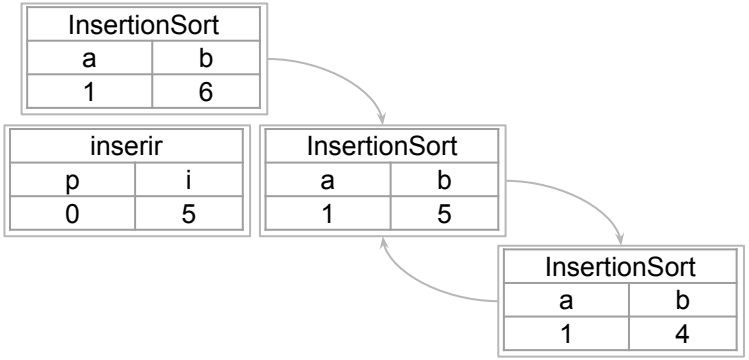
i	**1**	**2**	**3**	**4**	5	6
v[i]	**8**	**15**	**23**	**42**	4	16




```
função insertionSort (v,a,b)
se a ≥ b
      retorne v
insertionSort(v,a,b-1)
inserir(v,a,b)
retorne v
```

```
função inserir (v,a,b)
p ← buscar(v[b],v,a,b-1)
i ← b
enquanto i > p + 1
      trocar(v, i, i - 1)
      i ← i - 1
retorne v
```

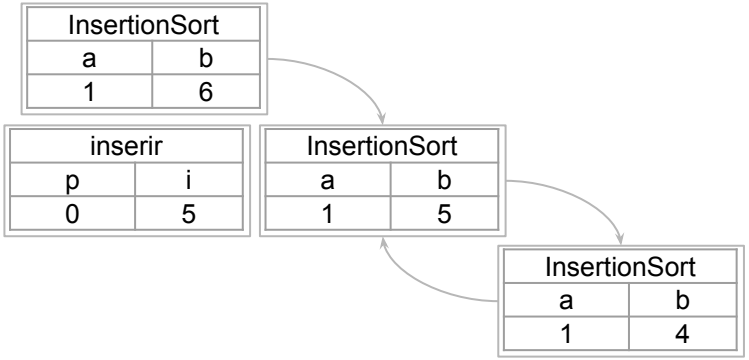
i	**1**	**2**	**3**	**4**	5	6
v[i]	**8**	**15**	**23**	**42**	4	16




```
função insertionSort (v,a,b)
se a ≥ b
      retorne v
insertionSort(v,a,b-1)
inserir(v,a,b)
retorne v
```

```
função inserir (v,a,b)
p ← buscar(v[b],v,a,b-1)
i ← b
enquanto i > p + 1
      trocar(v, i, i - 1)
      i ← i - 1
retorne v
```

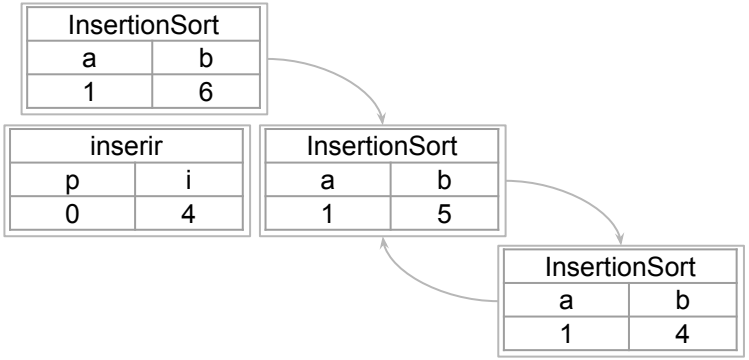
i	**1**	**2**	**3**	**4**	5	6
v[i]	**8**	**15**	**23**	**42**	4	16




```
função insertionSort (v,a,b)
se a ≥ b
      retorne v
insertionSort(v,a,b-1)
inserir(v,a,b)
retorne v
```

```
função inserir (v,a,b)
p ← buscar(v[b],v,a,b-1)
i ← b
enquanto i > p + 1
      trocar(v, i, i - 1)
      i ← i - 1
retorne v
```

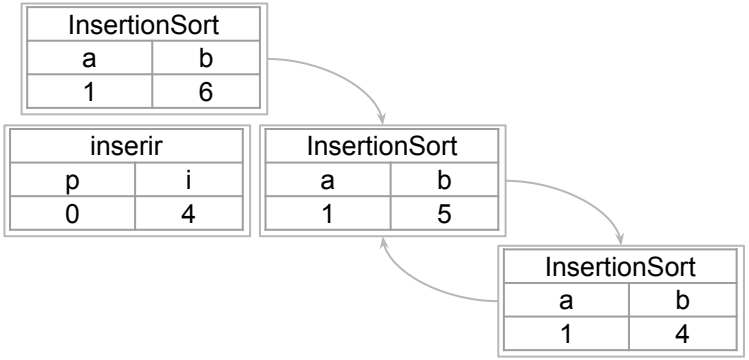
i	**1**	**2**	**3**	4	**5**	6
v[i]	**8**	**15**	**23**	4	**42**	16




```
função insertionSort (v,a,b)
se a ≥ b
      retorne v
insertionSort(v,a,b-1)
inserir(v,a,b)
retorne v
```

```
função inserir (v,a,b)
p ← buscar(v[b],v,a,b-1)
i ← b
enquanto i > p + 1
      trocar(v, i, i - 1)
      i ← i - 1
retorne v
```

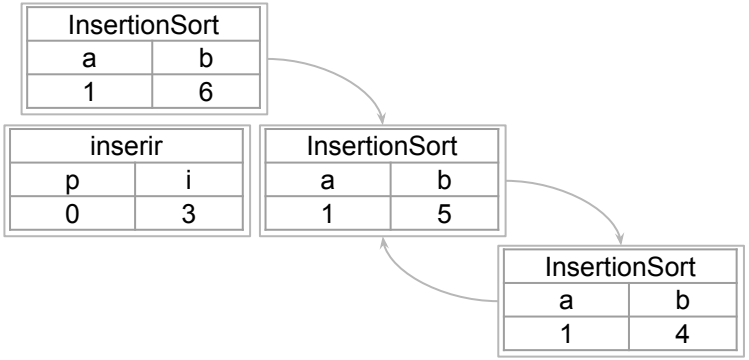
i	**1**	**2**	**3**	4	**5**	6
v[i]	**8**	**15**	**23**	4	**42**	16




```
função insertionSort (v,a,b)
se a ≥ b
      retorne v
insertionSort(v,a,b-1)
inserir(v,a,b)
retorne v
```

```
função inserir (v,a,b)
p ← buscar(v[b],v,a,b-1)
i ← b
enquanto i > p + 1
      trocar(v, i, i - 1)
      i ← i - 1
retorne v
```

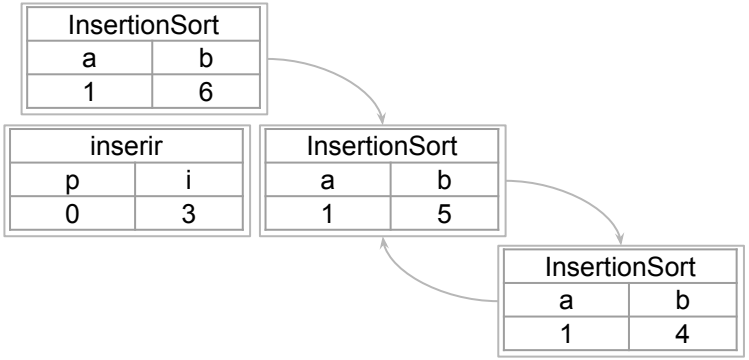
i	**1**	**2**	3	**4**	**5**	6
v[i]	**8**	**15**	4	**23**	**42**	16




```
função insertionSort (v,a,b)
se a ≥ b
      retorne v
insertionSort(v,a,b-1)
inserir(v,a,b)
retorne v
```

```
função inserir (v,a,b)
p ← buscar(v[b],v,a,b-1)
i ← b
enquanto i > p + 1
      trocar(v, i, i - 1)
      i ← i - 1
retorne v
```

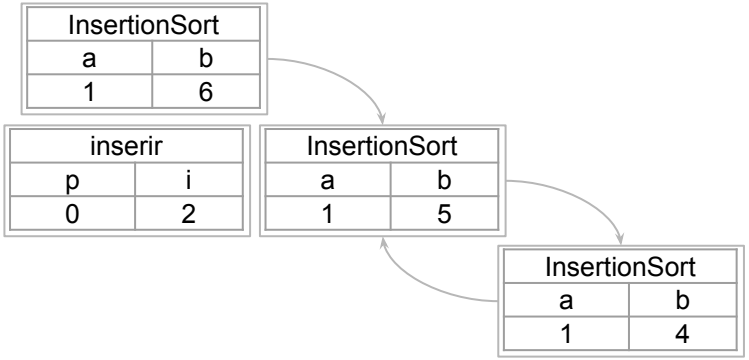
i	**1**	**2**	3	**4**	**5**	6
v[i]	**8**	**15**	4	**23**	**42**	16




```
função insertionSort (v,a,b)
se a ≥ b
      retorne v
insertionSort(v,a,b-1)
inserir(v,a,b)
retorne v
```

```
função inserir (v,a,b)
p ← buscar(v[b],v,a,b-1)
i ← b
enquanto i > p + 1
      trocar(v, i, i - 1)
      i ← i - 1
retorne v
```

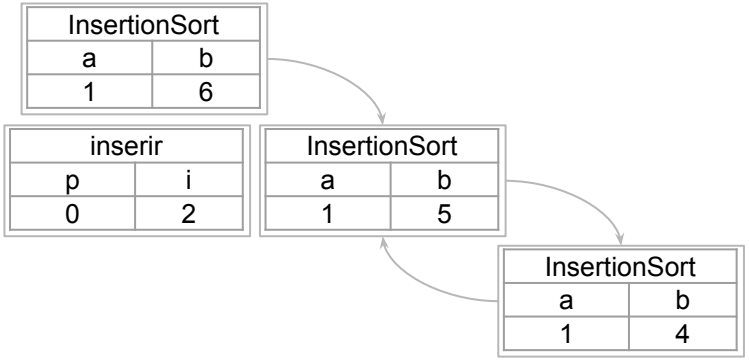
i	**1**	2	**3**	**4**	**5**	6
v[i]	**8**	4	**15**	**23**	**42**	16




```
função insertionSort (v,a,b)
se a ≥ b
      retorne v
insertionSort(v,a,b-1)
inserir(v,a,b)
retorne v
```

```
função inserir (v,a,b)
p ← buscar(v[b],v,a,b-1)
i ← b
enquanto i > p + 1
      trocar(v, i, i - 1)
      i ← i - 1
retorne v
```

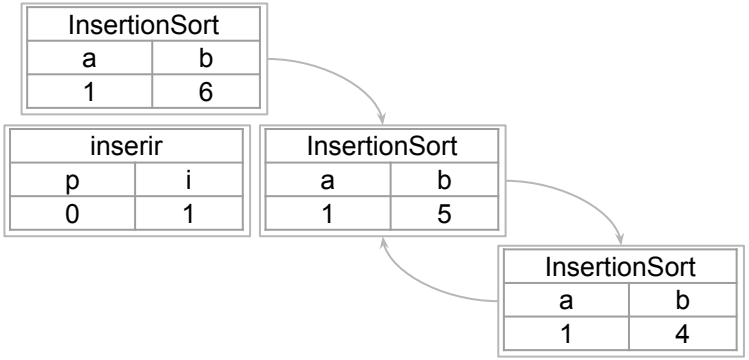
i	**1**	2	**3**	**4**	**5**	6
v[i]	**8**	4	**15**	**23**	**42**	16




```
função insertionSort (v,a,b)
se a ≥ b
      retorne v
insertionSort(v,a,b-1)
inserir(v,a,b)
retorne v
```

```
função inserir (v,a,b)
p ← buscar(v[b],v,a,b-1)
i ← b
enquanto i > p + 1
      trocar(v, i, i - 1)
      i ← i - 1
retorne v
```

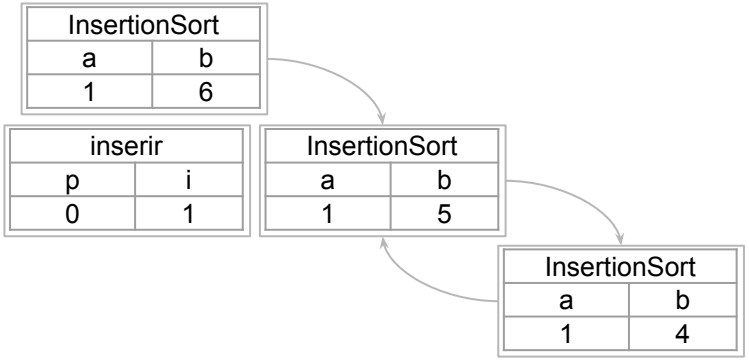
i	1	**2**	**3**	**4**	**5**	6
v[i]	4	**8**	**15**	**23**	**42**	16




```
função insertionSort (v,a,b)
se a ≥ b
      retorne v
insertionSort(v,a,b-1)
inserir(v,a,b)
retorne v
```

```
função inserir (v,a,b)
p ← buscar(v[b],v,a,b-1)
i ← b
enquanto i > p + 1
      trocar(v, i, i - 1)
      i ← i - 1
retorne v
```

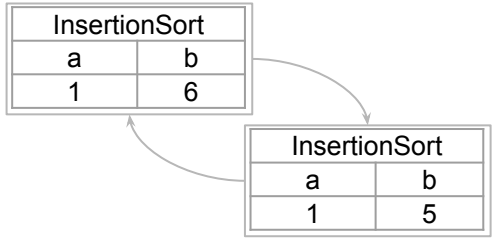
i	1	**2**	**3**	**4**	**5**	6
v[i]	4	**8**	**15**	**23**	**42**	16




```
função insertionSort (v,a,b)
se a ≥ b
      retorne v
insertionSort(v,a,b-1)
inserir(v,a,b)
retorne v
```

```
função inserir (v,a,b)
p ← buscar(v[b],v,a,b-1)
i ← b
enquanto i > p + 1
      trocar(v, i, i - 1)
      i ← i - 1
retorne v
```

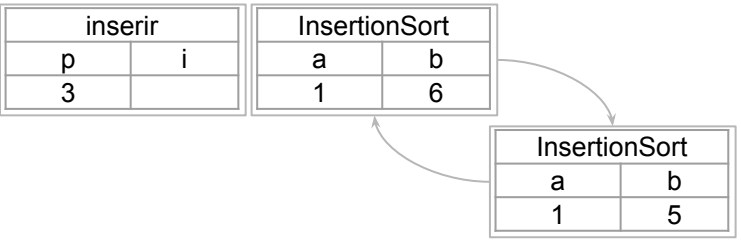
i	**1**	**2**	**3**	**4**	**5**	6
v[i]	**4**	**8**	**15**	**23**	**42**	16




```
função insertionSort (v,a,b)
se a ≥ b
      retorne v
insertionSort(v,a,b-1)
inserir(v,a,b)
retorne v
```

```
função inserir (v,a,b)
p ← buscar(v[b],v,a,b-1)
i ← b
enquanto i > p + 1
      trocar(v, i, i - 1)
      i ← i - 1
retorne v
```

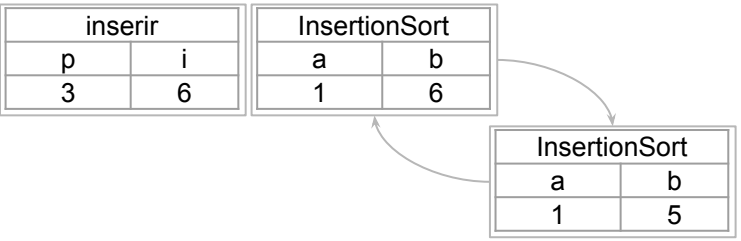
i	**1**	**2**	**3**	**4**	**5**	6
v[i]	**4**	**8**	**15**	**23**	**42**	16




```
função insertionSort (v,a,b)
se a ≥ b
      retorne v
insertionSort(v,a,b-1)
inserir(v,a,b)
retorne v
```

```
função inserir (v,a,b)
p ← buscar(v[b],v,a,b-1)
i ← b
enquanto i > p + 1
      trocar(v, i, i - 1)
      i ← i - 1
retorne v
```

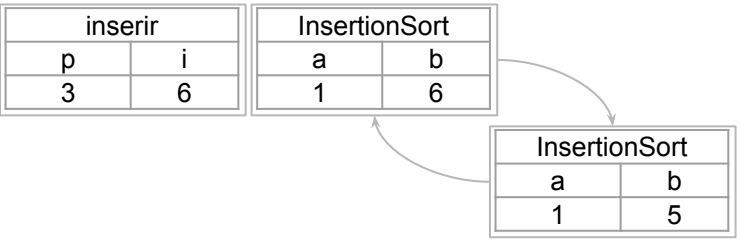
i	1	2	3	4	5	6
v[i]	4	8	15	23	42	16




```
função insertionSort (v,a,b)
se a ≥ b
      retorne v
insertionSort(v,a,b-1)
inserir(v,a,b)
retorne v
```

```
função inserir (v,a,b)
p ← buscar(v[b],v,a,b-1)
i ← b
enquanto i > p + 1
      trocar(v, i, i - 1)
      i ← i - 1
retorne v
```

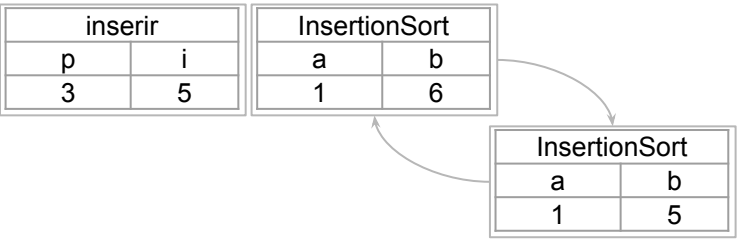
i	1	2	3	4	5	6
v[i]	4	8	15	23	42	16




```
função insertionSort (v,a,b)
se a ≥ b
      retorne v
insertionSort(v,a,b-1)
inserir(v,a,b)
retorne v
```

```
função inserir (v,a,b)
p ← buscar(v[b],v,a,b-1)
i ← b
enquanto i > p + 1
      trocar(v, i, i - 1)
      i ← i - 1
retorne v
```

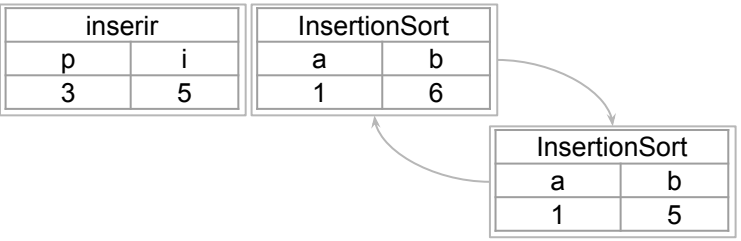
i	1	2	3	4	5	6
v[i]	4	8	15	23	16	42




```
função insertionSort (v,a,b)
se a ≥ b
      retorne v
insertionSort(v,a,b-1)
inserir(v,a,b)
retorne v
```

```
função inserir (v,a,b)
p ← buscar(v[b],v,a,b-1)
i ← b
enquanto i > p + 1
      trocar(v, i, i - 1)
      i ← i - 1
retorne v
```

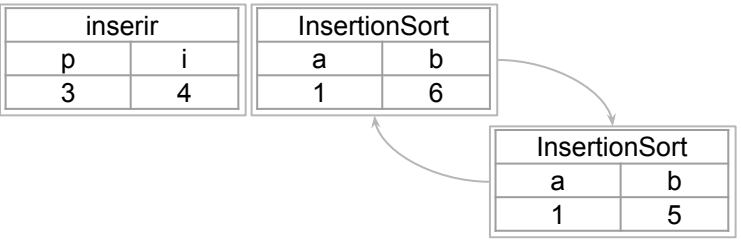
i	1	2	3	4	5	6
v[i]	4	8	15	23	16	42




```
função insertionSort (v,a,b)
se a ≥ b
      retorne v
insertionSort(v,a,b-1)
inserir(v,a,b)
retorne v
```

```
função inserir (v,a,b)
p ← buscar(v[b],v,a,b-1)
i ← b
enquanto i > p + 1
      trocar(v, i, i - 1)
      i ← i - 1
retorne v
```

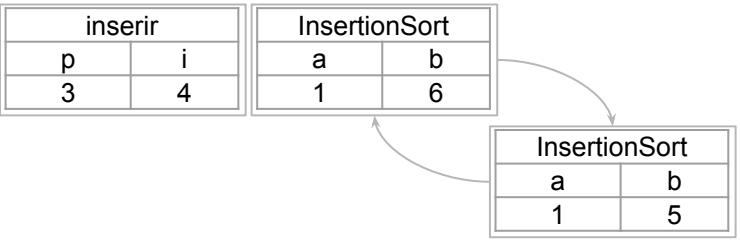
i	1	2	3	4	5	6
v[i]	4	8	15	16	23	42




```
função insertionSort (v,a,b)
se a ≥ b
      retorne v
insertionSort(v,a,b-1)
inserir(v,a,b)
retorne v
```

```
função inserir (v,a,b)
p ← buscar(v[b],v,a,b-1)
i ← b
enquanto i > p + 1
      trocar(v, i, i - 1)
      i ← i - 1
retorne v
```

i	1	2	3	4	5	6
v[i]	4	8	15	16	23	42




```
função insertionSort (v,a,b)
se a ≥ b
      retorne v
insertionSort(v,a,b-1)
inserir(v,a,b)
retorne v
```

```
função inserir (v,a,b)
p ← buscar(v[b],v,a,b-1)
i ← b
enquanto i > p + 1
      trocar(v, i, i - 1)
      i ← i - 1
retorne v
```

i	1	2	3	4	5	6
v[i]	4	8	15	16	23	42

InsertionSort	
a	b
1	6

```
função insertionSort (v,a,b)
se a ≥ b
      retorne v
insertionSort(v,a,b-1)
inserir(v,a,b)
retorne v
```

```
função inserir (v,a,b)
p ← buscar(v[b],v,a,b-1)
i ← b
enquanto i > p + 1
      trocar(v, i, i - 1)
      i ← i - 1
retorne v
```

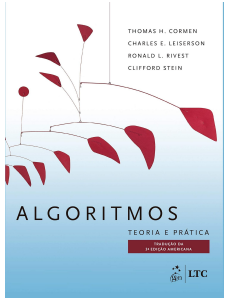
i	1	2	3	4	5	6
v[i]	4	8	15	16	23	42

Exercícios

1. Implemente o insertion sort recursivo visto em sala em C
2. Implemente uma versão iterativa do insertion sort

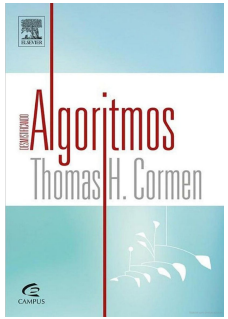
Referências

T. Cormen, C. Leiserson, R. Rivest, C. Stein. Algoritmos: Teoria e Prática. 3a ed. 2012

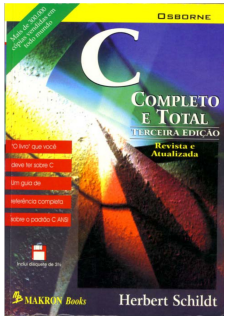


R. Sedgewick, K. Wayne. Algorithms Part I. 4a ed. 2014

T. Cormen. Desmistificando algoritmos. 2017.



H. Schildt. C completo e total. 1996



Licença