> “A matemática requer uma dose pequena, não de genialidade, mas de liberdade de imaginação, que em uma dose maior, seria loucura” (Angus K. Rodgers).

Eficiência e Recorrências

Paulo Ricardo Lisboa de Almeida

DSBD

UFPR
UNIVERSIDADE FEDERAL DO PARANÁ

Eficiência

Na Ciência da Computação buscamos algoritmos **eficientes** para resolver os problemas.

Vamos estabelecer eficiência como: recursos consumidos/quantidade de tarefa realizada.

Mas que recursos são esses que desejamos gastar eficientemente?

Exemplos de Recursos

Joules;

Tempo;

Memória;

Transferência de dados (comunicação);

...

Eficiência

Eficiência de algoritmos e análise de recursão via recorrência são temas estudados profundamente em **Matemática Discreta** e em **Análise de Algoritmos.**

Mas vamos começar a olhar para esses temas para entender melhor nossos algoritmos.

Muitas das provas apresentadas serão **informais.**

Servirão como uma introdução ao tema.

Valor Mínimo

Para encontrar o valor mínimo de um vetor, precisamos realizar comparações.

A questão é, **quantas comparações entre elementos são necessárias?**

Vamos definir a quantidade de comparações como o recurso sendo medido.

```
função minimoVetor (v,a,b)
entrada: vetor v indexado por [a..b], com a ≤ b
saída: m ∈ [a..b], tal que v[m] ≤ v[j] ∀j ∈ [a..b]
j ← a
m ← a
enquanto j < b
    j ← j + 1
    se v[j] < v[m]
       m ← j
retorne m
```

Obs.: essa é a versão **iterativa**

Valor Mínimo

Para um vetor v de tamanho $|v| = n = b-a+1$.

O loop inicia as comparações em $a+1$, e termina em b.

São realizadas então $b - (a + 1) + 1 = b-a$ comparações entre elementos.

```
função minimoVetor (v,a,b)
entrada: vetor v indexado por [a..b], com a ≤ b
saída: m ∈ [a..b], tal que v[m] ≤ v[j] ∀j ∈ [a..b]
j ← a
m ← a
enquanto j < b
    j ← j + 1
    se v[j] < v[m]
       m ← j
retorne m
```

Valor Mínimo

Para um vetor v de tamanho $|v| = n = b-a+1$.

O loop inicia as comparações em $a+1$, e termina em b.

São realizadas então $b - (a + 1) + 1 = b-a$ comparações entre elementos.

Termina em b

Inicia em $a+1$

```
função minimoVetor (v,a,b)
entrada: vetor v indexado por [a..b], com a ≤ b
saída: m ∈ [a..b], tal que v[m] ≤ v[j] ∀j ∈ [a..b]
j ← a
m ← a
enquanto j < b
    j ← j + 1
    se v[j] < v[m]
       m ← j
retorne m
```

Versão Recursiva

Vamos verificar a versão recursiva do algoritmo.

Comece com um teste de mesa para v = [28, 12, 1, 25, 10, 1].

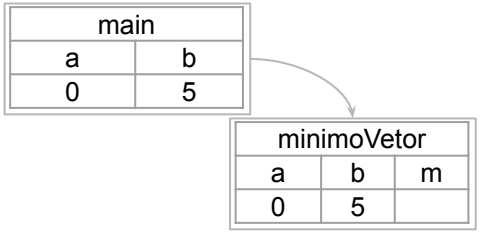
```
função main ()
     v ← [28, 12, 1, 25, 10, 1]
     m ← minimoVetor(v,0,5)
     imprima(m, v[m])


função minimoVetor (v,a,b)
entrada: vetor v indexado por [a..b], com a ≤ b
saída: m ∈ [a..b], tal que v[m] ≤ v[j] ∀j ∈ [a..b]
se a = b
    retorne a
m ← minimoVetor(v,a,b-1)
Se v[b] < v[m]
     m ← b
retorne m
```

main	
a	b
0	5

```
função main ()
     v ← [28, 12, 1, 25, 10, 1]
     m ← minimoVetor(v,0,5)
     imprima(m, v[m])
```

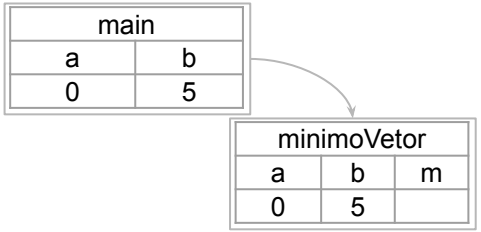
```
função minimoVetor (v,a,b)
entrada: vetor v indexado por [a..b], com a ≤ b
saída: m ∈ [a..b], tal que v[m] ≤ v[j] ∀j ∈ [a..b]
se a = b
    retorne a
m ← minimoVetor(v,a,b-1)
Se v[b] < v[m]
     m ← b
retorne m
```

i	0	1	2	3	4	5
v[i]	28	12	1	25	10	1




```
função main ()
     v ← [28, 12, 1, 25, 10, 1]
     m ← minimoVetor(v,0,5)
     imprima(m, v[m])


função minimoVetor (v,a,b)
entrada: vetor v indexado por [a..b], com a ≤ b
saída: m ∈ [a..b], tal que v[m] ≤ v[j] ∀j ∈ [a..b]
se a = b
    retorne a
m ← minimoVetor(v,a,b-1)
Se v[b] < v[m]
     m ← b
retorne m
```

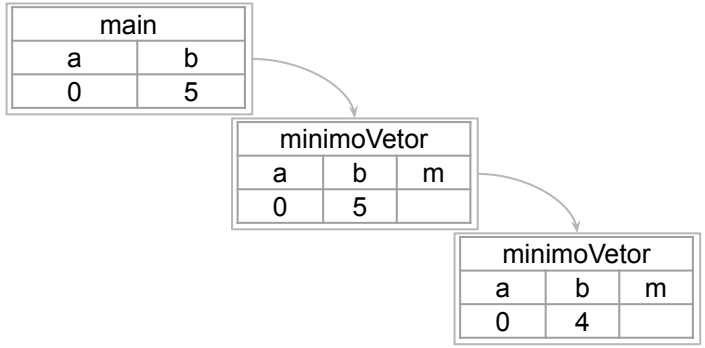
i	**0**	**1**	**2**	**3**	**4**	**5**
v[i]	28	12	1	25	10	1




```
função main ()
     v ← [28, 12, 1, 25, 10, 1]
     m ← minimoVetor(v,0,5)
     imprima(m, v[m])
```

```
função minimoVetor (v,a,b)
entrada: vetor v indexado por [a..b], com a ≤ b
saída: m ∈ [a..b], tal que v[m] ≤ v[j] ∀j ∈ [a..b]
se a = b
    retorne a
m ← minimoVetor(v,a,b-1)
Se v[b] < v[m]
     m ← b
retorne m
```

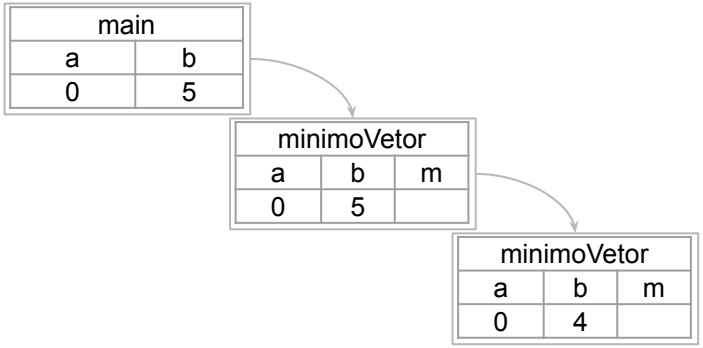
i	**0**	**1**	**2**	**3**	**4**	**5**
v[i]	28	12	1	25	10	1




```
função main ()
     v ← [28, 12, 1, 25, 10, 1]
     m ← minimoVetor(v,0,5)
     imprima(m, v[m])
```

```
função minimoVetor (v,a,b)
entrada: vetor v indexado por [a..b], com a ≤ b
saída: m ∈ [a..b], tal que v[m] ≤ v[j] ∀j ∈ [a..b]
se a = b
    retorne a
m ← minimoVetor(v,a,b-1)
Se v[b] < v[m]
     m ← b
retorne m
```

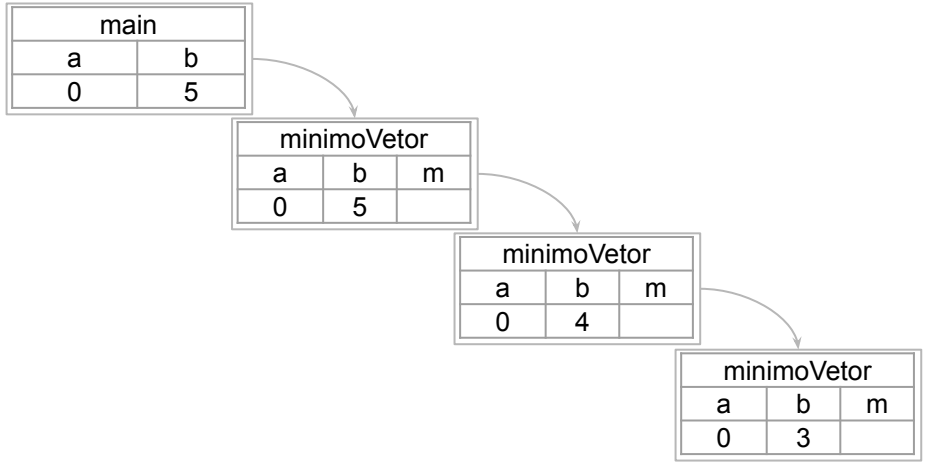
i	**0**	**1**	**2**	**3**	**4**	5
v[i]	28	12	1	25	10	1




```
função main ()
     v ← [28, 12, 1, 25, 10, 1]
     m ← minimoVetor(v,0,5)
     imprima(m, v[m])
```

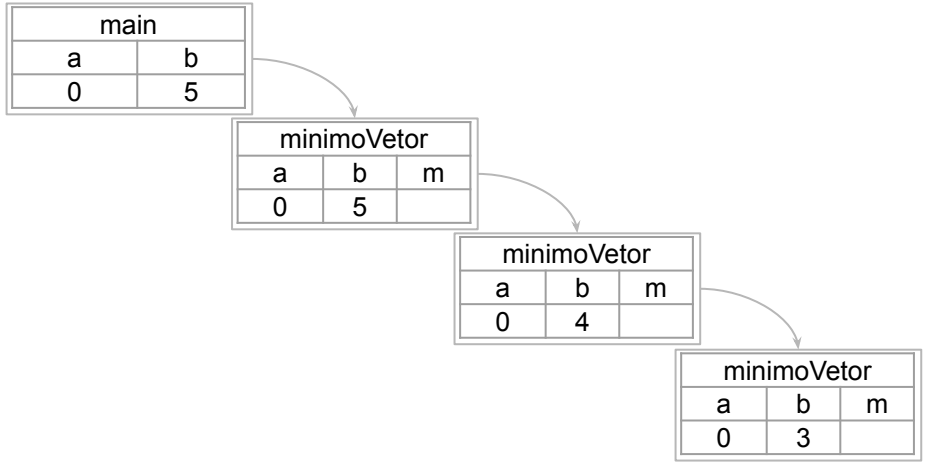
```
função minimoVetor (v,a,b)
entrada: vetor v indexado por [a..b], com a ≤ b
saída: m ∈ [a..b], tal que v[m] ≤ v[j] ∀j ∈ [a..b]
se a = b
    retorne a
m ← minimoVetor(v,a,b-1)
Se v[b] < v[m]
     m ← b
retorne m
```

i	**0**	**1**	**2**	**3**	**4**	5
v[i]	28	12	1	25	10	1



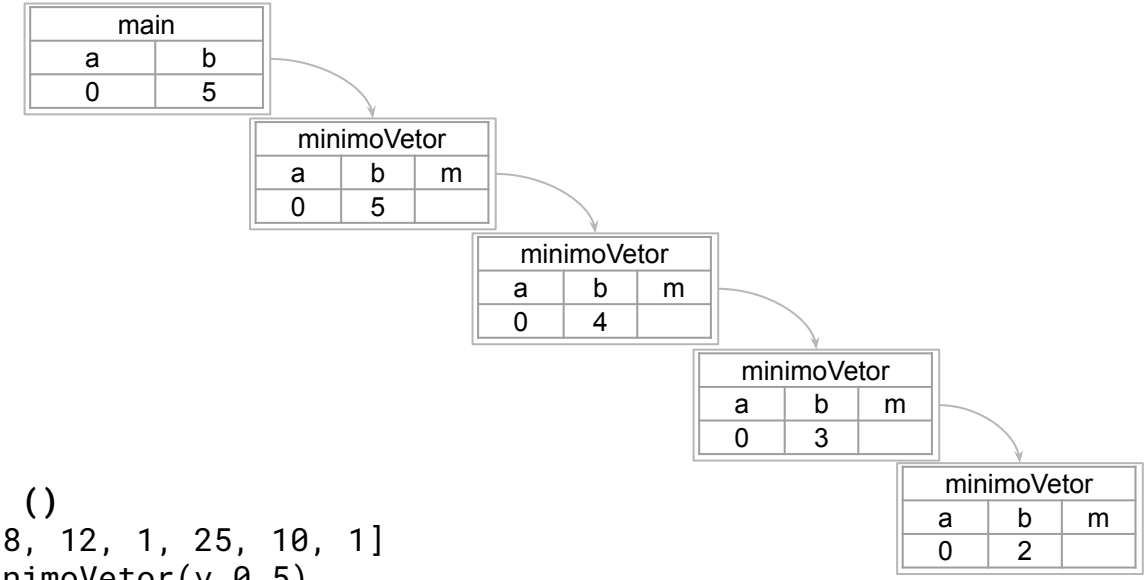

```
função main ()
     v ← [28, 12, 1, 25, 10, 1]
     m ← minimoVetor(v,0,5)
     imprima(m, v[m])


função minimoVetor (v,a,b)
entrada: vetor v indexado por [a..b], com a ≤ b
saída: m ∈ [a..b], tal que v[m] ≤ v[j] ∀j ∈ [a..b]
se a = b
    retorne a
m ← minimoVetor(v,a,b-1)
Se v[b] < v[m]
     m ← b
retorne m
```

i	**0**	**1**	**2**	**3**	4	5
v[i]	28	12	1	25	10	1




```
função main ()
     v ← [28, 12, 1, 25, 10, 1]
     m ← minimoVetor(v,0,5)
     imprima(m, v[m])


função minimoVetor (v,a,b)
entrada: vetor v indexado por [a..b], com a ≤ b
saída: m ∈ [a..b], tal que v[m] ≤ v[j] ∀j ∈ [a..b]
se a = b
    retorne a
m ← minimoVetor(v,a,b-1)
Se v[b] < v[m]
     m ← b
retorne m
```

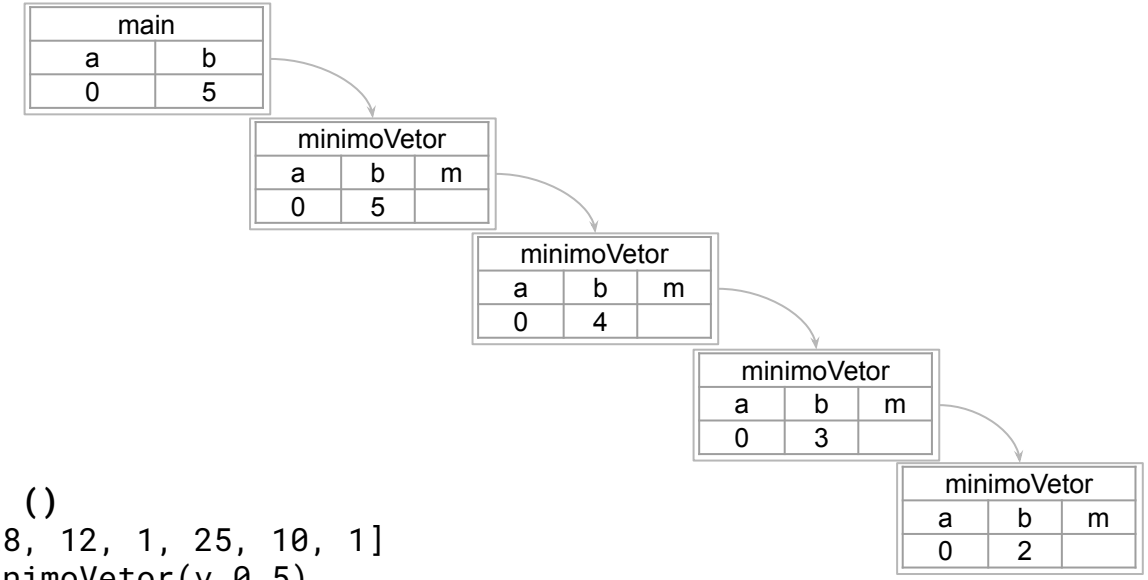
i	**0**	**1**	**2**	**3**	4	5
v[i]	28	12	1	25	10	1




```
função main ()
     v ← [28, 12, 1, 25, 10, 1]
     m ← minimoVetor(v,0,5)
     imprima(m, v[m])
```

```
função minimoVetor (v,a,b)
entrada: vetor v indexado por [a..b], com a ≤ b
saída: m ∈ [a..b], tal que v[m] ≤ v[j] ∀j ∈ [a..b]
se a = b
    retorne a
m ← minimoVetor(v,a,b-1)
Se v[b] < v[m]
     m ← b
retorne m
```

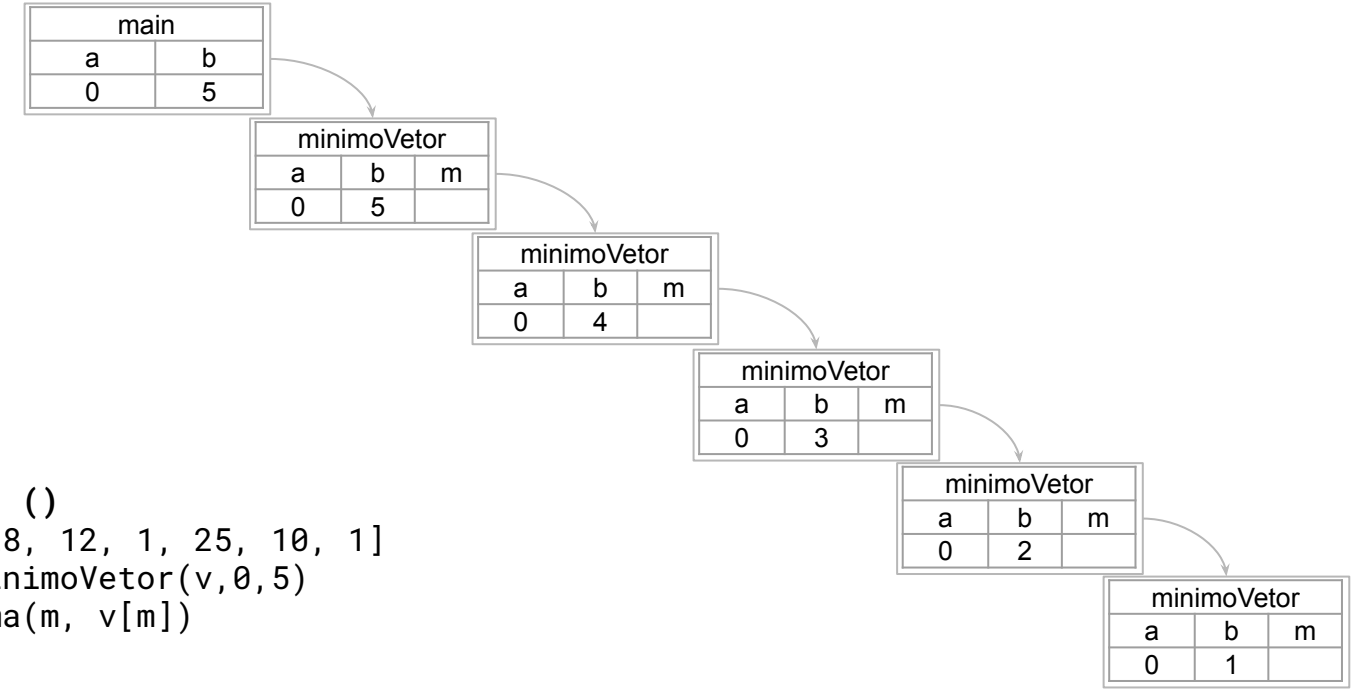
i	**0**	**1**	**2**	3	4	5
v[i]	28	12	1	25	10	1




```
função main ()
     v ← [28, 12, 1, 25, 10, 1]
     m ← minimoVetor(v,0,5)
     imprima(m, v[m])
```

```
função minimoVetor (v,a,b)
entrada: vetor v indexado por [a..b], com a ≤ b
saída: m ∈ [a..b], tal que v[m] ≤ v[j] ∀j ∈ [a..b]
se a = b
    retorne a
m ← minimoVetor(v,a,b-1)
Se v[b] < v[m]
     m ← b
retorne m
```

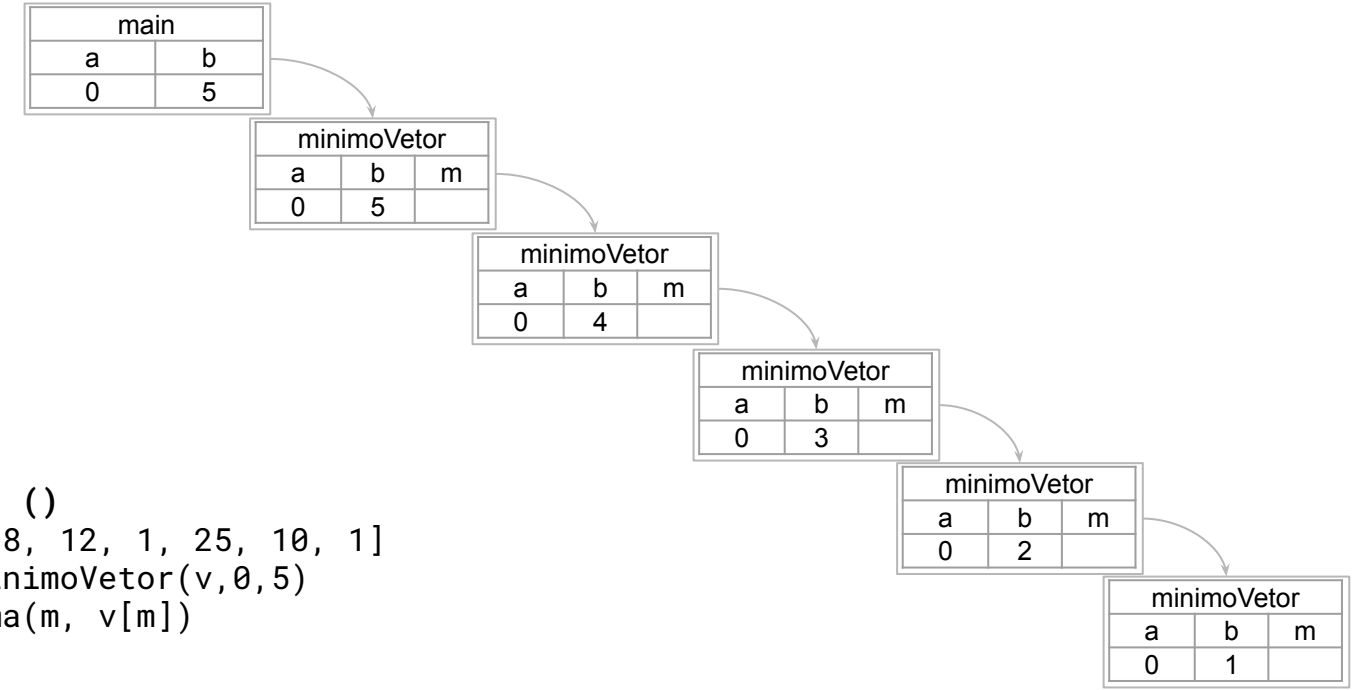
i	**0**	**1**	**2**	3	4	5
v[i]	28	12	1	25	10	1




```
função main ()
     v ← [28, 12, 1, 25, 10, 1]
     m ← minimoVetor(v,0,5)
     imprima(m, v[m])
```

```
função minimoVetor (v,a,b)
entrada: vetor v indexado por [a..b], com a ≤ b
saída: m ∈ [a..b], tal que v[m] ≤ v[j] ∀j ∈ [a..b]
se a = b
    retorne a
m ← minimoVetor(v,a,b-1)
Se v[b] < v[m]
     m ← b
retorne m
```

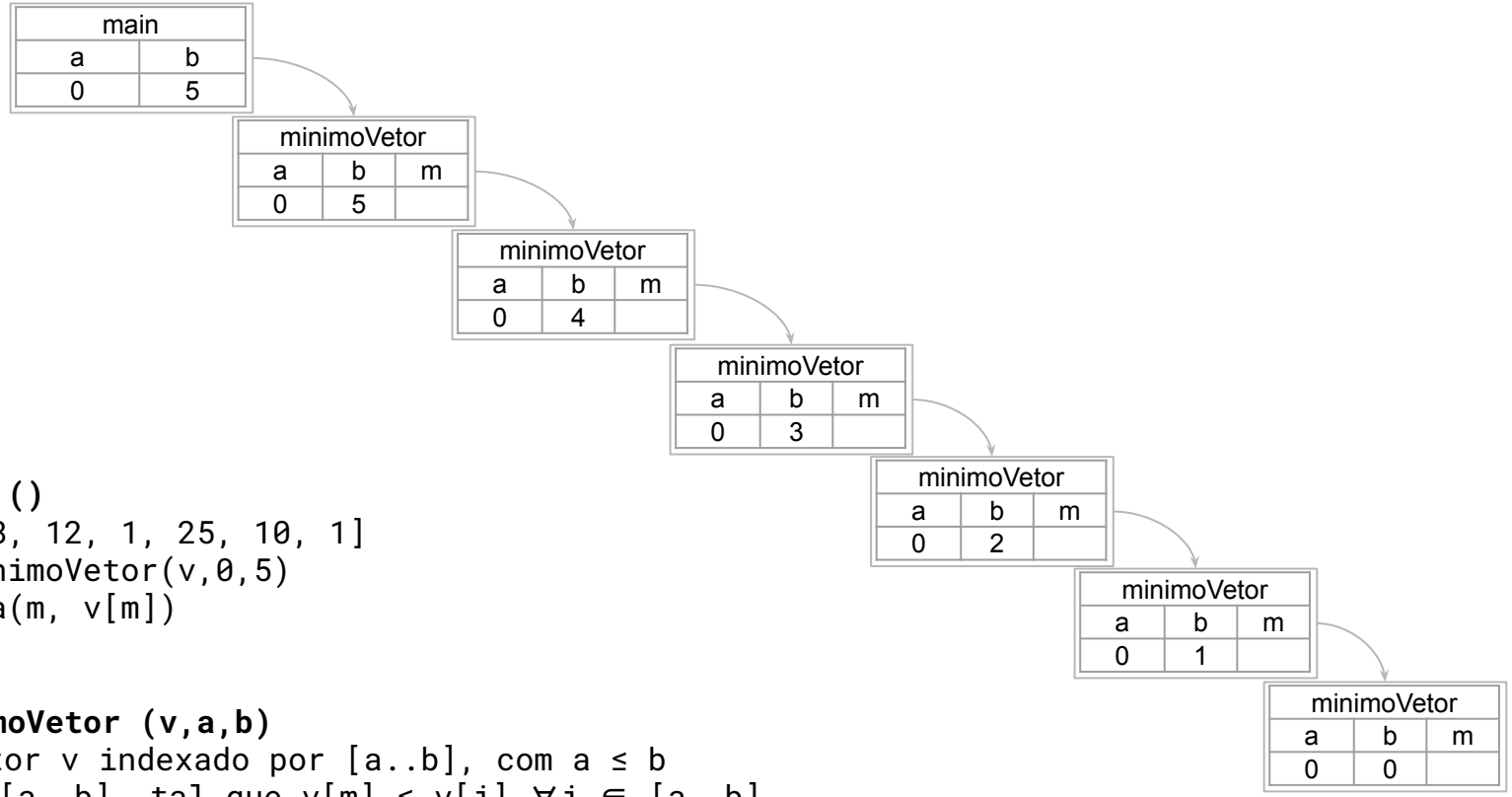
i	**0**	**1**	2	3	4	5
v[i]	28	12	1	25	10	1




```
função main ()
     v ← [28, 12, 1, 25, 10, 1]
     m ← minimoVetor(v,0,5)
     imprima(m, v[m])
```

```
função minimoVetor (v,a,b)
entrada: vetor v indexado por [a..b], com a ≤ b
saída: m ∈ [a..b], tal que v[m] ≤ v[j] ∀j ∈ [a..b]
se a = b
    retorne a
m ← minimoVetor(v,a,b-1)
Se v[b] < v[m]
     m ← b
retorne m
```

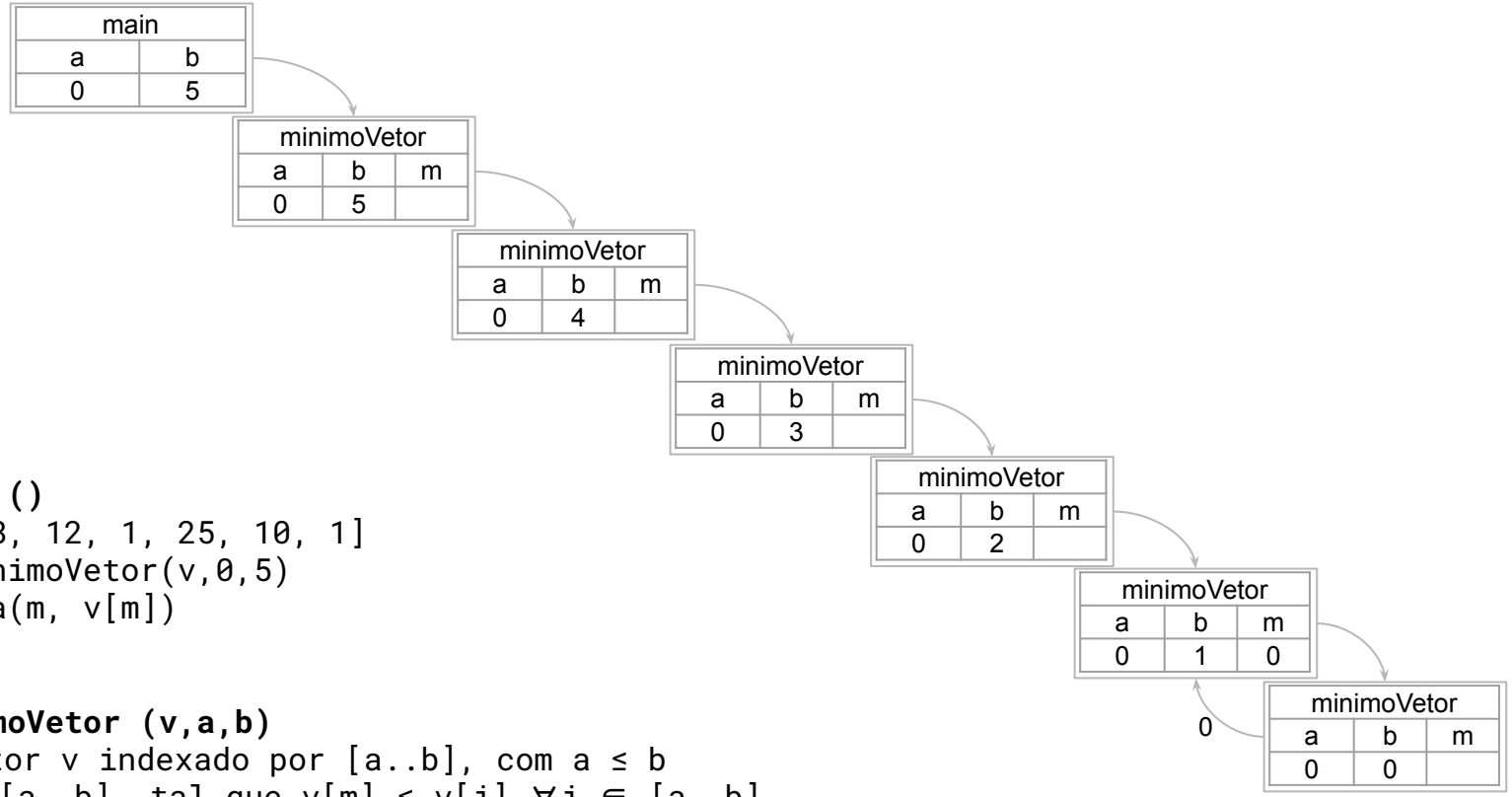
i	**0**	**1**	2	3	4	5
v[i]	28	12	1	25	10	1




```
função main ()
     v ← [28, 12, 1, 25, 10, 1]
     m ← minimoVetor(v,0,5)
     imprima(m, v[m])
```

```
função minimoVetor (v,a,b)
entrada: vetor v indexado por [a..b], com a ≤ b
saída: m ∈ [a..b], tal que v[m] ≤ v[j] ∀j ∈ [a..b]
se a = b
    retorne a
m ← minimoVetor(v,a,b-1)
Se v[b] < v[m]
     m ← b
retorne m
```

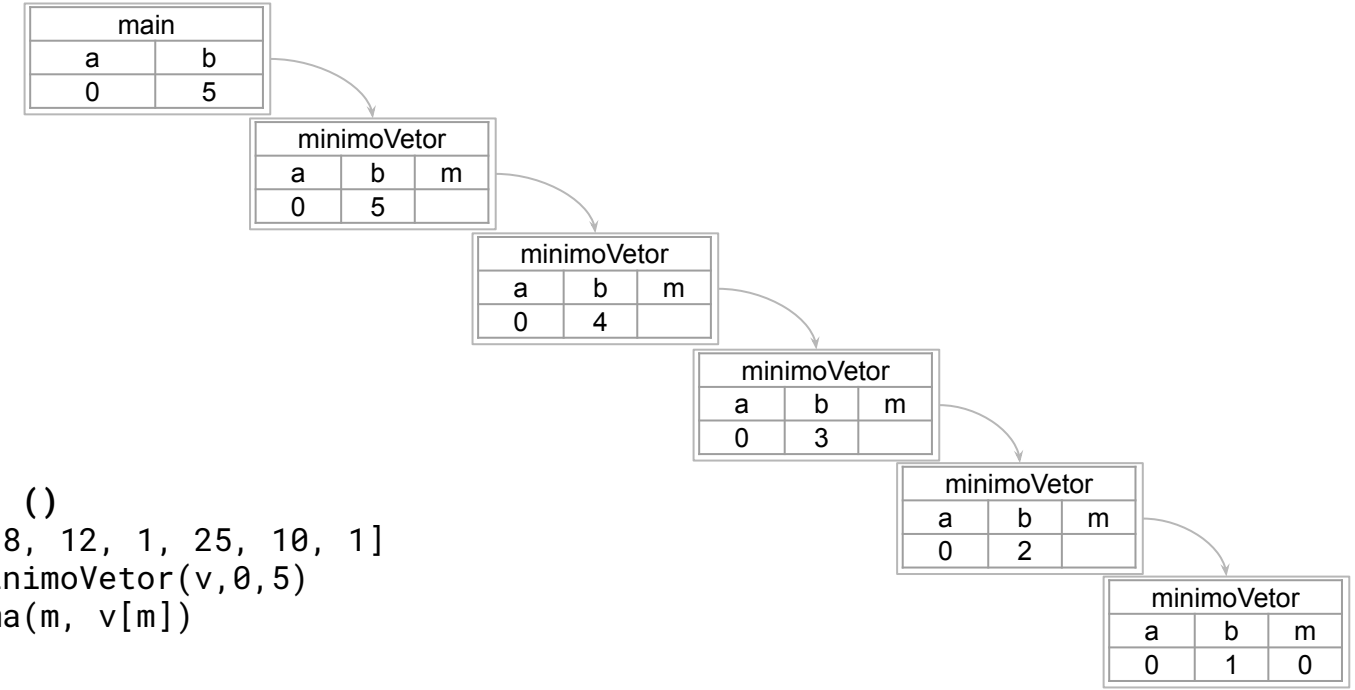
i	**0**	1	2	3	4	5
v[i]	28	12	1	25	10	1




```
função main ()
     v ← [28, 12, 1, 25, 10, 1]
     m ← minimoVetor(v,0,5)
     imprima(m, v[m])
```

```
função minimoVetor (v,a,b)
entrada: vetor v indexado por [a..b], com a ≤ b
saída: m ∈ [a..b], tal que v[m] ≤ v[j] ∀j ∈ [a..b]
se a = b
    retorne a
m ← minimoVetor(v,a,b-1)
Se v[b] < v[m]
     m ← b
retorne m
```

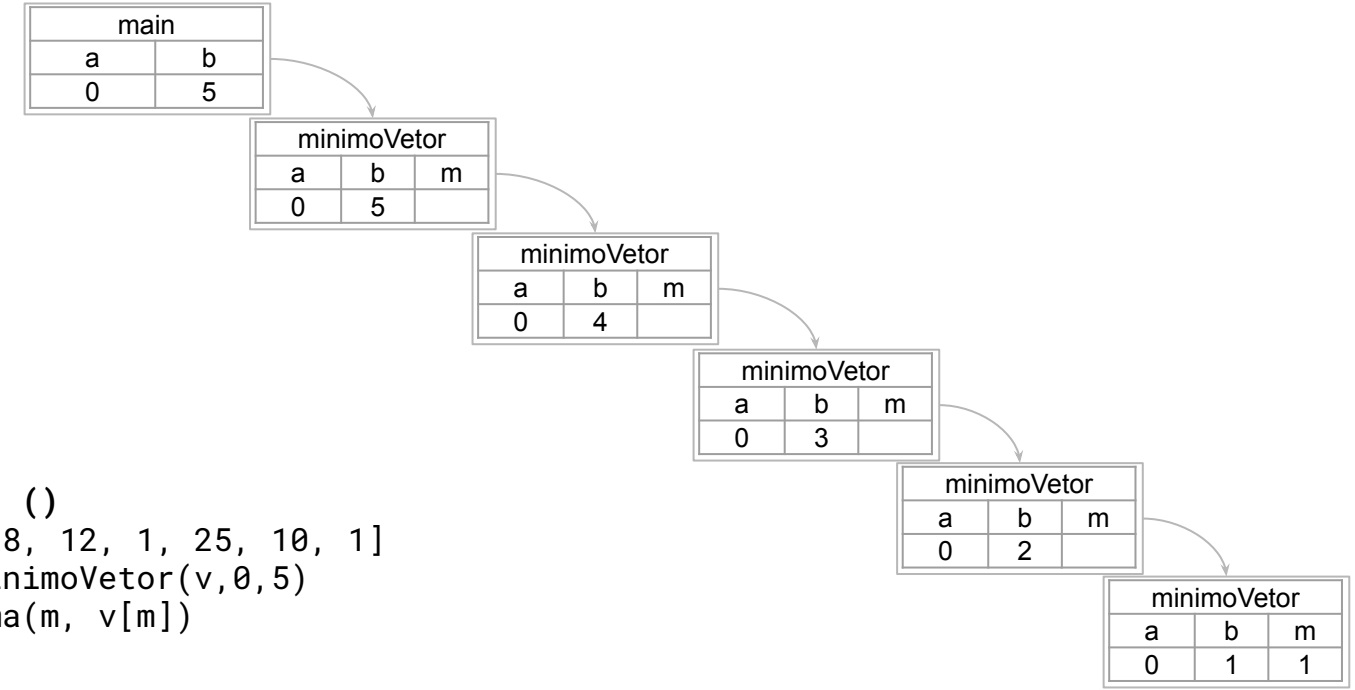
i	**0**	1	2	3	4	5
v[i]	28	12	1	25	10	1




```
função main ()
     v ← [28, 12, 1, 25, 10, 1]
     m ← minimoVetor(v,0,5)
     imprima(m, v[m])
```

```
função minimoVetor (v,a,b)
entrada: vetor v indexado por [a..b], com a ≤ b
saída: m ∈ [a..b], tal que v[m] ≤ v[j] ∀j ∈ [a..b]
se a = b
    retorne a
m ← minimoVetor(v,a,b-1)
Se v[b] < v[m]
     m ← b
retorne m
```

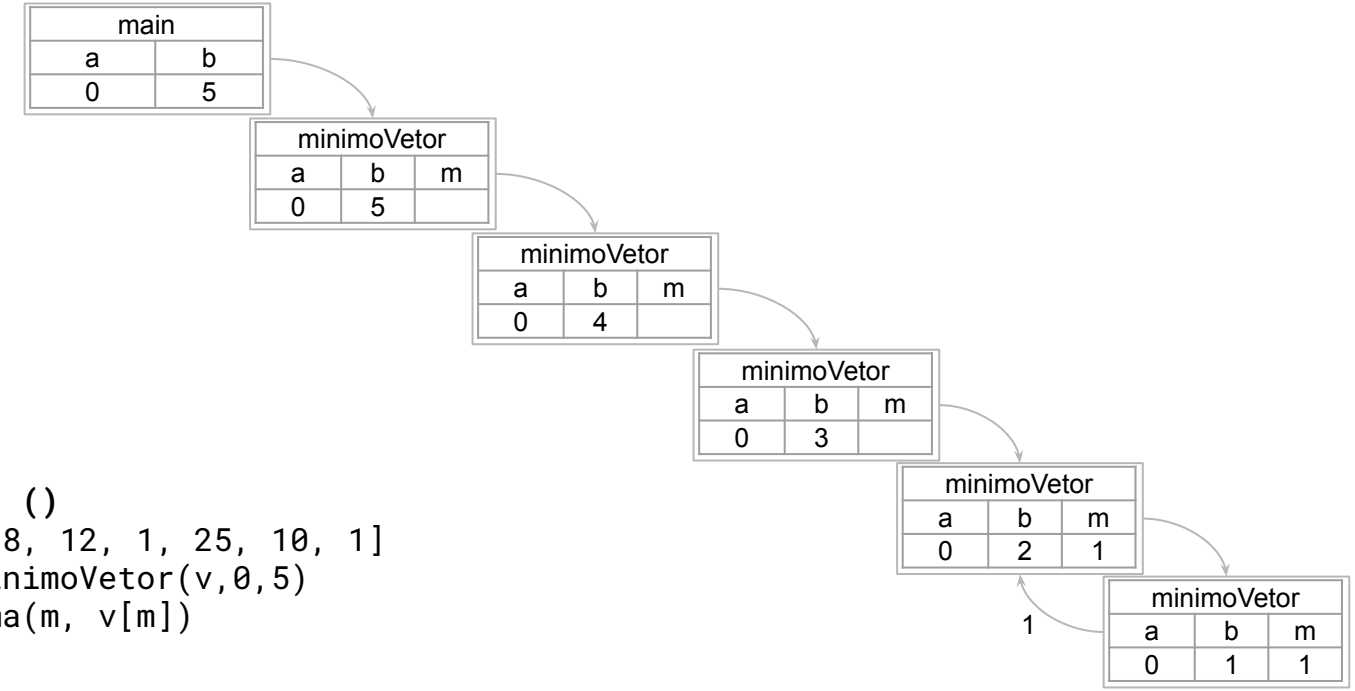
i	**0**	**1**	2	3	4	5
v[i]	28	12	1	25	10	1




```
função main ()
     v ← [28, 12, 1, 25, 10, 1]
     m ← minimoVetor(v,0,5)
     imprima(m, v[m])
```

```
função minimoVetor (v,a,b)
entrada: vetor v indexado por [a..b], com a ≤ b
saída: m ∈ [a..b], tal que v[m] ≤ v[j] ∀j ∈ [a..b]
se a = b
    retorne a
m ← minimoVetor(v,a,b-1)
Se v[b] < v[m]
     m ← b
retorne m
```

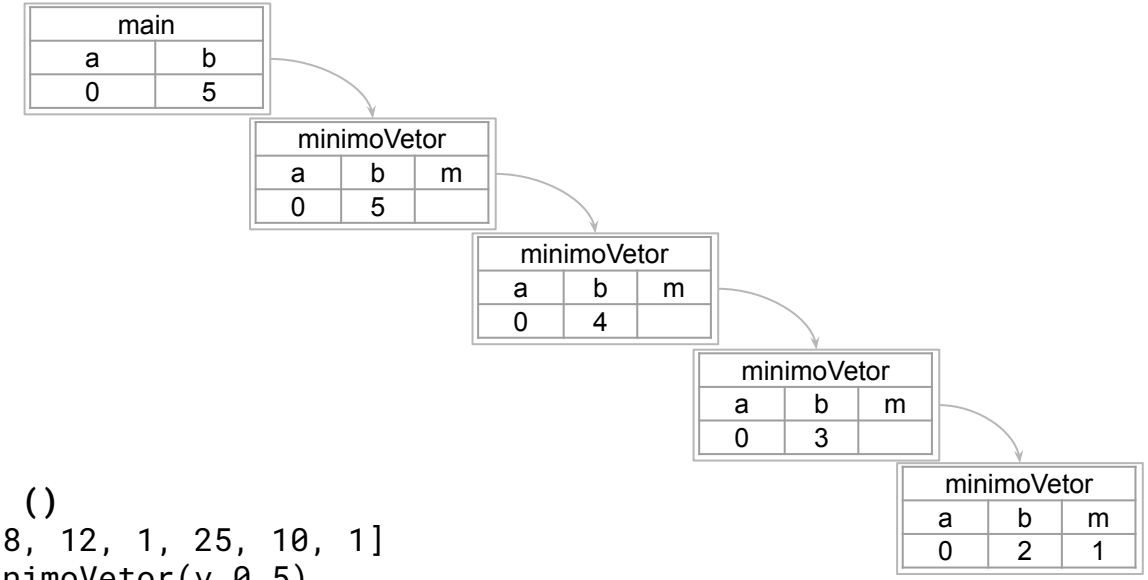
i	**0**	**1**	2	3	4	5
v[i]	28	12	1	25	10	1




```
função main ()
     v ← [28, 12, 1, 25, 10, 1]
     m ← minimoVetor(v,0,5)
     imprima(m, v[m])
```

```
função minimoVetor (v,a,b)
entrada: vetor v indexado por [a..b], com a ≤ b
saída: m ∈ [a..b], tal que v[m] ≤ v[j] ∀j ∈ [a..b]
se a = b
    retorne a
m ← minimoVetor(v,a,b-1)
Se v[b] < v[m]
     m ← b
retorne m
```

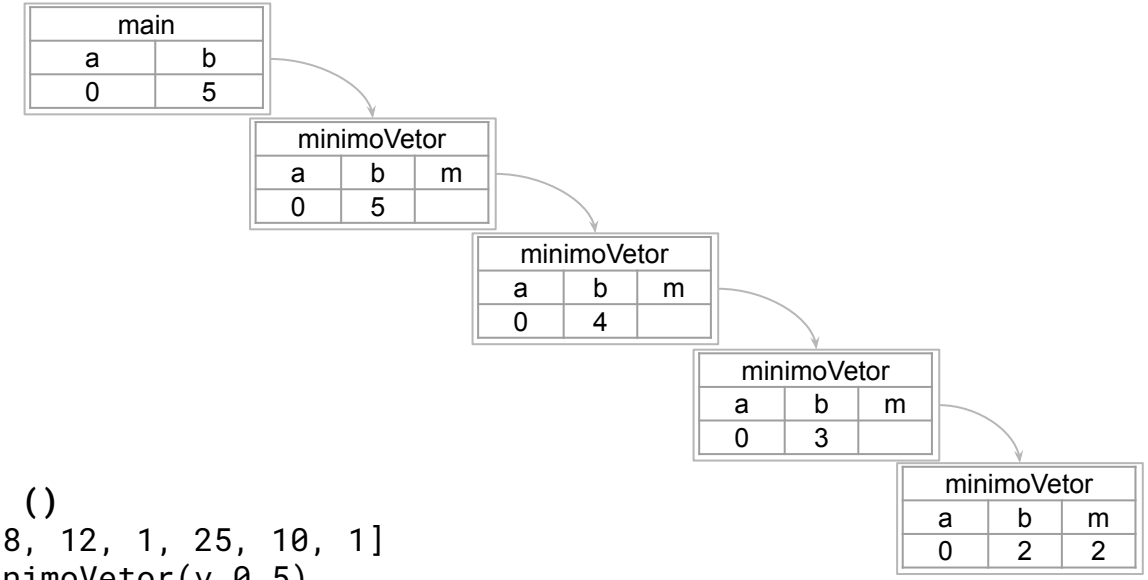
i	**0**	**1**	2	3	4	5
v[i]	28	12	1	25	10	1




```
função main ()
     v ← [28, 12, 1, 25, 10, 1]
     m ← minimoVetor(v,0,5)
     imprima(m, v[m])
```

```
função minimoVetor (v,a,b)
entrada: vetor v indexado por [a..b], com a ≤ b
saída: m ∈ [a..b], tal que v[m] ≤ v[j] ∀j ∈ [a..b]
se a = b
    retorne a
m ← minimoVetor(v,a,b-1)
Se v[b] < v[m]
     m ← b
retorne m
```

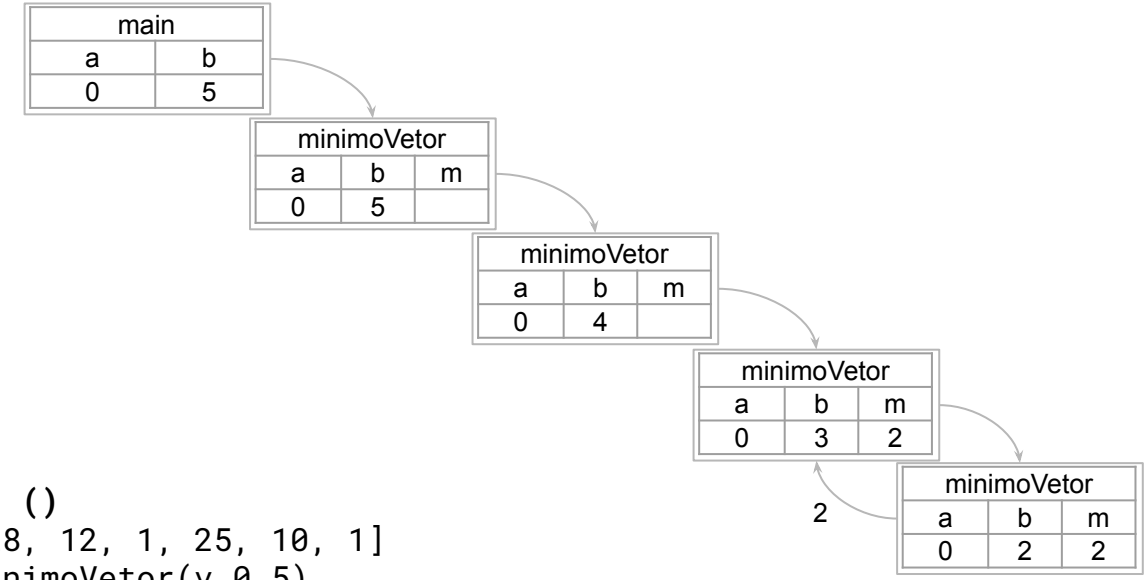
i	**0**	**1**	**2**	3	4	5
v[i]	28	12	1	25	10	1




```
função main ()
     v ← [28, 12, 1, 25, 10, 1]
     m ← minimoVetor(v,0,5)
     imprima(m, v[m])
```

```
função minimoVetor (v,a,b)
entrada: vetor v indexado por [a..b], com a ≤ b
saída: m ∈ [a..b], tal que v[m] ≤ v[j] ∀j ∈ [a..b]
se a = b
    retorne a
m ← minimoVetor(v,a,b-1)
Se v[b] < v[m]
     m ← b
retorne m
```

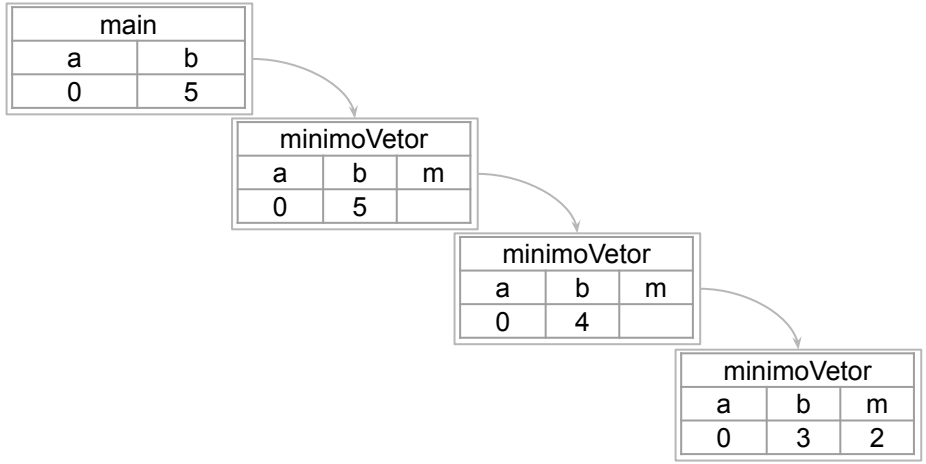
i	**0**	**1**	**2**	3	4	5
v[i]	28	12	1	25	10	1




```
função main ()
     v ← [28, 12, 1, 25, 10, 1]
     m ← minimoVetor(v,0,5)
     imprima(m, v[m])
```

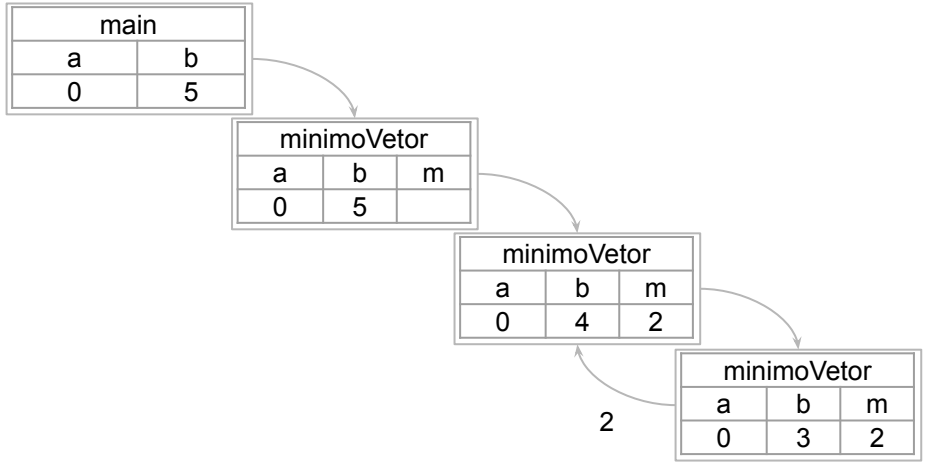
```
função minimoVetor (v,a,b)
entrada: vetor v indexado por [a..b], com a ≤ b
saída: m ∈ [a..b], tal que v[m] ≤ v[j] ∀j ∈ [a..b]
se a = b
    retorne a
m ← minimoVetor(v,a,b-1)
Se v[b] < v[m]
     m ← b
retorne m
```

i	**0**	**1**	**2**	3	4	5
v[i]	28	12	1	25	10	1



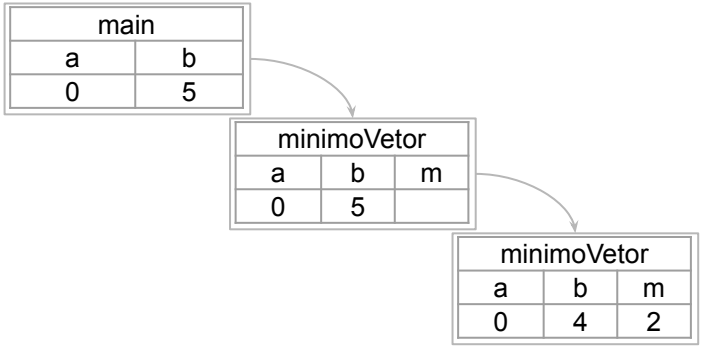

```
função main ()
     v ← [28, 12, 1, 25, 10, 1]
     m ← minimoVetor(v,0,5)
     imprima(m, v[m])


função minimoVetor (v,a,b)
entrada: vetor v indexado por [a..b], com a ≤ b
saída: m ∈ [a..b], tal que v[m] ≤ v[j] ∀j ∈ [a..b]
se a = b
    retorne a
m ← minimoVetor(v,a,b-1)
Se v[b] < v[m]
     m ← b
retorne m
```

i	**0**	**1**	**2**	**3**	4	5
v[i]	28	12	1	25	10	1



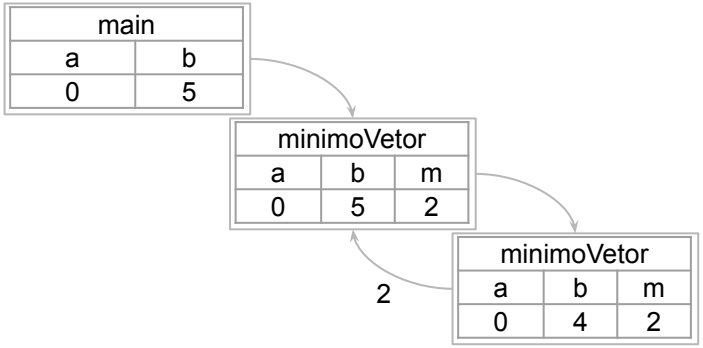

```
função main ()
     v ← [28, 12, 1, 25, 10, 1]
     m ← minimoVetor(v,0,5)
     imprima(m, v[m])


função minimoVetor (v,a,b)
entrada: vetor v indexado por [a..b], com a ≤ b
saída: m ∈ [a..b], tal que v[m] ≤ v[j] ∀j ∈ [a..b]
se a = b
    retorne a
m ← minimoVetor(v,a,b-1)
Se v[b] < v[m]
     m ← b
retorne m
```

i	**0**	**1**	**2**	**3**	4	5
v[i]	28	12	1	25	10	1




```
função main ()
     v ← [28, 12, 1, 25, 10, 1]
     m ← minimoVetor(v,0,5)
     imprima(m, v[m])


função minimoVetor (v,a,b)
entrada: vetor v indexado por [a..b], com a ≤ b
saída: m ∈ [a..b], tal que v[m] ≤ v[j] ∀j ∈ [a..b]
se a = b
    retorne a
m ← minimoVetor(v,a,b-1)
Se v[b] < v[m]
     m ← b
retorne m
```

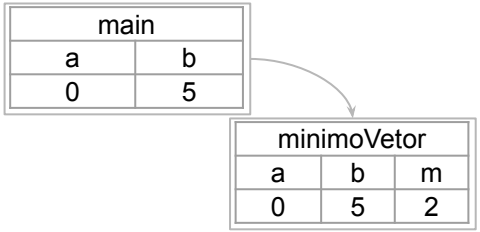
i	**0**	**1**	**2**	**3**	**4**	5
v[i]	28	12	1	25	10	1




```
função main ()
     v ← [28, 12, 1, 25, 10, 1]
     m ← minimoVetor(v,0,5)
     imprima(m, v[m])
```

```
função minimoVetor (v,a,b)
entrada: vetor v indexado por [a..b], com a ≤ b
saída: m ∈ [a..b], tal que v[m] ≤ v[j] ∀j ∈ [a..b]
se a = b
    retorne a
m ← minimoVetor(v,a,b-1)
Se v[b] < v[m]
     m ← b
retorne m
```

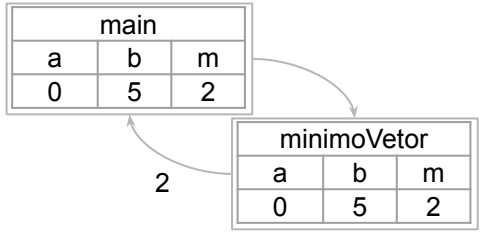
i	**0**	**1**	**2**	**3**	**4**	5
v[i]	28	12	1	25	10	1




```
função main ()
     v ← [28, 12, 1, 25, 10, 1]
     m ← minimoVetor(v,0,5)
     imprima(m, v[m])
```

```
função minimoVetor (v,a,b)
entrada: vetor v indexado por [a..b], com a ≤ b
saída: m ∈ [a..b], tal que v[m] ≤ v[j] ∀j ∈ [a..b]
se a = b
    retorne a
m ← minimoVetor(v,a,b-1)
Se v[b] < v[m]
     m ← b
retorne m
```

i	**0**	**1**	**2**	**3**	**4**	**5**
v[i]	28	12	1	25	10	1




```
função main ()
     v ← [28, 12, 1, 25, 10, 1]
     m ← minimoVetor(v,0,5)
     imprima(m, v[m])
```

```
função minimoVetor (v,a,b)
entrada: vetor v indexado por [a..b], com a ≤ b
saída: m ∈ [a..b], tal que v[m] ≤ v[j] ∀j ∈ [a..b]
se a = b
    retorne a
m ← minimoVetor(v,a,b-1)
Se v[b] < v[m]
     m ← b
retorne m
```

i	**0**	**1**	**2**	**3**	**4**	**5**
v[i]	28	12	1	25	10	1

main		
a	b	m
0	5	2

Saída:
2 1

```
função main ()
     v ← [28, 12, 1, 25, 10, 1]
     m ← minimoVetor(v,0,5)
     imprima(m, v[m])
```

```
função minimoVetor (v,a,b)
entrada: vetor v indexado por [a..b], com a ≤ b
saída: m ∈ [a..b], tal que v[m] ≤ v[j] ∀j ∈ [a..b]
se a = b
    retorne a
m ← minimoVetor(v,a,b-1)
Se v[b] < v[m]
     m ← b
retorne m
```

i	0	1	2	3	4	5
v[i]	28	12	1	25	10	1

Pergunta

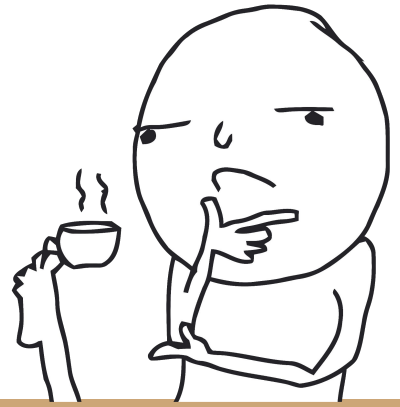
O número de comparações é o mesmo da versão iterativa e recursiva?

```
função minimoVetor (v,a,b)
entrada: vetor v indexado por [a..b], com a ≤ b
saída: m ∈ [a..b], tal que v[m] ≤ v[j] ∀j ∈ [a..b]
se a = b
    retorne a
m ← minimoVetor(v,a,b-1)
Se v[b] < v[m]
     m ← b
retorne m
```

Recursivo.

```
função minimoVetor (v,a,b)
entrada: vetor v indexado por [a..b], com a ≤ b
saída: m ∈ [a..b], tal que v[m] ≤ v[j] ∀j ∈ [a..b]
j ← a
m ← a
enquanto j < b
    j ← j + 1
    se v[j] < v[m]
       m ← j
retorne m
```

Iterativo.



Recorrências

Analisando a versão **recursiva** da função minimoVetor.

Vamos responder quantas comparações são necessárias nessa versão do algoritmo.

```
função minimoVetor (v,a,b)
entrada: vetor v indexado por [a..b], com a ≤ b
saída: m ∈ [a..b], tal que v[m] ≤ v[j] ∀j ∈ [a..b]
se a = b
    retorne a
m ← minimoVetor(v,a,b-1)
Se v[b] < v[m]
    m ← b
retorne m
```

Recorrências

Primeiro, lembrando que o tamanho do vetor é $|(v, a, b)| = b - a + 1 = n.$

```
função minimoVetor (v,a,b)
entrada: vetor v indexado por [a..b], com a ≤ b
saída: m ∈ [a..b], tal que v[m] ≤ v[j] ∀j ∈ [a..b]
se a = b
    retorne a
m ← minimoVetor(v,a,b-1)
Se v[b] < v[m]
     m ← b
retorne m
```

Recorrências

Primeiro, lembrando que o tamanho do vetor é $|(v, a, b)| = b - a + 1 = n.$

Para o caso base, qual o valor de *n*, e quantas **comparações de elementos do vetor** são necessárias?

```
função minimoVetor (v,a,b)
entrada: vetor v indexado por [a..b], com a ≤ b
saída: m ∈ [a..b], tal que v[m] ≤ v[j] ∀j ∈ [a..b]
se a = b
    retorne a
m ← minimoVetor(v,a,b-1)
Se v[b] < v[m]
     m ← b
retorne m
```

Recorrências

Primeiro, lembrando que o tamanho do vetor é $|(v, a, b)| = b - a + 1 = n.$

Para o caso base, qual o valor de *n*, e quantas **comparações de elementos do vetor** são necessárias?

> n = 1.
>
> São necessárias 0 comparações entre elementos.

```
função minimoVetor (v,a,b)
entrada: vetor v indexado por [a..b], com a ≤ b
saída: m ∈ [a..b], tal que v[m] ≤ v[j] ∀j ∈ [a..b]
se a = b
    retorne a
m ← minimoVetor(v,a,b-1)
Se v[b] < v[m]
     m ← b
retorne m
```

Recorrências

E para os casos não base?

```
função minimoVetor (v,a,b)
entrada: vetor v indexado por [a..b], com a ≤ b
saída: m ∈ [a..b], tal que v[m] ≤ v[j] ∀j ∈ [a..b]
se a = b
    retorne a
m ← minimoVetor(v,a,b-1)
Se v[b] < v[m]
     m ← b
retorne m
```

Recorrências

E para os casos não base?

n > *1*.

São necessárias 1 + numeroComparacoes(minimoVetor(v,a,b-1)).

```
função minimoVetor (v,a,b)
entrada: vetor v indexado por [a..b], com a ≤ b
saída: m ∈ [a..b], tal que v[m] ≤ v[j] ∀j ∈ [a..b]
se a = b
    retorne a
m ← minimoVetor(v,a,b-1)
Se v[b] < v[m]
     m ← b
retorne m
```

Recorrências

Definindo uma função de custo $C(n)$

Recorrências

Definindo uma função de custo $C(n)$

$$C(n) = \begin{cases} 0, & \text{se } n = 1, \\ 1 + C(n-1), & \text{se } n > 1 \end{cases}$$

Recorrências

C(n) é uma função definida de forma recursiva. Para calcular C(n) precisamos calcular C(n-1), mas para calcular C(n-1) precisamos calcular C(n-2), ...

Isso é uma **recorrência**.

$$C(n) = \begin{cases} 0, & \text{se } n = 1, \\ 1 + C(n-1), & \text{se } n > 1 \end{cases}$$

Recorrências

Exemplo: qual o custo para C(5)?

$$C(n) = \begin{cases} 0, & \text{se } n = 1, \\ 1 + C(n-1), & \text{se } n > 1 \end{cases}$$

Recorrências

Vamos expandir a função para resolver a equação para o caso geral C(n).

$$\forall n > 0,$$

$$C(n) = 1 + C(n-1) = 1 + (1 + C(n-2)) = 1 + (1 + (1 + C(n-3))) = \ldots$$

$$= C(n) = 1 + C(n-1) = 2 + C(n-2)) = 3 + C(n-3))) = \ldots$$

$$= \mu + C(n-\mu)$$

$$C(n) = \begin{cases} 0, & \text{se } n = 1, \\ 1 + C(n-1), & \text{se } n > 1 \end{cases}$$

Recorrências

Continuamos recursivamente, até atingir o caso base, onde sabemos a resposta.
Temos o caso base para n = 1, logo:

$$n - \mu = 1$$

Implicando assim que.

$$\mu = n - 1$$

$$C(n) = \begin{cases} 0, & \text{se } n = 1, \\ 1 + C(n-1), & \text{se } n > 1 \end{cases}$$

Substituindo

$$C(n) = \mu + C(n - \mu)$$

$$\mu = n - 1$$

$$C(n) = \begin{cases} 0, & \text{se } n = 1, \\ 1 + C(n-1), & \text{se } n > 1, \end{cases}$$

$$C(n) = \mu + C(n - \mu) = n - 1 + C(n - (n - 1))$$

$$= n - 1 + C(1) = n - 1$$

Comparando

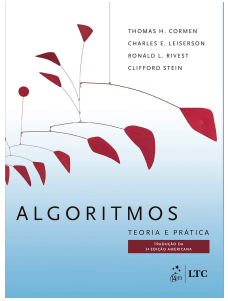
Comprovamos então que tanto a versão iterativa quanto a versão recursiva do algoritmo para encontrar o índice do menor valor custam $n-1$ comparações.

Exercícios

1. Considere o algoritmo para calcular n! (dado na aula passada).
    a. Na versão **recursiva** do algoritmo, expresse M(n) como uma recorrência para calcular o número de multiplicações realizadas;
    b. Resolva a recorrência.
2. Considere o algoritmo para calcular o somatório dos elementos de um vetor (exercício da aula passada).
    a. Na versão **recursiva** do algoritmo, expresse S(n) como uma recorrência para calcular o número de somas realizadas;
    b. Resolva a recorrência.

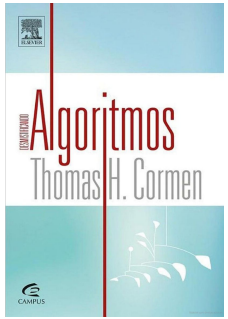
Referências

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

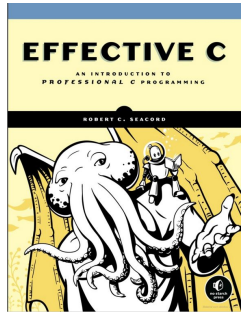


R. Sedgewick, K. Wayne. Algorithms Part I. 4a ed. 2014

T. Cormen. Desmistificando algoritmos. 2017.



R. C. Seacord. Effective C: An Introduction to Professional C. 2020.



Licença