

A diferença entre as notações de complexidade  $O$ ,  $\Omega$  e  $\Theta$

" $O$ " denota um limite superior.

" $\Omega$ " denota um limite inferior.

" $\Theta$ " representa os  $O$  e  $\Omega$  ao mesmo tempo. Ou seja,  $O = \Omega = \Theta$

Resposta tiradas de

<https://homepages.dcc.ufmg.br/~cunha/teaching/20121/aeds2/complexity.pdf>

## Exercício (5)

- Calcule o número de subtrações que o código abaixo realiza:

```
int i = 10;
```

```
while (i >= 7){
```

```
    i--;
```

```
}
```

→ 4

$O=(1)$


$\Omega=(1)$

$\Theta=(1)$

## Exercício (6)

- Calcule o número de subtrações que o código abaixo realiza:

```
...  
for (int i = 5; i >= 2; i--){  
    a--;  
}
```



$$O=(1)$$

$$\Omega=(1)$$

$$\Theta=(1)$$

# Exercício (7)

- Calcule o número de subtrações que o código abaixo realiza:

```
...  
for (int i = 0; i < 5; i++){  
    if (i % 2 == 0){  
        a--;  
        b--;  
    } else {  
        c--;  
    }  
}
```

Handwritten annotations on the code:

- For the `if` condition `(i % 2 == 0)`, a handwritten note  $\rightarrow 0, 2, 4$  indicates the values of `i` that satisfy the condition. A bracket under these values points to the `a--;` and `b--;` statements, with a calculation  $3 \times 2 = 6$  indicating 6 subtractions.
- For the `else` block, a handwritten note  $\rightarrow 1, 3$  indicates the values of `i` that do not satisfy the condition. A bracket under these values points to the `c--;` statement, with a calculation  $2$  indicating 2 subtractions.
- A large bracket on the right side of the code blocks points to a circled number **8**, representing the total number of subtractions.

$$O=(1)$$

$$\Omega=(1)$$

$$\Theta=(1)$$

## Noções de Complexidade

## Exercício (8)

- Calcule o número de subtrações que o código abaixo realiza:

```
...  
for (int i = 0; i < n; i++){  
    for (int j = 0; j < n; j++){  
        a--;  
    }  
}
```



$$O=(n^2)$$

$$\Omega=(n^2)$$

$$\Theta=(n^2)$$

# Exercício (9)

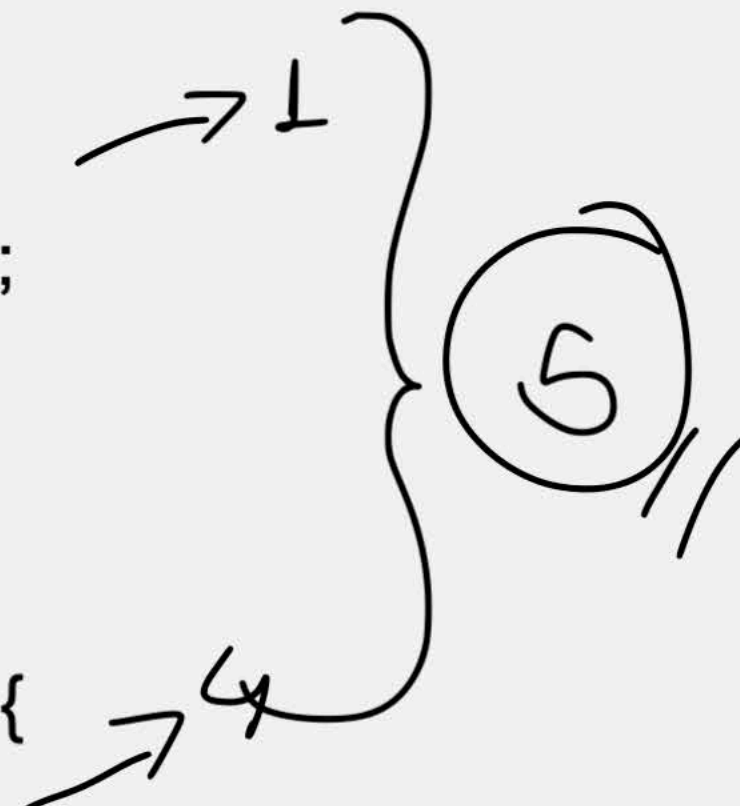
- Calcule o número de subtrações que o código abaixo realiza:

```
int i = 1, b = 10;
```

```
while (i > 0){  
    b--;  
    i = i >> 1;  
}
```

```
i = 0;
```

```
while (i < 15){  
    b--;  
    i += 2;  
}
```



$$O=(1)$$

$$\Omega=(1)$$

$$\Theta=(1)$$

## Exercício (10)

- Calcule o número de multiplicações que o código abaixo realiza:

```
for (int i = 0; i < n; i++)  
  for (int j = 0; j < n - 3; j++)  
    a *= 2;
```

$$O=(n^2)$$

$$\Omega=(n^2)$$

$$\Theta=(n^2)$$

$$n \cdot (n - 3) = n^2 - 3n$$



# Exercício (11)

- Calcule o número de multiplicações que o código abaixo realiza:

```
for (int i = n - 7; i >= 1; i--)  
    for (int j = 0; j < n; j++)  
        a *= 2;
```

$$O=(n^2)$$

$$\Omega=(n^2)$$

$$\Theta=(n^2)$$

$$(n - 7)n = n^2 - 7n$$



## Exercício (12)

- Calcule o número de multiplicações que o código abaixo realiza:

```
for (int i = n; i > 0; i /= 2)  
    a *= 2;
```

→  $\log_2(n)$

$O = (\lg(n))$   
 $\Omega = (\lg(n))$   
 $\Theta = (\lg(n))$

**Dica: Na pasta fonte, veja o código Log.java**

## Exercício (13)

- Calcule o número de multiplicações que o código abaixo realiza:

```
for (int i = n+4; i > 0; i >>= 1)  
    a *= 2;
```

$$\rightarrow \lfloor \log_2(n+4) \rfloor + 1$$

$$O = (\lg(n))$$

$$\Omega = (\lg(n))$$

$$\Theta = (\lg(n))$$

- Calcule o número de multiplicações que o código abaixo realiza:

```
for (int i = n - 7; i >= 1; i--)  
    for (int j = n - 7; j >= 1; j--)  
        a *= 2;
```

$$(n - 7)(n - 7) = (n - 7)^2 //$$

$$O = (n^2)$$

$$\Omega = (n^2)$$

$$\Theta = (n^2)$$

## Exercício (15)

- Calcule o número de multiplicações que o código abaixo realiza:

```
for (int i = n + 1; i > 0; i /= 2)  
    a *= 2;
```

$$\lfloor \log_2(n+1) \rfloor$$

$$O=(\lg(n))$$

$$\Omega=(\lg(n))$$

$$\Theta=(\lg(n))$$

# Exercício (16)

- Calcule o número de multiplicações que o código abaixo realiza:

```
for (int i = n; i > 1; i /= 2)  
    a *= 2;
```

$$\lfloor \log_2(n) \rfloor - 1$$

$$O=(\lg(n))$$

$$\Omega=(\lg(n))$$

$$\Theta=(\lg(n))$$

## Exercício (17)

- Calcule o número de multiplicações que o código abaixo realiza:

```
for (int i = 1; i < n; i *= 2)  
    a *= 2;
```

$$\left\lfloor \frac{n}{2} \right\rfloor$$

$$O=(n)$$

$$\Omega=(n)$$

$$\Theta=(n)$$



# Exercício (18)

- Calcule o número de multiplicações que o código abaixo realiza:

```
for (int i = 1; i <= n; i *= 2)  
    a *= 2;
```

$$\frac{n}{2} + 1$$

$$O=(n)$$

$$\Omega=(n)$$


$$\Theta=(n)$$



## Exercício Resolvido (1)

- Calcule o número de subtrações que o código abaixo realiza:

```
...  
a--;  
a -= 3;  
a = a - 2;
```



$$O=(1)$$

$$\Omega=(1)$$

$$\Theta=(1)$$

# Exercício Resolvido (2)

- Calcule o número de adições que o código abaixo realiza:

```
...  
if (a + 5 < b + 3){  
    i++;  
    ++b;  
    a += 3;  
} else {  
    j++;  
}
```

melhor caso

3 vezes

$O=(1)$

pior caso

$\Omega=(1)$

$\Theta=(1)$

5 vezes

## Exercício Resolvido (3)

- Calcule o número de adições que o código abaixo realiza:

```
...
if (a + 5 < b + 3 || c + 1 < d + 3){
    i++;
    ++b;
    a += 3;
} else {
    j++;
}
```

→ V | ? → melhor caso  
 → F | F → 5 vezes  
 → F | V → pior caso  
 → 7 vezes

$O=(1)$

$\Omega=(1)$

$\Theta=(1)$

# Exercício Resolvido (4)

- Calcule o número de subtrações que o código abaixo realiza:

```
...  
for (int i = 0; i < 4; i++){  
    a--;  
}
```

4

$O=(1)$

$\Omega=(1)$

$\Theta=(1)$

# Exercício Resolvido (5)

- Calcule o número de subtrações que o código abaixo realiza:

```
...
for (int i = 0; i < n; i++){
    a--;
    b--;
}
```

$$(2 \cdot n)$$

$$O=(n)$$

$$\Omega=(n)$$

$$\Theta=(n)$$

Sua resposta deve ser em função de n



# Exercício Resolvido (6)

- Calcule o número de subtrações que o código abaixo realiza:

```
int i = 0, b = 10;
```

```
while (i < 3){
```

```
    i++;
```

```
    b--;
```

```
}
```

3

$O=(1)$

$\Omega=(1)$

$\Theta=(1)$

## Exercício Resolvido (7)

- Calcule o número de subtrações que o código abaixo realiza:

```
...  
for (int i = 3; i < n; i++){  
    a--;  
}
```

$$(n - 3)$$

$$O=(n)$$

$$\Omega=(n)$$

$$\Theta=(n)$$



# Exercício Resolvido (8)

- Calcule o número de subtrações que o código abaixo realiza:



```
int a = 10;  
  
for (int i = 0; i < 3; i++){  
    for (int j = 0; j < 2; j++){  
        a--;  
    }  
}
```

$$3 \cdot 2 \cdot 1 = 6$$

$$O=(1)$$

$$\Omega=(1)$$

$$\Theta=(1)$$

- Calcule o número de multiplicações que o código abaixo realiza:

```
for (int i = n; i > 0; i /= 2)  
    a *= 2;
```

$$\lg(n)$$

$$O = (\lg(n))$$

$$\Omega = (\lg(n))$$

$$\Theta = (\lg(n))$$

# Exercício Resolvido (10)

• Faça um método que receba um número inteiro  $n$  e efetue o número de subtrações pedido em:

a)  $3n + 2n^2$

b)  $5n + 4n^3$

c)  $\lg(n) + n$

d)  $2n^3 + 5$

e)  $9n^4 + 5n^2 + n/2$

f)  $\lg(n) + 5 \lg(n)$

a)  $3n + 2n^2$      $O = (n^2)$ ,  $\Omega = (n^2)$ ,  $\Theta = (n^2)$

```
for(int i = 0; i < n; i++){
    a--;
    b--;
    c--;
}
for(int i = 0; i < n; i++){
    for(int j = 0; j < n; j++){
        a--;
        b--;
    }
}
```

b)  $5n + 4n^3$      $O = (n^3)$ ,  $\Omega = (n^3)$ ,  $\Theta = (n^3)$

```
for(int i = 0; i < n; i++){
    for(int i = 0; i < 5; i++){
        a--;
    }
}
for(int i = 0; i < n; i++){
    for(int j = 0; j < n; j++){
        for(int k = 0; k < n; k++){
            for(int i = 0; i < 4; i++){
                a--;
            }
        }
    }
}
```

c)  $\lg(n) + n$      $O = (\lg(n))$ ,  $\Omega = (\lg(n))$ ,  $\Theta = (\lg(n))$

```
for(int i = n; n > 0; i/=2){
    a--;
}
for(int i = 0; i < n; i++){
    a--;
}
```

d)  $2n^3 + 5$      $O = (n^3)$ ,  $\Omega = (n^3)$ ,  $\Theta = (n^3)$

```
for(int i = 0; i < n; i++){
    for(int j = 0; j < n; j++){
```

```

        for(int k = 0; k < n; k++){
            a--;
            b--;
        }
    }
}
for(int i = 0; i < 5; i++){
    a--;
}

```

e)  $9n^4 + 5n^2 + n/2$       $O = (n^4)$ ,  $\Omega = (n^4)$ ,  $\Theta = (n^4)$

```

for(int i = 0; i < n; i++){
    for(int j = 0; j < n; j++){
        for(int k = 0; k < n; k++){
            for(int l = 0; l < n; l++){
                for(int m = 0; m < 9; m++){
                    a--;
                }
            }
        }
    }
}
for(int i = 0; i < n; i++){
    for(int j = 0; j < n; j++){
        for(int k = 0; k < 5; k++){
            a--;
        }
    }
}
for(int i = 0; i < n; i+=2){
    a--;
}

```

f)  $\lg(n) + 5\lg(n)$       $O = (\lg(n))$ ,  $\Omega = (\lg(n))$ ,  $\Theta = (\lg(n))$

```

for(int i = n; n > 0; i/=2){
    a--;
}
for(int i = n; n > 0; i/=2){
    for(int j = 0; j < 5; j++){
        a--;
    }
}

```

## Noções de Complexidade

## Exercício Resolvido (11)

- Encontre o menor valor em um *array* de inteiros

```
int min = array[0];  
  
for (int i = 1; i < n; i++){  
    if (min > array[i]){  
        min = array[i];  
    }  
}
```

1º) Qual é a operação relevante?

*Comparação*

2º) Quantas vezes ela será executada?

*$(n - 1)$*