

# Contagem de operações:

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

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

R = 3

---

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

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

R = Melhor caso DUAS do IF e UMA do else = 3

Pior caso DUAS do IF e TRES dentro do IF = 5

---

- 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++;  
}
```

R = Pior dos casos 8

Melhor dos casos 5

---

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

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

R = 4, pois nesse caso o número de subtrações é igual a N, definido neste for por 4.

---

## CALCULAR O CUSTO:

- Será o custo da condição mais o número de iterações multiplicado pela soma dos custos da condição e da lista a ser repetida

```
while ( condição() ){  
    lista();  
}
```

**Custo:**  $\text{condição()} + n \times [\text{lista()} + \text{condição()}]$

*onde  $n$  é o número de vezes que o laço será repetido*

- Será o número  $n$  de iterações multiplicado pela soma dos custos da lista de comandos e da condição

```
do {  
    lista();  
} while ( condição() );
```

**Custo:**  $n \times [\text{lista()} + \text{condição()}]$

*onde  $n$  é o número de vezes que o laço será repetido*

---

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

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

Sua resposta deve ser em função de n

$$R = 2n$$

Pois o FOR vai se repetir N vezes e cada vez que ele repetir ele vai subtrair DUAS VEZES, se ali fosse a--; b--; c--; seria a resposta 3n.

---

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

```
int i = 0, b = 10;  
  
while (i < 3){  
    i++;  
    b--;  
}
```

$$R = 3 \text{ subtrações}$$

---

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

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

$$R = N - 3$$

$N - 3$  pois o valor inicial do FOR não começou do 0, se o valor de  $N$  for 5, então o código vai repetir um número de subtração igual  $5 - 3 = 2$ , sendo 2 o tanto de repetições feitas.

---

- Quando tivermos uma estrutura de repetição em que o contador começa com **a**, repete enquanto menor que  $n$  e é incrementado em uma unidade, faremos **( $n - a$ )** iterações

```
for (int i = a; i < n ; i++){  
    lista();  
}
```

---

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

```
int i = 10;  
  
while (i >= 7){  
    i--;  
}
```

$$R = 3 + 1$$

Pois como vemos o sinal de maior e igual  $\geq$ , então temos que subtrair o valor de  $i$  por 7 e somar por MAIS UM resultando no tanto de repetições feitas.

---

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

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

$$R = 5$$

Segue o mesmo raciocínio de cima,  $5 - 2 = 3$ . Portanto como temos que executar dois tipos de subtração diferentes então somamos esses dois tipos na nossa conta  $3 + 2 = 5$

---

- 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--;  
    }  
}
```

R = Pior dos casos: 2

Melhor dos casos: 1

---

- 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--;  
    }  
}
```

R = 6

Pois o primeiro for repetir 3 vezes e como não tem nenhuma restrição o segundo tem que executar 2 vezes, então apenas multiplicamos  $3 \times 2 = 6$ , como só a uma subtração dentro do for secundário, fica  $3 \times 2 \times 1 = 6$ .

---

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

```
int a = 10, b = 10, c = 10, d = 10;

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

$$R = 3 \times 2 \times 4 = 20$$

---

- 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--;
    }
}
```

$$R = N \times N \times 1 = 2n \cdot 1$$

---



- 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;
}
```

>> significa || em outras palavras OU.

R = ?

---

- 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;
```

R =  $2n$

---

- 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;
```

$$R = ((n - 7) + 1) \times (J) \times 1$$

---

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

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

$$R = (N / 2) \times 1$$

---

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

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

$$R = ???$$

---

- 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;
```

$$R = ((N - 7) + 1) \times ((N - 7) + 1) \times 1$$


---


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

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

$$R = (N / 2) \times 1$$

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

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



Quando  $n$  é uma potência de 2, realizamos  $\lg(n) + 1$  multiplicações

Se  $n = 8$ , efetuamos a multiplicação quando  $i$  vale 8, 4, 2, 1

$n = 16$ , ..... 16, 8, 4, 2, 1

$n = 32$ , ..... 32, 16, 8, 4, 2, 1

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



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

Para um valor qualquer de n, temos  $\lfloor \lg(n) \rfloor + 1$  multiplicações

n = 7, .....	7, 3, 1
Se n = 8, efetuamos a multiplicação quando i vale	8, 4, 2, 1
n = 9, .....	9, 4, 2, 1
n = 15, .....	15, 7, 3, 1
n = 16, .....	16, 8, 4, 2, 1
n = 17, .....	17, 8, 4, 2, 1
n = 31, .....	31, 15, 7, 3, 1
n = 32, .....	32, 16, 8, 4, 2, 1
n = 33, .....	33, 16, 8, 4, 2, 1

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

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

$$R = ((N + 1) / 2) \times 1$$

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

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

$$R = \log(n)$$

Pois se N for 8, temos que fazer:

$$i \times 2 = 2$$

$$2 \times 2 = 4$$

$$4 \times 2 = 8$$

que dessa equação temos que  $\log(8) = 3$ , então são feitas 3 repetições de multiplicação

---

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

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

$$R = \log(n) + 1$$

Seguindo a explicação de cima, seria  $\log(8) + 1 = 4$