

# Unidade I:

## Introdução - Noções de Complexidade




**PUC Minas**

Instituto de Ciências Exatas e Informática  
Departamento de Ciência da Computação

# Agenda

- Exercícios iniciais
- Contagem de operações
- Noção sobre as notações  $O$ ,  $\Omega$  e  $\Theta$

- **Exercícios iniciais** 
- Contagem de operações
- Noção sobre as notações  $O$ ,  $\Omega$  e  $\Theta$

## Exercício (1)

- Resolva as equações abaixo:

a)  $2^0 =$

d)  $2^3 =$

g)  $2^6 =$

j)  $2^9 =$

b)  $2^1 =$

e)  $2^4 =$

h)  $2^7 =$

k)  $2^{10} =$

c)  $2^2 =$

f)  $2^5 =$

i)  $2^8 =$

l)  $2^{11} =$

## Exercício (2)

- Resolva as equações abaixo:

a) $\lg(2048) =$	d) $\lg(256) =$	g) $\lg(32) =$	j) $\lg(4) =$
b) $\lg(1024) =$	e) $\lg(128) =$	h) $\lg(16) =$	k) $\lg(2) =$
c) $\lg(512) =$	f) $\lg(64) =$	i) $\lg(8) =$	l) $\lg(1) =$

---

Nota:  $\lg(n)$  é a mesma coisa que o logaritmo de  $n$  na base dois, ou seja,  $\log_2(n)$

## Exercício (3)

- Resolva as equações abaixo:

a)  $\lceil 4,01 \rceil =$

b)  $\lfloor 4,01 \rfloor =$

c)  $\lceil 4,99 \rceil =$

d)  $\lfloor 4,99 \rfloor =$

e)  $\lceil \lg(16) \rceil =$

f)  $\lfloor \lg(16) \rfloor =$

g)  $\lg(17) =$

h)  $\lceil \lg(17) \rceil =$

i)  $\lfloor \lg(17) \rfloor =$

j)  $\lg(15) =$

k)  $\lceil \lg(15) \rceil =$

l)  $\lfloor \lg(15) \rfloor =$

## Exercício (4)

- Plote um gráfico com todas as funções abaixo:

a)  $f(n) = n$

b)  $f(n) = n^2$

c)  $f(n) = n^3$

d)  $f(n) = \text{sqrt}(n)$

e)  $f(n) = \lg(n) = \log_2(n)$


f)  $f(n) = 3n^2 + 5n - 3$

g)  $f(n) = -3n^2 + 5n - 3$

h)  $f(n) = |-n^2|$

i)  $f(n) = 5n^4 + 2n^2$

j)  $f(n) = n * \lg(n)$

- Exercícios iniciais
- **Contagem de operações** 
- Noção sobre as notações  $O$ ,  $\Omega$  e  $\Theta$



# Exercício Resolvido (1)

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

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

## Exercício Resolvido (1)

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



```
...  
a--;  
a -= 3;  
a = a - 2; //três subtrações
```

## Exercício Resolvido (2)

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

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

## Exercício Resolvido (2)

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



```
... // Melhor caso
if (a + 5 < b + 3){ // 2
    i++;
    ++b;
    a += 3;
} else {
    j++; // 1
}
```

## Exercício Resolvido (2)

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



```
... // Pior caso
if (a + 5 < b + 3){ // 2
    i++;
    ++b;           // 3
    a += 3;
} else {
    j++;
}
```

# Cenários Possíveis

- **Melhor caso**: menor “tempo de execução” para todas entradas possíveis de tamanho  $n$
- **Pior caso**: maior “tempo de execução” para todas entradas possíveis
- **Caso médio (ou esperado)**: média dos tempos de execução para todas as entradas possíveis (abordado em Grafos / PAA)

# Contagem de Operações com Condicional

- Será o custo da condição mais ou o da lista de verdadeira ou o da falsa

```
if ( condição() ){  
    listaVerdadeiro();  
} else {  
    listaFalso();  
}
```

**Melhor caso:**  $\text{condição()} + \min(\text{listaVerdadeiro()}, \text{listaFalso}())$

**Pior caso:**  $\text{condição()} + \max(\text{listaVerdadeiro()}, \text{listaFalso}())$

## 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++;  
}
```



## 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++;  
}
```

**Resposta:** O número máximo de adições acontece quando a primeira condição do if é falsa e a segunda, verdadeira. Se a primeira condição for verdadeira, o Java nem executa a segunda condição (ver AND\_OR.java)

A	B	OR
F	x	x
T	x	T

A	B	AND
F	x	F
T	x	x

## Exercício Resolvido (4)

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

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

## Exercício Resolvido (4)

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



```
...  
for (int i = 0; i < 4; i++){ // Faremos as subtrações quando i vale  
    a--; // 0, 1, 2, 3  
}
```

# Contagem de Operações com Repetição

- 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

# Contagem de Operações com Repetição

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

```
for (int  $i = 0$ ;  $i < n$  ;  $i++$ ){  
    lista();  
}
```

## Exercício Resolvido (5)

- 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  $n$

## Exercício Resolvido (5)

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



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

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--;
```

```
}
```



## Exercício Resolvido (6)

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



```
int i = 0, b = 10;    // Faremos subtrações
                      // quando o valor de i igual
while (i < 3){        // a 0, 1 e 2
    i++;
    b--;
}
```

## Exercício Resolvido (7)

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

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

## Exercício Resolvido (7)

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



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

Se  $n = 6$ , temos subtrações quando  $i$  vale 3, 4, 5 ( $6 - 3 = 3$ , vezes)

$n = 7$

3, 4, 5, 6 ( $7 - 3 = 4$  vezes)

....

$n = 10$

3, 4, 5, 6, 7, 8, 9 ( $10 - 3 = 7$  vezes)

# Contagem de Operações com Repetição

- 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  $i = a; i < n ; i++$ ){  
     $lista()$ ;  
}
```

## Exercício (5)

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

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

## Exercício (6)

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

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

## 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--;  
    }  
}
```

## 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--;  
    }  
}
```



## 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--;  
    }  
}
```

Solução fácil:  $3 \times 2 \times 1$

## 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--;  
    }  
}
```

Solução fácil:  $3 \times 2 \times 1$

## 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--;  
    }  
}
```

Solução fácil:  $3 \times 2 \times 1$

## 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--;  
    }  
}
```

Solução fácil:  $3 \times 2 \times 1$

## 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--;  
    }  
}
```

i	j	sub
0		0

Solução difícil

## Exercício Resolvido (8)

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



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

i	j	sub
0		

## 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--;  
    }  
}
```

i	j	sub
0	0	

## Exercício Resolvido (8)

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



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

i	j	sub
0	0	



## 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--;  
    }  
}
```

i	j	sub
0	0	1

## 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--;  
    }  
}
```

i	j	sub
0	1	1

## Exercício Resolvido (8)

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



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

i	j	sub
0	1	1

## 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--;  
    }  
}
```

i	j	sub
0	1	2

## 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--;  
    }  
}
```

i	j	sub
0	2	2

## Exercício Resolvido (8)

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



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

i	j	sub
0	2	2

## 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--;  
    }  
}
```

i	j	sub
1		2

## Exercício Resolvido (8)

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



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

i	j	sub
1		2



## 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--;  
    }  
}
```

i	j	sub
1	0	2

## Exercício Resolvido (8)

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



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

i	j	sub
1	0	2

## 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--;  
    }  
}
```

i	j	sub
1	0	3

## 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--;  
    }  
}
```

i	j	sub
1	1	3

## Exercício Resolvido (8)

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



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

i	j	sub
1	1	3

## 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--;  
    }  
}
```

i	j	sub
1	1	4

## 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--;  
    }  
}
```

i	j	sub
1	2	4

## Exercício Resolvido (8)

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



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

i	j	sub
1	2	4



## 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--;  
    }  
}
```

i	j	sub
2		4

## Exercício Resolvido (8)

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



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

i	j	sub
2		4

## 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--;  
    }  
}
```

i	j	sub
2	0	4

## Exercício Resolvido (8)

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



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

i	j	sub
2	0	4

## 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--;  
    }  
}
```

i	j	sub
2	0	5

## 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--;  
    }  
}
```

i	j	sub
2	1	5

## Exercício Resolvido (8)

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



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

i	j	sub
2	1	5

## 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--;  
    }  
}
```

i	j	sub
2	1	6



## 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--;  
    }  
}
```

i	j	sub
2	2	6

## Exercício Resolvido (8)

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



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

i	j	sub
2	2	6

## 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--;  
    }  
}
```

i	j	sub
3		6

## Exercício Resolvido (8)

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



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

i	j	sub
3		6

## 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--;  
    }  
}
```

## 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;  
}
```

## 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;
```

## 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;
```



## Exercício (12)

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

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

**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;
```

## Exercício (14)

- 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;
```

## Exercício Resolvido (9)

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

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

## Exercício Resolvido (9)

- 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

# Exercício Resolvido (9)



- 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

# Contagem de Operações com Repetição

- Quando tivermos uma estrutura de repetição em que o escopo de busca é sistematicamente dividido pela metade, temos um custo logarítmico

```
for (int i = n; i > 0; i /= 2){  
    lista();  
}
```

## Exercício (15)

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

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



## Exercício (16)

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

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

## Exercício (17)

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

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

## Exercício (18)

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

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

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

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



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

## 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?

2º) Quantas vezes ela será executada?

## 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?

R: Comparação entre elementos do *array*

2º) Quantas vezes ela será executada?

## 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?

R: Comparação entre elementos do *array*

2º) Quantas vezes ela será executada?

R: Se tivermos  $n$  elementos:  $T(n) = n - 1$




## 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];  
    }  
}
```

3º) O nosso  $T(n) = n - 1$  é para qual dos três casos?

- Exercícios iniciais
- Contagem de operações
- **Noção sobre as notações  $O$ ,  $\Omega$  e  $\Theta$**  

# Noção sobre as Notações $O$ , $\Omega$ e $\Theta$

- “Mão na roda” e podem ser lidas como aproximadamente
- Abordadas mais à frente da disciplina (Unidade III)
- Neste ponto, apresentamos “somente” noções sobre as notações

# Noção sobre as Notações $O$ , $\Omega$ e $\Theta$

- A partir deste ponto, utilizaremos as notações  $O$ ,  $\Omega$  e  $\Theta$  para identificar a complexidade (número de operações) de um algoritmo, assim:
  - Um algoritmo que realiza **1** operação é  **$O(1)$ ,  $\Omega(1)$  e  $\Theta(1)$**
  - Um algoritmo que realiza  **$n$**  operações é  **$O(n)$ ,  $\Omega(n)$  e  $\Theta(n)$**
  - Um algoritmo que realiza  **$n^2$**  operações é  **$O(n^2)$ ,  $\Omega(n^2)$  e  $\Theta(n^2)$**
  - Um algoritmo que realiza  **$\lg(n)$**  operações é  **$O(\lg(n))$ ,  $\Omega(\lg(n))$  e  $\Theta(\lg(n))$**

# Noção sobre as Notações $O$ , $\Omega$ e $\Theta$

- Nas notações, ignoramos as constantes:
  - Um algoritmo que realiza **2**, **3** ou **5** operações é  **$O(1)$** ,  **$\Omega(1)$**  e  **$\Theta(1)$**
  - Um algoritmo que realiza  **$2n$** ,  **$3n$**  ou  **$5n$**  operações é  **$O(n)$** ,  **$\Omega(n)$**  e  **$\Theta(n)$**
  - Um algoritmo que realiza  **$2n^2$** ,  **$3n^2$**  ou  **$5n^2$**  operações é  **$O(n^2)$** ,  **$\Omega(n^2)$**  e  **$\Theta(n^2)$**
  - Um algoritmo que realiza  **$2\lg(n)$** ,  **$3\lg(n)$**  ou  **$5\lg(n)$**  operações é  **$O(\lg(n))$** ,  **$\Omega(\lg(n))$**  e  **$\Theta(\lg(n))$**

# Noção sobre as Notações $O$ , $\Omega$ e $\Theta$

- Nas notações, ignoramos termos com menor crescimento:
  - Um algoritmo que realiza  $3n + 2n^2$  operações é  $O(n^2)$ ,  $\Omega(n^2)$  e  $\Theta(n^2)$
  - Um algoritmo que realiza  $5n + 4n^3$  operações é  $O(n^3)$ ,  $\Omega(n^3)$  e  $\Theta(n^3)$
  - Um algoritmo que realiza  $\lg(n) + n$  operações é  $O(n)$ ,  $\Omega(n)$  e  $\Theta(n)$
  - Um algoritmo que realiza  $2n^3 + 5$  operações é  $O(n^3)$ ,  $\Omega(n^3)$  e  $\Theta(n^3)$
  - Um algoritmo que realiza  $9n^4 + 5n^2 + n/2$  operações é  $O(n^4)$ ,  $\Omega(n^4)$  e  $\Theta(n^4)$
  - Um algoritmo que realiza  $\lg(n) + 5 \lg(n)$  operações é  $O(\lg(n))$ ,  $\Omega(\lg(n))$  e  $\Theta(\lg(n))$

## Trabalho Teórico 5

- Pergunta 1: Qual é a diferença entre as notações  $O$ ,  $\Omega$  e  $\Theta$ ? Pesquise!!!
- Pergunta 2: Qual é a notação  $O$ ,  $\Omega$  e  $\Theta$  para todos os exercícios feitos nesta Unidade?