

# Exercício Resolvido (1): Resolva as Equações

a)  $2^0 = 1$

d)  $2^3 = 8$

g)  $2^6 = 64$

j)  $2^9 = 512$

b)  $2^1 = 2$

e)  $2^4 = 16$

h)  $2^7 = 128$

k)  $2^{10} = 1024$

c)  $2^2 = 4$

f)  $2^5 = 32$

i)  $2^8 = 256$

l)  $2^{11} = 2048$

# Exercício Resolvido (2): Resolva as Equações

a)  $\lg(2048) = 11$     d)  $\lg(256) = 8$     g)  $\lg(32) = 5$     j)  $\lg(4) = 2$

b)  $\lg(1024) = 10$     e)  $\lg(128) = 7$     h)  $\lg(16) = 4$     k)  $\lg(2) = 1$

c)  $\lg(512) = 9$     f)  $\lg(64) = 6$     i)  $\lg(8) = 3$     l)  $\lg(1) = 0$

# Exercício Resolvido (3): Resolva as Equações

$$a) \overline{4,01} = 5$$

$$d) \underline{4,99} = 4$$

$$g) \lg(17) = 4,087$$

$$j) \lg(15) = 3,907$$

$$b) \underline{4,01} = 4$$

$$e) \overline{\lg(16)} = 4$$

$$h) \overline{\lg(17)} = 5$$

$$k) \overline{\lg(15)} = 4$$

$$c) \overline{4,99} = 5$$

$$f) \underline{\lg(16)} = 4$$

$$i) \underline{\lg(17)} = 4$$

$$l) \underline{\lg(15)} = 3$$

## Exercício Resolvido (4): Plote os Gráficos

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

1,25E+9

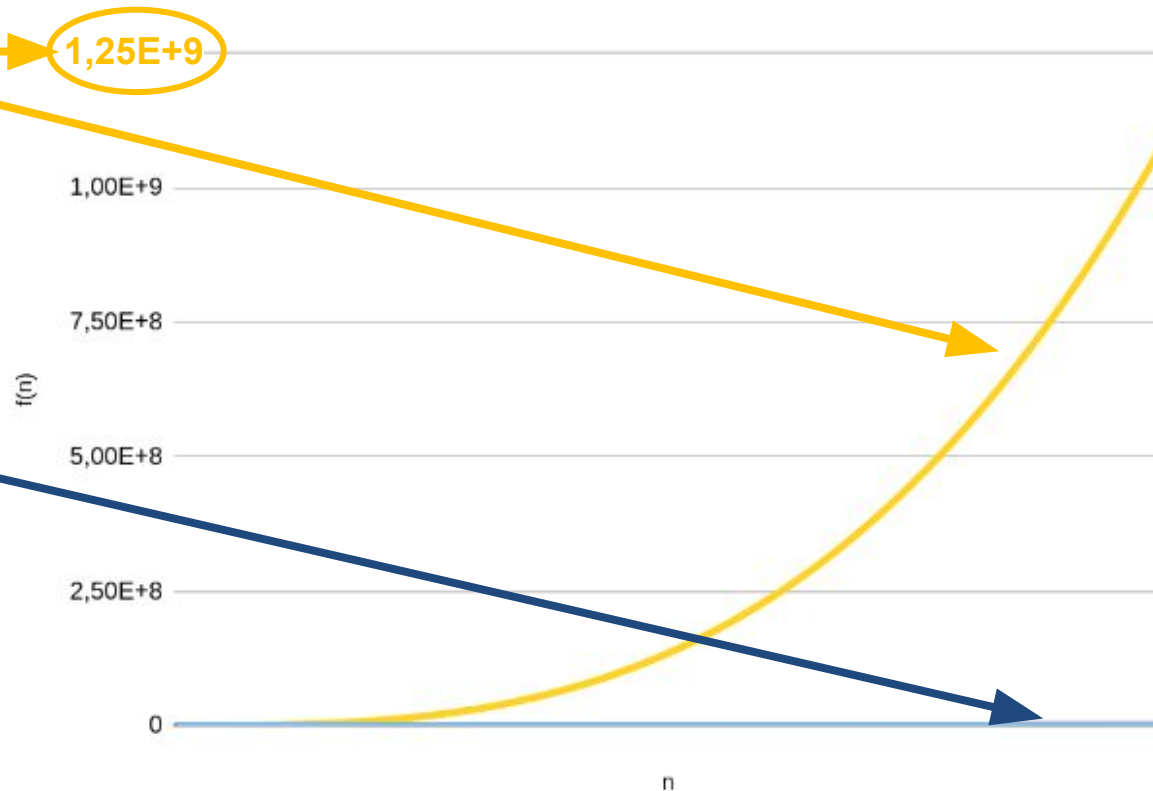
b)  $f(n) = n^2$

c)  $f(n) = n \times \lg(n)$

d)  $f(n) = n$

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

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



## Exercício Resolvido (4): Plote os Gráficos

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

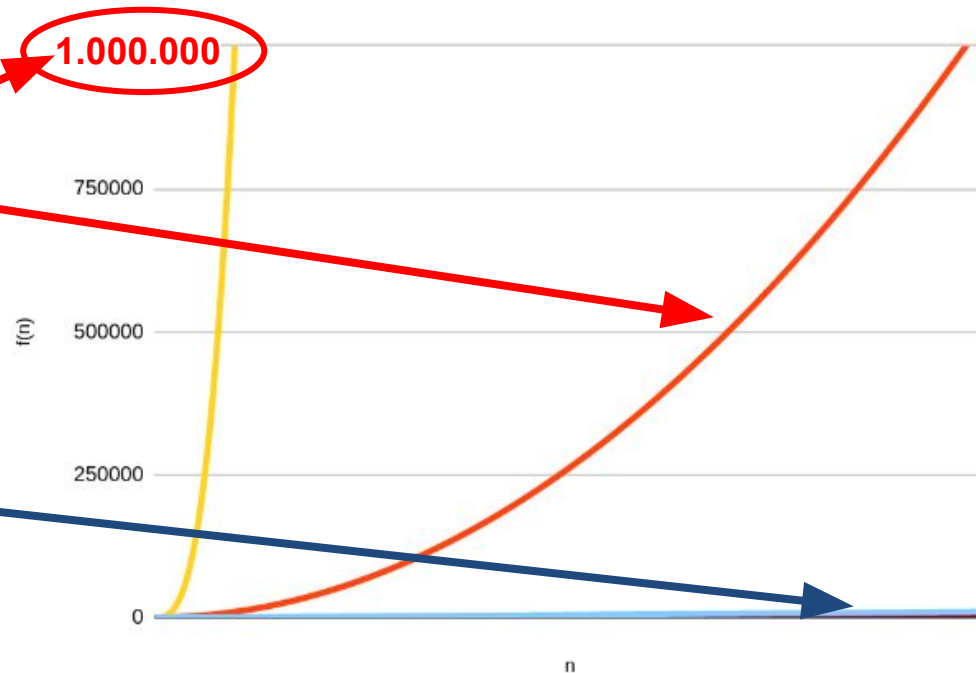
b)  $f(n) = n^2$

c)  $f(n) = n \times \lg(n)$

d)  $f(n) = n$

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

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



## Exercício Resolvido (4): Plote os Gráficos

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

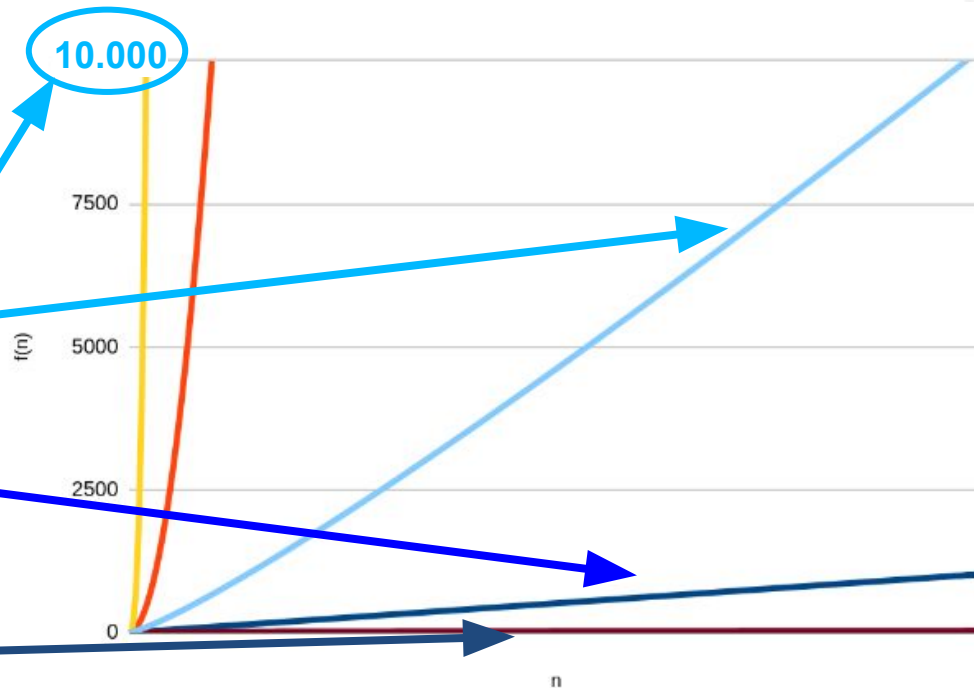
b)  $f(n) = n^2$

c)  $f(n) = n \times \lg(n)$

d)  $f(n) = n$

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

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



## Exercício Resolvido (4): Plote os Gráficos

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

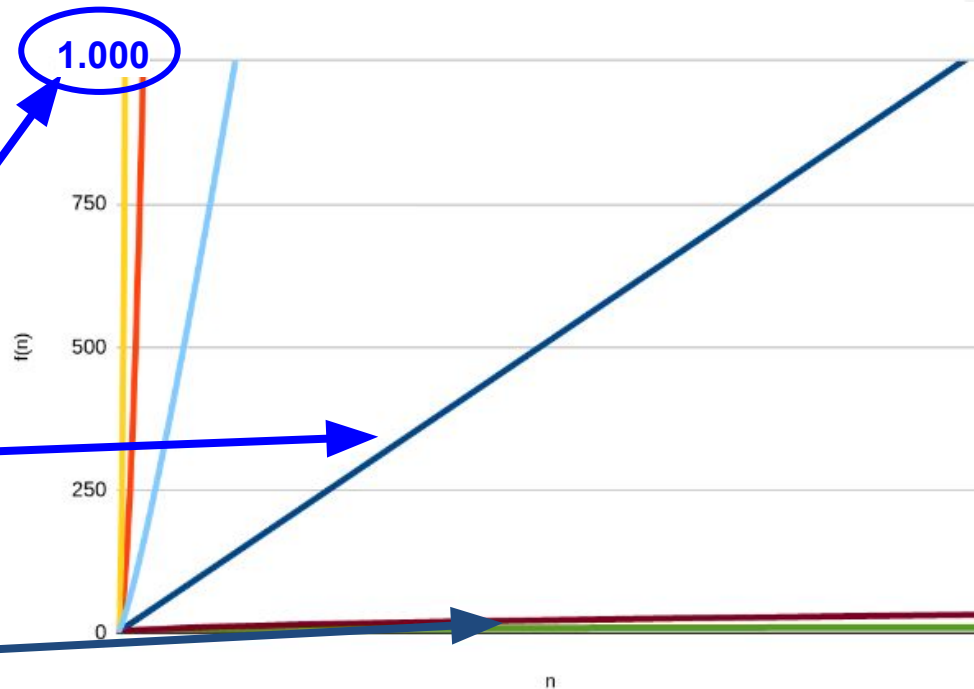
b)  $f(n) = n^2$

c)  $f(n) = n \times \lg(n)$

d)  $f(n) = n$

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

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



## Exercício Resolvido (4): Plote os Gráficos

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

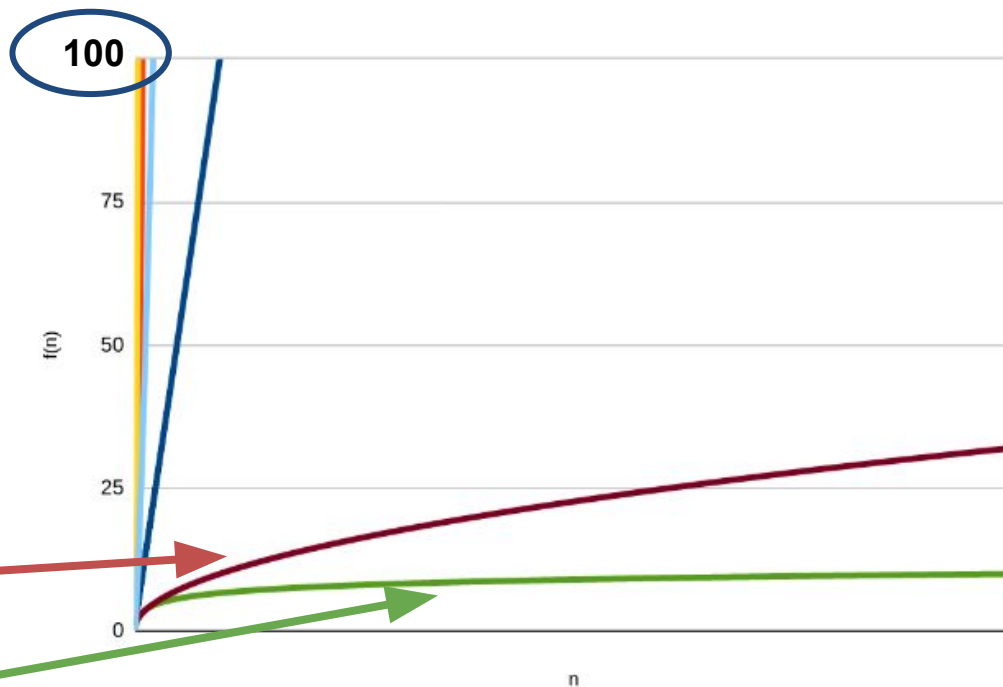
b)  $f(n) = n^2$

c)  $f(n) = n \times \lg(n)$

d)  $f(n) = n$

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

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



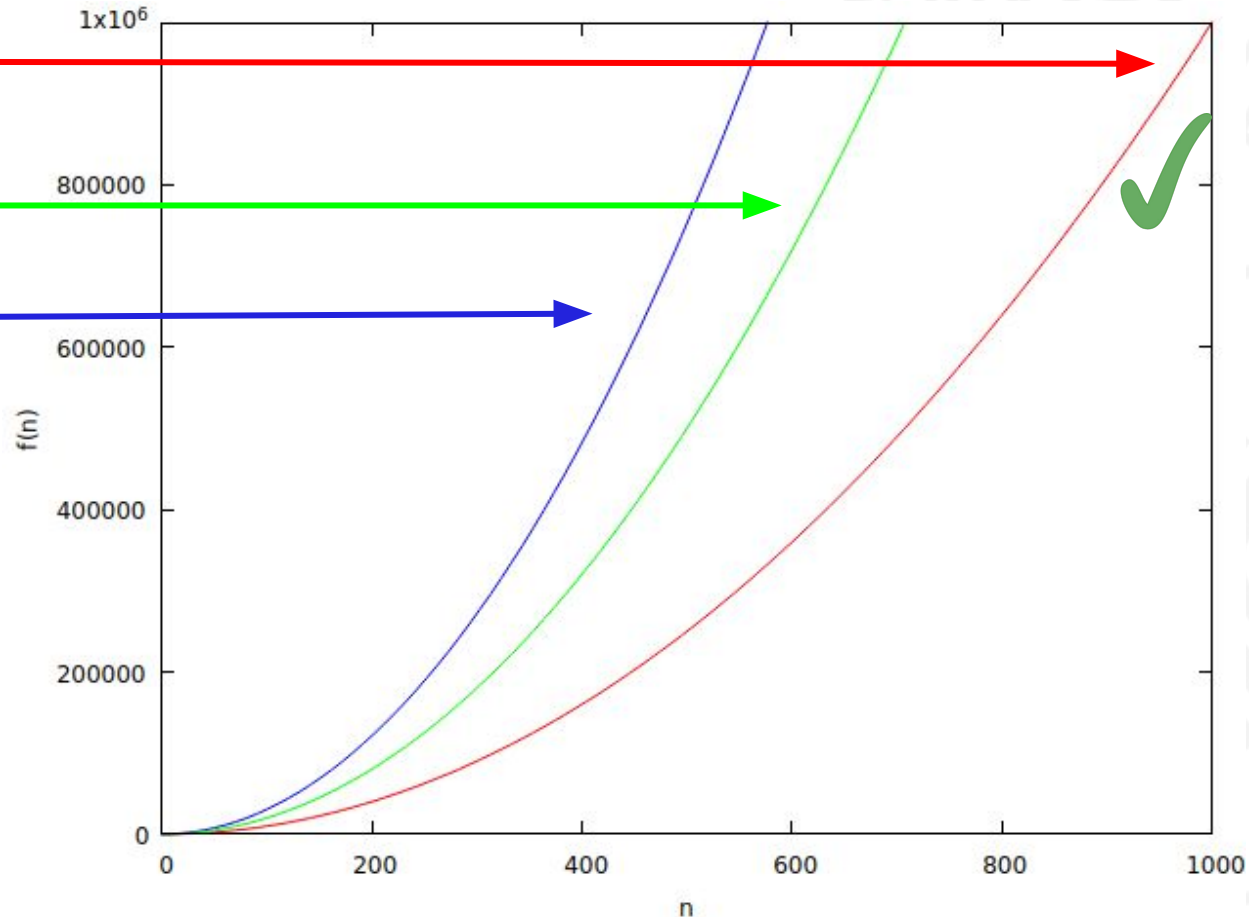


## Exercício Resolvido (5): Plote os Gráficos

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

b)  $f(n) = 2 \times | -n^2 |$

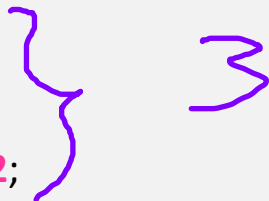
c)  $f(n) = 3n^2 + 5n - 3$



# Exercício Resolvido (6)

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

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



# Exercício Resolvido (7)

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

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

(Pior Caso)

# Exercício Resolvido (8)

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

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

Handwritten annotations in purple:

- A bracket above `a - 5` with the number `2`.
- A bracket above `b - 3` with the number `2`.
- A bracket to the right of `i--;`, `--b;`, and `a -= 3;` with the number `3`.
- An arrow pointing to `j--;` with the number `1`.

# Exercício Resolvido (9)

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

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

3

# Exercício Resolvido (10)

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

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

$2n$

**Observação:** Sua resposta deve ser em função de  $n$

# Exercício Resolvido (11)

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

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

$n - 3$

# Exercício Resolvido (12)

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

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

3



# Exercício Resolvido (13)

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



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

}

# Exercício Resolvido (14)

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

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

3 \* 2  
6

# Exercício Resolvido (16)

- 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)  $2n^4 + 2n^2 + n/2$

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 (16)

- 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)  $2n^4 + 2n^2 + n/2$

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

```
...  
i = 0;  
while (i < n){  
    i++;  
    a--; b--; c--; d--; e--;  
}  
for (i = 0; i < n; i++){  
    for (j = 0; j < n; j++){  
        for (k = 0; k < n; k++, a--, b--, c--, d--);  
    }  
}
```

# Exercício Resolvido (16)

- 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)  $2n^4 + 2n^2 + n/2$

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

```
...  
i = 1;  
while (i < n){  
    i *= 2;  
    a--;  
}  
for (i = 0; i < n; i++){  
    a--;  
}
```

# Exercício Resolvido (16)

- 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)  $2n^4 + 2n^2 + n/2$

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

```
...  
for (i = 0; i < n; i++){  
    for (j = 0; j < n; j++){  
        for (k = 0; k < n; k++, a--, b--);  
    }  
}  
  
a--; b--; c--; d--; e--;
```

# Exercício Resolvido (16)

- 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)  $2n^4 + 2n^2 + n/2$

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

```
...
for (i = 0; i < n; i++){
    for (j = 0; j < n; j++){
        a--; b--;
        for (k = 0; k < n; k++){
            for (l = 0; l < n; l++, c--; d--);
        }
    }
    if (i % 2 == 0) e--;
}
```

# Exercício Resolvido (16)

- 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)  $2n^4 + 2n^2 + n/2$

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

```
...  
i = 1;  
while (i < n){  
    i *= 2;  
    a--; b--; c--; d--; e--; f--;  
}
```



# Exercício Resolvido (17)

- 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: A operação relevante sendo feita são as comparações entre as iterações das variáveis `min` e `array[i]` ao longo do loop.

2º) Quantas vezes ela será executada?

R:  
Ela será executada  $n - 1$  vezes.

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

R: Todos os casos.

# Exercício Resolvido (17)

- 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];  
    }  
}
```

4º) O nosso algoritmo é ótimo? Por que?

Sim, pois todos os elementos da array estão sendo comparados com a condição indicada no `if`.

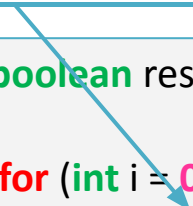
Pause!

# Exercício Resolvido (18)

- Apresente a função de complexidade de tempo (número de comparações entre elementos do array) da pesquisa sequencial no melhor e no pior caso

```
boolean resp = false;

for (int i = 0; i < n; i++){
    if (array[i] == x){
        resp = true;
        i = n;
    }
}
```



**Melhor caso:** elemento desejado na primeira posição

$$t(n) = 1$$

**Pior caso:** elemento desejado não está no array ou está na última posição

$$t(n) = n$$

## Exercício Resolvido (19)

- Apresente a função de complexidade de tempo (número de comparações entre elementos do array) da pesquisa binária no melhor e no pior caso

```
boolean resp = false;
int dir = n - 1, esq = 0, meio, diferenca;
while (esq <= dir) {
    meio = (esq + dir) / 2;
    diferenca = (x - array[meio]);
    if (diferenca == 0){
        resp = true;
        esq = n;
    } else if (diferenca > 0){
        esq = meio + 1;
    } else {
        dir = meio - 1;
    }
}
```

Melhor caso: elemento desejado está na posição  $[(esq+dir)/2]$

$$t(n) = 1$$

Pior caso: elemento desejado não está no array ou está na última posição procurada; contudo, lembrando que cada iteração reduz o espaço de busca pela metade

$$t(n) = \lg(n)$$

# Exercício Resolvido (20)

- Explique porque o Algoritmo de Seleção realiza  $m(n) = 3n - 3$  movimentações de registros

```
for (int i = 0; i < (n - 1); i++) {  
    int menor = i;  
    for (int j = (i + 1); j < n; j++){  
        if (array[menor] > array[j]){  
            menor = j;  
        }  
    }  
    swap(menor, i);  
}
```

```
void swap(int a, int b) {  
    int temp = array[a];  
    array[a] = array[b];  
    array[b] = temp;  
}
```

# Exercício Resolvido (20)

- Explique porque o Algoritmo de Seleção realiza  $m(n) = 3n - 3$  movimentações de registros

```
for (int i = 0; i < (n - 1); i++) {  
    int menor = i;  
    for (int j = (i + 1); j < n; j++){  
        if (array[menor] > array[j]){  
            menor = j;  
        }  
    }  
    swap(menor, i);  
}
```

Todos os casos: o laço repete (n-1) vezes sendo que cada repetição chama o swap que faz 3 movimentações, resultando em:

$$t(n) = 3 \times (n-1) = 3n - 3$$

```
void swap(int a, int b) {  
    int temp = array[a];  
    array[a] = array[b];  
    array[b] = temp;  
}
```

## Exercício Resolvido (21)

- Modifique o código do Algoritmo de Seleção para que ele contabilize o número de movimentações de registros

```
for (int i = 0; i < (n - 1); i++) {  
    int menor = i;  
    for (int j = (i + 1); j < n; j++){  
        if (array[menor] > array[j]){  
            menor = j;  
        }  
    }  
    swap(menor, i);  
}
```

Isso pode ser feito ao adicionar a variável

`int mov = 0;`

posterior dessa variável por meio de

# Exercício Resolvido (22)

- Explique porque o Algoritmo de Seleção realiza  $c(n) = \frac{n^2}{2} - \frac{n}{2}$  comparações entre registros

```
for (int i = 0; i < (n - 1); i++) {  
    int menor = i;  
    for (int j = (i + 1); j < n; j++){  
        if (array[menor] > array[j]){  
            menor = j;  
        }  
    }  
    swap(menor, i);  
}
```

## CONSIDERAÇÕES INICIAIS:

- 1) Comparações desejadas: no if
- 2) Laço externo repete (n-1) vezes, para: 0, 1, 2, ..., n-2
- 3) Laço interno repete n - (i+1) vezes, para: i+1, i+2, i+3, ..., n-1



# Exercício Resolvido (22)

- Explique porque o Algoritmo de Seleção realiza  $c(n) = \frac{n^2}{2} - \frac{n}{2}$  comparações entre registros

```
for (int i = 0; i < (n - 1); i++) {  
    int menor = i;  
    for (int j = (i + 1); j < n; j++){  
        if (array[menor] > array[j]){  
            menor = j;  
        }  
    }  
    swap(menor, i);  
}
```

i	0	1	2	...	n-2
$c(i) = (n - (i+1))$	n-1	n-2	n-3	...	1

## CONSIDERAÇÕES INICIAIS:

- 1) Comparações desejadas: no if
- 2) Laço externo repete (n-1) vezes, para: 0, 1, 2, ..., n-2
- 3) Laço interno repete  $n - (i+1)$  vezes, para: i+1, i+2, i+3, ..., n-1

# Exercício Resolvido (22)

- Explique porque o Algoritmo de Seleção realiza  $c(n) = \frac{n^2}{2} - \frac{n}{2}$  comparações entre registros

```
for (int i = 0; i < (n - 1); i++) {  
    int menor = i;  
    for (int j = (i + 1); j < n; j++){  
        if (array[menor] > array[j]){  
            menor = j;  
        }  
    }  
    swap(menor, i);  
}
```

i	0	1	2	...	n-2
c(i) = (n - (i+1))	n-1	n-2	n-3	...	1

RESPOSTA:

$$c(n) = \sum_{i=0}^{n-2} (n - i - 1)$$

# Exercício Resolvido (23)

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

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

Pior caso:  $2n$

Melhor caso (else):  $n$

# Exercício (1)

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

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

4

# Exercício (2)

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

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

4

# Exercício (3)

- 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--;  
    }  
}
```

Melhor caso:

1

Pior caso (else):

2

# Exercício (4)

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

```
...  
int i = 10, b = 10;  
while (i > 0){           4  
    b--;  
    i = i >> 1;  
}  
i = 0;  
while (i < 15){          8  
    b--;  
    i += 2;  
}
```

# Exercício (5)

- 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;  
    }  
}
```

$n * (n - 3)$



# Exercício (6)

- 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;  
    }  
}
```

$(n - 7) * n$

# Exercício (7)

- 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;  
    }  
}
```

$(n - 7) * (n - 7)$

# Exercício (8)

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

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

$\lg(n) - 1$

# Exercício (9)

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

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

$\lg(n + 1)$

# Exercício (10)

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

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

# Exercício (11)

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

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

$\lg(n) + 1$

# Exercício (12)

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

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

$\lg(n + 4)$