

RESOLVA AS EQUAÇÕES

2)

a) $\lg(2048) = 11$

b) $\lg(1024) = 10$

c) $\lg(512) = 9$

d) $\lg(256) = 8$

e) $\lg(128) = 7$

f) $\lg(64) = 6$

g) $\lg(32) = 5$

h) $\lg(16) = 4$

i) $\lg(8) = 3$

j) $\lg(4) = 2$

k) $\lg(2) = 1$

l) $\lg(1) = 0$

3)

a) $\lceil \lg(16) \rceil = 5$

b) $\lfloor \lg(16) \rfloor = 4$

c) $\lceil \lg(16) \rceil = 5$

d) $\lfloor \lg(16) \rfloor = 4$

e) $\lceil \lg(16) \rceil = 4$

f) $\lfloor \lg(16) \rfloor = 4$

g) $\lg(17) = 4.08$

h) $\lceil \lg(17) \rceil = 5$

i) $\lfloor \lg(17) \rfloor = 4$

j) $\lg(15) = 3.90$

k) $\lceil \lg(15) \rceil = 4$

l) $\lfloor \lg(15) \rfloor = 3$

4)



$f(x) = x^3 \sim \dots$

$f(x) = x^2 \sim \dots$

$f(x) = x \cdot \lg(x) \sim \dots$

$f(x) = x \sim \dots$

$f(x) = \sqrt{x} \sim \dots$

$f(x) = \lg(x) \sim \dots$

CONTAGEM DE OPERAÇÕES

1) Calcule o número de operações do código abaixo:

...

a--;

a-=3;

a=a-2;

3

b) if (a-5 > b-3) {

 a--;

 --b;

 a-=3;

} else {

 j--;

}

MELHOR CASO

3

PIOR CASO

5

c) if (a-5 < b-3 || c-1 < d-3) {

 a--;

 --b;

 a-=3;

} else {

 j--;

}

MELHOR CASO

3

PIOR CASO

7

d) for (int i=0; i<4; i++) {

 a--;

}

4

E) for (int i=0; i < N; i++) {

 a--;

 b--;

}

$\int^N N \times 2$

F) for (int i=3; i < N; i++) {

 a--;

}

$\int^{N-3} N-3$

G) int i=0, b=10

while (i < 3) {

 i++;

 b--;

}

$\int^3 3 \text{ vezes}$

H) int i=0, b=10;

do {

 i++;

 b--;

} while (i < 3);

$\int^3 3 \text{ vezes}$

I) for (int i=0; i < 3; i++) {

 for (int j=0; j < 2; j++) {

 a--;

 }

}

$\int^6 6$

2) Calcular o número de multiplicações que o código abaixo realiza:

a) $\text{for/itt } i = r; i > 0; i /= 2 \{$

$\quad a * = 2;$

$\}$

$\log_2(r) + 1$ vezes

3) Faça um método que receba um número inteiro n e retorne o número de subtrações realizadas em:

a) $3n + 2n^2$

$i = 0;$

$\text{while}(i < n) \{$

$\quad i++;$

$\quad a--; b--; c--;$

$\}$

$\text{for}(i = 0; i < n; i++) \{$

$\quad \text{for}(j = 0; j < i; j++) \{$

$\quad \quad a--; b--;$

$\quad \}$

$\}$

B)

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

C) D) E) F) Não conseguiu realizar os exercícios.

- 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?

1) A operação relevante é a comparação de $\text{min} > \text{array}[i]$ 2)

$N-1$

- Da mesma forma que calculamos o custo de um churrasco:
 - Carne: 400 gramas por pessoa (preço médio do kg R\$ 20,00 - picanha, asinha, coraçãozinho ...)
 - Cerveja: 1,2 litros por pessoa (litro R\$ 3,80)
 - Refrigerante: 1 litro por pessoa (Garrafa 2 litros R\$ 3,50)

Exercício: Monte a função de complexidade (ou custo) do nosso churrasco

RESPOSTA:

$N * 400/1000 * 20 + N * 1200/1000 * 3,8 + N * 1 * 3,50/2$

- 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?

RESPOSTA: Sim, porque precisamos comparar com todos os elementos do array para garantir que encontremos o menor.

- 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;
    }
}
```

RESPOSTA:

MELHOR CASO: 1

PIOR CASO: 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 buscado se encontra no meio do vetor.

PIOR CASO: Elemento não está no vetor, ou está na última posição procurada.

- 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;
}
```

RESPOSTA: Porque a cada repetição ele executa 3 movimentos, temos como quantidade de repetições $n-1$, essa -1 repetição equivale a -3 movimentos, resultando em $3n-3$.

- 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);
}
```

RESPOSTA:

```
int cont = 0;
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;
        }
    }
    cont += 3;
    swap(menor, j);
}
```

- 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);  
}
```

RESPOSTA: Não compreendi

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

NOÇÕES SOBRE A NOTAÇÃO Θ

- 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--;  
    }  
}
```

RESPOSTA:

Melhor Caso: $f(n) = n$

Pior Caso: $f(n) = 2n$ Ou

seja, $\Theta(n)$.

EXERCICIOS NÃO RESOLVIDOS

Exercício (1)

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

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

$$(10-7+1) * 1 = 4$$

Exercício (2)

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

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

$$(5-2+1)*1 = 4$$

PUC Minas Virtual

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: $(N-1)*1$

PIOR CASO: $((N-1)/2)*2$

PUC Minas Virtual

Exercício (4)

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

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

12 subtrações

PUC Minas Virtual

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;  
    }  
}
```

$n * n - 3 * 1$

PUC Minas Virtual

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;  
    }  
}
```

$$n - 7 + 1 * n * 1$$

Exercício (8)

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

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

$$\lg(n)$$

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)  
    a *= 2;  
}
```

$\log(n)$

PUC Minas Virtual

Exercício (11)

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

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

$\log(n)$

PUC Minas Virtual