Exercício Resolvido (1): Resolva as Equações

a) 2^{10} =

b) $\lg(1024)$ =

c) $\lg(17)$ =

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

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

PUC Minas Virtual

a) 2*2=4	64*2=128		b)1024/2=512	32/2=16		c)17/2=8.5
4*2=8	128*2=256		512/2=256	16/2=8		8.5/2=4.25
8*2=16	256*2=512		256/2=128	8/2=4		4.25/2=2.125
16*2=32	512*2=1024		128/2=64	4/2=2		2.125/21.0625
32*2=64			64=/2=32	2/2=1		responta =4.087
			respota=10			
d) 5			e) 4			

Exercício Resolvido (2): Plote os Gráficos

a) f(n) = n^3

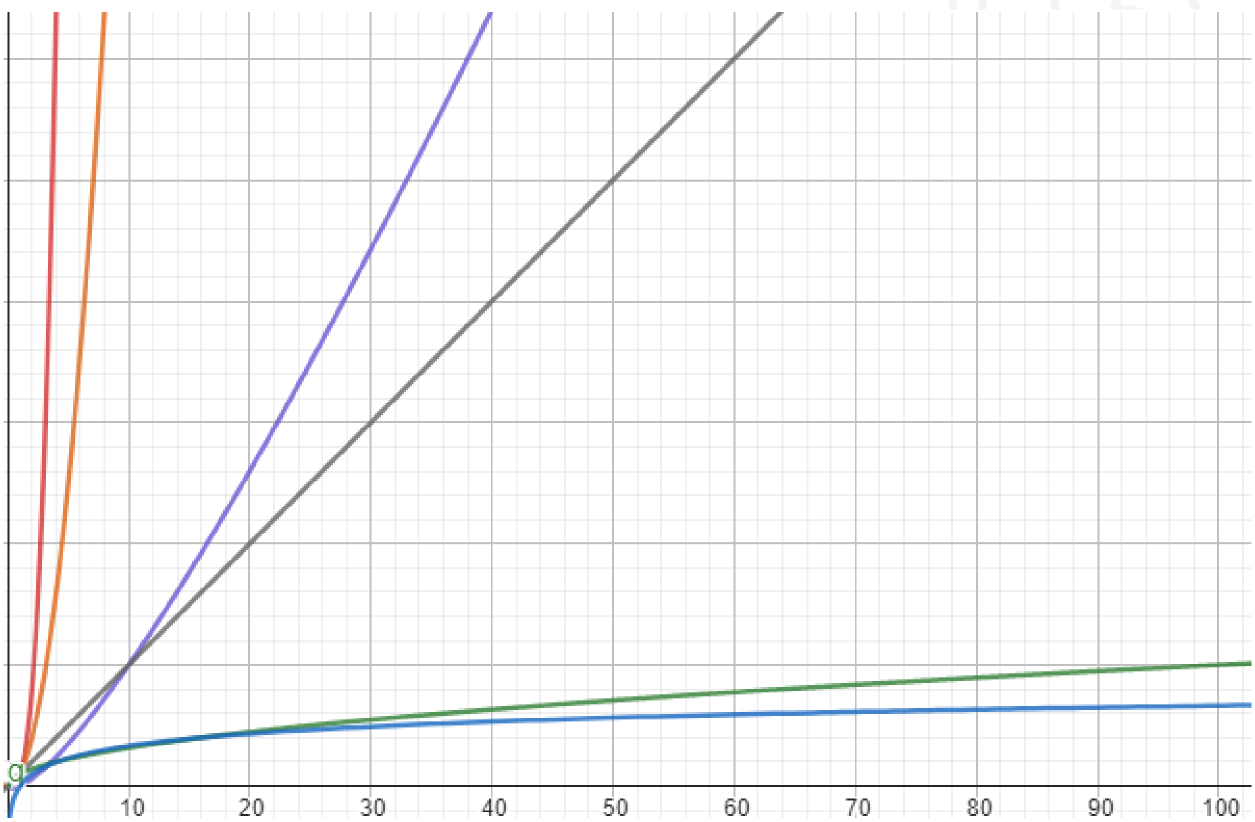
b) f(n) = n^2

c) f(n) = n x lg(n)

d) f(n) = n

e) f(n) = sqrt(n)

f) f(n) = lg(n)



Exercício Resolvido (3)

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

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

Pior caso 5, $\Theta(1)$
Melhor 3, $\Theta(1)$ subtrações

Exercício Resolvido (4)

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

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

O número de subtrações é 2n, $\Theta(n)$

Exercício Resolvido (5)

- 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--;
                b--;
                c--;
        }
}
```

O número de subtrações será $3n^2$, $\Theta(n^2)$

Exercício Resolvido (6)

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

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

O número de subtrações sera $\lfloor(\lg(n))\rfloor + 1$, $\Theta(\lg n)$

Exercício Resolvido (7): Pesquisa Sequencial

- 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;
        }
}
```

Este algoritmo é ótimo?

melhor caso de comparações entre elementos é
Melhor caso: 1 se estiver na primeira posição
Pior: n se tiver na última posição

Exercício Resolvido (8)

- Um aluno deve procurar um valor em um *array* de números reais. Ele tem duas alternativas. Primeiro, executar uma pesquisa sequencial. Segundo, ordenar o *array* e, em seguida, aplicar uma pesquisa binária. O que fazer?

A melhor opção para o aluno é executar a pesquisa sequencial tem custo $\Theta(n)$ enquanto ordenar o array e realizar a busca binaria tem custo $\Theta(n * \lg n)$ e $\Theta(\lg n)$

Exercício Resolvido (9)

Responda se as afirmações são verdadeiras ou falsas:

a) $3n^2 + 5n + 1$ é $O(n)$:

b) $3n^2 + 5n + 1$ é $O(n^2)$:

c) $3n^2 + 5n + 1$ é $O(n^3)$:

d) $3n^2 + 5n + 1$ é $\Omega(n)$:

e) $3n^2 + 5n + 1$ é $\Omega(n^2)$:

f) $3n^2 + 5n + 1$ é $\Omega(n^3)$:

g) $3n^2 + 5n + 1$ é $\Theta(n)$:

h) $3n^2 + 5n + 1$ é $\Theta(n^2)$:

i) $3n^2 + 5n + 1$ é $\Theta(n^3)$:

a) falsa
b) verdadeira
c) verdadeira
d) verdadeira
e) verdadeira
f) falsa
g) falsa
h) verdadeira
i) falsa

Exercício Resolvido (10)

- Sabendo que o Algoritmo de Seleção faz $\Theta(n^2)$ comparações entre registros, quantas dessas comparações temos no código abaixo? Justifique

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

O for roda n vezes
Algoritmo de Seleção $\Theta(n^2)$
Logo = n * $\Theta(n^2)$= $\Theta(n^3)$

Exercício Resolvido (11)

- Dado $f(n) = 3n^2 - 5n - 9$, $g(n) = n.lg(n)$, $l(n) = n.lg^2(n)$ e $h(n) = 99n^8$, qual é a ordem de complexidade das operações abaixo (use a notação Θ):
  a) h(n) + g(n) - f(n)
  b) $\Theta(h(n)) + \Theta(g(n)) - \Theta(f(n))$
  c) f(n) *x* g(n)
  d) g(n) *x* l(n) + h(n)
  e) f(n) *x* g(n) *x* l(n)
  f) $\Theta(\Theta(\Theta(\Theta(f(n)))))$

a) $[99^8] + [n*lg(n)]-[3n^2 - 5n - 9] = \Theta(n^8)$
b) $\Theta(n^8) + \Theta(n * lg(n)) - \Theta(n^2) = \Theta(n^8)$
c) $\Theta(n^2) * \Theta(n * lg(n)) = \Theta(n^3 * lg(n))$
d) $\Theta(n * lg(n)) * \Theta(n * lg^2(n)) + \Theta(n^8) = \Theta(n^8)$
e) $\Theta(n^2) * \Theta(n * lg(n)) * \Theta(n * lg^2(n)) = \Theta(n^4 * lg^3(n))$
f) $\Theta(n)$

Exercício Resolvido (12)

Dada a definição da notação O:

a) Mostre os valores de c e m tal que, para $n \geq m$, $|3n^2 + 5n + 1| \leq c \times |n^2|$, provando que $3n^2 + 5n + 1$ é $O(n^2)$

b) Mostre os valores de c e m tal que, para $n \geq m$, $|3n^2 + 5n + 1| \leq c \times |n^3|$, provando que $3n^2 + 5n + 1$ é $O(n^3)$

c) Prove que $3n^2 + 5n + 1$ **não é** $O(n)$

Exercício Resolvido (13)

- Apresente a função e a ordem de complexidade para o **número de comparações de registros** no **pior** e **melhor** caso

```
void imprimirMaxMin(int [] array, int n){
        int max, min;
        if (array[0] > array[1]){
                max = array[0];
                min = array[1];
        } else {
                max = array[1];
                min = array[0];
        }
        for (int i = 2; i < n; i++){
                if (array[i] > max){
                        max = array[i];
                } else if (array[i] < min){
                        min = array[i];
}       }       }
```


Função: Melhor: 1+(n-2) Pior: 1+2(n-2)

Ordem: todos os casos O(n), Ω(n) e **Θ**(n).

Exercício Resolvido (14)

Apresente a função e a ordem de complexidade para o número de movimentações de registros no pior e melhor caso

```
void imprimirMaxMin(int [] array, int n){
    int max, min;
    if (array[0] > array[1]){
        max = array[0];
        min = array[1];
    } else {
        max = array[1];
        min = array[0];
    }
    for (int i = 2; i < n; i++){
        if (array[i] > max){
            max = array[i];
        } else if (array[i] < min){
            min = array[i];
}       }       }
```

Função: Melhor: 2 Pior: 2+(n-2)

Ordem: Melhor: O(1), Ω(1) e **Θ**(1) Pior: O(n), Ω(n) e **Θ**(n).

Exercício Resolvido (15)

- Apresente a função e a ordem de complexidade para o número de subtrações para o pior e melhor caso

```
i = 0;
while (i < n) {
    i++;    a--;
}
if (b > c) {
    i--;
} else {
    i--;
    a--;
}
```

Função: Melhor: n+1 Pior: n+2

Ordem: todos os casos O(n), Ω(n) e **Θ**(n).

Exercício Resolvido (16)

- Apresente a função e a ordem de complexidade para o **número de subtrações** para o **pior** e **melhor** caso

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

Função: todos os casos (2n+1)n
Ordem: todos os casos $O(n^2)$, $\Omega(n^2)$ e $\boldsymbol{\Theta}(n^2)$.

Exercício Resolvido (17)

- Apresente a função e a ordem de complexidade para o **número de subtrações** para o **pior** e **melhor** caso

```
for (i = 0; i < n; i++) {
        for (j = 1; j <= n; j *= 2) {
                b--;
        }
}
```

Função: todos os casos $(\lfloor lg(n) \rfloor+1)*n=n*\lfloor lg(n) \rfloor+n$
Ordem: todos os casos $O(n * \lg(n))$, $\Omega(n * \lg(n))$ e $\boldsymbol{\Theta}(n * \lg(n))$.

Exercício Resolvido (18)

- Apresente o tipo de crescimento que melhor caracteriza as funções abaixo (Khan Academy, adaptado)

	Constante	Linear	Polinomial	Exponencial
3n		X		
1	X			
(3/2)n		X		
$2n^3$			X	
2^n				X
$3n^2$			X	
1000	X			
$(3/2)^n$				X

Exercício Resolvido (19)

- Classifique as funções $f_1(n) = n^2$, $f_2(n) = n$, $f_3(n) = 2^n$, $f_4(n) = (3/2)^n$, $f_5(n) = n^3$ e $f_6(n) = 1$ de acordo com o crescimento, do mais lento para o mais rápido (Khan Academy, adaptado)

$$f_6(n) = 1 < f_2(n) = n < f_1(n) = \mathrm{n}^2 < f_5(n) = \mathrm{n}^3 < f_4(n) = (3/2)^n < f_3(n) = 2^{\mathrm{n}}$$

Exercício Resolvido (20)

- Classifique as funções $f_1(n) = n.\log_6(n)$, $f_2(n) = \lg(n)$, $f_3(n) = \log_8(n)$, $f_4(n) = 8n^2$, $f_5(n) = n.\lg(n)$, $f_6(n) = 64$, $f_7(n) = 6n^3$, $f_8(n) = 8^{2n}$ e $f_9(n) = 4n$ de acordo com o crescimento, do mais lento para o mais rápido (Khan Academy, adaptado)

$$f_6(n) = 64 < f_3(n) = log_8(n) < f_2(n) = \lg(n) < f_9(n) = 4\mathrm{n} < f_1(n) = \mathrm{n} * log_6(n) <$$
$$f_5(n) = \mathrm{n} * \lg(\mathrm{n}) < f_4(n) = 8\mathrm{n}^2 < f_7(n) = 6n^3 < f_8(n) = 8^{2\mathrm{n}}$$

Exercício Resolvido (21)

- Faça a correspondência entre cada função f(n) com sua g(n) equivalente, em termos de O. Essa correspondência acontece quando f(n) = O(g(n)) (Khan Academy, adaptado)

f(n)	**g(n)**
$n + 30$	n^4
$n^2 + 2n - 10$	$3n - 1$
$n^3 \times 3n$	$\lg(2n)$
$\lg(n)$	$n^2 + 3n$

Exercício Resolvido (21)

- Faça a correspondência entre cada função f(n) com sua g(n) equivalente, em termos de O. Essa correspondência acontece quando f(n) = O(g(n)) (Khan Academy, adaptado)

f(n)	**g(n)**
$n + 30$	n^4
$n^2 + 2n - 10$	$3n - 1$
$n^3 \times 3n$	$\lg(2n)$
$\lg(n)$	$n^2 + 3n$

Exercício (1)

- Encontre o maior e menor valores em um *array* de inteiros e, em seguida, encontre a função de complexidade de tempo para sua solução

```
int menor=array[0];
int maior=array[0];
for(int i=0;i<array.length;i++){
    if(menor>array[i]){
        menor=array[i];
    }if(maior<array[i]){
        maior=array[i];
    }
}
```

Função de complexidade de tempo: O(n)

Exercício (2)

- Considerando o problema de encontrar o maior e menor valores em um *array*, veja os quatro códigos propostos e analisados no livro do Ziviani

Exercício (3)

- Preencha verdadeiro ou falso na tabela abaixo:

	$\Theta(1)$	$\Theta(\lg n)$	$\Theta(n)$	$\Theta(n.\lg(n))$	$\Theta(n^2)$	$\Theta(n^3)$	$\Theta(n^5)$	$\Theta(n^{20})$
$f(n) = \lg(n)$								
$f(n) = n \,.\, \lg(n)$								
$f(n) = 5n + 1$								
$f(n) = 7n^5 - 3n^2$								
$f(n) = 99n^3 - 1000n^2$								
$f(n) = n^5 - 99999n^4$								

Exercício (4)

- Preencha verdadeiro ou falso na tabela abaixo:

	$O(1)$	$O(\lg n)$	$O(n)$	$O(n.\lg(n))$	$O(n^2)$	$O(n^3)$	$O(n^5)$	$O(n^{20})$
$f(n) = \lg(n)$								
$f(n) = n \,.\, \lg(n)$								
$f(n) = 5n + 1$								
$f(n) = 7n^5 - 3n^2$								
$f(n) = 99n^3 - 1000n^2$								
$f(n) = n^5 - 99999n^4$								

Exercício (5)

- Preencha verdadeiro ou falso na tabela abaixo:

	$\Omega(1)$	$\Omega(\lg n)$	$\Omega(n)$	$\Omega(n.\lg(n))$	$\Omega(n^2)$	$\Omega(n^3)$	$\Omega(n^5)$	$\Omega(n^{20})$
$f(n) = \lg(n)$								
$f(n) = n \,.\, \lg(n)$								
$f(n) = 5n + 1$								
$f(n) = 7n^5 - 3n^2$								
$f(n) = 99n^3 - 1000n^2$								
$f(n) = n^5 - 99999n^4$								

Exercício (6)

- Dada a definição da notação Ω:

   a) Mostre os valores de c e m tal que, para $n \geq m$, $|g(n)| \geq c \times |f(n)|$, provando que $3n^2 + 5n + 1$ é $\Omega(n^2)$

   b) Mostre os valores de c e m tal que, para $n \geq m$, $|g(n)| \geq c \times |f(n)|$, provando que $3n^2 + 5n + 1$ é $\Omega(n)$

   c) Prove que $3n^2 + 5n + 1$ **não é** $\Omega(n^3)$

Exercício (7)

- Dada a definição da notação Θ:

   a) Mostre um valor para c_1, c_2 e m tal que, para $n \geq m$, $c_1 \times |f(n)| \leq |g(n)| \leq c_2 \times |f(n)|$, provando que $3n^2 + 5n +1$ é $\Theta(n^2)$

   b) Prove que $3n^2 + 5n +1$ **não** é $\Theta(n)$

   c) Prove que $3n^2 + 5n +1$ **não** é $\Theta(n^3)$

Exercício (8)

- Suponha um sistema de monitoramento contendo os métodos telefone, luz, alarme, sensor e câmera, apresente a função e ordem de complexidade para o **pior** e **melhor** caso: (a) **método alarme**; (b) **outros métodos**.

```
void sistemaMonitoramento() {
        alarme(((telefone() == true && luz() == true)) ? 0 : 1);
        for (int i = 2; i < n; i++){
                if (sensor(i- 2) == true){
                        alarme (i - 2);
                } else if (camera(i- 2) == true){
                        alarme (i - 2 + n);
} } }
```

Alarme: Melhor caso O(1) Pior caso O(n)
Para os outros métodos: Melhor caso O(1) Pior caso O(n)

Exercício (10)

- Anteriormente, verificamos que quando desejamos pesquisar a existência de ***um*** elemento em um *array* de números reais é adequado executar uma pesquisa sequencial cujo custo é Θ(n). Nesse caso, o custo de ordenar o *array* e, em seguida, aplicar uma pesquisa binária é mais elevado, Θ(n *x* lg(n)) + Θ(lg(n)) = Θ(n *x* lg(n)). Agora, supondo que desejamos efetuar ***n*** pesquisas, responda qual das duas soluções é mais eficiente

Se o número de pesquisas for baixo o melhor é a pesquisa sequencial que o custo de pesquisa seria $\Theta(n^2)$, caso o número de pesquisas for um número alto, é mais eficiente ordenar o array primeiro e depois fazer uma pesquisa binaria