Exercício Resolvido (1): Resolva as Equações

a) 2^0 =

b) 2^1 =

c) 2^2 =

d) 2^3 =

e) 2^4 =

f) 2^5 =

g) 2^6 =

h) 2^7 =

i) 2^8 =

j) 2^9 =

k) 2^{10} =

l) 2^{11} =

	exercicio	resposta
A	2^0	1
B	2^1	2
C	2^2	4
D	2^3	8
E	2^4	16
F	2^5	32
G	2^6	64
H	2^7	128
I	2^8	256
J	2^9	512
K	2^10	1024
L	2^11	2048

Exercício Resolvido (2): Resolva as Equações

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) =

	exercicio	resposta
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

Exercício Resolvido (3): Resolva as Equações

a) $\lceil 4{,}01 \rceil =$ d) $\lfloor 4{,}99 \rfloor =$ g) $\lg(17) =$ j) $\lg(15) =$

b) $\lfloor 4{,}01 \rfloor =$ e) $\lceil \lg(16) \rceil =$ h) $\lceil \lg(17) \rceil =$ k) $\lceil \lg(15) \rceil =$

c) $\lceil 4{,}99 \rceil =$ f) $\lfloor \lg(16) \rfloor =$ i) $\lfloor \lg(17) \rfloor =$ l) $\lfloor \lg(15) \rfloor =$

	exercicio	resposta
A	⌈ 4.01 ⌉	5
B	⌊4.01⌋	4
C	⌈4.99⌉	5
D	⌊4.99⌋	4
E	⌈lg(16)⌉	4
F	⌊lg(16)⌋	4
G	lg(17)	4.087
H	⌈lg(17)⌉	5
I	⌊lg(17)⌋	4
J	lg(15)	3.907
K	⌈lg(15)⌉	4
L	⌊lg(15)⌋	3

Exercício Resolvido (4): Plote os Gráficos

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

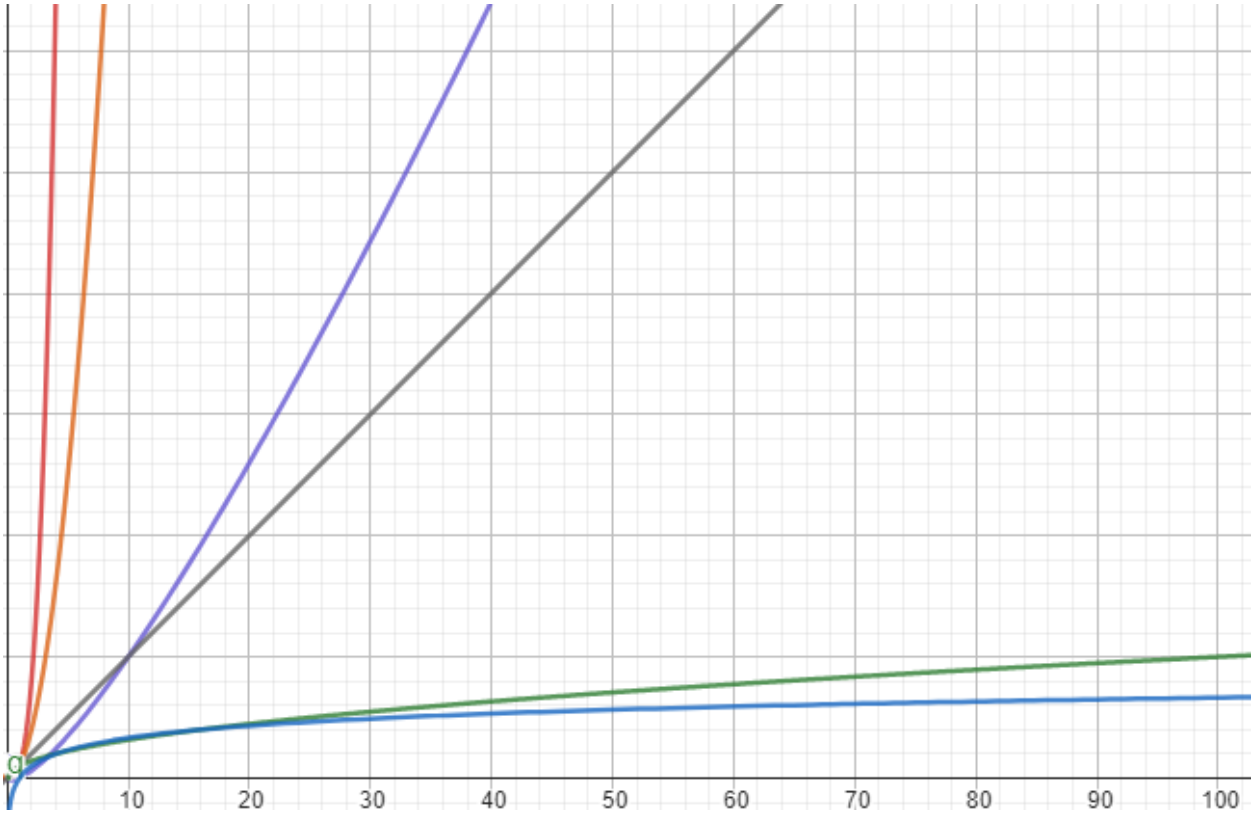
b) $f(n) = n^2$

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

d) $f(n) = n$

e) $f(n) = sqrt(n)$

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



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

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

3 subtrações

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 5 e no melhor 3 subtrações

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--;
}
```

Pior caso 7 e no melhor 5 subtrações

Exercício Resolvido (9)

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

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

4 subtrações

Exercício Resolvido (10)

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

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

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

O numero de subtrações vai ser igual 2n

Exercício Resolvido (11)

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

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

O numero de subtrações vai ser igual a (n-3) vezes

Exercício Resolvido (12)

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

```
...
int i = 0, b = 10;

while (i < 3){
        i++;
        b--;
}
```

O numero de subtrações vai ser igual a 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);
```

O numero de subtrações vai ser 3

- 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--;
        }
}
```

O numero de subtrações vai ser 3*2*1 que é igual a 6

Exercício Resolvido (15)

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

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

O numero de multiplicações vai ser o piso de lg(n)+1

Como Calcular a Complexidade de um Algoritmo

- 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

f(n)=20(n*0,4)+3,8(n*1,2)+n(3,5/2)

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];
        }
}
```

```
Run | Debug
public static void main (String[] args){
    int[] array;
    array = new int[10];

    int menor=array[0];
    for(int i=1;i<array.length;i++){
        if(menor>array[i]){
            menor=array[i];
        }
    }
}
```

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?

2º) Quantas vezes ela será executada?

1- O if que compara os elementos do array
2- N – 1 vezes

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

Todos os casos pois precisamos comparar todos os elementos do array

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

Pois precisamos testar todos os elementos do array

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 1 vez e no pior n vezes

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


Melhor caso vai ser 1 comparacao e a pior vai ser 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;
}
```


Pois ele seleciona o numero da posição do vetor e seleciona o menor numero, e depois para realizar o swap ele seleciona um numero para a variável 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);
}
```

Devemos criar uma variável int igual a 0 e abaixo do swap colocamos para a variável ela mesma mais 3

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--;
        }
}
```

PUC Minas Virtual

No melhor caso é n ou **Θ**(n)
No pior caso é 2n ou **Θ**(n)

Exercício (1)

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

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

4 subtrações

Exercício (2)

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

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

4*2 = 8

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--;
        }
}
```

9 subtrações

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

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;
        }
}
```

O numero de multiplicações é 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;
        }
}
```

O numero de multiplicações é n * (n - 7)

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;
        }
}
```

O numero de multiplicações é (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;
}
```

O numero de multiplicações é 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;
}
```

o numero de multiplicações é 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;
}
```

O numero de multiplicações é lg(n)

Exercício (11)

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

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

O numero de multiplicações é 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;
}
```

O numero de multiplicações é lg(n+4)+1