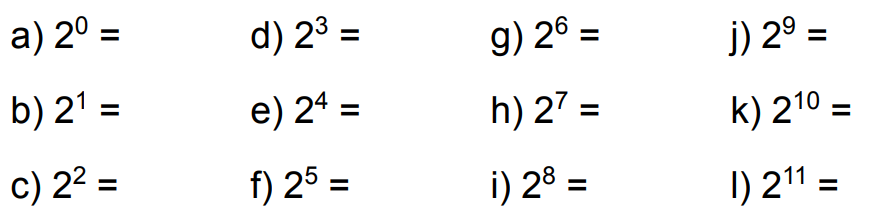
**• Resolva as equações abaixo:**



A) 1

B) 2

C) 4

D) 8

E) 16

F) 32

G) 64

H) 128

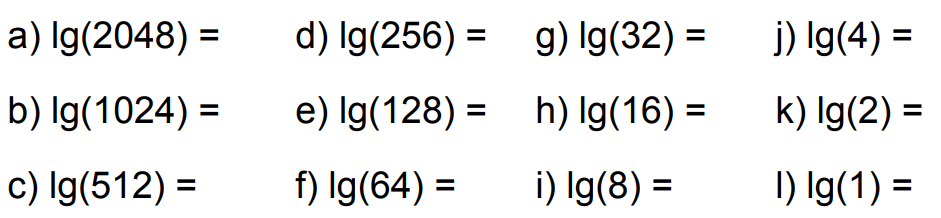
I) 256

J) 512

K) 1024

l) 2048

• Resolva as equações abaixo:



A) 11

B) 10

C) 9

D) 8

E) 7

F) 6

G) 5

H) 4

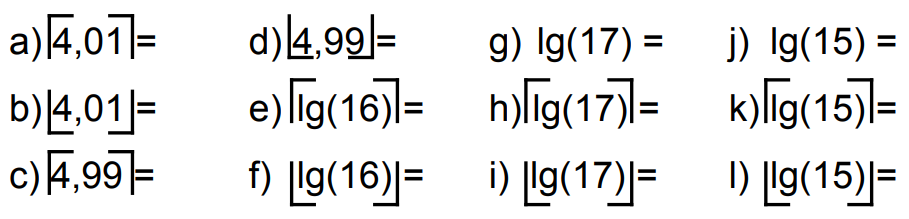
I) 3

J) 2

K) 1

l) 0

**• Resolva as equações abaixo:**



A) 5

B) 4

C) 5

D) 4

E) 4

F) 4

G) 4.0875

H) 5

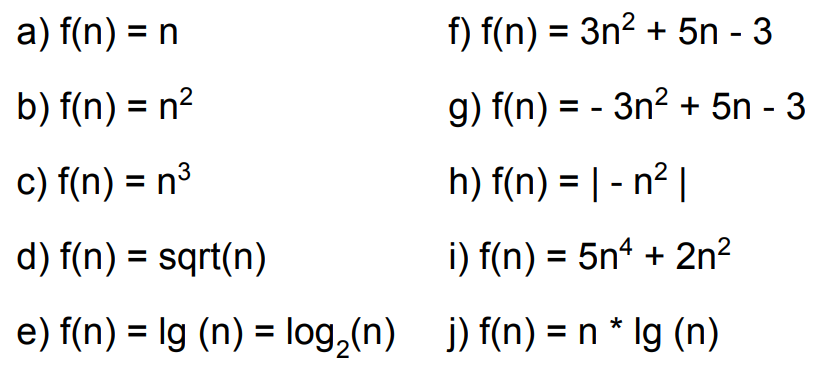
I) 4

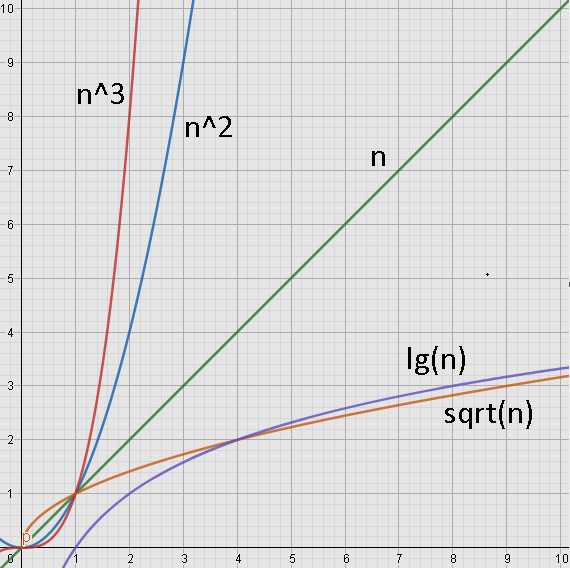
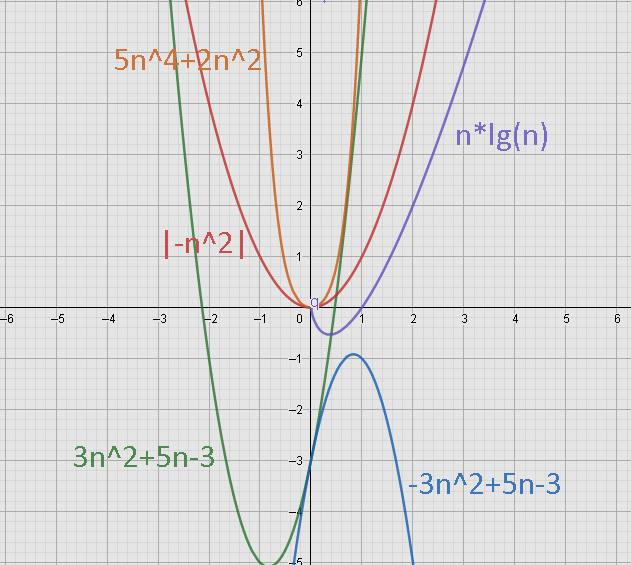
J) 3.9069

K) 4

l) 3

**• Plote um gráfico com todas as funções abaixo:**



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

a--;

a -= 3;

a = a - 2;

R: 3 subtracoes

*O(1)*

**• Calcule o número de adições que o código abaixo realiza:**

if (a + 5 < b + 3){

i++;

++b;

a += 3;

} else {

j++;

}

R = melhor caso 3, pior caso 5

*O(1)*

**• 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++;

}

R = melhor caso 5, pior caso 7

*O(1)*

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

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

a--;

}

R = ocorreram 4 subtracoes, uma para cada valor de i {0, 1, 2, 3}

*O(1)*

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

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

a--;

b--;

}

R = 2n subtrações

*O(n)*

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

int i = 0, b = 10;

while (i < 3){

i++;

b--;

}

R = 3 subtracoes, uma para cada valor de i {0, 1, 2}

*O(1)*

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

for (int i = 3; i < n; i++){

a--;

}

R = n-3 subtracoes

*O(n)*

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

int i = 10;

while (i >= 7){

i--;

}

R = 4 subtracoes

*O(1)*

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

for (int i = 5; i >= 2; i--){

a--;

}

R = 8 subtracoes

*O(1)*

**• 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--;

}

}

R = 8 subtracoes

*O(1)*

**• 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--;

}

}

R = 3\*2\*1 = 6 subtracoes

*O(1)*

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

int a = 10, b = 10, c = 10, d = 10;

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

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

a--;

b--;

c--;

d--;

}

}

R = 3\*2\*4 = 24 subtracoes

*O(1)*

**• 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--;

}

}

R = n\*n = n2 subtracoes

*O(n2)*

**• 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;

}

R = 1 + 5 = 6 subtracoes

*O(1)*

**• 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;

R = n\*(n-3) = n2 -3n multiplicacoes

*O(n2)*

**• 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;

R = (n-7)\*n = n2 -7n multiplicações

*O(n2)*

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

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

a \*= 2;

R = ⌊lg(n)⌋+1 multiplicacoes

*O(lg(n))*

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

for (int i = n+4; i > 0; i >>= 1)

a \*= 2;

R = ⌊lg(n+4)⌋+1 multiplicacoes

*O(lg(n))*

**• 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;

R = (n-7)2 multiplicacoes

*O(n2)*

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

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

a \*= 2;

R = ⌊lg(n)⌋+1 multiplicacoes

*O(lg(n))*

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

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

a \*= 2;

R = ⌊lg(n+1)⌋+1 multiplicacoes

*O(lg(n))*

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

for (int i = n+1; i > 1; i /= 2)

a \*= 2;

R = ⌊lg(n+1)⌋ multiplicações

*O(lg(n))*

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

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

a \*= 2;

R = 2⌊lg(n)⌋ multiplicações

*O(lg(n))*

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

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

a \*= 2;

R = 2⌊lg(n)⌋+1 multiplicacoes

*O(lg(n))*

**• Faça um método que receba um número inteiro n e efetue o número de subtrações pedido em:**

**a) 3n+2n2**

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

a--;

b--;

c--;

}

for(int i =0; i<n; i++)

for(int j =n; j>0; j--)

a--;

*O(n2)*

**b) 5n+4n3**

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

a--;

b--;

c--;

d--;

e--;

}

for(int i = n; i>0; i--)

for(int j = n; j>0; j--)

for(intk = n; k>0; k--)

a--;

*O(n3)*

**c)lg(n) + n**

for(int i =0; i<n; i++)

a--;

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

a--;

*O(lg(n))*

**d)2n3 +5**

a--;

b--;

c--;

d--;

e--;

for(int i = 0; i<n; i++)

for(int j = 0; j<n; j++)

for(int k = n; k>0; k--)

a--;

*O(n3)*

**e)9n4 + 4n2 + n/2**

for(int i = n; i>0; i--){

for(int j = n; j>0; j--){

a--; b--; c--; e--;

for(int k = n; k>0; k--){

for(int l = n; l>0; l--){

a--; b--; c--; d--; e--;

}}}}

for(int i = o; i<n/2; i++)

a--;

*O(n4)*

**f)lg(n) + 5lg(n)**

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

a--;

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

a--; b--; c--; d--; e--;

}

*O(lg(n))*

**• 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];

}

}

*O(n)*

**1º) Qual é a operação relevante?**

R = Comparacao entre elementos do array

**2º) Quantas vezes ela será executada?**

R = Se tiver N elementos, n-1 vezes

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

R = Neste exemplo todos

**• Qual é o número total de comparações (i < n) no código abaixo?**

int min = array[0];

for (int i = 1; i < n; i++){

if (min > array[i]){

min = array[i];

}

}

R = n – 1 + 1 = n

*O(n)*

**• Qual é o número total de comparações no código abaixo?**

int min = array[0];

for (int i = 1; i < n; i++){

if (min > array[i]){

min = array[i];

}

}

R = n + (n-1) = 2n - 1

*O(n)*

**• Pergunta 1: Qual é a diferença entre as notações O, Ω e 𝚯? Pesquise!!!**

As notações apresentadas são respectivamente chamadas de

**O** = Big O

**Ω** = Big Omega

**𝚯** = Big Theta

Elas servem para mostrar os diferentes grais de complexidade de crescimento de um algoritimo

Mas todo algoritimo tem sua dificuldade diferenciada entre pior caso, melhor caso e pior caso, que e exatamente o que elas representam, sendo o Big O para o pior, Big Omega para o melhor e o Big Theta para representar o caso médio

**• Pergunta 2: Qual é a notação O, Ω e 𝚯 para todos os exercícios feitos nesta Unidade 1b?**

Para uma melhor facilidade de visualização o grau de complexidade foi previamente colocado nos próprios exercicios