Unidade III:

Fundamentos de Análise de Algoritmos

Exercício Resolvido (1):

Resolva as equações abaixo:

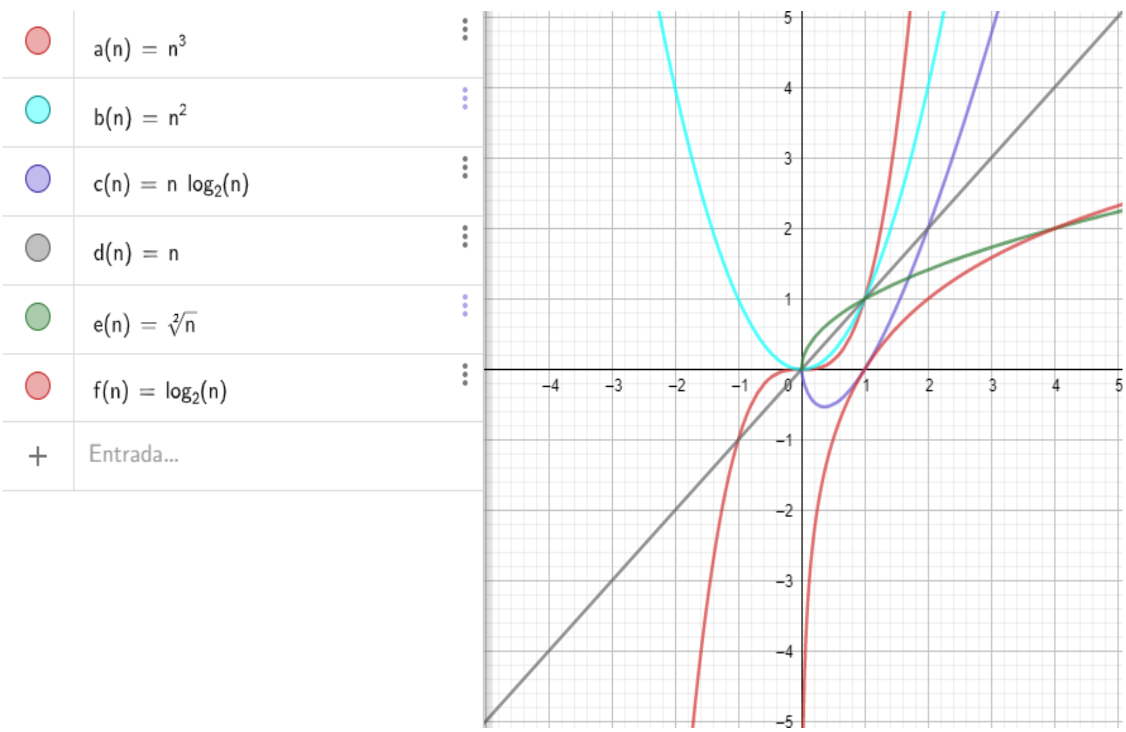
1. 2^{10} = 1024
2. lg(1024) = 10
3. lg(17) = 4.087463
4. teto lg(17) = 5
5. piso lg(17) = 4

Exercício Resolvido (2):

Plote um gráfico com todas as funções abaixo:

a) f(n) = n^3
b) f(n) = n^2
c) f(n) = n*lg(n)
d) f(n) = n
e) f(n) = sqrt(n)
f) f(n) = lg(n)

Resposta:



Exercício Resolvido (3):

Calcule o número de subtrações que o código abaixo realiza:

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

Resposta:

Melhor caso: f(n) = n (O(n), Ω(n) e Θ(n))

Pior caso: f(n) = 2n (O(2n), Ω(2n) e Θ(2n))

Exercício Resolvido (4):

Calcule o número de subtrações que o código abaixo realiza:

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

Resposta:

Haverá n-3 subtrações, logo Θ(n-3)

Exercício Resolvido (5):

Calcule o número de multiplicações que o código abaixo realiza:

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

Resposta:

Quando n for potência de 2 haverá lg(n) + 1 multiplicações.

Quando n não for potência de 2 haverá piso (lg(n)) + 1 multiplicações.

Exercício Resolvido (6):

Outra forma de compreender o código anterior é executando o mesmo:

```
class Log {
  public static void main (String[] args) {
    int[] n = {4,5,6,7,8,9,10,11,12,13,14,15,16,17,31,32,33,63,64,65};
    int cont;

    for(int k = 0; k < n.length; k++){
      System.out.print("\n[n = " + n[k] + "] => ");
      cont = 0;
      for(int i = n[k]; i > 0; i /= 2){
        System.out.print(" " + i);
        cont++;
      }
      System.out.print(" (" + cont + " vezes)");
    }
    System.out.print("\n");
  }
}
```

Resposta:

Código pode ser encontrado na pasta ***exercicios_praticos***

Exercício Resolvido (7 - 8):

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];
    }
}
```

1. Qual é a operação relevante?

   R. Comparação de arrays

2. Quantas vezes ela será executada?

   R. Será executada n-1 vezes.

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

   R. Para os três casos.

4, o nosso algoritmo é ótimo? Por que?

   R. Sim, porque o melhor caso é igual ao pior e o caso médio.

Exercício:

Monte a função de complexidade (ou custo) do nosso 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)

Resposta:

f(n) = 4/10 * 20 * n + 1.2 * 3.8 * n + 3.5/2 * n
f(n) = n(4/10 * 20 + 1.2 * 3.8 + 3.5/2)
f(n) = n * 14.31

Exercício Resolvido (9):

Responda:

```
boolean resp = false;

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

a) Qual é a operação relevante?
   R. Comparação entre elementos do array
b) Quantas vezes ela será executada?
   R. Melhor caso - f(n) = 1
   Pior caso - f(n) = n
   Caso médio - f(n) = (n+1)/2
c) O nosso algoritmo é o melhor?
   R. Depende, caso saibamos que o array está ordenado, não, neste caso é possível fazer a pesquisa binaria.
   Caso contrário, sim, pois devemos percorrer todos elementos até acharmos o elemento desejado.

Exercício Resolvido (10):

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?

Resposta:

O aluno deve escolher a primeira opção, pois ela tem o consumo de O(n), enquanto a segunda opção tem o custo de O(n*lg n) para a ordenação e O(lg n) para pesquisa binaria.

Exercício Resolvido (11):

Responda se as afirmações são verdadeiras ou falsas:

a) $3n^2 + 5n + 1$ é $O(n)$ = falsa
b) $3n^2 + 5n + 1$ é $O(n^2)$ = verdadeiro
c) $3n^2 + 5n + 1$ é $O(n^3)$ = verdadeiro
d) $3n^2 + 5n + 1$ é $\Omega(n)$ = verdadeiro
e) $3n^2 + 5n + 1$ é $\Omega(n^2)$ = verdadeiro
f) $3n^2 + 5n + 1$ é $\Omega(n^3)$ = falsa
g) $3n^2 + 5n + 1$ é $\Theta(n)$ = falsa
h) $3n^2 + 5n + 1$ é $\Theta(n^2)$ = verdadeiro
i) $3n^2 + 5n + 1$ é $\Theta(n^3)$ = falsa

Exercício Resolvido (12):

Apresente a função e a complexidade para os números de comparações e movimentações de registros para o pior e melhor caso.

```
void imprimirMaxMin( int [] array, int n){
    int maximo, minimo;

    if (array[0] > array[1]){
        maximo = array[0];    minimo = array[1];
    } else {
        maximo = array[1];    minimo = array[0];
    }

    for (int i = 2; i < n; i++){
        if (array[i] > maximo){
            maximo = array[i];
        } else if (array[i] < minimo){
            minimo = array[i];
        }
    }
}
```

Resposta:

Função de complexidade:

	Comparações	Movimentações
Melhor caso	f(n) = n-1	f(n) = 2
Pior caso	f(n) = 1+2(n-2)	f(n) = 2+(n-2)

Complexidade:

	Comparações	Movimentações
Melhor caso	$\Theta(n)$	$\Theta(1)$
Pior caso	$\Theta(n)$	$\Theta(n)$

Exercício Resolvido (13):

Apresente a função e a 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--;
}
```

Resposta:

Função de complexidade:

	Subtrações
Melhor caso	$f(n) = n + 1$
Pior caso	$f(n) = n + 2$

Complexidade:

	Subtrações
Melhor caso	$\Theta(n)$
Pior caso	$\Theta(n)$

Exercício Resolvido (14):

Apresente a função e a 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--;
}
```

Resposta:

Função de complexidade:

	Subtrações
Melhor caso	$f(n) = n(2n + 1)$
Pior caso	$f(n) = n(2n + 1)$

Complexidade:

	Subtrações
Melhor caso	$\Theta(n^2)$
Pior caso	$\Theta(n^2)$

Exercício Resolvido (15):

Apresente a função e a 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--;
    }
}
```

Resposta:

Função de complexidade:

	Subtrações
Ambos	$f(n) = n*lg(n) + n$

Complexidade:

	Subtrações
Ambos	$\Theta(n*lg(n))$

Exercício Resolvido (16):

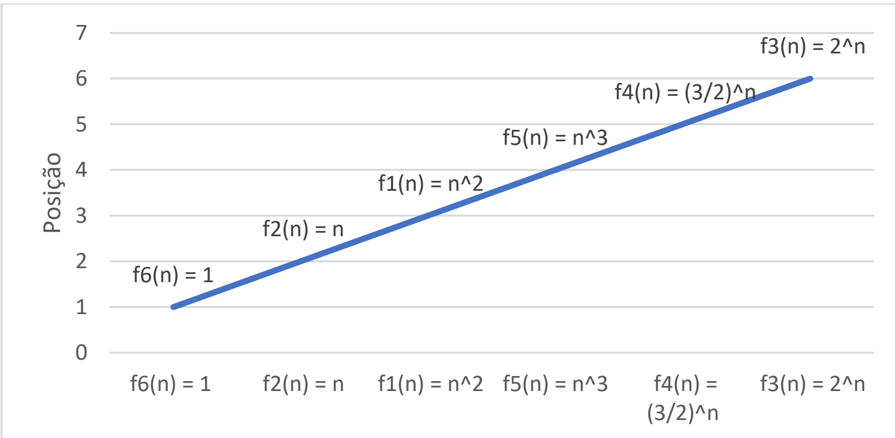
Apresente o tipo de crescimento que melhor caracteriza as funções abaixo (Khan Academy, adaptado)

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

Exercício Resolvido (17):

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).

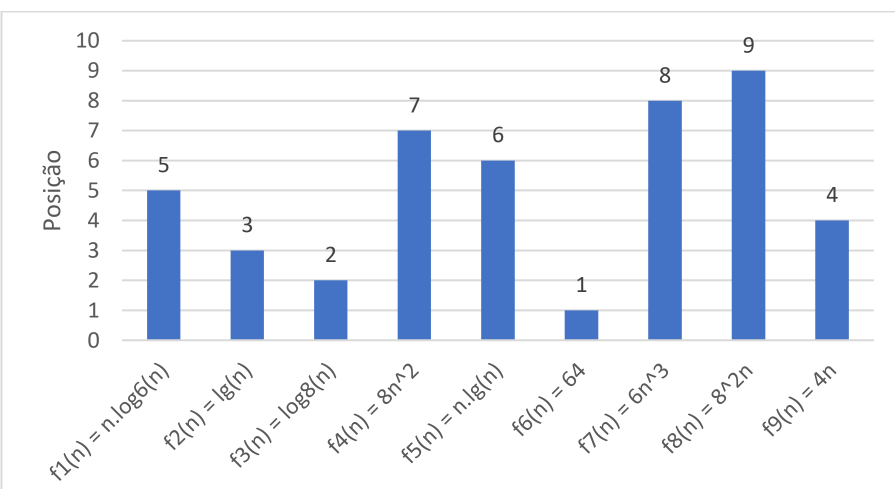
Resposta:



Exercício Resolvido (18):

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)

Resposta:



Exercício Resolvido (19):

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

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

Resposta:

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

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.

Função de complexidade:

F(n) = n-1 => O(n)

Obs.

O código, Exer_01.java se encontra na pasta ***exercícios_praticos.***

Exercício (2):

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

Exercício (3):

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**	**V**	**V**	**V**	**V**	**V**	**V**	**V**
f(n) = n * lg(n)	**F**	**F**	**F**	**V**	**V**	**V**	**V**	**V**
f(n) = 5n + 1	**F**	**F**	**V**	**V**	**V**	**V**	**V**	**V**
$f(n) = 7n^5 - 3n^2$	**F**	**F**	**F**	**F**	**F**	**F**	**V**	**V**
$f(n) = 99n^3 - 1000n^2$	**F**	**F**	**F**	**F**	**F**	**V**	**V**	**V**
$f(n) = n^5 - 99999n^4$	**F**	**F**	**F**	**F**	**F**	**F**	**V**	**V**

Exercício (4):

Preencha verdadeiro ou falso na tabela abaixo:

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

Exercício (5):

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**	**F**	**V**	**F**	**F**	**F**	**F**	**F**
$f(n) = n * \lg(n)$	**F**	**F**	**V**	**F**	**F**	**F**	**F**	**F**
$f(n) = 5n + 1$	**F**	**F**	**V**	**F**	**F**	**F**	**F**	**F**
$f(n) = 7n^5 - 3n^2$	**F**	**F**	**F**	**F**	**F**	**F**	**V**	**F**
$f(n) = 99n^3 - 1000n^2$	**F**	**F**	**F**	**F**	**F**	**V**	**F**	**F**
$f(n) = n^5 - 99999n^4$	**F**	**F**	**F**	**F**	**F**	**F**	**V**	**F**

Exercício (6):

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:

a) $f(n) + g(n) - h(n) = O(n^8)$

b) $O(f(n) + O(g(n)) - O(h(n)) = O(n^8)$

c) $f(n) \times g(n) = O(n^2 * n) = O(n^3)$

d) $g(n) \times l(n) + h(n) = O(n * n) + O(n^8) = O(n^8)$

e) $f(n) \times g(n) \times l(n) = O(n^2 * n * n^8) = O(n^{11})$

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

Exercício (7):

Dada a definição da notação O:

• Mostre um valor c e outro m tal que, para n ≥ m, |3n2+ 5n +1| ≤ c x |n2|, provando que 3n2 + 5n +1 é O(n2)

• Mostre um valor c e outro m tal que, para n ≥ m, |3n2 + 5n +1| ≤ c x |n3|, provando que 3n2 + 5n +1 é O(n3)

• Prove que 3n2 + 5n +1 não é O(n)

Exercício (8):

Dada a definição da notação Ω:

• Mostre um valor c e outro m tal que, para n ≥ m, |g(n)| ≥ c x |f(n)|, provando que 3n2 + 5n +1 é Ω(n2)

• Mostre um valor c e outro m tal que, para n ≥ m, |g(n)| ≥ c x |f(n)|, provando que 3n2 + 5n +1 é Ω(n)

• Prove que 3n2 + 5n +1 não é Ω(n3)

Exercício (9):

Dada a definição da notação Θ:

• Mostre um valor para c1, c2 e m tal que, para n ≥ m, c1 x |f(n)| ≤ |g(n)| ≤ 2x |f(n)|, provando que 3n2 + 5n +1 é Θ(n2)

• Prove que 3n2 + 5n +1 não é Θ(n)

• Prove que 3n2 + 5n +1 não é Θ(n3)

Exercício (10):

Faça um resumo sobre Teoria da Complexidade, Classes de Problemas P, NP e NP-Completo. Use LaTeX e siga o modelo de artigos da SBC (sem abstract, resumo e seções) com no máximo duas página.

Exercício (11):

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() {
        if (telefone() == true && luz() == true){
                alarme(0);
        } else {
                alarme(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);
} } }
```

Resposta:

Função

	Melhor caso	Pior caso
a	a(n) = 1	a(n) = n - 1
Telefone()	t(n) = 1	t(n) = 1
Luz()	l(n) = 0	l(n) = 1
Sensor()	s(n) = n-2	s(n) = n-2
Câmera()	c(n) = 0	c(n) = n-2

Complexidade

	Melhor caso	Pior caso
a	$\Theta(1)$	$\Theta(n)$
Telefone()	$\Theta(1)$	$\Theta(1)$
Luz()	$\Theta(0)$	$\Theta(1)$
Sensor()	$\Theta(n-2)$	$\Theta(n-2)$
Câmera()	$\Theta(0)$	$\Theta(n-2)$

Exercício (12):

Apresente um código, defina duas operações relevantes e apresente a função e a complexidade para as operações escolhidas no pior e melhor caso.

Código:

```
static void quadrado(int a, int[] array){
    int temp;
    for(int i = 1; i < a; i++)
        for(int j = 0; j < a; j++){
            if(j == array[0]){
                temp = array[0];
                temp++;
                array[0] = temp;
            } // end if
        } // end for
} // end quadrado()
```

Resposta:

Função:

Melhor e Pior caso:

COMP	$f(n) = n(n-1)$
MOV	$f(n) = 2n$

Complexidade:

Melhor e Pior caso:

$O(n^2)$, $\Theta(n^2)$, OMEGA(n^2)

Exercício (13):

No Exercício Resolvido (10), 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 $\Theta(n)$. Nesse caso, o custo de ordenar o array e, em seguida, aplicar uma pesquisa binária é mais elevado, $\Theta(n * lg(n)) + \Theta(lg(n)) = \Theta(n * lg(n))$. Agora, supondo que desejamos efetuar n pesquisas, responda qual das duas soluções é mais eficiente

Resposta:

Caso seja desejado realizar **n** pesquisas, seria melhor fazer a ordenação dos elementos do array e depois realizar a **n** pesquisas.