Unidade III:

Fundamentos de Análise de Algoritmos

**Exercício Resolvido (1):**

Resolva as equações abaixo:

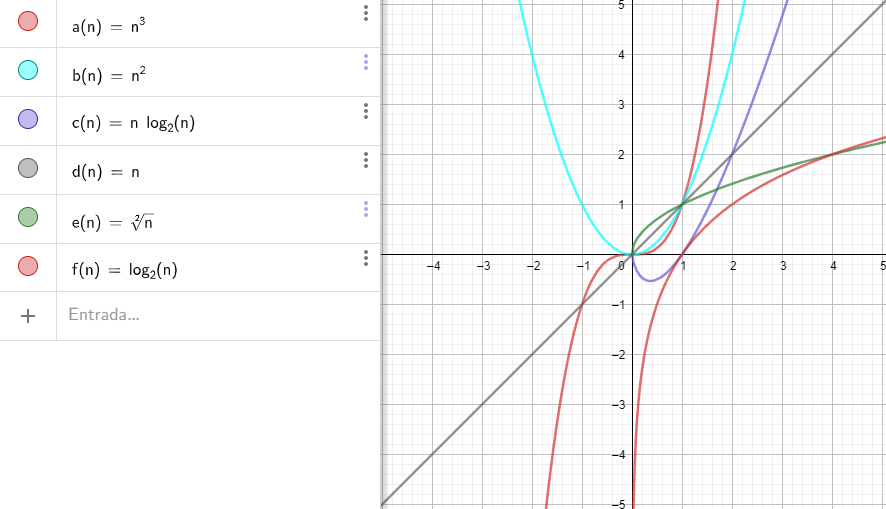
1. 210 = 1024
2. lg(1024) = 10
3. lg(17) = 4.087463
4. teto lg(17) = 5
5. piso lg(17) = 4

**Exercício Resolvido (2):**

Plote um gráfico com todas as funções abaixo:

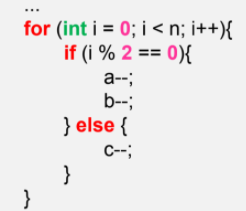
1. f(n) = n3
2. f(n) = n2
3. f(n) = n\*lg(n)
4. f(n) = n
5. f(n) = sqrt(n)
6. f(n) = lg(n)

**Resposta:**

****

**Exercício Resolvido (3):**

Calcule o número de subtrações que o código abaixo realiza:

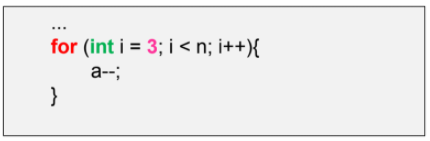
**Resposta:**

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:

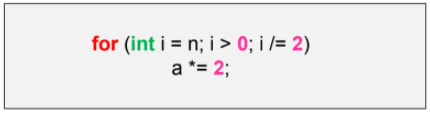
****

**Resposta:**

Haverá n-3 subtrações, logo Θ(n-3)

**Exercício Resolvido (5):**

Calcule o número de multiplicações que o código abaixo realiza:

****

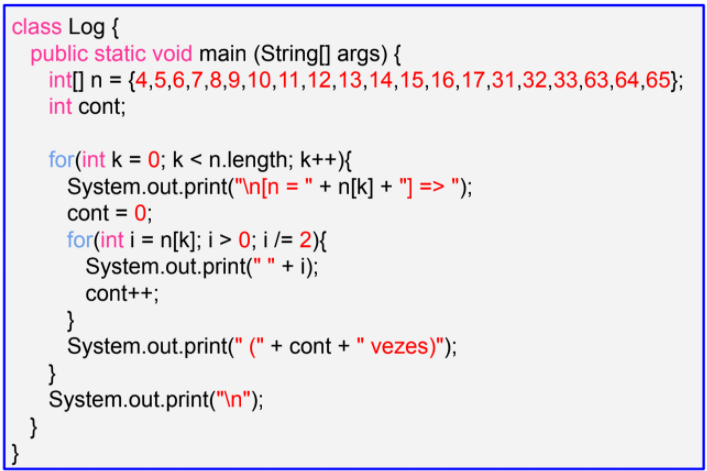
**Resposta:**

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:

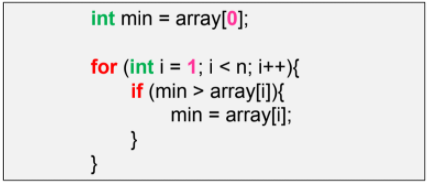
****

**Resposta:**

Código pode ser encontrado na pasta ***exercicios\_praticos***

**Exercício Resolvido (7 - 8):**

Encontre o menor valor em um array de inteiros:

1.  Qual é a operação relevante?

**R.** Comparação de arrays

1. 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)

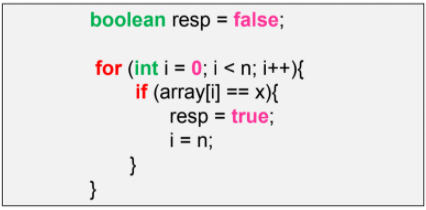
• Refrigerante: 1 litro por pessoa (Garrafa 2 litros R$ 3,50)

**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:

****

1. Qual é a operação relevante?

**R.** Comparação entre elementos do array

1. Quantas vezes ela será executada?

**R.** Melhor caso - f(n) = 1

Pior caso - f(n) = n

Caso médio - f(n) = (n+1)/2

1. 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.

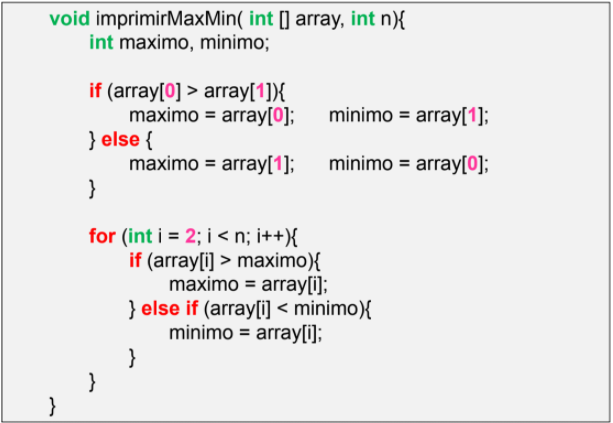
**Exercício Resolvido (11):**

Responda se as afirmações são verdadeiras ou falsas:

1. 3n2 + 5n + 1 é O(n) = falsa
2. 3n2 + 5n + 1 é O(n2) = verdadeiro
3. 3n2 + 5n + 1 é O(n3) = verdadeiro
4. 3n2 + 5n + 1 é Ω(n) = verdadeiro
5. 3n2 + 5n + 1 é Ω(n2) = verdadeiro
6. 3n2 + 5n + 1 é Ω(n3) = falsa
7. 3n2 + 5n + 1 é Θ(n) = falsa
8. 3n2 + 5n + 1 é Θ(n2) = verdadeiro
9. 3n2 + 5n + 1 é Θ(n3) = 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.



**Resposta:**

Função de complexidade:

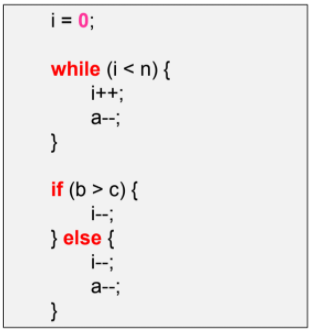
|  |  |  |
| --- | --- | --- |
|  | Comparações | Movimentações |
| Melhor caso | f(n) = n-1 | f(n) = 2 |
| Pior caso | f(n) = 1+2(n-2) | f(n) = 2+(n-2) |

Complexidade:

|  |  |  |
| --- | --- | --- |
|  | Comparações | Movimentações |
| Melhor caso | Ɵ(n) | Ɵ(1) |
| Pior caso | Ɵ(n) | Ɵ(n) |

**Exercício Resolvido (13):**

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

**Resposta:**

Função de complexidade:

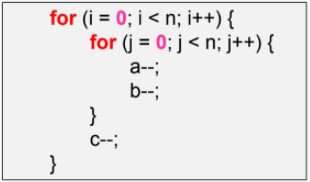
|  |  |
| --- | --- |
|  | Subtrações |
| Melhor caso | f(n) = n + 1 |
| Pior caso | f(n) = n + 2 |

|  |  |
| --- | --- |
|  | Subtrações |
| Melhor caso | Ɵ(n) |
| Pior caso | Ɵ(n) |

Complexidade:

**Exercício Resolvido (14):**

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

**Resposta:**

Função de complexidade:

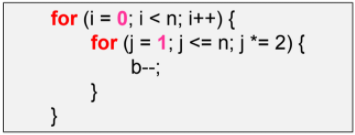
|  |  |
| --- | --- |
|  | Subtrações |
| Melhor caso | f(n) = n(2n + 1) |
| Pior caso | f(n) = n(2n + 1) |

|  |  |
| --- | --- |
|  | Subtrações |
| Melhor caso | Ɵ(n2) |
| Pior caso | Ɵ(n2) |

Complexidade:

**Exercício Resolvido (15):**

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

**Resposta:**

Função de complexidade:

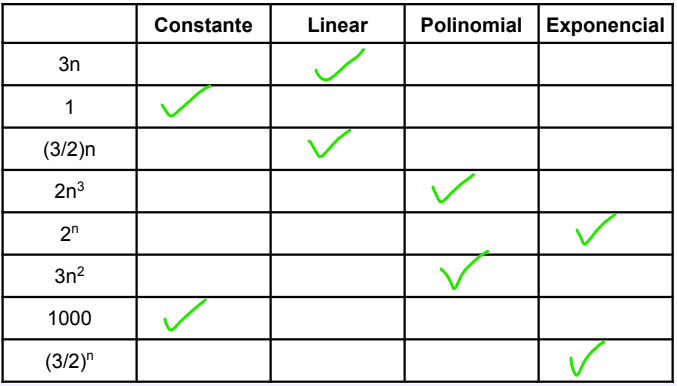
|  |  |
| --- | --- |
|  | Subtrações |
| Ambos | f(n) = n\*lg(n) + n |

|  |  |
| --- | --- |
|  | Subtrações |
| Ambos | Ɵ(n\*lg(n)) |

Complexidade:

**Exercício Resolvido (16):**

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



**Exercício Resolvido (17):**

Classifique as funções f1(n) = n2, f2(n) = n, f3(n) = 2n, f4(n) = (3/2)n, f5(n) = n3 e f6(n) = 1 de acordo com o crescimento, do mais lento para o mais rápido (Khan Academy, adaptado).

**Resposta:**

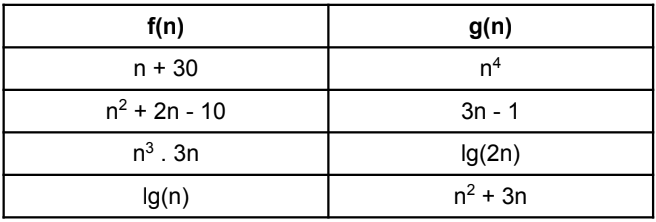
**Exercício Resolvido (18):**

Classifique as funções f1(n) = n.log6(n), f2(n) = lg(n), f3(n) = log8(n), f4(n) = 8n2, f5(n) = n.lg(n), f6(n) = 64, f7(n) = 6n3, f8(n) = 82n e f9(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)



**Resposta:**

|  |  |
| --- | --- |
| **f(n)** | **g(n)** |
| n+30 | 3n - 1 |
| n2 + 2n - 10 | n2 + 3n |
| n3 \* 3n | n4 |
| 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(n2) | O(n3) | O(n5) | O(n20) |
| 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) = 7n5 - 3n2 | **F** | **F** | **F** | **F** | **F** | **F** | **V** | **V** |
| f(n) = 99n3 - 1000n2 | **F** | **F** | **F** | **F** | **F** | **V** | **V** | **V** |
| f(n) = n5 - 99999n4 | **F** | **F** | **F** | **F** | **F** | **F** | **V** | **V** |

**Exercício (4):**

Preencha verdadeiro ou falso na tabela abaixo:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Ω(1) | Ω(lg n) | Ω(n) | Ω(n\*lg(n))) | Ω(n2) | Ω(n3) | Ω(n5) | Ω(n20) |
| 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) = 7n5 - 3n2 | **V** | **V** | **V** | **V** | **V** | **V** | **V** | **F** |
| f(n) = 99n3 - 1000n2 | **V** | **V** | **V** | **V** | **V** | **V** | **F** | **F** |
| f(n) = n5 - 99999n4 | **V** | **V** | **V** | **V** | **V** | **V** | **V** | **F** |

**Exercício (5):**

Preencha verdadeiro ou falso na tabela abaixo:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Θ(1) | Θ(lg n) | Θ(n) | Θ(n\*lg(n))) | Θ(n2) | Θ(n3) | Θ(n5) | Θ(n20) |
| 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) = 7n5 - 3n2 | **F** | **F** | **F** | **F** | **F** | **F** | **V** | **F** |
| f(n) = 99n3 - 1000n2 | **F** | **F** | **F** | **F** | **F** | **V** | **F** | **F** |
| f(n) = n5 - 99999n4 | **F** | **F** | **F** | **F** | **F** | **F** | **V** | **F** |

**Exercício (6):**

Dado **f(n)**=3n2-5n-9, **g(n)** = n\*lg(n), **l(n)** = n.lg2(n) e **h(n)** = 99n8, qual é a ordem de complexidade das operações:

a) f(n) + g(n) - h(n) = O(n8)

b) O(f(n) + O(g(n)) - O(h(n)) = O(n8)

c) f(n) x g(n) = O(n2 \* n) = O(n3)

d) g(n) x l(n) + h(n) = O(n \* n) + O(n8) = O(n8)

e) f(n) x g(n) x l(n) = O(n2 \* n \* n8) = O(n11)

f) O(O(O(O(f(n))))) = O(n2)

**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 Ɵ(n2)

• Prove que 3n2 + 5n +1 não é Θ(n)

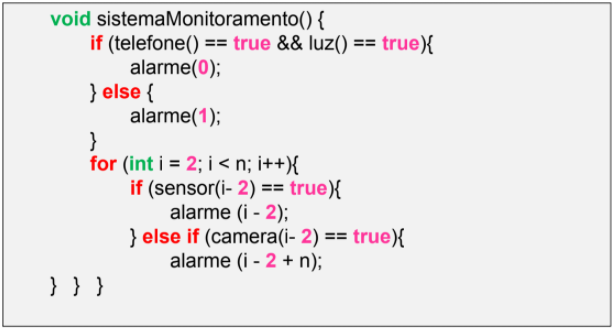
• Prove que 3n2 + 5n +1 não é Θ(n3)

**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.



**Resposta:**

|  |  |  |
| --- | --- | --- |
|  | **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** |

**Função Complexidade**

|  |  |  |
| --- | --- | --- |
|  | **Melhor caso** | **Pior caso** |
| **a** | Θ**(1)** | Θ**(n)** |
| **Telefone()** | Θ**(1)** | Θ**(1)** |
| **Luz()** | Θ**(0)** | Θ**(1)** |
| **Sensor()** | Θ**(n-2)** | Θ**(n-2)** |
| **Câmera()** | Θ**(0)** | Θ**(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(n2), Θ(n2), OMEGA(n2) |

**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 Θ(n). Nesse caso, o custo de ordenar o array e, em seguida, aplicar uma pesquisa binária é mais elevado, Θ(n \* lg(n)) + Θ(lg(n)) = Θ(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.