Exercícios Teóricos – u01: Fundamentos de Análise de Complexidade

Catarina F. M. Castro (803531) – AEDs II

Exercícios resolvidos

1-

a) 2^{10} = 1024

b) lg(1024) = 10

c) lg(17) = 4,08

d) $\lceil \lg(17) \rceil$ = 5

e) $\lfloor \lg(17) \rfloor$ = 4

2- Plote os Gráficos:

a) f(n) = n^3

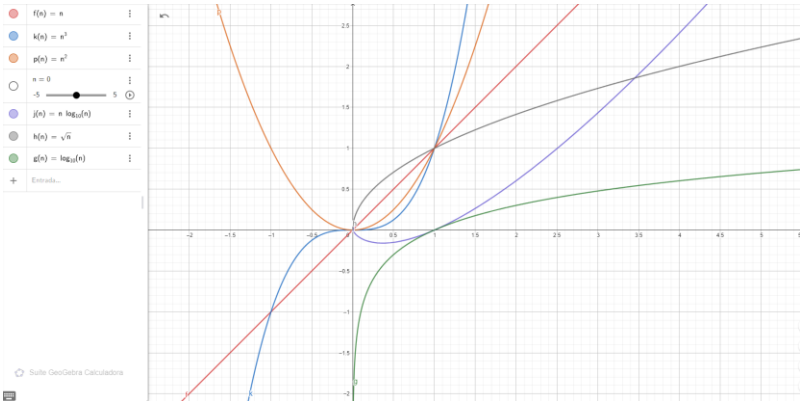
b) f(n) = n^2

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

d) f(n) = n

e) f(n) = sqrt(n)

f) f(n) = lg(n)



3-

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

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

Melhor caso: $f(n) = 3, \theta(1)$
Pior caso: $f(n) = 5, \theta(1)$

4-

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

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

O código realiza $f(n) = 2n, \theta(n)$ subtrações.

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 código realiza $f(n) = 3n^2, \theta(n^2)$

6-

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

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

O código realiza $f(n) = piso[\lg(n)] + 1, \theta(\ln(n))$

7-

- 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?

A função de complexidade desse algoritmo no melhor caso é $t(n) = 1$, que acontece quando o elemento buscado está na primeira posição. Já, no pior caso, a função de

complexidade é $t(n) = n$, que ocorre quando se testam todas as posições, apenas para descobrir que o elemento estava na última posição ou não estava no *array*. Visto isso, esse algoritmo é ótimo, já que é preciso testar todos os elementos para garantir a resposta.

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?

Nessa situação, o aluno deve optar por procurar o elemento por meio de uma pesquisa sequencial, que possui custo $\theta(n)$. Isso ocorre pois ordenar o array e depois aplicar o pesquisa binária teria o custo $\theta(n * lgn)$, e seria menos eficiente.

9-

- Responda se as afirmações são verdadeiras ou falsas:

a) $3n^2 + 5n + 1$ é $O(n)$: Falsa ⟶ $O > n^2$

b) $3n^2 + 5n + 1$ é $O(n^2)$: Verdadeira

c) $3n^2 + 5n + 1$ é $O(n^3)$: Verdadeira

d) $3n^2 + 5n + 1$ é $\Omega(n)$: Verdadeira

e) $3n^2 + 5n + 1$ é $\Omega(n^2)$: Verdadeira

f) $3n^2 + 5n + 1$ é $\Omega(n^3)$: Falsa ⟶ $\Omega < n^2$

g) $3n^2 + 5n + 1$ é $\mathbf{\Theta}(n)$: Falsa ⟶ $\theta = n$

h) $3n^2 + 5n + 1$ é $\mathbf{\Theta}(n^2)$: Verdadeira

i) $3n^2 + 5n + 1$ é $\mathbf{\Theta}(n^3)$: Falsa ⟶ $\theta = n^2$

10-

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

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

O algoritmo de seleção faz $n * \theta(n^2)$ comparações, ou seja $\theta(n^3)$ comparações.

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) $h(n) + g(n) - f(n)$ $= [99n^8] + [n \cdot lg(n)] - [3n^2 - 5n - 9] = \Theta(n^8)$

b) $\Theta(h(n)) + \Theta(g(n)) - \Theta(f(n))$ $= \theta(n^8) + \theta(n \cdot lg(n)) - \theta(n^2) = \theta(n^8)$

c) $f(n) \times g(n)$ $= \theta(n^2) \cdot \theta(n \cdot lg(n)) = \theta(n^3 \cdot lg(n))$

d) $g(n) \times l(n) + h(n)$ $= \theta(n \cdot lg(n)) \cdot \theta(n \cdot lg^2(n)) + \theta(n^8) = \theta(n^8)$

e) $f(n) \times g(n) \times l(n)$ $= \theta(n^2) \cdot \theta(n \cdot lg(n)) \cdot \theta(n \cdot lg^2(n)) = \theta(n^4 \cdot lg^3(n))$

f) $\Theta(\Theta(\Theta(\Theta(f(n)))))$ $= \theta(n)$

12-

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 $O(n^2)$

Para que $O(n^2)$, é preciso que $c > 3$. Um exemplo seria $c = 4$ e $m = 5{,}2$.

13-

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 $O(n^3)$

Assim como a questão anterior, para que $O(n^3)$, é preciso que $c > 3$, visto que a notação O indica qualquer complexidade cujo crescimento seja maior ou igual ao limite justo. Um exemplo seria $c = 4$ e $m = 5{,}7$.

14-

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

Não existe nenhum par (c, m) em que, tal que $n \geq m$, $|3n2 + 5n + 1| \leq c\,x\,|n|$ seja verdadeira.

15-

- 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];
}       }       }
```


No melhor caso, a função de complexidade é $f(n) = 1 + (n - 2)$. Já no pior caso, a função é $f(n) = 1 + 2(n - 2)$. Em ambos os casos, a ordem de complexidade é $\theta(n)\ \Omega(n)\ O(n)$.

16-

- 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];
}       }       }
```

No melhor caso, a função de complexidade é $f(n) = 2$, com ordem de complexidade $\theta(1)$, $\Omega(1)$, e $O(1)$. Já no pior caso, a função é $f(n) = 2 + (n - 2)$, com ordem $\theta(n)$, $\Omega(n)$, e $O(n)$.

17-

- 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--;
}
```

Melhor caso: $f(n) = n + 1$, com ordem $\theta(n)$, $\Omega(n)$, e $O(n)$.
Pior caso: $f(n) = n + 2$, com ordem $\theta(n)$, $\Omega(n)$, e $O(n)$.

18-

- 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--;
}
```

Todos os casos: $f(n) = (2n + 1)n$, com ordem $\theta(n^2)$, $\Omega(n^2)$, e $O(n^2)$.

19-

- 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--;
        }
}
```

Todos os casos: $f(n) = n + piso(\lg(n))$, com ordem $\theta(n * \lg(n))$, $\Omega(n * \lg(n))$, e $O(n * \lg(n))$.

20-

- 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	✓			
$(3/2)^n$				✓

21-

- 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)

A primeira função $f1$ possui crescimento polinomial, $f2$ possui crescimento linear, $f3$ e $f4$ possuem crescimento exponencial, $f5$ possui crescimento polinomial e, por fim, $f6$ possui crescimento constante.

Dessa forma:

$$f6 < f2 < f1 < f5 < f4 < f3$$

22-

- 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)

$$f6 < f3 < f2 < f9 < f1 < f5 < f4 < f7 < f8$$

23-

- 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)
1 n + 30	3 n^4
2 $n^2 + 2n - 10$	1 3n - 1
3 n^3 x 3n	4 lg(2n)
4 lg(n)	2 $n^2 + 3n$

Exercícios não-resolvidos

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

```
void maxMin1 () {
      max = min = array[0];
      for (int i = 1; i < n; i ++) {
            if (array[i] > max) max = array[i];
            if (array[i] < min) min = array[i];
      }
}
```

Todos os casos: $f(n) = 2(n-1)$

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

3-

- Preencha verdadeiro ou falso na tabela abaixo:

	Θ(1)	Θ(lg n)	Θ(n)	Θ(n.lg(n))	Θ(n^2)	Θ(n^3)	Θ(n^5)	Θ(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) = $n^5 - 99999n^4$							✓	

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) = $n^5 - 99999n^4$							✓	✓

5-

- Preencha verdadeiro ou falso na tabela abaixo:

	Ω(1)	Ω(lg n)	Ω(n)	Ω(n.lg(n))	Ω(n^2)	Ω(n^3)	Ω(n^5)	Ω(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) = $n^5 - 99999n^4$	✓	✓	✓	✓	✓	✓	✓	

6-

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 $\Omega(n^2)$

$c = 3$ e $m = 1$

7-

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 $\Omega(n)$

$c = 3$ e $m = 1$

8-

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

Não existem valores para c e m que satisfaçam as condições necessárias para, tal que $n \geq m, |3n2 + 5n + 1| \geq c\ x\ |n3|$ seja considerado verdadeiro

9-

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 $\Theta(n^2)$

$c = 3$ ou $c2 = 4$ e $m = 5.2$.

10-

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

Não existem valores para c e m que satisfaçam as condições necessárias para que a afirmação seja considerada verdadeira.

11-

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

Não existem valores para c e m que satisfaçam as condições necessárias para que a afirmação seja considerada verdadeira.

12-

- 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);
} } }
```

No método alarme():
- A função de complexidade é $f(n) = 1 + (n - 2)$ no pior caso e é $f(n) = 1$ no melhor caso.
- A ordem de complexidade é $\theta(n)$, $\Omega(n)$, e $O(n)$ no pior caso e $\theta(1)$, $\Omega(1)$, e $O(1)$.

Em outros métodos:
- A função de complexidade é $f(n) = 2 + 2x(n - 2)$ no pior caso e $f(n) = 2 + (n - 2)$ no melhor caso.
- A ordem de complexidade é $\theta(n)$, $\Omega(n)$, e $O(n)$ em todos os casos.

13-

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

```
public class ExercicioComplexidade {
   public static void main(String[] args){
         // Declaração do Scanner
         Scanner in = new Scanner(System.in);

         // Declaração de variáveis
         int n = in.nextInt();
         int arr = new vet[n];
         int cont = 0;

         // Preenchimento do vetor
         for(int i=0, i<n, i++){
               arr[i] = nextInt();

               // Contagem dos números pares
               if(arr[i] % 2 == 0){
                     cont++;
               }
         }
   }
}
```

No melhor caso, são feitas $f(n) = n$ operações, com um nível de complexidade de $\theta(n)$. Já no pior caso, são feitas $f(n) = 2n$ operações, também com um nível de complexidade $\theta(n)$.

14-

- 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

As n pesquisas sequenciais terão um custo de $n * \theta(n) = \theta(n^2)$
As n pesquisas binárias terão um custo de $\theta\big(n\ x\ lg(n)\big) + \ n * \theta(lg(n)) \ = \ \theta(n * lg(n))$