Pontificia Universidade Catolica de Minas Gerais
ICEI
Ciencias da Computacao
Algoritmos e Estruturas de Dados II

Trabalho Teorico 6

Aluno(a): Jose Fernando Rossi Junior

Professor: Rodrigo Richard

Belo Horizonte
09/2021

- **Ex. Res. 1**
  1. $2^10 = 1024$
  2. $lg(1024) = 10$
  3. $lg(17) = 4,087$
  4. $\lceil 17 \rceil = 5$
  5. $\lfloor 17 \rfloor = 4$
- **Ex. Res. 2**

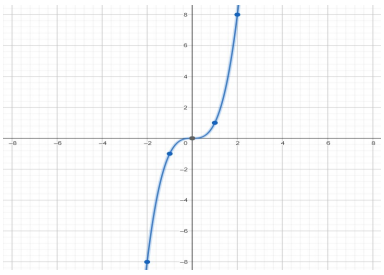

Figura 1: Grafico: n^3.

1.

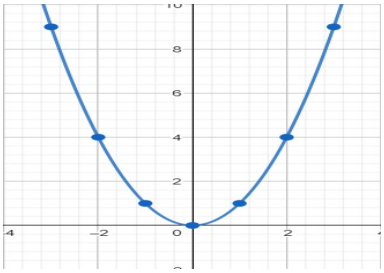

Figura 2: Grafico: n^2.

2.

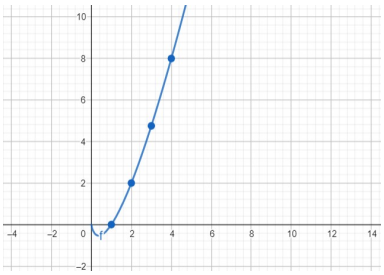

Figura 3: Grafico: $n * lg(n)$.

3.

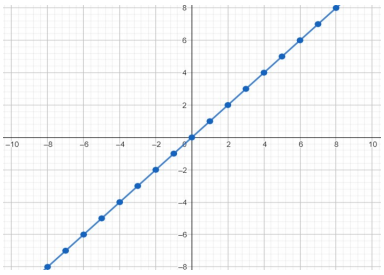

Figura 4: Grafico: n.

4.

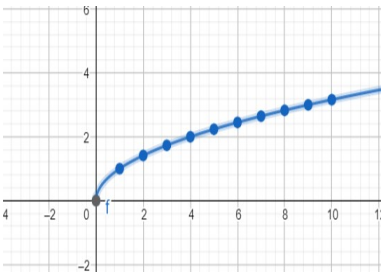

Figura 5: Grafico: $\sqrt{n}$.

5.

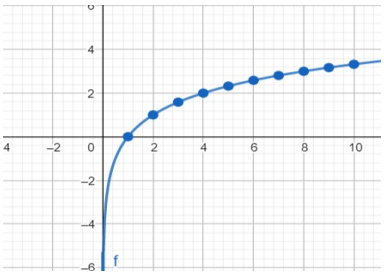

Figura 6: Grafico: $\lg n$.

6.

- **Ex. Res. 3**
  - $(2 * \frac{n}{2}) + (\frac{n}{2}) = \frac{3n}{2}$
- **Ex. Res. 4**
  - $n - 3$
- **Ex. Res. 5**
  - $lg(n) + 1$
- **Ex. Res. 6**

```
[n= 4] =>  4 2 1 (3 vezes)
[n= 5] =>  5 2 1 (3 vezes)
[n= 6] =>  6 3 1 (3 vezes)
[n= 7] =>  7 3 1 (3 vezes)
[n= 8] =>  8 4 2 1 (4 vezes)
[n= 9] =>  9 4 2 1 (4 vezes)
[n= 10] =>  10 5 2 1 (4 vezes)
[n= 11] =>  11 5 2 1 (4 vezes)
[n= 12] =>  12 6 3 1 (4 vezes)
[n= 13] =>  13 6 3 1 (4 vezes)
[n= 14] =>  14 7 3 1 (4 vezes)
[n= 15] =>  15 7 3 1 (4 vezes)
[n= 16] =>  16 8 4 2 1 (5 vezes)
[n= 17] =>  17 8 4 2 1 (5 vezes)
[n= 31] =>  31 15 7 3 1 (5 vezes)
[n= 32] =>  32 16 8 4 2 1 (6 vezes)
[n= 33] =>  33 16 8 4 2 1 (6 vezes)
[n= 63] =>  63 31 15 7 3 1 (6 vezes)
[n= 64] =>  64 32 16 8 4 2 1 (7 vezes)
[n= 65] =>  65 32 16 8 4 2 1 (7 vezes)
```

Figura 7: Resultado do metodo apresentado na questao

- **Ex. Res. 7**

  1. Comparacao entre elementos do array.
  2. $n-1$ vezes.
  3. Para todos os tres casos.

- **Ex. 1**

  - $(0,4n*20)+(1,2n*3,8)+(n*\frac{3,5}{2})$
    $8n+4,56n+1,75n$
    $14,31n$

- **Ex. Res. 8**

  1. Comparacao de elementos.
  2. $n-1$ vezes.
  3. Para os tres casos.
  4. Sim, pois o custo minimo sempre sera esse, $n-1$.

- **Ex. Res. 9**

  1. Comparacao de elementos.
  2. No melhor caso, 1 vez.
     No pior caso, n vezes.
     No caso medio, $\frac{n+1}{2}$ vezes.
  3. Sim, pois nesse caso temos de testar todos os elementos do array.

- **Ex. 2**

1.
```java
/*
* Retorna o maior elemento do vetor
*/
public static int maiorE(int[] array){
  int maior = array[0];
  for(int i=1; i<array.length; i++){
    if (array[i] > maior)
    maior = array[i];
    }
    return maior;
}
```

2.

```java
/*
* Retorna o menor elemento do vetor
*/
public static int menorE(int[] array){
    int menor = array[0];
    for(int i=1; i<array.length; i++){
        if (array[i] < menor)
        menor = array[i];
    }
    return menor;
}
```

3. $O(n-1)$

• Ex. Res. 10

– Nesse caso, e mais vantajoso utilizar a pesquisa sequencial, ja que a mesma tem custo $O(n)$. Em contrapartida, a ordenacao e posterior pesquisa binaria tem custo $O(n * lg(n))$, sendo $O(n)$ o custo da ordenacao e $O(lg(n))$ da busca binaria.

• Ex. Res. 11

1. Falsa. Correcao: $O(n^2)$.
2. Verdadeira.
3. Verdadeira.
4. Verdadeira.
5. Verdadeira.
6. Falsa. Correcao: $\Omega(n^2)$.
7. Falsa. Correcao: $\Theta(n^2)$.
8. Verdadeira.
9. Falsa. Correcao: $\Theta(n^2)$.

- **Ex. 3**

	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

Figura 8: Tabela True or False para notacao de complexidade.

- **Ex. 4**

	Ω(1)	Ω(lg n)	Ω(n)	Ω(n.lg(n))	$\Omega(n^2)$	$\Omega(n^3)$	$\Omega(n^5)$	$\Omega(n^{20})$
f(n) = lg(n)	V	V	F	F	F	F	F	F
f(n) = n . lg(n)	V	V	V	V	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

Figura 9: Tabela True or False para notacao de complexidade.

- **Ex. 5**[2.0cm]

	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	V	F	F	F	F	F	F
f(n) = n . lg(n)	F	F	F	V	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

Figura 10: Tabela True or False para notacao de complexidade.

Exercicio 10 - Resumo

- Notacao O - Pior caso de um algoritmo. (Teto)
- Notacao Ω - Melhor caso de um algoritmo. (Piso)
- Notacao Θ - Caso medio de um algoritmo. (Media)[1.0cm]
- Geralmente, quando vamos analisar a complexidade e custo de um algoritmo, utilizamos a notacao O, pois ela nos fornece a pior possibilidade de tempo do algoritmo.
- Outro ponto interessante de abordar, e o fato de que quanto maior a inclinacao da curva da funcao em relacao ao eixo Y, mais caro e o algoritmo. Sendo assim, algoritmos com custo n^3, n^2 e afins, sao caros, ja que a inclinacao e alta. Em contrapartida algoritmos com custo $lg(n)$ ou n, sao algoritmos mais baratos, ja que a inclinacao da curva e menor.
- Em suma, algoritmos constantes, lineares e logaritmicos sao melhores em termos de eficiencia. Ja algoritmos exponenciais, cubicos e quadraticos sao menos recomendados em termos de eficiencia.

- **Ex. Res. 12**
  1. Pior caso $\rightarrow 1 + 2n - 4 = O(n), \Omega(n), \Theta(n)$
  2. Melhor caso $\rightarrow 1 + (n-2) * 0 = O(1), \Omega(1), \Theta(1)$
- **Ex. Res. 13**
  1. Pior caso $\rightarrow n + 2 = O(n), \Omega(n), \Theta(n)$
  2. Melhor caso $\rightarrow n + 1 = O(n), \Omega(n), \Theta(n)$
- **Ex. Res. 14**
  1. Geral $\rightarrow 2n^2 + n = O(n^2), \Omega(n^2), \Theta(n^2)$
- **Ex. Res. 15**
  1. Geral $\rightarrow n * lg(n) + n = O(n * lg(n)), \Omega(n * lg(n)), \Theta(n * lg(n))$
- **Ex. Res. 16**

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

Figura 11: Tabela para crescimento da funcao.

- **Ex. Res. 17**
  1. f_6
  2. f_2
  3. f_1
  4. f_5
  5. f_4
  6. f_3
- **Ex. Res. 17**
  1. f_6
  2. f_3
  3. f_2
  4. f_9
  5. f_1
  6. f_5
  7. f_4
  8. f_7
  9. f_8
- **Ex. Res. 19**

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

Figura 12: Tabela para comparacao de funcoes Θ.

- **Ex. 13**
  - Nesse caso o custo seria menor, ja que o custo da ordenacao e pesquisa binaria de n elementos seria $\Theta(n * lg(n))$, ja para pesquisa sequencial seria $\Theta(n^2)$.