

# Trabalho 1 – Anexar UM documento no MOODLE com TODAS respostas em ORDEM.

Data da entrega: 04/08/2021. Nome do documento: <Nome\_do\_aluno>\_Trabalho1.pdf

Formato do documento: PDF

Nome: Kauan Peçanha Lira \_\_\_\_\_ Data: 04/08/2022 \_\_\_\_\_

- 1) (1,0) Ordene as seguintes funções em ordem crescente, de acordo com as suas taxas de crescimento assintótico. Sugestão: vocês podem fazer um pequeno programa e estabelecer valores grandes de  $n$  e verificar as taxas de crescimento.

$4n \cdot \log(n) + 2n$ ,  $2^{10}$ ,  $2^{\log(n)}$ ,  
 $3n + 100\log(n)$ ,  $4n$ ,  $2^n$ ,  $3^{3n}$ ,  $n^2 + 10n$ ,  $n^3$ ,  $n \cdot \log(n)$

Resolução:

Primeiro, foi necessário colocar tais fórmulas, no Excel, afim de se facilitar os cálculos. Desta forma:

Fórmulas	Classificação	1	5	10	20	30	100	1000	100000	10000000
$4n \cdot \log(n) + 2n$	linear	2	23.97940009	60	144.0823997	237.25	1000	14000	2200000	300000000
$2^{**}(10)$	constante	1024.00	1024.00	1024.00	1024.00	1024.00	1024.00	1025.00	1026.00	1027.00
$2^{**}\log(n)$	exponencial	1	1.62334541	2	2.464047377	2.783926756	4	8	32	128
$3^n + 100 \cdot \log(n)$	linear	3	84.89700043	130	190.1029996	237.7121255	500	3300	300500	30000700
$4^n$	linear	4	20	40	80	120	400	4000	400000	40000000
$2^{**}(n)$	exponencial	2	32	1024	1048576	1073741824	1.26765E+30	1.0715E+301	#NUM!	#NUM!
$3^{**}(3n)$	exponencial	27	14348907	2.05891E+14	4.23912E+28	8.72796E+42	1.3689E+143	#NUM!	#NUM!	#NUM!
$(n^{**}(2)) + 10^n$	exponencial	11	75	200	600	1200	11000	1010000	10001000000	1E+14
$n^{**}(3)$	exponencial	1	125	1000	8000	27000	1000000	1000000000	1E+15	1E+21
$n \cdot \log(n)$	linear	0	3.494850022	10	26.02059991	44.31363764	200	3000	500000	700000000

Análise da Coluna K (ordem crescente)	
32	$2^{**}\log(n)$
1026	$2^{**}(10)$
300500	$3^n + 100 \cdot \log(n)$
400000	$4^n$
5000000	$n \cdot \log(n)$
22000000	$4^n \cdot \log(n) + 2^n$
10001000000	$(n^{**}(2)) + 10^n$
1E+15	$n^{**}(3)$
#NUM!	$2^{**}(n)$
#NUM!	$3^{**}(3n)$
Resultado	Função Correspondente

Análise da Coluna L (ordem crescente)	
128	$2^{**}\log(n)$
1027	$2^{**}(10)$
30000700	$3^n + 100 \cdot \log(n)$
40000000	$4^n$
70000000	$n \cdot \log(n)$
300000000	$4^n \cdot \log(n) + 2^n$
1E+14	$(n^{**}(2)) + 10^n$
1E+21	$n^{**}(3)$
#NUM!	$2^{**}(n)$
#NUM!	$3^{**}(3n)$
Resultado	Função Correspondente

Podendo ser acessada seguinte:

**Favor ver planilha Trab\_Alg\_Est\_Dados.xlsx**

Pode-se notar que ha tres categorias de Complexidade:

- Constante
- Linear
- Exponencial

Desta forma, sabe-se que a hierarquia, em questão de complexidade, é: Constante, Linear, Exponencial. Desta forma, analisando os dados expostos nesta tabela acima, pode-se perceber que estas Complexidades seguem esta ordem:

Observação: A longo prazo, a complexidade logarítmica será superior à constante, e somente por isso, a constante será classificada como a mais inferior dentre todas as listadas aqui. Não somente isso, bem como, quando se analisa as duas maiores funções, sabe-se facilmente que  $3^{**}(3n)$  é superior à  $2^{**}(n)$  pelo simples fato de que, além dos resultados serem sempre maiores para aquele, sabe-se que sua base é superior à seu inferior em uma unidade, e que seu expoente é superior ao de seu inferior em 2 unidades. Desta forma, infere-se facilmente que  $3^{**}(3n)$  sempre será superior à  $2^{**}(n)$ .

Notação: \* = (multiplicação) ; \*\* = (exponenciação);

Ordem decrescente:

- $2^{**}(10) = 2^{10} = \text{menor}$
- $2^{**}\log(n) = 2^{\log(n)}$
- $3^n + 100 \cdot \log(n) = 3 \cdot n + 100 \cdot \log(n)$
- $4^n = 4 \cdot n$

- $n \cdot \log(n) = n \cdot \log(n)$
- $4 \cdot n \cdot \log(n) + 2 \cdot n = 4 \cdot n \cdot \log(n) + 2 \cdot n$
- $(n \cdot (2)) + 10 \cdot n = n^2 + 10 \cdot n$
- $n \cdot (3) = n^3$
- $2 \cdot (n) = 2^n$
- $3 \cdot (3n) = 3^{3n} = \text{maior}$

2) (1,0) **Mostre** que  $f(n) = \log_y n$  é  $O(\log_x n)$  para  $x, y > 1$ ; ou seja, a base logarítmica não afeta a complexidade assintótica da função/ algoritmo.

Resolução:

De acordo com a Notação O-Grande, esta relação existiria para todo  $n \geq N$ , bem como para  $x, y > 1$ . Isto confere como verdade, uma vez que, em uma análise assintótica de ordem superior, independentemente do logaritmando determinado na análise do algoritmo, isto sempre será  $O(\log(n))$ , após se abstrair os termos de menor importância. Entretanto, é importante se salientar que  $x$  e  $y$  devem ser maiores que 1 para que isto seja verdadeiro.

3) (1,0) Encontre o menor limite assintótico superior para o algoritmo abaixo, escrito em C. Mostre como você chegou a esse resultado.

```
int f(int n) {
    int i, j, k, sum = 0;
    for (i=1; i < n; i *= 3) {
        for (j = n; j > 0; j /= 2) {
            for (k = j; k < n * n; k += 2) {
                sum += (i + j * k);
            }
        }
    }
}
```

Resolução:

Programa em C

```
#include<stdio.h>
#include<stdlib.h>
```

```
int f(int n)
{
    int i, j, k, sum = 0; //4 declaração 4x

    for(i=1;i<n;i*3) //3 inicialização, comparação e multiplicação
    {
        for(j = n; j>0; j/=2) //3 atribuição, comparação, e multiplicação
        {
            for(k=j;k<n*n;k+=2) //3 atribuição, comparação e incrementação
            {
                sum +=(i+j*k); //3(atribuição, soma e multiplicação)
            }
        }
    }
}
```

Observa-se que, poderia ser denotado  $f(n) = 4 + (3n)(3n)(3n+3)$ , sendo  $f(n) = 27n^3 + 27n^2 + 4$ . Entretanto,

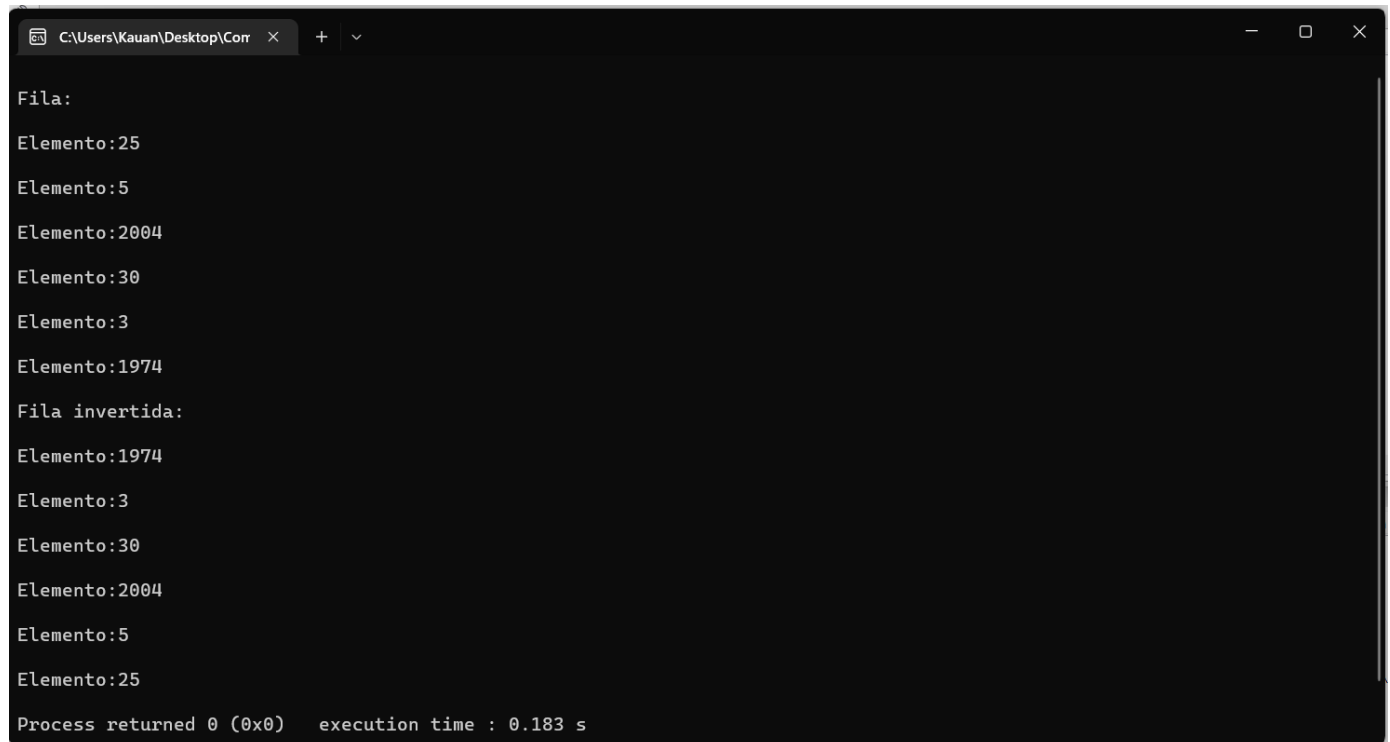
sabe-se que deve-se abstrair os termos constantes, os termos de menores graus, sendo então  $f(n) = 27n^3 + 27n^2 + 4$  o  $O(n^3)$ .

- 4) (1,0) Escreva um **método** para inverter a ordem dos elementos de uma **fila**, usando uma **pilha** como estrutura auxiliar.

Resolução:

Favor ver o arquivo `questao4.cpp` no drive.

Resultado:



```
C:\Users\Kauan\Desktop\Corr >
Fila:
Elemento:25
Elemento:5
Elemento:2004
Elemento:30
Elemento:3
Elemento:1974
Fila invertida:
Elemento:1974
Elemento:3
Elemento:30
Elemento:2004
Elemento:5
Elemento:25
Process returned 0 (0x0)   execution time : 0.183 s
```

- 5) (2,0) Implemente um programa orientado a objetos de uma lista simplesmente encadeada. No método main crie um objeto SequentialList representando uma **lista seqüencial geral** de números inteiros. Faça também as seguintes operações:

- a) inserir 10 números
- b) imprimir apenas os números primos da lista;
- c) inserir o número 14 no fim da lista;
- d) ler um número e inseri-lo no meio da lista (a lista deve conter número par de elementos).
- e) ler um número e procurá-lo na lista, imprimindo a posição de sua primeira ocorrência a partir do início; se não estiver na lista, imprimir uma mensagem adequada.

Apresentar o programa e o resultado

Resolução:

Favor ver o arquivo `questao5.cpp` no Drive.

Resultado:

```
C:\Users\Kauan\Desktop\Corr x + v
Adicionando 10 numeros
1
2
3
4
5
6
7
8
9
10
Pressione qualquer tecla para continuar. . .

Adicionando 14 ao fim da lista
1
2
3
4
5
6
7
8
9
10
14
Pressione qualquer tecla para continuar. . .

Process returned 0 (0x0)   execution time : 6.745 s
Press any key to continue.
```

6) (2,0) Seja a classe SequentialList criada anteriormente, crie um novo método chamado tranferirMaior() que transfere o nó de maior valor para o início da lista, mantendo os demais na mesma ordem original. Se não for possível fazer a movimentação, imprimir mensagem com o motivo específico, ou seja: *Lista vazia, Lista com um só nó ou Maior valor já está no início da lista.*

Resolução: Não consegui fazer este programa, professora.

7) (2,0) Implemente uma **lista duplamente encadeada** com as operações básicas de inserção, remoção, impressão e busca. Inclua um método que **retorne** o **n-ésimo** elemento da lista (ou informe que o n-ésimo elemento não existe). Por exemplo, se a lista tem 7 elementos, e se eu quero saber qual valor do quinto elemento, ela me retornará o valor do quinto elemento da minha lista; mas se eu quiser o oitavo elemento, o método retornará o valor -1 e conterà uma mensagem de posição inexistente.

Resolução: Não consegui fazer este programa, professora.

Apresentar o programa e o resultado.