



# BCC241 – PROJETO E ANÁLISE DE ALGORITMOS

## LISTA DE EXERCÍCIOS E1



1. Em cada item indique se  $f=O(g)$  ou  $f = \Omega(g)$

f	g
a. $n^{-100}$	$n^{-200}$
b. $n^{1/2}$	$n^{2/3}$
c. $100n + \log n$	$n + (\log n)^2$
d. $10 \log n$	$\log n^2$
e. $n^2$	$n \log n$
f. $n^{1/2}$	$5^{\log_2 n}$
g. $2^n$	$2^{n+1}$
h. $n!$	$2^n$

2. Exercício 0.4 a e b de DPV.

3. Qual é o menor valor de  $n$  tal que um algoritmo cujo tempo de execução é  $100n^2$  funciona mais rápido que um algoritmo cujo tempo de execução é  $2^n$  na mesma máquina?

4. Você dispõe de dois algoritmos A1 e A2 para resolver um mesmo problema. A função de complexidade dos mesmos é  $n^2 + n$  e  $10^3 n \log n$ , respectivamente. Qual algoritmo você escolheria? Discuta todas as possibilidades.

5. Para cada função  $f(n)$  e cada tempo  $t$  na tabela a seguir, determine o maior tamanho  $n$  de um problema que pode ser resolvido no tempo  $t$ , supondo-se que o algoritmo para resolver o problema demore  $f(n)$  microssegundos.

$f(n) \setminus t$	1 segundo	1 minuto	1 hora	1 dia
$\lg n$				
$\sqrt{n}$				
$n$				
$n \lg n$				
$n^2$				
$n^3$				
$2^n$				
$n!$				

6. Usando divisão e conquista multiplique 10011011 and 10111010.

7. Exercício 2.4 do DPV.

8. Exercício 2.5 do DPV. Até e)

9. Exercício 2.12 do DPV.

10. Exercício 2.16 do DPV.

11. Forneça o algoritmo que informe a mediana de um vetor em  $O(n)$ .

12. Seja o seguinte trecho de pseudo-código:

```
a = 0;
for i=0..n:
  for j=1..i:
    for k=j+1..j+i:
      a = a + 1;
```

Defina o valor de  $a$  em função de  $n$ .

13. Prove por indução:

a) 
$$\sum_{i=1}^n \frac{1}{i(i+1)} = \frac{n}{n+1}$$

b) 
$$\sum_{i=1}^n a^i = \frac{a^{n+1} - a}{a - 1}$$

c) 
$$\sum_{j=1}^n j^2 = \frac{n(n+1)(2n+1)}{6}$$

14. Prove que  $T(n) = T\left(\frac{n}{2}\right) + 1 = O(\log n)$ .

15. Responda com V ou F. Justifique sua resposta por item.

	Um algoritmo de ordem de complexidade polinomial é sempre preferível a um algoritmo de ordem de complexidade exponencial.
	Se o pior caso do tempo de computação T de um algoritmo A é $\Omega(n^2)$ , então é possível que T seja $O(n)$ para algumas entradas de A.
	$\log n^2 = \Theta(\log n + 5)$
	$3^n = O(2^n)$
	$n^2 \log n = \Omega(n\sqrt{n})$
	$T(n) = 64T(n/8) + n^2 = \Theta(n^2)$
	Sempre que $f=O(h)$ e $g=O(h)$ , $f=O(g)$
	Se $f \neq g$ e $f=O(g)$ então $g=O(f)$
	As funções $n \cdot \log(n)$ e $n \cdot \log(n \cdot n)$ possuem a mesma ordem de complexidade
	$\log(n^c)$ é $\theta(\log(n))$ para qualquer constante $c > 0$
	$2^{100}$ é $O(1)$
	$2^{n-1} = O(2^n)$
	$2^n = \omega(2^{n-1})$

16. Encontre a complexidade de cada um dos seguintes trechos de programa:

**a) Dois loops em seqüência:**

```
for (i = 0; i < N; i++) {
    seqüência de comandos
}
for (j = 0; j < M; j++) {
    seqüência de comandos
}
```

O que acontece se trocarmos a complexidade do segundo loop por N ao invés de M?

**b) Um loop aninhado seguido por um loop não aninhado:**

```
for (i = 0; i < N; i++) {
    for (j = 0; j < N; j++) {
        seqüência de comandos;
    }
}
for (k = 0; k < N; k++) {
    seqüência de comandos
}
```

**c) Um loop aninhado onde o número de vezes do loop mais interno depende do valor do índice no loop mais externo:**

```
for (i = 0; i < N; i++) {
    for (j = i; j < N; j++) {
        seqüência de comandos
    }
}
```

17. Encontre a complexidade de melhor caso e de pior caso do seguinte trecho de programa:

```
if (x==y) {
    for (i = 0; i < N; i++) {
        for (j = 0; j < N; j++) {
            seqüência de comandos;
        }
    }
    for (i = 0; i < N; i++) {
        seqüência de comandos
    }
}
else {
    for (i = 0; i < N; i++) {
        for (j = 0; j < N; j++) {
            for (k = 0; k < N; k++) {
                seqüência de comandos;
            }
        }
    }
}
```

18.