BCC241 – Projeto e Análise de Algoritmos
Lista de Exercícios E1

1. Em cada item indique e explique se f=O(g), f = Ω(g), f=o(g), f=ω(g) ou f=Θ(g)

	f	g
a.	n -100	n-200
b.	$n^{1/2}$	$n^{2/3}$
c.	log n	$\log^2 n$
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. Preencha a tabela a seguir, informando o tempo gasto por um programa, para resolver um problema com instâncias de tamanho n. Considere o tempo para cada instrução/passo de 10^{-6} segundos e a função de complexidade especificada em cada linha.

	Tamanho da instância (N)					
Função de Complexidade	10	30	40	50	60	70
N + 10						
$2N^2+N + 5$						
$2^N + N^2$						

4. 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 e o tempo de cada instrução é 10^{-6} segundos.

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

5. A tabela a seguir mostra o tamanho da maior instância de um problema que um computador atual consegue resolver em 1 hora, dado a função de complexidade do algoritmo. Preencha a tabela mostrando o maior tamanho de instância que se consegue resolver em uma hora em um computador 100 vezes e 1000 vezes mais rápido que o atual

Maior instância que um computador resolve em 1 hora			
Função de complexidade	Computador Atual	Computador 100x mais rápido	Computador 1000x mais rápido
N	N		
n^2	M		
n^4	P		
4^n	R		

6. Você dispõe de dois algoritmos A1 e A2 para resolver um mesmo problema. As funções de complexidade dos mesmos são $n^2 + n$ *e* $10^3 n \; log \; log \, n$, respectivamente. Qual algoritmo você escolheria? Discuta todas as possibilidades.

7. 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.*

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

9. 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 ≠g e f=O(g) então g=O(f)
	As funções n.log(n) e n.log(n.n) possuem a mesma ordem de complexidade
	$\log(n^c)$ é $\theta(\log(n))$ para qualquer constante c>0
	2^{100} é O(1)
	$2^{n-1} = O(2^n)$
	$2^n = \omega(2^{n-1})$

10. 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 // O(1)
}
for (j = 0; j < M; j++) {
      seqüência de comandos // O(1)
}
```

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; // O(1)
      }
}
for (k = 0; k < N; k++) {
      seqüência de comandos  // O(1)
}
```

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  // O(1)
      }
}
```

11. 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; // O(1)
            }
      }
      for (i = 0; i < N; i++) {
            seqüência de comandos // O(1)
      }
}
else {
      for (i = 0; i < N; i++) {
            for (j = 0; j < N; j++) {
                  for (k = 0; k < N; k++) {
                        seqüência de comandos;  O(1)
                  }
            }
      }
}
```

12. Analise os algoritmos considerando inteiros grandes, ou seja, o tamanho da instância de entrada corresponde à quantidade de bits utilizada, das Figuras 1.1, 1.2, 1.4 e 1.5 de DPV.

[DPV] DASGUPTA, Sanjoy; PAPADIMITRIOU, Christos H.; VAZIRANI, Umesh Virkumar. Algoritmos. São Paulo: McGraw-Hill, 2009. Disponível em: http://algorithmics.lsi.upc.edu/docs/Dasgupta-Papadimitriou-Vazirani.pdf