BCC241 – Projeto e Análise de Algoritmos
Lista de Exercícios E2

1. Exercício 2.4 do DPV.

2. Exercício 2.5 do DPV.

3. Exercício 2.12 do DPV.

4. Exercício 2.14 do DPV.

5. Exercício 2.16 do DPV.

6. 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.*

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

8. Encontre a complexidade do seguinte procedimento que imprime todos os nós de uma árvore binária:

```
function inOrder(no)
input:  no da árvore
output: impressão da subárvore com raiz em nó

if (existe filho esquerda): inOrder(no.esquerda);
imprime(no);
if (existe filho direita): inOrder(no.direita);
```

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

10. 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.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)
	$2^{n-1} = O(2^n)$
	$2^n = \omega(2^{n-1})$

11. 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
      }
}
```

12. 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;
                  }
            }
      }
}
```