1 - Utilize as definições das notações assintóticas e avalie se são verdadeiras ou falsas as seguintes afirmativas:

a) $3n^3 + 2n + 1 = O(n^3)$

$$3n^3 + 2n + 1 \leq cn^3$$

$$3 + 2/n^2 + 1/n^3 \leq c$$

Existe c= 3+2+1=6 e m = 1 que atendem à desigualdade, portanto a afirmação é verdadeira.

b) Insertion-Sort = $O(n^2)$

c) $3n^2 + n = \Omega(n)$

$$3n^3 + n >= c \times n$$

$$3n^2 + 1 >= c$$

Existe c= 3+1=4 e m = 1 que atendem à desigualdade, portanto a afirmação é verdadeira.

d) $2n^3 = \Theta(n^2)$

Parte 1: Big O

$$2n^3 <= c \times n^2$$

$$2n <= c$$

Como c é uma constante e n representa uma variável tendendo ao crescimento, então não existe um valor para c que garanta que este seja maior que n, para qualquer valor de n. A afirmação é falsa.

Parte 2: Ω

$$2n^3 >= c \times n^2$$

$$2n >= c$$

Existe c= 2x1=2 e m = 1 que atendem à desigualdade, portanto a afirmação é verdadeira.

As duas partes (Parte 1 e Parte 2) necessitam ser verdadeiras. Como a Parte 1 é falsa, logo a afirmação é falsa.

2 - Analise os trechos de código abaixo e estime o tempo de execução para o melhor e o pior caso de cada um.

a) Somatório de uma matriz de dimensões nxnxn.

```
função SomatórioMatriz3D(matriz[][][]:int)
soma=0
i = 0
enquanto i < tamanho_linha(matriz[][][])
   j=0
   enquanto j < tamanho_coluna(matriz[][][])
       k=0
        enquanto k < tamanho_nivel(matriz[][][])
            soma = soma + matriz[][][]
             k = k + 1
        j= j + 1
   i = i + 1
retorna(soma)
```

A estimativa do tempo de execução é cúbica ($T(n) = n^3$)

b) Algoritmo de ordenação por inserção.

	custo	quantidade
função OrdenaPorInserção(vetor[]:int)		
para j = 2 até tamanho(vetor)		
chave = vetor[j]		
i = j-1		
enquanto i > 0 e vetor[i] > chave		
vetor[i+1] = vetor [i]		
i = i-1		
vetor [i+ 1] = chave		

Resposta:

	custo	quantidade
função OrdenaPorInserção(vetor[]:int)		
para j = 2 até tamanho(vetor)	op1	n
chave = vetor[j]	op2	n-1
i = j-1	op3	n-1
enquanto i > 0 e vetor[i] > chave	op4	$\sum_{j=2}^{n} tj$
vetor[i+1] = vetor [i]	op5	$\sum_{j=2}^{n} (tj-1)$
i = i-1	op6	$\sum_{j=2}^{n} (tj-1)$
vetor [i+ 1] = chave	op7	n-1

Obs.: Considere o termo tj representando o número de vezes da execução do teste do loop interno. Este valor irá variar de acordo com o melhor e o pior caso. Visão inicial dos custos de processamento:

T(n)= n op1 + (n-1) op2 + (n-1) op3 + $\sum_{j=2}^{n} tj$ op4 + $\sum_{j=2}^{n}(tj-1)$ op5 + $\sum_{j=2}^{n}(tj-1)$ op6 + (n-1) op7

O Melhor caso ocorre quando o vetor está ordenado. Neste caso, para cada valor de j=2, 3, ...,n o teste da chave com o valor i do vetor (vetor[i]>chave) será verdadeiro quando i tem o valor inicial igual j-1. Deste modo conclui-se que t_j=1 para j=2, 3, ..., n. Assim os somatórios serão resolvidos como abaixo, para os valores de N e (n-1), sendo que a estimativa do tempo de execução resulta linear em n.

$\sum_{j=2}^{n} tj = (n-1)$ $\quad\quad$ $\sum_{j=2}^{n}(tj-1) = 0$

T(n)= n op1 + (n-1) op2 + (n-1) op3 + (n-1) op4 + (n-1) op7

T(n)= n op1 + n op2 – op2 + n op3 -op3 + n op4 - op4 + n op7 - op7

T(n)= n (op1 + op2 + op3 + op4 + op7) – (op2 + op3 + op4 + op7)

O pior caso ocorre quando o vetor estiver ordenado de forma inversa. Neste caso será realizada a comparação do elemento vetor[i] e da chave para todos os elementos do vetor em análise (ou seja, o vetor[1...j-1]), portanto t_j=j para j=2, 3, .., n. Assim temos:

$\sum_{j=2}^{n} tj = \frac{n(n+1)}{2} - 1$ $\quad\quad$ $\sum_{j=2}^{n}(tj-1) = \frac{n(n+1)}{2} - n$

Neste caso,

T(n)= n op1 + (n-1) op2 + (n-1) op3 + $\sum_{j=2}^{n} tj$ op4 + $\sum_{j=2}^{n}(tj-1)$ op5 + $\sum_{j=2}^{n}(tj-1)$ op6 + (n-1) op7

T(n)= n op1 + n op2 - op2 + n op3 - op3 + $\left[\frac{n(n+1)}{2} - 1\right]$ op4 + $\left[\frac{n(n+1)}{2}\text{-n}\right]$op5+ $\left[\frac{n(n+1)}{2}\text{-n}\right]$op6+n op7 - op7

T(n)= n op1 + n op2 - op2 + n op3 - op3 + $\left[\frac{n(n+1)}{2} - 1\right]$ op4 + $\left[\frac{n(n+1)}{2}\text{-n}\right]$op5+$\left[\frac{n(n+1)}{2}\text{-n}\right]$op6+ n op7 - op7

T(n) = $\left[\frac{\text{op4}}{2} + \frac{\text{op5}}{2} + \frac{\text{op6}}{2}\right]$ n² + $\left[\text{op1} + \text{op2} + \text{op3} + \frac{\text{op4}}{2} + \frac{\text{op5}}{2} + \frac{\text{op6}}{2} + \text{op7}\right]$ n – (op2+op3+op4+op7)

A estimativa do tempo de execução é quadrática T(n) = n²

3 - Encontre a forma fechada dos somatórios a seguir:

a)

$$\sum_{i=1}^{n} i2^{i-1}$$

$$Sn=\sum_{i=1}^{n} i2^{i-1}$$

$$Sn+(n+1)2^{n}=\sum_{i=1}^{n+1} i2^{i-1}$$

$$Sn+(n+1)2^{n}=1+\sum_{i=2}^{n+1} i2^{i-1}$$

$$Sn+(n+1)2^{n}=1+\sum_{i=1}^{n} (i+1)2^{i+1-1}$$

$$Sn+(n+1)2^{n}=1+\sum_{i=1}^{n} (i+1)2^{i}$$

$$Sn+(n+1)2^{n}=1+\sum_{i=1}^{n} i2^{i}+2^{i}$$

$$Sn+(n+1)2^{n}=1+\sum_{i=1}^{n} i2^{i}+\sum_{i=1}^{n} 2^{i}$$ → o segundo somatório já foi resolvido em aula

$$Sn+(n+1)2^{n}=1+(\sum_{i=1}^{n} i2^{i-1}\cdot 2^{1})+2^{n+1}-2$$

$$Sn+(n+1)2^{n}=1+2(\sum_{i=1}^{n} i2^{i-1})+2^{n+1}-2$$

$$Sn+(n+1)2^{n}=1+2\,Sn+2^{n+1}-2$$

$$(n+1)2^{n}-2^{n+1}-1+2=2\,Sn-Sn$$

$$Sn=(n+1)2^{n}-2^{n+1}+1$$ → **resposta final**

b)

$$\sum_{i=1}^{n} \frac{2^{i+1}}{5^i}$$

$$Sn=\sum_{i=1}^{n} \frac{2^{i+1}}{5^i}$$

$$Sn+1=Sn+\frac{(2^{n+2})}{5^{n+1}}=\sum_{i=1}^{n+1} \frac{2^{i+1}}{5^i}$$

$$Sn+\frac{(2^{n+2})}{5^{n+1}}=\frac{4}{5}+\sum_{i=2}^{n+1} \frac{2^{i+1}}{5^i}$$

$$Sn+\frac{(2^{n+2})}{5^{n+1}}=\frac{4}{5}+\sum_{i=1}^{n} \frac{2^{i+1+1}}{5^{i+1}}$$

$$Sn+\frac{(2^{n+2})}{5^{n+1}}=\frac{4}{5}+\sum_{i=1}^{n} \frac{2^{i+2}}{5^{i+1}}$$

$$Sn+\frac{(2^{n+2})}{5^{n+1}}=\frac{4}{5}+\sum_{i=1}^{n} \frac{2^{i+1}}{5^i}\cdot\frac{2^1}{5^1}$$

$$Sn+\frac{(2^{n+2})}{5^{n+1}}=\frac{4}{5}+\frac{2}{5}\sum_{i=1}^{n} \frac{2^{i+1}}{5^i}$$

$$Sn+\frac{(2^{n+2})}{5^{n+1}}=\frac{4}{5}+\frac{2}{5}Sn$$

$$\frac{(2^{n+2})}{5^{n+1}}-\frac{4}{5}=\frac{2}{5}Sn-Sn$$

$$\frac{(2^{n+2})}{5^{n+1}}-\frac{4}{5}=\frac{2}{5}Sn-Sn$$

$$\frac{-3}{5}Sn=\frac{2^{n+2}}{5}n+1-\frac{4}{5}$$

$$Sn=\frac{\frac{2^{n+2}}{5^{n+1}}-\frac{4}{5}}{\frac{-3}{5}}$$

$$Sn=(\frac{2^{n+2}}{5^{n+1}}-\frac{4}{5})\cdot\frac{-5}{3}$$

$$Sn=\frac{-5}{3}(\frac{2^{n+2}}{5^{n+1}})+\frac{4}{3}$$ → **resposta final**

c)

$$\sum_{i=1}^{n} 7i-3$$

$$Sn=\sum_{i=1}^{n} 7i-3$$

$$Sn=\sum_{i=1}^{n} 7i-\sum_{i=1}^{n} 3$$

$$Sn=7(\sum_{i=1}^{n} i)-3n$$ → esse somatório já foi resolvido em aula

$$Sn=7\frac{(n(n+1))}{2}-3n$$

$$Sn=\frac{7n^2+7n}{2}-3n$$

$$Sn=\frac{7n^2+7n-6n}{2}$$

$$Sn=\frac{7n^2+n}{2}$$ → **resposta final**

4 - Aplique o método da substituição para encontrar a forma fechada da equação recorrente abaixo.

$$T(n)=\begin{cases} a, n=1 \\ 4T(n-1)+a, n>1 \end{cases}$$

Dica:
$a*4^{k-1} + a*4^{k-2} + a*4^{k-3} + ... + 4a + a = a*(4^{k} - 1 / 3)$

- Calculando a complexidade:

$T(n) = 4T(n - 1) + a$
$T(n - 1) = 4T(n - 2) + a$
$T(n - 2) = 4T(n -3) + a$
$T(n - 3) = 4T(n -4) + a$

4. Técnica backward substitution:
$T(n) = 4T(n - 1) + a \Rightarrow 4^{1}T(n - 1) + a$
$T(n) = 4(4T(n - 2) + a) + a \Rightarrow 4^{2}T(n - 2) + 4a + a$
$T(n) = 16(4T(n -3) + a) + 4a + a \Rightarrow 4^{3}T(n - 3) + 16a + 4a + a$
$T(n) = 64(4T(n -4) + a) + 16a + 4a + a \Rightarrow 4^{4} + 64a + 16a + 4a + a$

5. Generalização:
$4^{k}*T(n - k) + a*4^{k-1} + a*4^{k-2} + a*4^{k-3} + ... + a*4^{3} +a*4^{2} + a*4^{1} + a*4^{0}$

6. Assume-se que $n – k = 1$, logo, $k = n - 1$.

$T(n) = 4^{k}*T(n - k) + a*4^{k-1} + a*4^{k-2} + a*4^{k-3} + ... + a*4^{3} +a*4^{2} + a*4^{1} + a*4^{0}$
$T(n) = 4^{k}*T(n - (n - 1)) + a*4^{k-1} + a*4^{k-2} + a*4^{k-3} + ... + a*4^{3} +a*4^{2} + a*4^{1} + a*4^{0}$
$T(n) = 4^{k}*T(1) + a*4^{k-1} + a*4^{k-2} + a*4^{k-3} + ... + a*4^{3} +a*4^{2} + a*4^{1} + a*4^{0}$

$T(n) = 4^{k}*a + a*4^{k-1} + a*4^{k-2} + a*4^{k-3} + ... + a*4^{3} +a*4^{2} + a*4^{1} + a*4^{0}$

$T(n) = a*(4^{k}) + a*((4^{k} - 1) / 3)$
$T(n) = a*(4^{n-1}) + a*(((4^{n-1}) - 1) / 3)$
$T(n) = a*(4^{n} / 4^{1}) + a*(((4^{n} / 4^{1}) - 1) / 3)$

5 - Encontre a equação recorrente para o algoritmo Merge-Sort e encontre a sua fórmula fechada.

```
Merge-Sort(A, p, r)
   if p < r then
      q←⌊(p+r)/2⌋
      Merge-Sort(A, p, q)
      Merge-Sort(A, q+1, r)
      Merge(A, p, q, r)
```

Dica 1: Considere $T_{Merge}(n) = n$.
Dica 2: $\mathbf{2^{log2(n)} = n}$, onde log2(n) corresponde a logaritmo de base 2.

Equação recorrente:

$$T(n) = \begin{cases} c, n \leqslant 1 \\ 2T(n/2) + dn, n > 1 \end{cases}$$

Método da substituição:

$$\begin{aligned} T(n) &= 2T(n/2) + dn \\ &= 2(2T(n/4) + dn/2) + dn \\ &= 4T(n/4) + dn + dn \\ &= 4(2T(n/8) + dn/4) + 2dn \\ &= 8T(n/8) + dn + 2dn \\ &= 8T(n/8) + 3dn \\ &\;\;. \\ &\;\;. \\ &\;\;. \end{aligned}$$

$$T(n) = 2^i T(n/2^i) + i(dn)$$

Sendo *i = log n* (base 2) (para eliminar a recorrência),

$$T(n) = 2^{\log(n)} T(n/2^{\log(n)}) + dn.\log(n)$$

$$T(n) = 2^{\log(n)} T(n/n) + dn.\log(n)$$

$$T(n) = n.c + dn.\log(n)$$

$$T(n) = dnlog(n) + cn$$

Então:

T(n) = O(n log n)