PUCPR
GRUPO MARISTA

SOMATIVA

Disciplina:	Complexidade de Algoritmos	**Ano/Semestre**:	2024/1
Professor:	Edson Emílio Scalabrin	**Metodologia:**	PBL
Resultado de Aprendizagem:	**RA1:** Emprega técnicas de contagem, soma e prova matemática na análise de algoritmos interativos para problemas de computação, de acordo com o contexto de aplicação e preceitos éticos.		
Indicador de desempenho:	**ID01**: Contabiliza instruções de algoritmos. **ID04**: Aplica notação assintótica. **ID05**: Análise complexidade termos de pior caso, melhor caso e caso médio.		

Problema 01: Dado o algoritmo de ordenação por inserção a seguir:

```
INSERTION-SORT(A)
1 for j ← 2 to comprimento[A]
2       do chave ← A[j]
3          ▷ Inserir A[j] na seqüência ordenada A[1..j − 1].
4          i ← j − 1
5          while i > 0 e A[i] > chave
6             do A[i + 1] ← A[i]
7                i ← i − 1
8          A[i + 1] ← chave
```

O primeiro passo para analisar sua complexidade consiste na contagem do número de vezes que uma instrução é executada. Cada instrução tem um custo associado.

A contagem de cada instrução do algoritmo de ordenação por inserção se encontra a seguir.

INSERTION-SORT(A)	*custo*	*vezes*
1 **for** $j \leftarrow 2$ **to** *comprimento*[A]	c_1	n
2 **do** *chave* $\leftarrow A[j]$	c_2	$n-1$
3 ▷ Inserir $A[j]$ na seqüência ordenada $A[1..j-1]$.	0	$n-1$
4 $i \leftarrow j-1$	c_4	$n-1$
5 **while** $i > 0$ e $A[i] >$ *chave*	c_5	$\sum_{j=2}^{n} t_j$
6 **do** $A[i+1] \leftarrow A[i]$	c_6	$\sum_{j=2}^{n} (t_j - 1)$
7 $i \leftarrow i-1$	c_7	$\sum_{j=2}^{n} (t_j - 1)$
8 $A[i+1] \leftarrow$ *chave*	c_8	$n-1$

Por uma questão de simplificação,

$$S = \sum_{j=2}^{n} (t_j - 1), para\ n = 4\ a\ somatório\ seria:$$

$$S = 1 + 2 + 3$$

Sabe-se que a soma dos números inteiros positivos é dada pela fórmula F1 a seguir:

$$S = \sum_{i=1}^{n} i \qquad sua\ fórmula\ fechada\ é \qquad F1 = \frac{n(n+1)}{2}$$

Dado que T(n), abaixo, é a soma das instruções do algoritmo de ordenação por inserção, pode-se reescrever T(n) como um polinômio $an^2 + bn + c$, onde a, b e c constantes que, mais uma vez, dependem dos custos de instrução c_i.

$$T(n) = c_1 n + c_2 (n-1) + c_4 (n-1) + c_5 \sum_{j=2}^{n} t_j + c_6 \sum_{j=2}^{n} (t_j - 1) + c_7 \sum_{j=2}^{n} (t_j - 1) + c_8 (n-1).$$

Tomando como ponto de partida T(n), **pede-se para**:

a) Rescrever T(n) na forma de $an^2 + bn + c$, *para facilitar o processo vamos assumir o valor 1 para cada uma das constantes:* $C_1, C_2, ..., C_8$

b) Indicar a expressão que representa o MELHOR CASO

c) Indicar a expressão que representa o PIOR CASO

d) Verificar f(n) = $\Theta(n^2)$

Problema 02: Dado o algoritmo de *bolha*, escrito em Python, a seguir:

```
def troca(vetor, i, j):
    aux = vetor[i]                  # C4.1
    vetor[i] = vetor[j]             # C4.2
    vetor[j] = aux                  # C4.3

def bolha(vetor):
    for i in range(0, len(vetor) - 1, 1):       # C1
        for j in range(i+1, len(vetor), 1):     # C2
            if vetor[ i ] > vetor[ j ]:         # C3
                troca(vetor, i, j)              # C4

X = [99,88,77,66,55,44,33,22,11]
bolha(X)
```

Pede-se para:

a) realizar a contagem de cada instrução do algoritmo *bolha*, indicando a contagem para cada linha, associando aos custos C_1, C_2, ..., C_{10}. Nota: realize a contagem apenas sobre as instruções claramente executadas. Explique os casos não incluídos na contagem.
b) expressar a soma T(n) das instruções do algoritmo de ordenação *bolha*, indicando cada parcela do somatório, associando cada parcela ao custo C_i.
c) encontrar a forma fechada de T(n)
d) Verificar **f(n) = $\Theta(n^2)$**

IMPORTANTE: para ambos problemas, a resolução de cada item solicitado deve ser mostrada de forma detalhada; não deixei nada como subentendido, ou seja, explique/explicite suas escolhas/hipóteses.