
**PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO GRANDE DO SUL**
**Escola Politécnica**

| | |
|---|---|
| **Curso:** Bacharelado em Engenharia de Computação<br>**Disciplina:** Algoritmos e Estruturas de Dados I<br>**Turma:** 14 **Data:** ___/___/2023<br><br>**Nome:** _____ | G1: P1 [X] P2 [ ]<br>Substituição [ ]<br>G2 [ ]<br>**Nota:** _____<br>**Prof. Roland Teodorowitsch** |

# Modelo de P1 da disciplina de Algoritmos e Estruturas de Dados I

Esta prova contém 4 página(s) e 9 questões, somando um um total de 10 pontos.
Questões adaptadas da P1 elaborada em 2023/1 pelo prof. Iaçanã Ianiski Weber.

1. (1 ponto) Analise os trechos de algoritmos abaixo e identifique suas respectivas classes de complexidade, marcando a única opção adequada para cada algoritmo.

   (a) Algoritmo I:

```
for (i=0; i < n; i++)
    for (j=0; j < n; j++)
        a += i*3 + aa[i][j];
```

   A. *quadrático* B. $\log n$ C. *exponencial* D. $n \log n$ E. *constante* F. *cúbico*

   (b) Algoritmo II:

```
for (i=1; i < n; i=i+i)
    b = b >> i;
```

   A. *quadrático* B. $\log n$ C. *exponencial* D. $n \log n$ E. *constante* F. *cúbico*

   (c) Algoritmo III:

```
for (i=0; i < n; i++)
    for (j=0; j < n; j++)
        for (k=0; k < n; k++)
            c += i + j + k;
```

   A. *quadrático* B. $\log n$ C. *exponencial* D. $n \log n$ E. *constante* F. *cúbico*

   (d) Algoritmo IV:

```
for (i=0; i < n; i++)
    for (j=i; j < i+5; j++)
        for (k=0; k < n; k++)
            d++;
```

   A. *quadrático* B. $\log n$ C. *exponencial* D. $n \log n$ E. *constante* F. *cúbico*

   (e) Algoritmo V:

```
for (i=0; i < 127; i++)
    for (j = 0; j < 127; j++)
        e[i][j] = 0;
```

   A. *quadrático* B. $\log n$ C. *exponencial* D. $n \log n$ E. *constante* F. *cúbico*

2. (.5 ponto) Considere as seguintes funções:

$$f_1(n) = O(n) \qquad f_2(n) = O(\log n) \qquad f_3(n) = O(2^n) \qquad f_4(n) = O(n^2)$$

A sequência que apresenta as funções acima ordenadas, pela sua taxa de crescimento, de forma crescente é:

A. $f_2 - f_1 - f_4 - f_3$
B. $f_3 - f_4 - f_1 - f_2$
C. $f_1 - f_3 - f_2 - f_4$
D. $f_1 - f_2 - f_3 - f_4$
E. $f_2 - f_1 - f_3 - f_4$

3. (1 ponto) Qual a notação $O$ dos algoritmos que tem as seguintes taxas de crescimento assintótica?
   (a) $3n \log n + 2n + 5$
   (b) $1000n \log n + 15n^3$
   (c) $8 \log n + n$
   (d) $500n^5 + 2^n$
   (e) $10 \log n + 1521$
   (f) $3n + 4500n$
   (g) $3n + 753 \log n + 4$

4. (.5 ponto) Para o método de ordenação *Quicksort*, a ordem de complexidade do pior caso e do caso médio, respectivamente, é?

A. $O(n^2)$ e $O(n^2)$
B. $O(n^2)$ e $O(n \log n)$
C. $O(n \log n)$ e $O(n^2)$
D. $O(n \log n)$ e $O(n \log n)$
E. $O(n \log n)$ e $O(n)$

5. (.5 ponto) Considere o vetor de números inteiros:

$$\{833, 93, 743, 752, 427, 608, 442\}$$

Um algoritmo de ordenação é utilizado para dispor os elementos do vetor em ordem crescente. A seguir é exibido o conteúdo do vetor ao final de cada iteração que realiza movimentações de elementos durante a execução do algoritmo.

$$\{93, 743, 752, 427, 608, 442, 833\}$$
$$\{93, 743, 427, 608, 442, 752, 833\}$$
$$\{93, 427, 608, 442, 743, 752, 833\}$$
$$\{93, 427, 442, 608, 743, 752, 833\}$$

As movimentações de elementos obtidas neste caso evidenciam que o algoritmo utilizado é:

A. QuickSort
B. InsertionSort
C. BubbleSort
D. MergeSort

6. (1 ponto) Considere uma lista de inteiros baseada em arranjo, implementada em C++ usando a classe a seguir:

```
class IntArrayList {
private:
  int numElements; // Número de elementos armazenado na lista (total ocupado)
  int maxElements; // Número máximo de elementos da lista (total alocado)
  int *list;       // Endereço da área alocada para a lista
public:
  // ...
  bool isSorted() const;  // Método a ser implementado!
};
```

Implemente o método `bool isSorted()`, que testa se a lista de inteiros baseada em arranjo está ordenada ou não. O método deve retornar `true` se o vetor estiver ordenado ou `false`, em caso contrário. A implementação deste método deve ter complexidade máxima de $O(n)$.

7. (1.5 pontos) Considere a execução do seguinte trecho de código:

```
stack *s = initStack();
queue *q = initQueue();
push(s, 55);
push(s, 99);
enqueue(q, 88);
enqueue(q, 56);
enqueue(q, 46);
```

A partir desta configuração inicial, apresente o conteúdo das estruturas `q` e `s` após cada chamada a seguir:

```
push(s, dequeue(q));
push(s, head(q));
enqueue(q, dequeue(q));
push(s, pop(s));
enqueue(q, pop(s));
```

8. (2 pontos) Implemente os métodos `enqueue` e `dequeue`, que, respectivamente, **inserem** e **removem** inteiros de uma **fila** que utiliza uma estrutura de simplesmente encadeada de nodos. Implemente estes métodos para a seguinte classe em C++:

```
class IntLinkedQueue {
private:
  struct Node {
    int data;
    Node *next;
    Node(int d) {  data = d;  next = nullptr;  }
  };
  Node *head, *tail;  // Ponteiros para início e fim da fila
public:
  // ...
  void enqueue(const int e);  // Método a ser implementado!
  bool dequeue(int &e);       // Método a ser implementado!
};
```

9. (2 pontos) Considere uma lista duplamente encadeada de inteiros, definida pela classe a seguir, e implemente o método `int lastIndexOf(const int info)` para esta classe.

```
class IntDoubleLinkedList {
private:
  struct Node {
    int data;
    Node *prev, *next;
    Node(int d) {  data = d;  prev = next = nullptr;  }
  };
  int numElements;    // Número de elementos na lista
  Node *head, *tail;  // Ponteiros para início e fim da lista
public:
  // ...
  int lastIndexOf(const int info) const;  // Método a ser implementado!
};
```

Esse método recebe como parâmetro um inteiro (`info`) e procura na lista duplamente encadeada pela última ocorrência deste valor, retornando o índice dessa ocorrência, ou -1 se a lista não contiver a informação. Observe que, nesta implementação de lista duplamente encadeada, há um campo com o número de elementos da lista, o que poderá ser útil.