PUC Minas

Algoritmos e Estruturas de Dados II
Lista de Exercícios de Revisão para a 1ª Prova - Não avaliativa

1) (Algoritmos e Estruturas de Dados , Complexidade de algoritmos, Analista de Sistemas Pleno, Processos, Petrobrás, CESGRANRIO). A respeito de funções e algoritmos, assinale a afirmativa correta.

a) O limite inferior de um algoritmo (O) é utilizado para a análise do pior caso de sua execução.

b) Uma função f(n) domina assintoticamente g(n), se existem duas constantes positivas c e n0, tais que, para n = n0, temos que $|g(n)| = c|f(n)|$.

c) A função $f(5 * \log_2 n)$ é $\Theta(n)$.

d) A função $f(5n^3 + 2n^2)$ é $O(n)$.

e) Se uma função g(n) é o limite superior justo de outra função f(n), então f(n) é O(g(n)) e g(n) é O(f(n)).

2) (Sistemas de Informação , Complexidade de algoritmos, Analista de Sistemas, TJ SP, VUNESP). Considerando o conceito de Complexidade de Algoritmos, representado por O(função), assinale a alternativa que apresenta, de forma crescente, as complexidades de algoritmos.

a) $O(2^n)$; $O(n^3)$; $O(n^2)$; $O(\log_2 n)$; $O(n*\log_2 n)$.
b) $O(n^2)$; $O(n^3)$; $O(2^n)$; $O(\log_2 n)$; $O(n*\log_2 n)$.
c) $O(n^3)$; $O(n^2)$; $O(2^n)$; $O(n*\log_2 n)$; $O(\log_2 n)$.
d) $O(\log_2 n)$; $O(n*\log_2 n)$; $O(n^2)$; $O(n^3)$; $O(2^n)$.
e) $O(n*\log_2 n)$; $O(\log_2 n)$; $O(2^n)$; $O(n^3)$; $O(n^2)$.

3) (Algoritmos e Estrutura de Dados, Complexidade de algoritmos, Técnico Judiciário em Tecnologia da Informação, TRT 19ª Região, FCC). Considere os seguintes algoritmos e suas complexidades na notação Big O:

Algoritmo A: O(log n)
Algoritmo B: $O(n^2)$
Algoritmo C: O(n * log n)

Considerando-se o pior caso de execução desses algoritmos, é correto afirmar que o algoritmo

a) A é o menos eficiente.
b) C é o menos eficiente.
c) A não é o mais eficiente nem o menos eficiente.
d) B é o menos eficiente.
e) C é o mais eficiente.

4) Dois vetores ordenados, contendo, cada um deles, N números inteiros, precisam ser unidos em outro vetor maior, que conterá os 2N números, que também serão armazenados de forma ordenada. A complexidade de tempo de melhor caso desse processo será, então,

a) O(1), pois é necessário fazer apenas uma cópia simples de cada um dos elementos originais.
b) O(log N), pois usa-se a busca binária para determinar qual será o próximo elemento copiado para o vetor de destino.
c) O(N), pois precisa-se fazer uma cópia de cada um dos elementos originais, o que implica uma varredura completa de cada vetor de origem.
d) O(N * log N), pois é necessário fazer uma busca de cada elemento para depois inseri-lo no vetor de destino.
e) O(N^2), pois, como há dois vetores, precisa-se fazer dois laços de forma aninhada (um dentro do outro), gerando uma multiplicação das quantidades de elementos.

5) Depois de pensar sobre determinado problema, João fez um rascunho de uma função, produzindo o algoritmo em pseudocódigo abaixo:

```
Funcao calculo_A(n)
K=0
se (n>1000) então
para i de 1 ate n faça
        para j de 1 ate n faça
                k = (k*k) + j
        fim para
fim para
senão
para j de 1 ate n faça
        k = (k*j) +2
fim para
fim se
retorne k
```

É correto afirmar que o algoritmo é (assinale todas as opções corretas):

a) O(n)
b) O(n^2)
c) O(n^3)
d) O(log n)
e) O(n * log n)

6) (CESPE / CEBRASPE - 2022 - DPE-RO - Analista da Defensoria Pública – Programação) A complexidade do algoritmo2 abaixo é (assinale todas as alternativas verdadeiras):

```
função algortimo2(n)
0: inicio
1: se n = 0 então
2:   retorne 0
3: senão
4:   se n = 1 então
5:    retorne 1
6:   senão
7:    penultimo = 0
8:    ultimo =1
9:    para i =2 até n faça
10:     atual = penultimo + ultimo
11:     penultimo = ultimo
12:     ultimo = atual
13:   fim para
14:   retorne atual
15:  fim se
16: fim se
17: fim
```

a) $O(2^n)$.
b) $O(n^2)$.
c) $O(n)$.
d) $O(\log(n))$.
e) $O(n*\log(n))$.

7) (COMPERVE - 2016 - UFRN - Analista de Tecnologia da Informação) Analise o algoritmo a seguir:

```
Algoritmo2:

função algo(n)
   i <- 1
   j <- 0
   para k de 1 até n faça
      x  <-  i + j
      i  <-  j
      j  <-  x
   retorne j
```

Em relação ao algoritmo exposto, é correto afirmar que:

a) o algoritmo é $\Theta(2^n)$.
b) o algoritmo é $\Theta(n)$.
c) o algoritmo é $\Theta(n^2)$.
d) o algoritmo é $\Theta(n^3)$.

8) (ACEP - 2019 - Prefeitura de Aracati - CE - Analista de Sistemas) Considere o trecho de pseudocódigo abaixo:

```
para i ← 0 até n passo 1 faça
  para i ← 0 até n passo 1 faça
  início
    p[i][j] ← 0;
    k ← 0;
    enquanto k < n faça
    início
      p[i][j] ← p[i][j] + a[i][k] * b[k][j];
      k ← k + 1;
  fim
```

A ordem de complexidade do trecho em questão é:

a) O(n)
b) $O(n^2)$
c) O(n*log n)
d) $O(n^3)$

9) (IF-SP - 2019 – Informática) A notação O é amplamente utilizada como ferramenta de análise para calcular a complexidade computacional de um algoritmo caracterizando seu tempo de execução e limites espaciais em função de um parâmetro *n*.
Considere o código de um método em Java contendo o algoritmo a seguir:

```
public static boolean saoDisjuntos(int[] a, int[] b) {
        for (int i=0; i < a.length; i++)
                for (int j=0; j < b.length; j++)
                        if (a[i] == b[j]) return false;
        return true;
}
```

Se cada um dos arranjos ***a*** e ***b*** do algoritmo acima tem tamanho *n*, então, o pior caso para o tempo de execução desse método é (assinale todas as alternativas corretas):

a) $O(2^n)$
b) O(n)
c) $O(n^2)$
d) O(log n)

10) (CCV-UFC - 2013 - UFC - Analista de Tecnologia da Informação - Arquitetura e Desenvolvimento de *Software*) O algoritmo a seguir, descrito em pseudocódigo, pode ser utilizado para ordenar um vetor ***A***[0..*n*].

```
Algoritmo (A[], n)

    VARIAVEIS
    var i, j, elemento;

    PARA j <- 1 ATÉ n FAÇA
        elemento <- A[j];
        i <- j – 1;

        ENQUANTO ((i >= 0) E (A[i] > elemento)) FAÇA
            A[i+1] <- A[i]
            A[i] <- elemento
            i <- i–1
        FIM_ENQUANTO

    FIM_PARA

FIM
```

No pior caso, a complexidade deste algoritmo é:

a) $O(n^2)$
b) $O(1)$
c) $O(n)$
d) $O(\log n)$
e) $O(n*\log n)$

11) Dada a sequência de números: 3 4 9 2 5 8 2 1 7 4 6 2 9 8 5 1, ordene-a em ordem não decrescente segundo os seguintes algoritmos, apresentando a sequência obtida após cada passo do algoritmo:

    a) *MergeSort*.
    b) *QuickSort*.
    c) *HeapSort*.

12) Dados três vetores ordenados, implemente uma função merge3_sort que intercale e retorne o vetor resultante ordenado (sugestão: baseie-se no algoritmo do *mergesort* original). Qual a complexidade desse algoritmo?

13) Faça uma tabela comparativa dos algoritmos vistos em aula considerando os seguintes aspectos: número de comparações (melhor, pior e caso médio), número de movimentações, facilidade de implementação e uso de memória auxiliar.

14) Faça um programa que leia *n* nomes e ordene-os utilizando o algoritmo *heapsort*. No final, o algoritmo deve mostrar todos os nomes ordenados.