Universidade Federal do Pará
Instituto de Ciências Exatas e Naturais
Faculdade de Computação
Análise de Algoritmos

Lista de Exercícios
Fundamentos

1. Sejam f, g e h funções reais positivas da variável inteira n. Com respeito às notações assintóticas, avalie as afirmativas abaixo.

I. $f = O(g)$ se e somente se $g = \omega(f)$.

II. $f = \Theta(g)$ se e somente se $f = O(g)$ e $f = \Omega(g)$.

III. $f = o(g)$ e $g = o(h)$ implicam $f = o(h)$.

IV. $nlog(n) + k = O(n)$, dado que k é uma constante positiva.

A análise permite concluir que somente

(A) a afirmativa III é verdadeira.
(B) as afirmativas I e IV são verdadeiras.
(C) as afirmativas II e III são verdadeiras.
(D) as afirmativas I, II e III são verdadeiras.
(E) as afirmativas II, III e IV são verdadeiras.

2. Um limite inferior para um problema P é uma função f, tal que a complexidade no tempo de pior caso de qualquer algoritmo que resolva P é $\Omega(f)$. Isto quer dizer que

(A) todo algoritmo que resolve P efetua $\Theta(f)$ passos.
(B) se existir um algoritmo A que resolve P com complexidade $O(f)$, então A é denominado algoritmo ótimo para P.
(C) todo algoritmo que resolve P é recursivo.
(D) todo algoritmo que resolve P é interativo.
(E) todo algoritmo que resolve P tem complexidade linear no mínimo.

3. O algoritmo para calcular um polinômio $a_n x^n + a_{n-1} x^{n-1} + ... + a_1 x + a_0$, quando $x = c$, pode ser expresso em pseudocódigo por:

```
 Polinomio(A, n, c)
1. power = 1
2. y = A[0]
3. para i = 1 até n faça
4.    power = power * c
5.    y = y + A[i] * power
6. retornar (y)
```

Seja $T(n)$ o tempo de execução do algoritmo acima descrito para as entradas $A[a_0...a_n]$, n e c. A ordem de $T(n)$ usando a notação Little-o é

(A) $T(n) = o(c)$.
(B) $T(n) = o(log(n))$.
(C) $T(n) = o(\sqrt{n})$.
(D) $T(n) = o(n)$.
(E) $T(n) = o(n^2)$.

4. Considere o algoritmo abaixo.

```
 PROC(n)
1. se n <= 1 então
2.    retornar (2)
3. senão
4.    retornar (PROC(n/2) + PROC(n/2))
5. fim se
```

Assinale a alternativa que indica o valor retornado pelo algoritmo considerando a entrada $n = 64$.

(A) 128.
(B) 130.
(C) 1.024.
(D) 4.096.
(E) 4.160.

5. O tempo de execução $T(n)$ de um algoritmo, em que n é o tamanho da entrada, é dado pela equação de recorrência $T(n) = 8T(n/2) + qn$ para todo $n > 1$. Dado que $T(1) = p$, e que p e q são constantes arbitrárias, a complexidade do algoritmo é

(A) $\Theta(log(n))$.
(B) $\Theta(n)$.
(C) $\Theta(nlog(n))$.
(D) $\Theta(n^2)$.
(E) $\Theta(n^3)$.

6. Resolva as relações de recorrência abaixo pelo Teorema Mestre.

(a) $T(1) = 1$.
$T(n) = 4T(n/4) + n$ para $n > 1$.

(b) $T(1) = 1$.
$T(n) = 7T(n/2) + n^2$ para $n > 1$.

(c) $T(1) = 1$.
$T(n) = 3T(n/9) + 2n$ para $n > 1$.

7. As definições recursivas apresentadas abaixo descrevem o tempo de execução de dois algoritmos recursivos: A e B:

$T_A(1) = 1$ e $T_A(n) = 2T_A(n-1) + 2$ para $n \geq 2$.

$T_B(1) = 1$ e $T_B(n) = T_B(n/2) + n$ para $n \geq 2$.

Assinale a alternativa correta.

(A) Os algoritmos não são eficientes.
(B) Os algoritmos são assintoticamente equivalentes.
(C) O algoritmo B tem complexidade exponencial no tempo.
(D) O algoritmo A é mais eficiente assintoticamente que o algoritmo B.
(E) O algoritmo B é mais eficiente assintoticamente que o algoritmo A.

8. Considere o algoritmo A abaixo.

```
Algoritmo A(n)
Entrada: n, inteiro, n > 0.
{
   se n = 1
       retornar (1);
   senão
       retornar (2 * A(n /2) + 1);
}
```

A complexidade no tempo de pior caso do algoritmo A é

(A) linear.
(B) logarítmica.
(C) quadrática.
(D) exponencial.
(E) $nlog(n)$.

9. O algoritmo recursivo abaixo soma os n primeiros números naturais.

```
Algoritmo Soma(n)
Entrada: n, inteiro, n > 0.
{
   se n = 1
       retornar (1);
   senão
       retornar (Soma(n - 1) + n);
}
```

A complexidade no tempo do algoritmo é

(A) linear.
(B) logarítmica.
(C) quadrática.
(D) exponencial.
(E) $nlog(n)$.

10. Suponha que f e g são funções reais positivas da variável inteira n. Verifique se as seguintes afirmações são verdadeiras ou falsas. Justifique sua resposta caso a afirmativa seja falsa.

(a) Se $f(n) = O(g(n))$, então $2^{f(n)} = O(2^{g(n)})$.

(b) $f(n) = O((f(n))^2)$.

(c) Se $f(n) = O(g(n))$, então $g(n) = \Omega(f(n))$.

(d) $f(n) + O(f(n)) = \Theta(f(n))$.

11. Escreva um algoritmo (em pseudocódigo) que realize busca binária de forma iterativa e o implemente numa linguagem de programação a sua escolha. Construa um gráfico mostrando a relação valor de entrada x tempo de execução do algoritmo implementado. Considerando uma análise assintótica em pior caso, explique se o desempenho do algoritmo implementado é superior, inferior ou igual ao do algoritmo que implementa busca binária de forma recursiva.

Para as questões **12** e **13**, entregue os seguintes itens considerando o algoritmo implementado para resolver o problema computacional:

- Uma captura de tela que mostre a compilação correta na plataforma de teste;
- O cálculo da complexidade no tempo usando notação assintótica; e
- Um gráfico ilustrando a análise empírica, ou seja, a relação valor de entrada x tempo de execução.

12. Resolva o seguinte problema computacional:

Problema: Ajude a Federação (#1588)
https://www.beecrowd.com.br/judge/pt/problems/view/1588

13. Resolva o seguinte problema computacional de forma **recursiva**:

Problema: A Lenda de Flavious Josephus (#1030)
https://www.beecrowd.com.br/judge/pt/problems/view/1030