Universidade Federal de Campina Grande
Centro de Engenharia Elétrica e Informática
Departamento de Sistemas e Computação
Graduação em Ciência da Computação

Análise assintótica e complexidade algoritmica

1. Quais as vantagens de se usar o método analítico para avaliar o desempenho de algoritmos?

2. Porque nos preocupamos com o pior caso de um algoritmo?

3. Escreva um algoritmo iterativo para calcular o n-ésimo termo da sequencia de Fibonacci. Em seguida calcule a complexidade do algoritmo.

4. Proponha um algoritmo que calcule o MDC de dois numeros dados e encontre o tempo de execução desse algoritmo. Dica: procure entender a idéia do algoritmo de Euclides e tente estimar seu tempo observando o argumento que modifica a cada chamada recursiva. Você pode propor um limite inferior e um superior para sua função. Considere exemplos com os números 20,15 e depois com os números 92,72.

5. Sejam $f(n)$ e $g(n)$ funções de inteiros nos inteiros positivos. O que significa dizer que $f(n) = O(g(n))$?

6. Considere as funções $f(n) = 3n^2 - n + 4$ e $g(n) = nlogn + 5$. Podemos afirmar que $f(n) + g(n) = O(n^2)$?

7. Julgue as seguintes afirmações

   1. If $f(n) = \Theta(g(n))$ and $g(n) = \Theta(h(n))$, then $h(n) = \Theta(f(n))$
   2. If $f(n) = O(g(n))$ and $g(n) = O(h(n))$, then $h(n) = \Omega(f(n))$
   3. If $f(n) = O(g(n))$ and $g(n) = O(f(n))$ then $f(n) = g(n)$
   4. $\frac{n}{100} = \Omega(n)$

8. Qual a relação entre as funções logarítmicas: $f(n) = log_2 n$ e $g(n) = log_3 n$?
9. Considere um algoritmo composto por comandos primitivos e dois loops aninhados com contadores independentes. O loop mais externo tem M repetições enquanto que o loop interno tem N repetições. O que podemos afirmar sobre o tempo do algoritmo? Qual seria o seu tempo se os loops fossem invertidos (loop mais interno se tornasse o mais externo e vice-versa)?
10. Qual a complexidade da expressão: $\sum_{i=0}^{n} i^3$? Dica: o somatório pode ser expresso pelo polinômio: $n^2(n + 1)^2/4$.

11. Quais das seguintes alternativas são verdadeiras para todas as funções $f(n)$ positivas? Dica: tente fazer $f(n)$ sendo polinomial, exponencial e logarítmica, quando couber.

    a. $f(n^2) = \theta(f(n)^2)$, quando $f(n)$ é polinomial
    b. $f(n^2) = o(f(n)^2)$
    c. $f(n^2) = O(f(n)^2)$, quando $f(n)$ é exponencial
    d. $f(n^2) = \Omega(f(n)^2)$

12. Calcula a complexidade exata das seguintes funções abaixo

```
function CALC( )
    a = 0
    i = N
    while i > 0 do
        a = a + i
        i = i/2
```

```
function CRAZY_FUNC(n,x,y)
    for i = 0, i < n, ++i do
        if x < y then
            for k = 0, k < n, ++k do
                print(k)
        else
            print(i)
```

```
function ALG(n)
    j = 1
    while j < n do
        j = j * 2
```

Análise de algoritmos recursivos

1. Utilize os métodos iterativo e mestre para resolver as seguintes equações abaixo:
    - $T(n) = 4T\left(\frac{n}{2}\right) + n$
    - $T(n) = 4T\left(\frac{n}{2}\right) + n^2$
    - $T(n) = 4T\left(\frac{n}{2}\right) + n^3$
2. Qual a solução de relação de recorrência $T(2n) = T(n) + n$?
3. O limite $\Omega(n. \log n)$ é uma boa solução para a relação $T(n) = T\left(\frac{n}{3}\right) + T\left(\frac{2n}{3}\right) + n$? Argumente usando o método iterativo.
4. Use o método mestre para calcular a solução da relação de recorrência $T(n) = T\left(\frac{n}{2}\right) + \Theta(1)$.

5. Suponha que, para entradas de tamanho n, voce tenha de escolher um entre dois algoritmos.
   a) O algoritmo A resolve problemas dividindo-os em dois subproblemas de tamanho n-1, recursivamente resolve cada subproblema e então combina as soluções em tempo O(1).
   b) Algoritmo B resolve problemas dividindo-os em nove subproblemas de tamanho n/3, recursivamente resolve cada subproblema e então combina as soluções em tempo O(n^2).

   Qual o consumo de tempo desses algoritmos? Qual algoritmo é assintoticamente mais eficiente?
6. A Torre de Hanoi é um problema clássico em computação. Ela é composta de três hastes (origem, intermediaria, destino) e discos de tamanhos diferentes com um furo no meio que encaixam na haste de origem. Sabe-se que, **um disco maior jamais pode ficar sobre um disco menor** (*propriedade que deve valer sempre no jogo*). O objetivo do jogo é fazer trocas (um disco de cada vez) usando as hastes até que todos os discos sejam movidos da haste de origem para a de destino sem ferir a propriedade do jogo. Um algoritmo para resolver este problema seria: se existe apenas um disco, mova ele da haste de origem para a haste de destino. Caso existam n discos, aplique este algoritmo nos $n - 1$ discos de cima da haste de origem para movê-los para a haste temporária, depois mova o disco restante da haste de origem para a haste de destino e em seguida aplique este algoritmo nos os $n - 1$ discos para movê-los da haste temporária para a haste de destino. Determine a relação de recorrência desse algoritmo e encontre sua solução.
7. Considere o seguinte algoritmo.

```
function ALGO(n)
    if n ≤ 1 then
        return 1
    for i = 1, ..., 8 do
        z = ALGO(n/2)
    for i = 1, ..., n^3 do
        z = 0
```

   a. Escreva a recorrência para o algoritmo.
   b. Calcule a solução usando os métodos iterativo e mestre.
   c. Troque o "8" por "7" e calcule a solução usando o método iterativo.

8. Calcule o tempo de um algoritmo cuja relação de recorrência é $T(n) = T(\sqrt[3]{n}) + \Theta(1)$.