Universidade Federal do Ceará - Campus Quixadá
QXD0010 – Estrutura de Dados – Turma 03A – 2021.1
Prof. Atílio Gomes

Noções de Análise de Algoritmos

1. Considere o seguinte Algoritmo 1 que apresenta a função SOMA-MATRIZES, que calcula a matriz quadrada C de dimensão $n \times n$ que é a soma de duas matrizes A e B ambas de mesma dimensão $n \times n$. Determine a função $f(n)$ que dá o número de passos que esse algoritmo executa em função do parâmetro n. Qual a complexidade deste algoritmo? Justifique a sua resposta?

Algoritmo 1 Soma de duas matrizes quadradas

```
Função SOMA-MATRIZES(int *A, int *B, int *C, int n)
    para i ← 0 até n − 1 faça
        para j ← 0 até n − 1 faça
            C[i][j] ← A[i][j] + B[i][j]
        end para
    end para
    Retorne C
end Função
```

2. Considere o Algoritmo 2 que multiplica duas matrizes quadradas A e B. Determine a função $f(n)$ que dá o número de passos que esse algoritmo executa em função do parâmetro n. Qual a complexidade deste algoritmo? Justifique a sua resposta?

Algoritmo 2 Multiplicação de duas matrizes quadradas

```
Função MULTIPLICA-MATRIZES(int *A, int *B, int *C, int n)
    para i ← 0 até n − 1 faça
        para j ← 0 até n − 1 faça
            C[i][j] ← 0
            para i ← 0 até n − 1 faça
                C[i][j] ← C[i][j] + A[i][j] × B[i][j]
            end para
        end para
    end para
    Retorne C
end Função
```

3. Sejam as funções de complexidade $a(n) = n^2 - n + 549$ e $b(n) = 49n + 49$ referentes a certos algoritmos A e B, respectivamente. Para que valores de n é melhor aplicar o Algoritmo A?

4. O que é a complexidade de pior caso de uma algoritmo? E o que é a complexidade de melhor caso? Qual a diferença entre elas?

5. Faça um algoritmo que verifique se os elementos de um vetor estão ordenados de forma ascendente. Qual a complexidade de pior caso e melhor caso do seu algoritmo? Justifique suas respostas.

6. Para cada uma das afirmações abaixo, justifique formalmente (usando definições, manipulações algébricas e implicações) se for verdade ou dê um contraexemplo se for falso.

   (a) $3n = O(n)$

   (b) $2n^2 - n = O(n^2)$

   (c) $\log 8n = O(\log 2n)$

   (d) $2^{n+1} = O(2^n)$

   (e) Se $f(n) = 17$, então $f(n) = O(1)$

   (f) Se $f(n) = 3n^2 - n + 4$, então $f(n) = O(n^2)$

7. O que significa dizer que um algoritmo executa em tempo proporcional a n?

8. Por muitas vezes damos atenção apenas à análise do pior caso dos algoritmos. Explique o porquê.

9. Qual algoritmo você prefere: um algoritmo que requer n^5 passos ou um que requer 2^n passos? Justifique sua resposta.

10. O que significa dizer que $g(n)$ é $O(f(n))$?

11. Seja V um vetor composto por n inteiros distintos. Escreva um algoritmo de complexidade $O(n^2)$ que ordene os elementos do vetor em ordem crescente. Prove que seu algoritmo tem a complexidade exigida.

12. Nos casos a seguir proponha $g(n)$ e $h(n)$ tal que $f(n) = O(g(n))$ e $f(n) = \Omega(h(n))$. Encontre também constantes positivas m e c válidas conforme definições:

   (a) $f(n) = 3n^2 + 2n$

   (b) $f(n) = \log_2(n^2) + 11$

   (c) $f(n) = nlog_2(n+1)$

   (d) $f(n) = 3^{2n} + 5^n$

13. A sequencia de Fibonacci é uma sequência de elementos $f_0, f_1, \ldots, f_n$, definida do seguinte modo:

$$f_j = \begin{cases} j, & \text{se } 0 \leq n \leq 1; \\ f_{j-1} + f_{j-2}, & \text{se } j > 1. \end{cases}$$

   Elaborar um algoritmo, não recursivo, para determinar o elemento f_n da sequência, cuja complexidade seja linear em n e prove este fato.

14. Considere a seguinte sequencia de elementos $g_1, \ldots, g_n$ para um dado valor de k.

$$g_j = \begin{cases} j-1, & \text{se } 1 \leq j \leq k; \\ g_{j-1} + g_{j-2}, & \text{se } j > k. \end{cases}$$

Elaborar um algoritmo para determinar o elemento g_n da sequencia, cuja complexidade seja $O(n)$.

15. Determine a complexidade de pior caso dos algoritmos a seguir:

Algoritmo 3 Função F

```
Função F(int L[ ], int n)
    s ← 0
    para i ← 0 até n − 2 faça
        para j ← i + 1 até n − 1 faça
            if L[i] > L[j] then
                s ← s + 1
            fim if
        fim para
    fim para
    retorne s
fim Função
```

Algoritmo 4 Função G

```
Função G(int n)
    k ← 0
    enquanto n > 0 faça
        n ← n/2
        k ← k + 1
    fim enquanto
    retorne k
fim Função
```

Algoritmo 5 Função H

```
Função H(int L[ ], int n)
    if n > 1 then
        x ← H(L, n − 1)
        if x > L[n] then
            retorne x
        else
            retorne L[n]
        fim if
    else if n == 1 then
        retorne L[1]
    fim if
fim Função
```

16. Mostrar que o algoritmo da Torre de Hanoi, visto em sala, requer exatamente $2^n - 1$ movimentos de disco para terminar.

17. Considere a seguinte generalização do problema Torre de Hanói. O problema agora consiste em n discos de tamanhos distintos e quatro pinos, respectivamente, o de origem, o de destino e dois pinos de trabalho (auxiliares). De resto, o problema é como no caso de três pinos. Isto é, de início, os discos se encontram todos no pino de origem, em ordem decrescente de tamanho, de baixo para cima. O objetivo é empilhar todos os discos no pino-destino, satisfazendo às condições:

    (i) apenas um disco pode ser movido de cada vez;

    (ii) qualquer disco não pode ser jamais colocado sobre outro de tamanho menor.

    Elaborar um algoritmo para resolver essa generalização. Determinar o número de movimentos de disco efetuados.