Lista de Exercícios ACH-2002
Professor Marcos L. Chaim

1. Suponha um algoritmo A e um algoritmo B, com funções de complexidade de tempo $a(n) = n^2 - n + 549$ e $b(n) = 49n + 49$, respectivamente. Determine que valores de n pertencentes ao conjuntos dos números naturais para os quais A leva menos tempo para executar do que B. (Retirado de [3])

2. O que significa dizer que $f(n)$ é $O(f(n))$? (Retirado de [3])

3. Explique a diferença entre $O(1)$ e $O(2)$? (Retirado de [3])

4. Qual algoritmo você prefere: um algoritmo que requer n^5 passos ou um algoritmo que requer 2^n passos? (Retirado de [3])

5. Indique se as afirmativas a seguir são verdadeiras ou falsas e justifique suas respostas.

   (a) $2^{n+1} = O(2^n)$
   (b) $2^{2n} = O(2^n)$
   (c) $f(n) = O(u(n))$ e $g(n) = O(v(v)) \Rightarrow f(n) + g(n) = O(u(n) + v(n))$
   (d) $(n+1)^5$ é $O(n^5)$
   (e) $n^3 \log n$ é $\Omega(n^3)$

   (Retirado de [3])

6. Sejam duas funções não negativas $f(n)$ e $g(n)$. Diz-se que $f(n) = \Theta(g(n))$ se $f(n) = O(g(n))$ e $g(n) = O(f(n))$. Mostre que $\max(f(n), g(n)) = \Theta(f(n) + g(n))$.(Retirado de [3])

7. Resolva as equações de recorrência:

   (a) $T(n) = T(n-1) + c$, sendo c constante e $n > 1$; T(1)=0.
   (b) $T(n) = T(n-1) + 2^n$, para $n > 1$; T(0)=1.
   (c) $T(n) = cT(n-1)$, sendo c, k constantes e $n > 0$; T(0)=k.

   (Retirado de [3])

8. Escreva um método em Java para obter o maior e o segundo maior elemento de um arranjo. Apresente também a análise desse algoritmo (função de complexidade). Você acha o algoritmo do seu método eficiente? Por quê? Procure comprovar as suas respostas. (Retirado de [2])

9. Seja A um arranjo de $n + 1$ elementos que contém n elementos, ou seja, há ainda uma posição livre no arranjo. Considere o problema de inserir um novo elemento em A assumindo que A[0] > A[1] > A[2] > ... > A[n-1].

   (a) Apresente um limite inferior para esta classe de problemas.
   (b) Apresente uma prova informal para o limite inferior.

(c) Escreva um método em Java para resolver o problema acima. O algoritmo do seu método é ótimo?

(Retirado de [3])

10. Um arranjo A contém $n-1$ inteiros não-repetidos com valores entre $0 \ldots n-1$, ou seja, existe um número desta faixa de valores que não está em A. Escreva um método em Java com tempo de execução $O(n)$ para encontrar esse número. É permitido que você use espaço adicional que seja $O(1)$ além do próprio arranjo A. (Retirado de [2])

11. Escreva duas definições para uma função *soma* que, dados dois números inteiros positivos a e b, usando apenas as operações mais simples de incrementar 1 e decrementar 1 (suponha que as operações de adicionar e de subtrair mais de uma unidade não são disponíveis). A primeira definição deve usar um comando de repetição e segunda definição deve ser recursiva. (Retirado de [1])

12. Considere o trecho de programa abaixo:

```
...
 for(i = 0; i < n; i++)
  for( j = 0; j < m; j++)
    {
    // trecho de programa cujo custo é O(3)
    }
...
```

Qual a complexidade assintótica temporal do trecho de programa acima?

13. Considere o trecho de programa abaixo:

```
...
  for(i = 0; i < n; i++)
   {
    // trecho de programa cujo custo é O(1)
   }

  for( j = 0; j < m; j++)
    {
    // trecho de programa cujo custo é O(3)
    }
...
```

Qual a complexidade assintótica temporal do trecho de programa acima?

14. Considere o trecho de programa abaixo:

```
...
  for(i=n-1; i > 0 ; i--)
     for(j=1; j <= i; j++)
          {
          // trecho de programa cujo custo é O(1)
          }
...
```

Qual a complexidade assintótica temporal do trecho de programa acima?

15. Bill tem um algoritmo `buscaEmMatriz` para encontrar um elemento `x` em matriz `A` $n \times n$. O algoritmo faz iterações sobre as linhas de `A` e usa o algoritmo de busca seqüencial em cada linha até que `x` seja encontrado ou que todas as linhas tenham sido examinadas. Qual é o tempo de execução de pior caso do algoritmo `buscaEmMatriz` em função de n? O algortimo é linear? Justifique. (Adaptado de GoodRich & Tamassia, pag. 122)

16. Alan e Bill estão discutindo sobre o desempenho de seus algoritmos de ordenação. Alan afirma que seu algoritmo executado em tempo $O(n\ log_2\ n)$ é *sempre* superior ao algoritmo de Bill, que é $O(n^2)$. Isto é verdade? Justifique. (Adaptado de GoodRich & Tamassia, pag. 124)

17. *Sistemas criptográficos* típicos para a transmissão segura de dados são baseados no fato de que não são conhecidos algoritmos eficientes para fatoração de números inteiros muito grandes. Assim, se podemos representar uma mensagem secreta como um grande número primo p, podemos transmitir pela rede o número $r = p.q$, onde $q > p$ é outro grande número primo que serve como *chave de encriptação*. Um espião que obtenha o número r transmitido pela rede teria de fatorá-lo para descobrir a mensagem secreta p. Usar a fatoração para descobrir a mensagem é bastante difícil sem que se conheça a chave q. Para entender o porquê, considere o seguinte algoritmo simples de fatoração:

    Para cada inteiro p tal que $1 < p < r$, verifique se p divide r. Se dividir, imprima "A mensagem secreta é p!" e pare; caso contrário, continue.

    (a) Suponha que o espião use o algoritmo acima em um computador que possa realizar em 1 microssegundo (ou seja, em um milionésimo de segundo) uma operação de divisão entre dois inteiros de 1000 bits cada um. Forneça uma estimativa do tempo necessário (no pior caso) para decifra a mensagem secreta se r tem 100 bits.

    (b) Qual a complexidade de pior caso do algoritmo acima? Já que a entrada do algoritmo é um grande número r, supondo que o tamanho n da entrada é o número de bytes necessários para armazenar r, ou seja, $n = (log_2\ r/8)$, e que cada divisão toma tempo proporcional a $O(n)$. (Retirado de GoodRich & Tamassia, pag. 125)

Referências

[1] Carlos Camarão and Lucília Figueiredo. *Programação de Computadores em Java*. LTC Editora, Rio de Janeiro, 2003.

[2] Michael T. Goodrich and Robert Tamassia. *Estrutura de dados e Algoritmos em Java*. Editora Bookman.

[3] Nívio Ziviani. *Projeto de Algoritmos – com implementações em C e Pascal*. Thomson Editora, 2a. edition, 2004.