Departamento de Ciência da Computação
MATA52 - Análise e Projeto de Algoritmos
MATD74 - Algoritmos e Grafos
Professores: George Lima e Rafael Melo

Universidade Federal da Bahia
Lista 1

Introdução a Algoritmos

1. Cormen et al..: 2.1-2
2. Cormen et al..: 2.1-3
3. Cormen et al..: 2.1-4
4. Um palíndromo é uma palavra que possui a propriedade de poder ser lida tanto da direita para a esquerda como da esquerda para a direita. Exemplos de palíndromo são: esse, mussum e osso. Implemente um algoritmo de reconhecimento de palíndromos de tamanho n.
5. O que é um algoritmo polinomial?
6. Por muitas vezes damos atenção apenas ao pior caso dos algoritmos? Explique o porque.
7. Descreva algoritmos polinomiais para cada um dos seguintes problemas:
   a) Quadrado perfeito
   b) Equação do segundo grau
8. Considere o seguinte problema de busca:

   Entrada Uma sequencia de n números em um vetor $A = (a_1, a_2, ..., a_n)$ e um valor v.

   Saida O índice i tal que $v = A[i]$ ou um valor especial $NULL$ caso v não esteja presente na sequência A.

   Escreve um algoritmo de busca linear que varre a sequência a procura de v. Prove que o seu algoritmo está correto por meio de uma invariante. Certifique-se que a invariante atende às três propriedades fundamentais (inicialização, manutenção e terminação).
9. Considere o seguinte problema de adição de números binários.

   Entrada Dois números inteiros A e B representados na forma de vetores de tamanhos n onde cada uma de suas posições contém os bits 0 ou 1.

   Saida Um número C representado na forma de um vetor de tamanho $n+1$ que armazena a soma dos números A e B.

   Escreve um algoritmo linear que resolva este problema. Prove a sua complexidade.
10. Seja o seguinte código do algoritmo INSERTION-SORT.

```
INSERTION-SORT(A)
   for j = 2 to A.length
      key = A[j]
      i = j - 1
      while i > 0 and A[i] > key
```

```
            A[i+1] = A[i]
            i = i - 1
        A[i+1] = key
}
```

É possível analisar a corretude de um trecho de código por meio de uma invariante. Encontrar uma invariante que avalie de forma completa e adequada a corretude de um trecho de código pode ser uma tarefa difícil, mas uma vez bem definida, basta verificar se essa se mantém verdadeira antes, durante e após a execução do código.

A seguinte invariante foi definida para o procedimento INSERTION-SORT:

> Os elementos em $A[1, ..., j-1]$ estão ordenados.

Justifique a validade dessa invariante.

11. Prove que o seguinte algoritmo para determinar o valor máximo de um vetor com n elementos está correto.[1]

```c
int max( int* a, int n ){
    int m = a[0];
    for ( int i = 1; i < n; i++ )
      if ( a[i] > m )
         m = a[i];
    return m;
}
```

12. Descreva um algoritmo para determinar o segundo menor elemento de um conjunto $S = \{s_1, s_2, \ldots, s_n\}$. Determinar exatamente o número de comparações efetuadas pelo algoritmo. O algoritmo será considerado tão melhor quanto menor for este número de comparações.

Crescimento de Funções

13. Disponha as seguintes funções em ordem crescente de complexidade assintótica:
    - $f_1(n) = 500n$
    - $f_2(n) = n \log_2 n$
    - $f_3(n) = n^5$
    - $f_4(n) = 2^n$
    - $f_5(n) = n^2$
    - $f_6(n) = 1$
    - $f_7(n) = \log_2 n$
    - $f_8(n) = \log_2(n^2)$
    - $f_9(n) = n^3$

14. Classifique as funções abaixo do ponto de vista de crescimento assintótico, i.e., se f_i vem antes de f_j na sua ordenação então $f_i = O(f_j)$:

$$2^{\sqrt{\lg n}} \quad 2^n \quad n^{5/4} \quad n \lg^3 n \quad n^{\lg n} \quad 2^{2^n} \quad 2^{n^2}$$

15. Dois algoritmos A e B possuem complexidade n^5 e 2^n, respectivamente. Em qual caso você utilizaria o algoritmo B ao invés do A? Exemplifique.

[1] Qual invariante este algoritmo mantém?

16. Algoritmos A e B possuem tempos de execução com complexidade $\Theta(n \log n)$ e $\Theta(n^{3/2})$, respectivamente. Qual deles é mais eficiente em termos assintóticos?

17. Um algoritmo tradicional e muito utilizado possui complexidade de $n^{1.5}$ enquanto um novo proposto é da ordem de $n \log n$. Qual utilizar?

18. Quais os significados da relações? $n^3 + 5n + 10 = n^3 + \Theta(n)$ e $n^3 + \Theta(n) = \Theta(n^3)$?

19. Ordene as seguintes funções por ordem de crescimento, ou seja, encontre uma ordenação $g_1, g_2, g_3, g_4, g_5, g_6, g_7, g_8$ das funções abaixo tal que $g_1 = \mathrm{O}(g_2)$, $g_2 = \mathrm{O}(g_3)$, $g_3 = \mathrm{O}(g_4)$, $g_4 = \mathrm{O}(g_5)$, $g_5 = \mathrm{O}(g_6)$, $g_6 = \mathrm{O}(g_7)$ e $g_7 = \mathrm{O}(g_8)$.

    a) $f_1(n) = n^\pi$
    b) $f_2(n) = \pi^n$
    c) $f_3(n) = \binom{n}{5}$
    d) $f_4(n) = \sqrt{2^{\sqrt{n}}} = 2^{\frac{1}{2}n^{1/2}}$
    e) $f_5(n) = \binom{n}{n-4}$
    f) $f_6(n) = 2^{\log^4 n}$
    g) $f_7(n) = n^{5(\log n)^2}$
    h) $f_8(n) = n^4 \binom{n}{4}$

    em que $\binom{n}{k} = \frac{n!}{k!(n-k)!}$, para $0 \leq k \leq n$.

20. Se $T(0) = T(1) = 1$, cada uma das seguintes recorrências define uma função T nos inteiros não negativos.

    a) $T(n) = 3T(\lfloor n/2 \rfloor) + n^2$.
    b) $T(n) = 2T(n-2) + 1$.
    c) $T(n) = T(n-1) + n^2$.

    Qual delas não pode ser limitada por uma função polinomial? Justifique a sua resposta.

21. Suponha que $f(n)$ e $g(n)$ sejam funções dos inteiros não-negativos nos inteiros não-negativos. Demonstre que as seguintes afirmações são equivalentes:

    a) Existem números reais $a > 1$, $b > 1$, $c > 1$ e existe um número inteiro $n_0 > 0$ tais que $a^{g(n)} \leq b^{f(n)} \leq c^{g(n)}$, para todo $n \geq n_0$;
    b) $f(n) = \Theta(g(n))$.

22. Considere que $f(n) = O(s(n))$ e $g(n) = O(r(n))$. Assinale Falso ou Verdade para as seguintes afirmações, demonstrando suas respostas:

    (a) $f(n) - g(n) = O(s(n) - r(n))$.
    (b) $f(n) + g(n) = O(s(n) + r(n))$.
    (c) $f(n) \times g(n) = O(s(n) \times r(n))$.
    (d) $f(n) \div g(n) = O(s(n) \div r(n))$.

23. Mostre que a notação ó-grande é transitiva, isto é, se $f(n) = \mathrm{O}(g(n))$ e $g(n) = \mathrm{O}(h(n))$, então $f(n) = \mathrm{O}(h(n))$.

24. Verdadeiro ou Falso? Prove.

   a) $10n = \mathrm{O}(n)$
   b) $10n^2 = \mathrm{O}(n)$
   c) $10n^{55} = \mathrm{O}(2^n)$
   d) $n^2 + 200n + 300 = \mathrm{O}(n^2)$
   e) $n^2 - 200n - 300 = \mathrm{O}(n)$
   f) $\frac{3n^2}{2} + \frac{7n}{2} - 4 = \mathrm{O}(n)$
   g) $\frac{3n^2}{2} + \frac{7n}{2} - 4 = \mathrm{O}(n^2)$
   h) $n^3 - 999999n^2 - 1000000 = \mathrm{O}(n^2)$
   i) $2^{n+1} = \mathrm{O}(2^n)$
   j) $3^n = \mathrm{O}(2^n)$
   k) $\log_2 n = \mathrm{O}(log_3 n)$
   l) $\log_3 n = \mathrm{O}(log_2 n)$
   m) $n = \mathrm{O}(2^n)$
   n) $\lg n = \mathrm{O}(n)$
   o) $n = \mathrm{O}(2^{n/4})$
   p) $4 \lg n = \mathrm{O}(n)$
   q) $100 \lg n - 10n + 2n \lg n = \mathrm{O}(n \lg n)$
   r) $\frac{3n^2}{2} + \frac{7n}{2} n^3 - 4 = \Theta(n^2)$
   s) $9999n^2 = \Theta(n^2)$
   t) $\frac{n^2}{1000} - 999n = \Theta(n^2)$
   u) $log_2 n + 1 = \Theta(log_{10} n)$

25. Qual a solução da expressão $\sum_{i=1}^{n} \Theta(i)$?

26. É correto ou incorreto afirmar que:

   (a) $(\log n)^a = O(n^b)$, para qualquer b e qualquer a positivo?
   (b) $\log_{10} n = O(\log_2 n)$?
   (c) $2^{n+1} = \mathrm{O}(2^n)$
   (d) $2^{2n} = O(2^n)$?
   (e) $f(n) = \Theta(f(n/2))$?
   (f) $\log n! = O(n \log n)$?
   (g) $\sum_1^n (1/2)^i = \Theta(1)$?
   (h) Se $T(\frac{n}{100}) = c\, n \log_2 n + n$, para algum $c > 0$, então $T(n) = O(n \log n)$?

27. Existem duas funções f e g, dos naturais nos reais positivos, tais que $f(n) \notin O(g(n))$ e $g(n) \notin O(f(n))$? Em caso afirmativo, apresente um exemplo. Em caso negativo, prove que tais funções não existem.

28. Prove ou apresente contra-exemplo: se $f(n) = O(g(n))$ e $g(n) = O(f(n))$, então segue que $f(n) = g(n)$ para todo n.

29. Considere as afirmativas abaixo onde $f(n)$ e $g(n)$ são funções assintoticamente positivas. Para cada uma delas, indique se a afirmativa é verdadeira ou falsa.

   a) Considere um algoritmo que faz 2^{2n} operações. Pode-se dizer que esse algoritmo é $\mathrm{O}(2^n)$.

   b) $f(n) = \Theta(f(n/2))$.

   c) Considere um algoritmo cuja função de complexidade é $f(n) = \lg(n)$. É correto afirmar que esse algoritmo é $\mathrm{O}(n)$ mas não é $\Theta(n)$.

   d) $f(n) + g(n) = \Theta(min(f(n), g(n)))$.

30. Demonstre as seguintes afirmações ou forneça contra-exemplos:

   (a) $100n^7 + 503n^5 = \Theta(n^7)$

   (b) $3n^4 + \Theta(n^2) = \Theta(n^4)$

   (c) Se $f(n) = \Theta(s(n))$ e $g(n) = \Theta(r(n))$, então $f(n) + g(n) = \Theta(f(n) + g(n))$

   (d) Se $f(n) = \Theta(s(n))$ e $g(n) = \Theta(r(n))$, então $f(n) - g(n) = \Theta(f(n) - g(n))$

   (e) $\log n! = O(n \log n)$.

31. Mostre que, para quaisquer constantes reais a e b, onde $b > 0$, $(n + a)^b = \Theta(n^b)$.

32. Nós podemos extender a notação vista em sala de aula para o caso de funções com dois parâmetros n e m que tendem ao infinito em proporções independentes. Para uma dada função $g(n, m)$, nós denotamos por $\mathrm{O}(n, m)$ o conjunto de funções

$$\begin{aligned}\mathrm{O}(g(n,m)) = \{f(n,m) :\ & \text{existem constantes positivas } c, n_0 \text{ e } m_0 \text{ tal que} \\ & 0 \leq f(n,m) \leq c \cdot g(n,m) \text{ para todo } n \geq n_0 oum \geq m_0\}\end{aligned}$$

   Dê as definições correspondentes para $\Omega(g(n, m))$ e $\Theta(g(n, m))$.

33. Usando a definição formal de Θ prove que $6n^3 \neq \Theta(n^2)$.

34. Sejam f e g duas sequências de números reais positivos. Prove que, se $\lim_{n\to\infty} \frac{f(n)}{g(n)} \in \mathbb{R}$, então $f(n) = O(g(n))$.

35. Determine a ordem de complexidade para cada função e justifique a sua resposta:

```
char* f(int n){
    char s[n];
    char x = 'x';
    for (int i = 0; i < n; i++)
        s[i] = x;
    return s;
}

char* g(int n){
    int i=-1;
    char s[n];
    char x = 'x';
    while (n != 0){
        if (n % 2 == 1)
            s[++i] = x;
```

```
        n = n/2;
    }
    return s;
}
```

36. Cormen et al., Capítulo 3, exercício 3.1-1

37. Cormen et al., Capítulo 3, exercício 3.1-2

38. Cormen et al., Capítulo 3, exercício 3.1-3

39. Cormen et al., Capítulo 3, exercício 3.1-4

40. Cormen et al., Capítulo 3, exercício 3.1-5

41. Cormen et al., Capítulo 3, exercício 3.1-6

42. Cormen et al., Capítulo 3, exercício 3.1-7

43. Cormen et al., Capítulo 3, exercício 3.1-8

44. Cormen et al., Capítulo 3, problema 3-1

45. Apresente a complexidade dos seguintes algoritmos em função de n. Justifique sua resposta:

   a)
   ```
   sum = 0; \\ 1 operacao
   for (int i = 0; i < n*n; i++) \\ N*N operacoes
       sum++; \\ 1 operacao
   ```

   Temos $1 + 1 \times N^2$ operações, o que nos leva a um algoritmo com complexidade assintótica $O(n^2)$.

   b)
   ```
   sum = 0;
   for (int i = 0; i < n; i++)
       for (int j = 0; j < n*n; j++)
           sum++;
   ```

   c)
   ```
   sum = 0;
   for (int i = 1; i < n; i*=2)
       sum++;
   ```

   d)
   ```
   sum = 0;
   for (int i = 0; i < n; i++)
       for (int j = n; j > 0; j/=2)
          sum++;
   ```

46. Qual a complexidade de tempo de execução dos algoritmos abaixo, em notação Θ?

(a)
```
// Recebe um vetor de inteiros com b−a+1 elementos
// Devolve o indice de um elemento particular apos processar o vetor
int part(int *p, int a, int b) {

  int v = p[a], l = a; r = b, w;

  while (l < r) {
    while(p[l] <= v && l <= b) l++;
    while(p[r] > v  && r >= a) r−−;
    if (l < r) {
      w = p[l];
      p[l] = p[r];
      p[r] = w;
    }
  }
  p[a] = p[r];
  p[r] = v;
  return(r);
}
```

```
// Recebe um vetor de inteiros p com n elementos
// Chama a funcao da questao anterior
int f(int *p, int n) {

  int a = (n+1)/2, b = (n+a)/2, i = 0;
  for (; a <= b; a++)
    i += part(p,a,b);
  return (i);
}
```

(47) Forneça a complexidade para os seguintes algoritmos [extraídos de Al Aho and Jeff Ullman (1994), capítulo 3]:

(a)
```
int PowersOfTwo(int n) {
  int i = 0;

  while (n%2 == 0) {
    n = n/2;
    i++;
  }
return i;
}
```

(b)
```
int prime(int n) {
  int i = 2;
  while (i*i <= n)
    if (n%i == 0) return FALSE;
    else i++;

  return TRUE;
}
```

Divisão e conquista

48. Separe um bando de $2n$ Orcs em dois times com n Orcs cada. Cada Orc tem um número marcado em suas costas que indica o quão habilidoso o Orc é em combate. Divida o

grupo da forma mais injusta possível para que o combate entre os dois times de Orcs seja extremamente sanguinário. Justifique sua escolha de divisão e explique como esta tarefa pode ser feita utilizando divisão e conquista em tempo $O(n \log n)$.

49. O tamanho das instâncias de um certo problema é medido por um parâmetro n. Tenho três algoritmos — A, B e C — para o problema:

    - A divide cada instância do problema em cinco subinstâncias de tamanho [n/2], resolve as subinstâncias e então combina as soluções em tempo O(n)
    - B divide cada instância do problema em duas subinstâncias de tamanho n - 1, resolve as subinstâncias e então combina as soluções em tempo O(1)
    - C divide cada instância do problema em nove subinstâncias de tamanho [n/3], resolve as subinstâncias e então combina as soluções em tempo $O(n^2)$

    Qual o consumo de tempo de cada um dos algoritmos? Qual dos algoritmo é assintoticamente mais eficiente no pior caso?

50. Cormen et. al., Capítulo 4, Exercício 4.4-1
51. Cormen et. al., Capítulo 4, Exercício 4.4-2
52. Cormen et. al., Capítulo 4, Exercício 4.4-3
53. Cormen et. al., Capítulo 4, Exercício 4.4-4
54. Cormen et. al., Capítulo 4, Exercício 4.5-1
55. Cormen et. al., Capítulo 4, Exercício 4.5-2
56. Cormen et. al., Capítulo 4, Exercício 4.5-3
57. Cormen et. al., Capítulo 4, Exercício 4.5-4
58. Cormen et. al., Capítulo 4, Exercício 4.5-5
59. Cormen et. al., Capítulo 4, Exercício 4.6-1
60. Cormen et. al., Capítulo 4, Exercício 4.6-2
61. Cormen et. al., Capítulo 4, Exercício 4.6-3
62. Cormen et. al., Capítulo 4, Exercício 4-1 (final do capítulo)
63. Cormen et. al., Capítulo 4, Exercício 4-2 (final do capítulo)
64. Resolva as seguintes recorrências:

    a) $T(n) = T(\frac{n}{3}) + T(\frac{2n}{3}) + \Theta(n)$
    b) $T(n) = \log n + T(\sqrt{n})$
    c) $T(n) = T(n-1) + 3n + 2$
    d) $T(n) = T(n-1) + n$
    e) $T(n) = 2T(n/2) + \lg(n)$
    f) $T(n) = 2T(n/2) + n\lg(n)$
    g) $T(n) = 2T(n/2) + n\lg^2(n)$

h) $T(n) = 4T(n/2) + n^2$

i) $T(n) = 2T(n/4) + \sqrt{n}$

j) $T(n) = T(\sqrt{n}) + 1$

65. Aplique o método da substituição para mostrar que $T(n) = O(g(n))$ para as seguintes relações:

    (a) $T(n) = 2T(n/2) + n$ e $g(n) = n \log n$.

    (b) $T(n) = 2T(n/2 + k) + n$, $k > 0$ e $g(n) = n \log n$.

    (c) $T(n) = 4T(n/2) + n$ e $g(n) = n^3$.

    (d) $T(n) = 4T(n/2) + n$ e $g(n) = n^2$.

    (e) $T(n) = T(n-1) + n$ e $g(n) = n^2$.

66. Mostre a árvore de recursão e encontre os limites assintóticos das seguintes recorrências. Para verificar a correção da sua resposta, use o método de substituição:

    (a) $T(n) = 2T(n/4) + \Theta(n^2)$.

    (b) $T(n) = 2T(n/2) + n^3$.

    (c) $T(n) = 3T(n/4) + n$.

    (d) $T(n) = T(n/2) + T(n/3) + n$.

    (e) $T(n) = T(n/2) + T(n-k) + 1$, onde k é uma constante tal que $0 < k < n$.

67. Usando o método mestre, resolva as seguintes recorrências:

    (a) $T(n) = 4T(n/2) + n$

    (b) $T(n) = 4T(n/2) + n^2$

    (c) $T(n) = 4T(n/2) + n^3$

68. Indique se as relações abaixo podem ser resolvidas pelo método mestre. Em caso positivo, forneça a solução. Caso contrário, forneça o motivo e resolva por outro método.

    (a) $T(n) = 2T(n/2) + \log n$

    (b) $T(n) = 2T(n/4) + \log n$

    (c) $T(n) = T(n/2) + n \log n$

    (d) $T(n) = 2T(n/2) + n \log n$

    (e) $T(n) = T(n/2) + T(n/3) + n$

    (f) $T(n) = 2T(n/2) + 2n \log n$

    (g) $T(n) = 2T(n/2) + 2^n$

69. Seja T uma função que leva números naturais em números reais. Suponha que T satisfaz a recorrência $T(n) = 2T(\frac{n}{2}) + 7n + 2$. Mostre que $T(n) = \Omega(n \log n)$.

70. Considere a recorrência

$$T(n) = \begin{cases} 1, & \text{se } n = 1. \\ T(n-1) + 1, & \text{se } n > 1 \text{ é ímpar} \\ 3T(n/2), & \text{se } n > 1 \text{ é par} \end{cases}$$

(a) Prove que, sempre que n é uma potência de dois, $T(n) = n^{\log_2 3}$.

(b) Prove que T é crescente.

(c) Prove que existe $k \geq 1$ tal que $k \cdot n^{\log_2 3} \geq (2n)^{\log_2 3}$ para todo $n \in \mathbf{N}$.

71. Considere a recorrência

$$U(n) = \begin{cases} 1, & \text{se } n = 1. \\ 2U(n-1), & \text{se } n > 1 \text{ é ímpar} \\ 3U(n/2), & \text{se } n > 1 \text{ é par} \end{cases}$$

Prove que, sempre que n é uma potência de dois, $U(n) = n^{\log_2 3}$.

72. Apresente o comportamento assintótico das seguintes equações de recorrência:

a) $T(n) = 2T(n/2) + \lg(n)$

b) $T(n) = 2T(n/2) + n\lg(n)$

c) $T(n) = 2T(n/2) + n\lg^2(n)$

d) $T(n) = 4T(n/2) + n^2$

Projeto de algoritmos

73. Considere o seguinte pseudocódigo:

```
//Pseudocodigo o algoritmo ingenuo para
//casamento de cadeias de caracteres
NAIVE-STRING-MATCHER(texto T, padrao P){
   n = T.lenght;
   m = P.lenght;
   for s = 0 to n-m do
      if P[1 .. m] == T[s+1 .. s+m]
      for( int j = 1; j <=i; j++)
         print "Padrao encontrado em" s-m
}
```

a) Conte as comparações que o algoritmo NAIVE-STRING-MATCHER realiza com o parão $P = 0001$ dentro do texto $T = 000010001010001$.

b) Suponha que sabemos que todos os caracteres em um padrão P são diferentes. Mostre como acelerar o algoritmo NAIVE-STRING-MATCHER para executar com tempo $\mathrm{O}(n)$ em um texto de tamanho n.

74. Descreva um algoritmo que, dados n inteiros com valores entre 1 e k, pré-processe esses inteiros e então (descontado o tempo do pré-processamento) responda perguntas da forma "dados a, b, quantos dos n inteiros estão no intervalo $[a\ ,\ b]$" em tempo $\mathrm{O}(1)$. O pré-processamento deve ser feito em tempo $\mathrm{O}(n + k)$.

75. Descreva um algoritmo que, dados n inteiros com valores de 1 a 10, pré-processe esses inteiros e então responda a pergunta da seguinte forma: "dados três inteiros (i, j, x) tais que $1 \leq i \leq j \leq n$ e $1 \leq x \leq 10$, quantos dos $j - i + 1$ inteiros no intervalo $[i, j]$ têm valor igual a x?". O pré-processamento deve ser feito em tempo $O(n)$ e cada consulta deve ser respondida em tempo $O(1)$.

76. Um elemento majoritário em um vetor de inteiros de tamanho n é um inteiro que ocorre em pelo menos $n/2$ posições do vetor. Nem todos os vetores têm um elemento majoritário, mas o elemento deve ser claramente único se ele existe.

   a) Projete um algoritmo de custo linear que identifica o elemento majoritário, se ele existir.

   b) Assuma agora que o vetor não contenha inteiros, mas contenha outro tipo de dados que permita apenas testes de igualdade. Qual é a complexidade para determinar se o vetor contem um elemento majoritário?