Elementos de Matemática

ESTE APÊNDICE APRESENTA tópicos de matemática que podem ser necessários para completo entendimento do texto principal.

B.1 Somatórios

Um **somatório**, representado pela letra grega Σ (sigma maiúsculo), consiste numa soma de uma sequência de números. Por exemplo, a soma $1 + 2 + \ldots + (n - 1) + n$ pode ser representada pelo somatório:

$$\sum_{i=1}^{n} i$$

A seguir serão enumeradas algumas propriedades e resultados notáveis de somatórios.

1. $\sum_{i=j}^{n} cf(i) = c\sum_{i=j}^{n} f(i)$, em que c é uma constante.

2. $\sum_{i=j}^{n} [f(i) \pm g(i)] = \sum_{i=j}^{n} f(i) \pm \sum_{i=j}^{n} g(i)$

3. $\sum_{i=j}^{n} 1 = n - j + 1$

4. $\sum_{i=0}^{n} i = \sum_{i=1}^{n} i = \frac{n(n+1)}{2}$

5. $\sum_{i=1}^{n} i^2 = \frac{n(n+1)(2n+1)}{6}$

6. $\sum_{i=j}^{k-1} a^i = \frac{a^j - a^k}{1-a} (j < k)$

Seja a_i $(0 \leq i \leq n)$ uma sequência de números. Então tem-se que:

7. $\sum_{i=1}^{n} (a_n - a_{n-1}) = a_n - a_0$

Esse último somatório recebe a denominação especial de **soma telescópica** ou **série telescópica**.

B.2 Polinômios

Um **polinômio** é uma expressão formada por números reais, denominados **coeficientes**, **variáveis** e **expoentes**, que são números inteiros não negativos. Essa expressão utiliza apenas as operações de soma, subtração, multiplicação e exponenciação em sua formação. Embora existam polinômios que utilizam mais de uma variável, o interesse aqui é restrito àqueles que usam apenas uma variável e podem, portanto, ser escritos como:

$$P(x) = a_n x^n + a_{n-1} x^{n-1} + \ldots + a_1 x + a_0$$

ou, em forma de somatório:

$$P(x) = \sum_{i=0}^{n} a_i x^i$$

Nessa definição, n é um número inteiro não negativo chamado **grau do polinômio** e a_0, a_1, ... , a_{n-1}, a_n são constantes reais (coeficientes do polinômio). Cada $a_i x^i$, $0 \leq i \leq n$, que constitui um polinômio é denominado **termo** e o expoente i é o **grau** do respectivo termo. O termo que possui grau igual a zero é denominado **termo constante**. Quando dois termos possuem o mesmo grau, diz-se que eles são **termos semelhantes**.

Polinômios são usados na definição de **funções polinomiais** e algumas dessas funções notáveis são apresentados a seguir:

- **Polinômio de grau zero**, também denominado **função constante**. Por exemplo:

  $$P(x) = 5$$

- **Polinômio de grau um**, também denominado **função linear** (ou **função afim**). Por exemplo:

  $$P(x) = 2x + 3$$

- **Polinômio de grau dois**, também denominado **função quadrática**. Por exemplo:

  $$P(x) = x^2 + x - 3$$

Um polinômio é **completo** quando todos os seus termos estão presentes. Por exemplo, todos os exemplos apresentados acima constituem polinômios completos, enquanto o polinômio:

$$P(x) = x^2 - 1$$

não é completo, pois lhe falta o termo de ordem 1.

Os termos de um polinômio podem ser arranjados à vontade, mas, tipicamente, eles são apresentados em ordem decrescente de graus, como tem sido o caso aqui.

O **valor** de um polinômio é o resultado obtido quando se substitui cada ocorrência de variável do polinômio por um valor real e se efetuam as devidas operações. Por exemplo, o valor do polinômio:

$$P(x) = x^2 + x - 3$$

quando sua variável é substituída por *2* é:

$$P(2) = 2^2 + 2 - 3 = 3$$

Dois ou mais polinômios podem ser **somados** ou **subtraídos** agrupando-se seus termos de mesmo grau e somando-se ou subtraindo-se os respectivos coeficientes. Por exemplo, a soma de:

$$P(x) = x^2 + x - 3$$

com:

$$Q(x) = x^4 - 2x^2 - x + 1$$

resulta em:

$$P(x) + Q(x) = x^4 - x^2 - 2$$

A **multiplicação** de polinômios usa as leis distributivas da multiplicação com relação à soma e subtração. Por exemplo, a multiplicação do polinômio:

$$P(x) = x - 1$$

pelo polinômio:

$$Q(x) = 2x^2 - x + 1$$

resulta no polinômio:

$$P(x).Q(x) = (x - 1)(2x^2 - x + 1) = 2x^3 - x^2 + x - 2x^2 + x - 1 = 2x^3 - 3x^2 + 2x - 1$$

Polinômios possuem várias outras propriedades e operações que não são de interesse neste livro.

B.3 O Método de Horner

O **método de Horner**[1] é tipicamente ensinado em cursos elementares de matemática como uma técnica empregada para calcular o quociente da divisão de um polinômio por outro. A mesma técnica, porém, tem várias outras utilidades práticas. Do ponto de vista de programação, a maior utilidade desse método está no cálculo do resultado da aplicação de uma função polinomial a um determinado valor.

Suponha, por exemplo, que se deseja calcular o resultado (i.e., o **valor numérico**) da aplicação da função polinomial $P(x)$ a seguir sobre *5* [i.e., deseja-se calcular $P(5)$]:

$$P(x) = 4x^4 - 5x^3 + 3x^2 + 2x - 1$$

A maneira mais rudimentar de realizar esse cálculo é a seguinte:

$$\begin{aligned} P(5) &= 4\times5^4 - 5\times5^3 + 3\times5^2 + 2\times5 - 1 \\ &= 4\times625 - 5\times125 + 3\times25 + 10 - 1 \\ &= 2500 - 625 + 75 + 10 - 1 \\ &= 1959 \end{aligned}$$

O cálculo ingênuo apresentado acima pode ser substancialmente simplificado usando-se o artifício manual mostrado na **Figura B–1**.

[1] Essa denominação é uma homenagem ao matemático britânico William George Horner que popularizou o método (mas não o inventou).

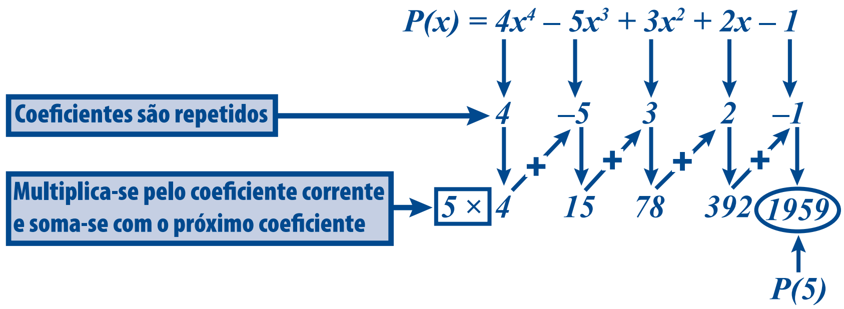

FIGURA B–1: CÁLCULO SIMPLIFICADO DE VALOR NUMÉRICO DE POLINÔMIO

A técnica informal descrita acima é baseada no fato de o polinômio:

$$P(x) = a_n x^n + a_{n-1} x^{n-1} + \ldots + a_1 x + a_0 \qquad [1]$$

poder ser escrito como:

$$P(x) = a_0 + x(a_1 + x(a_2 + \ldots + x(a_{n-1} + a_n x))) \qquad [2]$$

A prova dessa última afirmação é relativamente trivial e o leitor poderá encontrá-la em muitos livros de matemática elementar que versam sobre o assunto. Mais importante para o programador é saber como implementar esse método eficiente de cálculo.

Para entender como o método de Horner pode ser transformado em algoritmo, note que o valor numérico de um polinômio $P(x)$ para $x = x_0$ pode ser obtido por meio da seguinte sequência de constantes b_i:

$$\begin{aligned} b_n &= a_n \\ b_{n-1} &= a_{n-1} + b_n x_0 \\ &\vdots \\ b_0 &= a_0 + b_1 x_0 \end{aligned}$$

Substituindo iterativamente os valores b_i na **Fórmula [2]** apresentada acima, é fácil mostrar que $P(x_0) = b_0$.

A função `Horner()`, exibida no programa abaixo, implementa o método de Horner delineado acima para avaliar o valor numérico de um polinômio para um dado valor. Para simplificar, o conjunto dos números inteiros é considerado como domínio dos polinômios. Essa função retorna o referido valor calculado e seus parâmetros são:

- `pol` (entrada) — o polinômio representado por seus coeficientes armazenados num array
- `grau` (entrada) — o grau do polinômio
- `x` (entrada) — o valor para o qual o polinômio será calculado

```
int Horner(const int pol[], int grau, int x)
{
   int i,
       resultado;

      /* Inicia a sequência de números (bi) com o */
      /* coeficiente (an) do termo de maior grau   */
   resultado = pol[grau];

      /* Repetidamente, multiplica o valor do resultado   */
      /* por x e adiciona o valor do coeficiente seguinte */
   for (i = grau - 1; i >= 0; --i)
      resultado = resultado*x + pol[i];

   return resultado;
}
```

```
int main(void)
{
   int p[] = {-1, 2, 3, -5, 4}, /* Coeficientes do polinômio */
       grau; /* grau do polinômio */

   grau = sizeof(p)/sizeof(p[0]) - 1;

   printf( "\n\t>>> Valor de P(5): %d\n", Horner(p, grau, 5) );

   return 0;
}
```

Obter o valor numérico de um polinômio do modo tradicional (i.e., termo a termo) requer, no máximo, n adições e $(n^2 + n)/2$ multiplicações, o que significa dizer que essa operação tem custo temporal $\theta(n^2)$. Por outro lado, claramente, a função `Horner()`, que implementa a mesma operação usando o método de Horner, tem custo temporal $\theta(n)$.

B.4 Logaritmos

O logaritmo de um número real positivo x na base b, sendo $b > 0$ e $b \neq 1$, é definido como:

$$\boxed{\log_b x = y \text{ se e somente se } b^y = x}$$

A base b de um logaritmo pode ser omitida, mas, quando esse é o caso, subentendem-se valores diferentes para b, dependendo da área de aplicação. Por exemplo, em computação $log\ x$ significa $log_2\ x$, mas em muitas outras áreas de ciências, o significado de $log\ x$ é $log_{10}\ x$. Por outro lado, $ln\ x$ é o logaritmo natural (ou neperiano) de x; i.e., $ln\ x$ é o mesmo que $log_e\ x$, em que e é a constante de Napier.

Algumas propriedades importantes de logaritmos são as seguintes:

- **Regra do produto**: $log_b (xy) = log_b\ x + log_b\ y$
- **Regra do quociente**: $log_b (x/y) = log_b\ x - log_b\ y$
- **Regra do expoente**: $log_b\ x^p = p.log_b\ x$
- **Mudança de base**: $log_b\ x = (log_k\ x)/(log_k\ b)$
- **Definição alternativa**: $b^{\log_b x} = x$

B.5 Matrizes

Uma **matriz** $n \times m$ é uma tabela com n **linhas** e m **colunas** contendo **elementos** de algum conjunto. Por exemplo, a seguir tem-se uma matriz 2×3:

$$A_{2 \times 3} = \begin{vmatrix} -1 & 0 & 2 \\ 1 & 2 & -2 \end{vmatrix}$$

A seguir, serão apresentadas algumas definições básicas relacionadas a matrizes:

- Uma **matriz quadrada** é aquela cujo número de linhas é igual ao número de colunas.
- A **diagonal principal** de uma matriz quadrada $n \times n$ é a diagonal formada pelos elementos a_{ij} de modo que $i = j$, sendo $1 \leq i, j \leq n$.
- A **matriz identidade** I_n é a matriz quadrada $n \times n$ na qual todos os elementos na diagonal principal são iguais a 1 e os demais elementos são iguais a zero. Por exemplo, a matriz identidade I_2 é apresentada abaixo:

$$I_2 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

Soma e **subtração de matrizes** são definidas apenas quando as matrizes possuem as mesmas dimensões. Para se obter o resultado da soma ou subtração de duas matrizes, os elementos correspondentes são somados ou subtraídos, respectivamente. Por exemplo:

$$\begin{bmatrix} -1 & 0 & 2 \\ 1 & 2 & -2 \end{bmatrix} + \begin{bmatrix} -1 & 3 & 0 \\ -1 & 0 & -3 \end{bmatrix} = \begin{bmatrix} -2 & 3 & 2 \\ 0 & 2 & -5 \end{bmatrix}$$

Multiplicação de matrizes é um pouco mais complicada. Em primeiro lugar, as dimensões das matrizes a serem multiplicadas deve ser tal que o número de colunas da primeira matriz (operando esquerdo) deve ser igual ao número de linhas da segunda matriz (operando direito). Mais precisamente, se a matriz A for $n \times m$ e a matriz B for $m \times p$ é uma matriz C $n \times p$ em que cada elemento c_{ij} é dado por:

$$c_{ij} = \sum_{k=1}^{m} a_{ik} b_{kj}$$

Por exemplo:

$$\begin{bmatrix} -1 & 0 & 2 \\ 1 & 2 & -2 \end{bmatrix} \times \begin{bmatrix} 2 & 1 \\ 3 & 2 \\ 0 & -1 \end{bmatrix} = \begin{bmatrix} -2 & -3 \\ 8 & 6 \end{bmatrix}$$

É importante notar que a multiplicação de matrizes não é comutativa e que o produto de qualquer matriz $A_{m\,x\,n}$ pela matriz identidade I_n é a própria matriz $A_{m\,x\,n}$. Também, o produto da matriz identidade I_m pela matriz $A_{m\,x\,n}$ é $A_{m\,x\,n}$.

A **transposta** de uma matriz $A_{m\,x\,n}$ é a matriz $T_{n\,x\,m}$, tal que cada elemento da matriz transposta é dado por: $t_{ij} = a_{ji}$. Por exemplo, a transposta da matriz:

$$A_{2\,x\,3} = \begin{bmatrix} -1 & 0 & 2 \\ 1 & 2 & -2 \end{bmatrix}$$

é a matriz:

$$T_{3\,x\,2} = \begin{bmatrix} -1 & 1 \\ 0 & 2 \\ 2 & -2 \end{bmatrix}$$

B.6 Funções Piso e Teto

As funções **piso** e **teto** mapeiam números reais em números inteiros de tal modo que o piso de número real é sempre um número inteiro e o mesmo ocorre com o teto de um número real.

O piso de um número real x, representado como $\lfloor x \rfloor$, é o maior número inteiro menor do que ou igual a x. Por outro lado, o teto de um número real x, representado como $\lceil x \rceil$, é o menor número inteiro maior do que ou igual a x.

Essas definições parecem confusas, mas se tornam simples se você seguir o seguinte artifício para determinar o piso e o teto de um número real:

1. Determine entre quais números inteiros o número real em questão se encontra na reta real.
2. O menor desses números é o piso e o maior deles é o teto. Para não esquecer: o teto está sempre acima de sua cabeça e o teto está sempre abaixo dos seus pés.

Supondo que o número cujos piso e teto se desejam determinar seja *1,125*, a **Figura B–2** ilustra esse artifício:

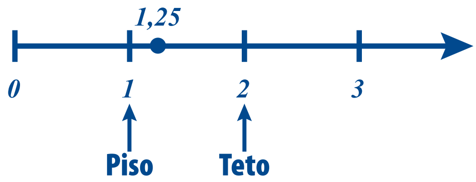

FIGURA B–2: PISO E TETO DE UM NÚMERO REAL POSITIVO

No caso de um número real negativo, determinar piso e teto pode parecer um pouco mais confuso, mas o procedimento acima descrito também se aplica nesse caso, como mostra a **Figura B–3**.

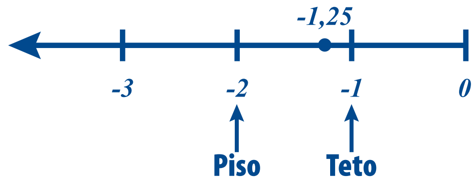

FIGURA B–3: PISO E TETO DE UM NÚMERO REAL NEGATIVO

Sendo x um número real e n um número inteiro, então as seguintes propriedades das funções piso e teto merecem destaque:

[1] $\lfloor x \rfloor = n$ se e somente se $n \leq x < n + 1$

[2] $\lfloor x \rfloor = n$ se e somente se $x - 1 < n \leq x$

[3] $\lceil x \rceil = n$ se e somente se $n - 1 < x \leq n$

[4] $\lceil x \rceil = n$ se e somente se $x \leq n < x + 1$

[5] $\lfloor x + n \rfloor = \lfloor x \rfloor + n$

[6] $\lceil x + n \rceil = \lceil x \rceil + n$

[7] $\lfloor x \rfloor = \lceil x \rceil = n$

B.7 Análise Combinatória

Análise combinatória é um ramo de matemática que se dedica a contar as diversas maneiras pelas quais elementos de um conjunto finito podem ser combinados. Essas formas de contagens são tipicamente divididas em três categorias:

- **Arranjos**, que são agrupamentos de elementos que dependem de ordenação. Para calcular o arranjo de n elementos considerados p a p, em que $p \leq n$, utiliza-se a fórmula:

$$A_{p,n} = \frac{n!}{(n-p)!}$$

Exemplo de arranjo: Suponha que num campeonato de futebol existam cinco equipes participantes, então, se o campeonato tiver dois turnos, de modo que os times se enfrentem duas vezes, então o número de confrontos entre essas equipes é determinado por:

$$A_{10,2} = \frac{10!}{(10-2)!} = \frac{10!}{8!} = 90$$

- **Combinações**, que são agrupamentos de elementos que não dependem de ordenação. Para calcular a combinação de n elementos considerados p a p, em que $p \leq n$, utiliza-se a fórmula:

$$C_{p,n} = \frac{n!}{p!(n-p)!}$$

Exemplo de combinação: O número de maneiras distintas que se podem agrupar as vogais da língua portuguesa duas a duas é dado por:

$$C_{5,2} = \frac{5!}{2!(5-2)!} = \frac{5!}{2!3!} = 10$$

- **Permutações**, que são agrupamentos ordenados de modo que o número de elementos de cada agrupamento é igual ao número de elementos disponíveis. A permutação de n elementos é calculada por meio da fórmula:

$$Pn = n!$$

Exemplo de permutação: O número de maneiras distintas que se podem apresentar as vogais da língua portuguesa, sem importar a ordem com que são apresentadas, é dado por:

$$P5 = 5! = 120$$

B.8 Indução Matemática

Indução matemática (finita) é um método de prova matemática utilizada para demonstrar teoremas sobre números inteiros não negativos ou números naturais. Esse método de prova tem duas formulações equivalentes que serão descritas a seguir.

B.8.1 Indução Fraca

Provavelmente, **indução fraca** é a técnica mais simples e mais utilizada de prova por indução. Ela segue os seguintes passos:

- **Passo 1 — Base da indução**. Mostre que o teorema é válido para o menor valor para o qual ele afirma ser válido. Por exemplo, se um teorema afirma que uma determinada propriedade é válida para $\forall n \in \mathbb{N}$, então o menor valor a que se refere esse passo é 0.
- **Passo 2 — Hipótese indutiva**. Assuma que o teorema vale para $n = k$.
- **Passo 3 — Indução**. Mostre que o teorema vale para $n = k + 1$.

Exemplo B.1 Mostre que:

$$\sum_{i=1}^{n} i = \frac{n(n+1)}{2}$$

Prova:

- **Base da indução**. Claramente, a igualdade vale para $n = 1$, pois obtém-se 1 como resultado nos dois lados da igualdade.
- **Hipótese indutiva**. Assuma que a seguinte igualdade é válida:

$$\sum_{i=1}^{k} i = \frac{k(k+1)}{2}$$

- **Passo indutivo**. Deve-se mostrar que a seguinte igualdade é válida usando-se a hipótese indutiva:

$$\sum_{i=1}^{k+1} i = \frac{(k+1)(k+1+1)}{2} = \frac{(k+1)(k+2)}{2}$$

Esse último somatório pode ser escrito como:

$$\sum_{i=1}^{k+1} i = \sum_{i=1}^{k} i + \sum_{i=k+1}^{k+1} i$$

Então, utilizando-se a hipótese indutiva, tem-se:

$$\sum_{i=1}^{k+1} i = \frac{k(k+1)}{2} + k + 1$$

Finalmente, resolvendo-se o lado direito dessa última igualdade, obtém-se:

$$\sum_{i=1}^{k+1} i = \frac{(k+1)(k+2)}{2}$$

■

Exemplo B.2 Mostre por indução que $T(n) = 5n^2 + 3n + 2$ é $O(n^2)$.

Prova: Deve-se mostrar que $5n^2 + 3n + 2 \leq c_0 n^2$, para cada $n \geq n_0$.

- **Base da indução**. Quando $c_0 = 6$ e $n_0 = 4$, tem-se que:

$$5n^2 + 3n + 2 = 5\cdot4^2 + 3\cdot4 + 2 = 94 \leq 96 = 6\cdot4^2 = c_0 n^2$$

- **Hipótese indutiva**:

$$5k^2 + 3k + 2 \leq 6k^2, \text{ para } k \geq 6$$

- **Passo indutivo**. Deve-se mostrar que $5\cdot(k+1)^2 + 3\cdot(k+1) + 2 \leq 6\cdot(k+1)^2$, ou seja, rearranjando os termos dessa última desigualdade, deve-se mostrar que:

$$5k^2 + 3k + 2 - 2(k-1) \leq 6k^2, \text{ para } k \geq 6 \text{ (a ser demonstrado) } (\dagger)$$

Somando-se $-2(k-1)$ aos dois lados da desigualdade que representa a hipótese indutiva, pode-se reescrevê-la como:

$$5k^2 + 3k + 2 - 2\cdot(k-1) \leq 6k^2 - 2\cdot(k-1), \text{ para } k \geq 6 \text{ (hipótese indutiva) } (\ddagger)$$

Agora, comparando-se as desigualdades (†) e (‡), resta mostrar que:

$$6k^2 - 2\cdot(k-1) \leq 6k^2, \text{ para } k \geq 6$$

É fácil mostrar que essa última desigualdade é valida para qualquer $k \geq 1$. Logo ela também é válida para $k \geq 6$. ■

B.8.2 Indução Forte

A diferença entre indução fraca e **indução forte** está no segundo passo da prova (i.e., na hipótese indutiva), que na indução forte é descrito como:

❒ **Passo 2 — Hipótese indutiva**. Assuma que o teorema vale para $n \leq k$.

Em outras palavras, enquanto a hipótese indutiva para indução fraca supõe que um teorema vale para um determinado valor k, a hipótese indutiva para indução forte assume que o teorema vale para um intervalo de valores (i.e., para qualquer n que seja menor do que um dado valor k).

Embora os dois tipos de prova por indução sejam equivalentes, existem situações nas quais o uso da versão forte de indução facilita a prova de alguns teoremas.

Exemplo B.3 Mostre por indução que qualquer inteiro $n \geq 2$ pode ser decomposto em fatores primos.

Prova:

- **Base da indução**. Claramente, a afirmação vale para $n = 2$, pois a decomposição de 2 é 2, que é um número primo.
- **Hipótese indutiva**. Suponha que a afirmação é válida para $n \leq k$.
- **Passo indutivo**. Deve-se mostrar que a afirmação é válida para $n = k + 1$.

  Se $k + 1$ for um número primo, sua decomposição é ele próprio, que é um número primo.

  Se $k + 1$ não for um número primo, ele é um número composto e pode ser expresso como:

  $$k + 1 = p \cdot q$$

  sendo p e q inteiros positivos e tais que $p < k + 1$ e $q < k + 1$.

  Como $p \leq k$ e $q \leq k$, de acordo com a hipótese indutiva, p e q possuem decomposições em fatores primos. Portanto, multiplicando-se essas duas decomposições, obtém uma decomposição de $k + 1$ em fatores primos, o que conclui a prova do teorema. ■

B.9 Relações de Recorrência

Uma **relação de recorrência** consiste em duas ou mais equações que descrevem uma sequência de números de tal modo que os primeiros termos da sequência são definidos explicitamente, enquanto os demais são definidos em função de seus antecessores. Por exemplo, as duas equações a seguir constituem uma das mais simples relações de recorrência:

$$F_n = \begin{cases} 0 & se\ n = 0 \\ F_{n-1} + 1 & se\ n \geq 1 \end{cases}$$

Nessa relação de recorrência, o primeiro termo é 0 e cada termo subsequente é obtido somando-se 1 ao termo anterior. Logo os termos dessa sequência correspondem exatamente ao conjunto dos números naturais.

Uma relação de recorrência pode ser aparecer em **notação de subscritos**, como no exemplo acima, ou em **notação de função**, como é mostrado a seguir:

$$f(n) = \begin{cases} 0 & se\ n = 0 \\ f(n-1) + 1 & se\ n \geq 1 \end{cases}$$

Em princípio, uma relação de recorrência não permite que se determine o valor de um termo arbitrário da sequência que ela representa. Por exemplo, se não fosse prenunciado que a relação de recorrência do exemplo acima representava os números naturais, talvez o leitor tivesse dificuldade de descobrir qual seria o centésimo termo dessa sequência.

Resolver uma relação de recorrência significa apresentá-la num formato (**solução**) que não envolve recursão. Tal formato é uma expressão, denominada **forma fechada**, que pode ser facilmente utilizada para determinar o enésimo termo da sequência que a relação de recorrência representa. A forma fechada da relação do exemplo acima é, simplesmente, n.

Não existe fórmula geral para resolução de relações de recorrência e, pior, nem sempre é simples encontrar uma fórmula que represente o n-ésimo termo da sequência que a relação de recorrência representa.

Esta seção apresentará apenas como são resolvidas relações de recorrência que aparecem neste livro.

B.9.1 Conjectura e Indução Matemática

A técnica de resolução de relações de recorrência que utiliza conjectura e indução matemática foi exposta na **Seção 6.11.6**. Essa é a técnica mais simples de resolução de relações de recorrência, mas, infelizmente, são poucas as relações de recorrência de interesse em estruturas de dados que podem ser resolvidas dessa maneira. Entretanto, por ser a mais simples, essa técnica deve ser a primeira a ser especulada.

B.9.2 Relações de Recorrência Homogêneas

Relações de recorrência **homogêneas** apresentam o seguinte formato:

$$f(n) = a_1 f(n-1) + a_2 f(n-2) + \dots + a_d f(n-d)$$

em que a_1, a_2, ..., a_d e d são constantes, sendo a constante d denominada a **ordem** da recorrência. Tipicamente, são especificados valores da função em para alguns valores denominados **condições de fronteira**.

Provavelmente, o exemplo mais notável de relação de recorrência homogênea é aquele que define a sequência de Fibonacci:

$$f(n) = \begin{cases} n & \text{se } n < 2 \\ f(n-1) + f(n-2) & \text{se } n \geq 2 \end{cases}$$

Na relação de recorrência de Fibonacci, $a_1 = a_2 = 1$ e as condições de fronteira são $f(0) = 0$ e $f(1) = 1$. A ordem dessa recorrência é $d = 2$.

Relações de recorrência homogêneas apresentam soluções da forma:

$$f(n) = x^n$$

Substituindo-se essa forma de solução na relação de recorrência, obtém uma equação denominada **equação característica**. Por exemplo, a equação característica da relação de recorrência de Fibonacci é:

$$x^n = x^{n-1} + x^{n-2}$$

Dividindo-se ambos os lados dessa equação por x^{n-2} e rearranjando seus termos, obtém-se:

$$x^2 - x - 1 = 0$$

Essa é uma equação que pode ser facilmente resolvida, resultando nas seguintes raízes:

$$x_1 = \frac{1+\sqrt{5}}{2} \quad \text{e} \quad x_2 = \frac{1-\sqrt{5}}{2}$$

Portanto têm-se duas soluções para a recorrência de Fibonacci, a saber:

$$f_1(n) = \left(\frac{1+\sqrt{5}}{2}\right)^n \text{ e } f_2(n) = \left(\frac{1-\sqrt{5}}{2}\right)^n.$$

Agora, em Matemática, demonstra-se um teorema que afirma que qualquer **combinação linear** das raízes de uma equação característica é uma solução para a recorrência. No caso da recorrência de Fibonacci, isso implica que:

$$f(n) = s\left(\frac{1+\sqrt{5}}{2}\right)^n + t\left(\frac{1-\sqrt{5}}{2}\right)^n$$

é uma solução para a recorrência. Os parâmetros s e t que aparecem na combinação linear podem ser determinados utilizando-se as condições de fronteira. Desse modo, no caso da recorrência de Fibonacci, obtém-se:

$$f(0) = s\left(\frac{1+\sqrt{5}}{2}\right)^0 + t\left(\frac{1-\sqrt{5}}{2}\right)^0 = s + t = 0 \tag{1}$$

e

$$f(1) = s\left(\frac{1+\sqrt{5}}{2}\right)^1 + t\left(\frac{1-\sqrt{5}}{2}\right)^1 = \left(\frac{1+\sqrt{5}}{2}\right)s + \left(\frac{1-\sqrt{5}}{2}\right)t = 1 \tag{2}$$

Resolvendo-se o sistema formado pelas equações **(1)** e **(2)**, obtém-se:

$$s = \frac{1}{\sqrt{5}} \text{ e } t = -\frac{1}{\sqrt{5}}.$$

Portanto a solução (forma fechada) para a recorrência de Fibonacci é dada por:

$$f(n) = \frac{1}{\sqrt{5}}\left(\frac{1+\sqrt{5}}{2}\right)^n - \frac{1}{\sqrt{5}}\left(\frac{1-\sqrt{5}}{2}\right)^n$$

B.10 Propriedades de Análise Assintótica

Esta seção apresentará demonstrações de teoremas enunciados no **Capítulo 6**.

Teorema 6.1 *$f(n)$ é $\theta(g(n))$* se e somente se *$g(n)$ é $\theta(f(n))$*.

Prova: ($\Rightarrow$) Suponha que $f(n)$ é $\theta(g(n))$. De acordo com a **Definição 6.3**, $f(n)$ é $O(g(n))$ e $f(n)$ é $\Omega(g(n))$. Como $f(n)$ é $O(g(n))$, então existem constantes $c_0 > 0$ e $n_0 \geq 0$ tais que $f(n) \leq c_0 g(n)$ para todo $n \geq n_0$. Mas, como $c_0 > 0$, tem-se que $g(n) \geq k_0 f(n)$, sendo $k_0 = 1/c_0$, para todo $n \geq n_0$. Isso mostra que $g(n)$ é $\Omega(f(n))$. Agora, como $f(n)$ é $\Omega(g(n))$, existem $c_0 > 0$ e $n_0 \geq 0$ tais que $f(n) \geq c_0 g(n)$ para todo $n \geq n_0$. Mas, como $c_0 > 0$, tem-se que $g(n) \leq k_0 f(n)$, sendo $k_0 = 1/c_0$, para todo $n \geq n_0$. Isso mostra que $g(n)$ é $O(f(n))$. Como $g(n)$ é $\Omega(f(n))$ e $g(n)$ é $O(f(n))$, tem-se que $g(n)$ é $\theta(f(n))$.

($\Leftarrow$) A segunda parte deste teorema pode ser demonstrada de modo semelhante à prova da primeira parte e é deixada como exercício para o leitor.

Corolário 6.1 *$f(n)$ é $\theta(g(n))$* se e somente se *$f(n)$ é $O(g(n))$* e *$g(n)$ é $O(f(n))$*.

Prova: ($\Rightarrow$) Suponha que $f(n)$ é $\theta(g(n))$. Então, por definição, $f(n)$ é $O(g(n))$ e, de acordo com o **Teorema 6.1**, $g(n)$ é $\theta(f(n))$, o que, por definição, significa que $g(n)$ é $O(f(n))$.

($\Leftarrow$) Suponha que $f(n)$ é $O(g(n))$ e $g(n)$ é $O(f(n))$. Como $g(n)$ é $O(f(n))$, tem-se que $f(n)$ é $\Omega(g(n))$ (v. prova do **Teorema 6.1**). Logo $f(n)$ é $\theta(g(n))$. ■

Corolário 6.2 *$f(n)$ é $\theta(g(n))$* se e somente se *$f(n)$ é $\Omega(g(n))$* e *$g(n)$ é $\Omega(f(n))$*.

Prova: Este corolário pode ser demonstrado de modo semelhante à prova do **Corolário 6.1**. Esta demonstração é deixada como exercício para o leitor.

Teorema 6.2 Suponha que a_0, a_1, ..., a_k sejam números reais, com $a_k \neq 0$. Então, tem-se que $T(n) = a_k n^k + a_{k-1} n^{k-1} + ... + a_1 n + a_0$ é $\theta(n^k)$.

Prova: Seja $c_0 = a_k + a_{k-1} + ... + a_1 + a_0$. Então, para qualquer n, tem-se que:

$T(n) = a_k n^k + a_{k-1} n^{k-1} + \ldots + a_1 n + a_0 \leq a_k n^k + a_{k-1} n^k + \ldots + a_1 n^k + a_0 n^k =$
$= (a_k + a_{k-1} + \ldots + a_1 + a_0) n^k = c_0 n^k$, o que mostra que $T(n)$ é $O(n^k)$. **(1)**

Agora, para qualquer n, tem-se também que:

$T(n) = a_k n^k + a_{k-1} n^{k-1} + \ldots + a_1 n + a_0 \geq a_k n^k$, o que mostra que $T(n)$ é $\Omega(n^k)$. **(2)**

Como, por **(1)**, $T(n)$ é $O(n^k)$ e, por **(2)**, $T(n)$ é $\Omega(n^k)$, conclui-se que $T(n)$ é $\theta(n^k)$. ■

Teorema 6.3 Se $T(n)$ é $c_1 f(n) + c_2$, então $T(n)$ é $\theta(f(n))$.

Prova: Para todo n, tem-se que $c_1 f(n) + c_2 \leq c_1 f(n) + c_2 f(n) = (c_1 + c_2) f(n)$. Portanto, considerando $c_0 = c_1 + c_2$, tem-se que $T(n)$ é $O(f(n))$.

Para todo n, tem-se que $c_1 f(n) + c_2 \geq c_1 f(n)$. Portanto $T(n)$ é $\Omega(f(n))$.

Como $T(n)$ é $O(f(n))$ e $T(n)$ é $\Omega(f(n))$, tem-se que $T(n)$ é $\theta(f(n))$. ■

Teorema 6.4 Se $T_1(n)$ é $\theta(f(n))$ e $T_2(n)$ é $\theta(g(n))$, então:

$$T_1(n) + T_2(n) \text{ é } \theta(max(f(n), g(n)))$$

Prova: Como, por hipótese, $T_1(n)$ é $\theta(f(n))$ e $T_2(n)$ é $\theta(g(n))$, então $T_1(n)$ é $O(f(n))$ e $T_2(n)$ é $O(g(n))$. Além disso, $T_1(n)$ é $\Omega(f(n))$ e $T_2(n)$ é $\Omega(g(n))$.

Como, $T_1(n)$ é $O(f(n))$ e $T_2(n)$ é $O(g(n))$, existem constantes $c_1 > 0$, $n_1 \geq 0$, $c_2 > 0$ e $n_2 \geq 0$ tais que:

$T_1(n) \leq c_1 f(n)$, para $n \geq n_1$ e $T_2(n) \leq c_2 g(n)$ para $n \geq n_2$.

Seja $n_0 = max(n_1, n_2)$. Então, tem-se que:

$T_1(n) + T_2(n) \leq c_1 f(n) + c_2 g(n)$ para $n \geq n_0$. **(1)**

Agora, por definição, $f(n) \leq max(f(n), g(n))$ e $g(n) \leq max(f(n), g(n))$ $\forall n$. **(2)**

Logo, em decorrência dos resultados **(1)** e **(2)**, obtém-se:

$T_1(n) + T_2(n) \leq c_1 \cdot max(f(n), g(n)) + c_2 \cdot max(f(n), g(n)) = (c_1 + c_2) max(f(n), g(n))$ para $n \geq n_0$.

Portanto, considerando-se $c_0 = c_1 + c_2$, $T_1(n) + T_2(n)$ é $O(max(f(n), g(n)))$.

Resta mostrar que $T_1(n) + T_2(n)$ é $\Omega(max(f(n), g(n)))$.

Como $T_1(n)$ é $\Omega(f(n))$ e $T_2(n)$ é $\Omega(g(n))$, então existem constantes $c_1 > 0$, $n_1 \geq 0$, $c_2 > 0$ e $n_2 \geq 0$ tais que:

$T_1(n) \geq c_1 f(n)$, para $n \geq n_1$ e $T_2(n) \geq c_2 g(n)$ para $n \geq n_2$.

Seja $n_0 = max(n_1, n_2)$. Então, tem-se que:

$T_1(n) + T_2(n) \geq c_1 f(n) + c_2 g(n)$ para $n \geq n_0$ **(3)**

Tem-se ainda que:

$max(f(n), g(n)) \leq f(n) + g(n) \Rightarrow c_1 f(n) + c_2 g(n) \geq max(f(n), g(n))$ **(4)**

Levando em consideração os resultados **(3)** e **(4)**, obtém-se

$T_1(n) + T_2(n) \geq c_1 f(n) + c_2 g(n) \geq max(f(n), g(n))$

Portanto $T_1(n) + T_2(n)$ é $\Omega(max(f(n), g(n)))$ ■

Teorema 6.5 Se $T_1(n)$ é $\theta(f(n))$ e $T_2(n)$ é $\theta(g(n))$, então $T_1(n) \cdot T_2(n)$ é $\theta(f(n) \cdot g(n))$.

Prova: Como $T_1(n)$ é $\theta(f(n))$ e $T_2(n)$ é $\theta(g(n))$, $T_1(n)$ é $O(f(n))$ e $T_2(n)$ é $O(g(n))$. Logo existem constantes $c_1 > 0$, $n_1 \geq 0$, $c_2 > 0$ e $n_2 \geq 0$ tais que:

$T_1(n) \leq c_1 f(n)$, para $n \geq n_1$ e $T_2(n) \leq c_2 g(n)$, para $n \geq n_2$.

Seja $n_0 = max(n_1, n_2)$. Então, tem-se que:

$T_1(n) T_2(n) \leq c_1 f(n) c_2 g(n) = c_1 c_2 f(n) g(n)$, para $n \geq n_0$.

Portanto, tomando-se $c_0 = c_1 c_2$, tem-se: $T_1(n) T_2(n)$ é $O(f(n) g(n))$.

Resta provar que $T_1(n)T_2(n)$ é $\Omega(f(n)g(n))$.
Como $T_1(n)$ é $\Omega(f(n))$ e $T_2(n)$ é $\Omega(g(n))$, existem constantes $c_1 > 0$, $n_1 \geq 0$, $c_2 > 0$ e $n_2 \geq 0$ tais que:
$T_1(n) \geq c_1 f(n)$, para $n \geq n_1$ e $T_2(n) \geq c_2 g(n)$, para $n \geq n_2$.
Seja $n_0 = max(n_1, n_2)$. Então, tem-se que:
$T_1(n)T_2(n) \geq c_1 f(n) c_2 g(n) = c_1 c_2 f(n) g(n)$, para $n \geq n_0$.
Portanto, tomando-se $c_0 = c_1 c_2$, tem-se: $T_1(n)T_2(n)$ é $\Omega(f(n)g(n))$. ■

Teorema 6.6 n^k é $O(c^n)$, para $k > 0$ e $c > 1$.

Prova: Deve-se mostrar que existem constantes $c_0 > 0$ e $n_0 \geq 0$ tais que:
$n^k \leq c_0 c^n$ para $n \geq n_0$.
Agora, considerando-se $c_0 = 1$ e $n_0 = 1$, pode-se mostrar por indução (tente fazer isso) que:
$n^k \leq c^n$ para $n \geq 1$. ■

Teorema 6.7 Se $T(n)$ é $\theta(f(n))$ e $f(n)$ é $\theta(g(n))$, então $T(n)$ é $\theta(g(n))$.

Prova: Se $T(n)$ é $\theta(f(n))$ e $f(n)$ é $\theta(g(n))$, então $T(n)$ é $O(f(n))$ e $f(n)$ é $O(g(n))$. Logo existem constantes $c_1 > 0$, $n_1 \geq 0$, $c_2 > 0$ e $n_2 \geq 0$ tais que:
$T(n) \leq c_1 f(n)$, para $n \geq n_1$ e $f(n) \leq c_2 g(n)$, para $n \geq n_2$.
Seja $n_0 = max(n_1, n_2)$. Então, para $n \geq n_0$, tem-se que:
$T(n) \leq c_1 f(n) \leq c_2 g(n) \Rightarrow f(n) \leq c_1/c_2 g(n)$.
Portanto, considerando-se $c_0 = c_1/c_2$, tem-se que: $T(n)$ é $O(g(n))$.
Resta mostrar que $T(n)$ é $\Omega(g(n))$.
Como $T(n)$ é $\Omega(f(n))$ e $f(n)$ é $\Omega(g(n))$, existem constantes $c_1 > 0$, $n_1 \geq 0$, $c_2 > 0$ e $n_2 \geq 0$ tais que:
$T(n) \geq c_1 f(n)$, para $n \geq n_1$ e $f(n) \geq c_2 g(n)$, para $n \geq n_2$.
Seja $n_0 = max(n_1, n_2)$. Então, para $n \geq n_0$, tem-se que:
$T(n) \geq c_1 f(n) \geq c_2 g(n) \Rightarrow f(n) \geq c_1/c_2 g(n)$.
Portanto, considerando-se $c_0 = c_1/c_2$, tem-se que: $T(n)$ é $\Omega(g(n))$. ■

Observação: O **Lema B.1** a seguir não aparece no **Capítulo 6**, mas será útil na prova do **Teorema 6.8**.

Lema B.1 $\forall\, x, a, b \in \mathbb{R}^+ \mid a \neq 1$ e $b \neq 1$, tem-se que $a^{\log_b x} = x^{\log_b a}$.

Prova:

$$a^{\log_b x} = a^{\left(\frac{\log_a x}{\log_a b}\right)} = \left(a^{\log_a x}\right)^{\left(\frac{\log_a a}{\log_a b}\right)} = x^{\log_b a}$$ ■

Teorema 6.8 Suponha que se tenha uma relação de recorrência da forma:

$$T(n) = \begin{cases} \text{não negativo} & \text{se } n = 1 \\ a \cdot T(n/2) + n & \text{se } n > 1 \end{cases}$$

Sendo a uma constante inteira não negativa, então tem-se que:
(i) Se $a < 2$, $T(n)$ é $\theta(n)$.
(ii) Se $a = 2$, $T(n)$ é $\theta(n \log n)$.
(iii) Se $a > 2$, $T(n)$ é $\theta(n^{\log a})$.

Prova: A discussão apresentada na **Seção 6.10** serve com prova dos **Casos (i)** e **(ii)**. No **Caso (iii)**, tem-se que, no nível i da árvore de recursão existem a^i nós, cada um dos quais corresponde a um subproblema de tamanho $n/2^i$. Logo, no nível i, o custo de processamento total é $a^i \cdot (n/2^i) = n \cdot (a/2)^i$. Efetuando-se o somatório sobre os $log\ n$ níveis, obtém-se como custo total:

$$T(n) = a^{\log n} \cdot T(1) + n \cdot \sum_{i=0}^{\log n - 1} \left(\frac{a}{2}\right)^i = a^{\log n} \cdot T(1) + n \cdot \frac{2 \cdot (n - a^{\log n})}{n \cdot (2-a)} = a^{\log n} \cdot T(1) + \frac{2 \cdot (n - a^{\log n})}{2-a}$$

Como $a > 2$ e a função $f(x) = log\ x$ é crescente, tem-se que $T(n)$ é $\theta(a^{log\ n})$ e, de acordo com o **Lema B.1**, $T(n)$ é $\theta(n^{log\ a})$, o que prova o **Caso (iii)**. ■

B.11 Propriedades de Árvores Binárias

Esta seção apresenta demonstrações de teoremas enunciados no **Capítulo 12**.

Teorema 12.1 **(i)** O número máximo de nós no nível i de uma árvore binária é 2^{i-1}, $i \geq 1$. (ii) O número máximo de nós numa árvore binária de profundidade p é $2^p - 1$, $p \geq 1$.

Prova:

(i) A prova é feita por indução sobre i (v. **Seção B.8**). Para $i = 1$, tem-se que $2^{i-1} = 2^{1-1} = 1$, pois só há um nó no nível 1, que é a raiz. Suponha agora que a parte (i) do teorema seja válida para $i = n$; isto é, que o número máximo de nós no nível n é dado por:

$$2^{n-1}$$

Então, deve-se mostrar que esta equação também vale para $i = n + 1$, ou seja, que o número máximo de nós no nível $n + 1$ é dado por:

$$2^n$$

Ora, como cada nó tem no máximo grau 2, então o número máximo de nós no nível $n + 1$ é 2 vezes o número máximo de nós no nível n, isto é, $2.2n\text{-}1 = 2n$. Assim a parte (i) da proposição está provada.

(ii) O número máximo de nós numa árvore binária de profundidade p é o somatório do número máximo de nós em todos os níveis da árvore, ou seja:

$$\sum_{i=1}^{k} 2^{i-1} = 2^k - 1$$

Exercício: A prova dessa última igualdade pode também ser feita por indução (v. **Seção B.8**). Como exercício, tente demonstrá-la.

Teorema 12.2 Para qualquer árvore binária, se n_0 é o número de nós terminais (folhas) e n_2 é o número de nós de grau 2, então $n_0 = n_2 + 1$.

Prova: Sejam n_1 o número de nós de grau 1 e n o número total de nós. Uma vez que qualquer nó possui grau menor do que ou igual a 2 tem-se que:

$n = n_0 + n_1 + n_2$ **(1)**

Agora, cada nó, exceto a raiz, possui uma ramificação que o liga a seu nó pai. Então, se r é o número de ramificações, tem-se $n = r + 1$. Mas, cada ramificação parte de um nó de grau um ou de um nó de grau dois. Assim $r = n_1 + 2n_2$ e, portanto:

$n = 1 + n_1 + 2n_2$ **(2)**

Resolvendo-se as equações **(1)** e **(2)**, obtém-se o resultado desejado:

$n_0 = n_2 + 1$ ■

Teorema 12.3 Se n é o número de nós de uma árvore estritamente binária e p é a sua profundidade, então a seguinte relação é válida: $2p - 1 \leq n \leq 2^p - 1$ $(p > 0)$.

Prova: Este teorema pode ser parafraseado assim: a menor árvore estritamente binária com profundidade p que existe possui $2^p - 1$ nós e a maior árvore estritamente binária com profundidade p que existe possui $2^p - 1$ nós.

Agora, a menor árvore estritamente binária com profundidade p possui dois nós em cada nível, exceto no primeiro nível, que só contém a raiz. Portanto o número de nós dessa árvore é dado por:

$$\left(\sum_{i=1}^{k} 2\right) - 1 = 2k - 1$$

Por outro lado, a maior árvore estritamente binária com profundidade p possui o número máximo de nós em cada nível que, de acordo com o **Teorema 12.1**, é dado por 2^{i-1}. Mas, de acordo com esse mesmo teorema, o número total de nós numa árvore dessa natureza é igual a $2^p - 1$. ■

Teorema 12.4 Numa árvore estritamente binária, o número total de nós n é dado por: $n = 2n_0 - 1$, sendo n_0 o número de folhas da árvore.

Prova: De acordo com o **Teorema 12.2**, tem-se que, para qualquer árvore binária:

$n_0 = n_2 + 1$ **(1)**

Mas, numa árvore estritamente binária, não existe nó com grau igual a um. Logo:

$n = n_0 + n_2$ **(2)**

Resolvendo-se as equações (1) e (2), obtém-se:

$n = 2n_0 - 1$ ■

Teorema 12.5 O número de nós de grau dois (n_2) de uma árvore binária perfeita mantém a seguinte relação com o número total de folhas (n_0): $n_2 = n_0 - 1$.

Prova: Seja p a profundidade de uma árvore binária perfeita. Então, o número total de nós (n) de uma árvore binária perfeita é, de acordo com a **Definição 12.7** e o **Teorema 12.1**, dado por:

$n = 2^p - 1 \Rightarrow n_0 + n_2 = 2^p - 1 \Rightarrow n_0 + n_2 + 1 = 2^p$ **(1)**

A segunda igualdade acima é decorrente do fato de só existirem nós de grau dois (n_2) e de grau zero (n_0). Agora, de acordo com a parte (i) do **Teorema 12.1**, tem-se que:

$n_0 = 2^{p-1} \Rightarrow 2n_0 = 2^p$ **(2)**

Resolvendo-se as equações **(1)** e **(2)** acima, obtém-se o resultado desejado:

$n_2 = n_0 - 1$ ■

Corolário 12.1 Numa árvore binária perfeita, o número de folhas (n_0) mantém a seguinte relação com o número total de nós (n):

$$n_0 = \frac{n+1}{2}$$

Prova: De acordo com o **Teorema 12.5**, tem-se:

$n_2 = n_0 - 1$ **(1)**

Agora, o número total de nós (n) de uma árvore binária perfeita é (v. acima) dado por:

$n = n_0 + n_2$ **(2)**

Resolvendo-se as equações **(1)** e **(2)** acima, obtém-se o resultado desejado. ■

Teorema 12.6 A profundidade p de uma árvore binária completa com n nós é dada por $p = \lfloor log_2\, n + 1 \rfloor$.

Prova: A menor árvore binária completa com profundidade p que existe possui apenas um nó no nível p, de modo que, de acordo com o **Teorema 12.1 (ii)**, o número mínimo de nós numa árvore binária completa com profundidade p é dado por:

$$n_{min} = 2^{p-1} - 1 + 1 = 2^{p-1}$$

A maior árvore binária completa com profundidade p que existe possui o número máximo de nós no nível p, que, de acordo com o **Teorema 12.1 (ii)**, é dado por:

$$n_{min} = 2^{p} - 1$$

Portanto o número de nós de uma árvore binária completa com profundidade p satisfaz a seguinte relação:

$$2^{p-1} \leq n \leq 2^{p} - 1$$

Desenvolvendo-se a primeira desigualdade, obtém-se:

$$p \leq log_2\, n + 1 \qquad \textbf{(1)}$$

Desenvolvendo-se a segunda desigualdade, obtém-se:

$$p \geq log_2\, (n + 1) \geq log_2\, n \qquad \textbf{(2)}$$

Combinando-se **(1)** e **(2)** e levando em conta a propriedade da função piso que afirma que:

$$m \leq x \leq m + 1 \Leftrightarrow m = \lfloor x \rfloor$$

obtém-se o resultado desejado:

$$p(n) = \lfloor \log_2 n + 1 \rfloor$$

■

Teorema 12.7 Se uma árvore binária completa com n nós for representada sequencialmente conforme foi descrito na **Seção 12.2.4**, então, para qualquer nó numerado por i, $0 \leq i \leq n - 1$, tem-se:

(i) $Pai(i)$ é numerado como $\lfloor (i - 1)/2 \rfloor$, se $i \neq 0$. Se $i = 0$, o nó i é a raiz da árvore, que não possui pai.

(ii) $FilhoEsquerdo(i)$ é numerado como $2 \cdot i + 1$, se $2 \cdot i + 1 \leq n$. Se $2 \cdot i + 1 > n$, então o nó i não possui filho da esquerda.

(iii) $FilhoDireito(i)$ é numerado como $2 \cdot i + 2$, se $2 \cdot i + 2 \leq n$. Se $2 \cdot i + 2 > n$, então o nó i não possui filho da direita.

Prova: Em primeiro lugar, deve-se notar que se a parte (ii) do teorema for provada, a parte (iii) torna-se uma consequência imediata, pois, quando um nó é numerado como $2 \cdot i + 1$, seu irmão direito, quando ele existe, é obviamente numerado como $2 \cdot i + 2$, devido ao esquema de numeração da esquerda para a direita. Além disso, tendo provado as partes (ii) e (iii), a parte (i) do teorema passa a ser consequência imediata dessa prova, visto que um nó que possui filhos numerados como $2 \cdot i + 1$ e $2 \cdot i + 2$ só pode ser numerado como $\lfloor (i - 1)/2 \rfloor$, devido ao esquema de numeração por níveis. Portanto resta demonstrar a parte (ii) do teorema e essa prova será feita por indução a seguir.

Para $i = 0$, o nó que apresenta essa numeração é a raiz e seu filho à esquerda é numerado com 1, a não ser que a árvore só possua raiz. (Não esqueça que, por hipótese, a árvore binária é completa.)

Como hipótese indutiva, suponha que a parte (ii) do teorema seja verdadeira para $i = k$; i.e., $FilhoEsquerdo(k)$ é numerado como $2 \cdot k + 1$. Então, deve-se mostrar que $FilhoEsquerdo(k + 1)$ é numerado como $2 \cdot (k + 1) + 1$.

Agora, de acordo com o esquema de numeração proposto, somando-se 2 a $FilhoEsquerdo(k)$, obtém-se $FilhoEsquerdo(k + 1)$. Mas, de acordo com a hipótese indutiva, $FilhoEsquerdo(k)$ é numerado como $2 \cdot k + 1$, de modo que a numeração de $FilhoEsquerdo(k + 1)$ é dada por $2 \cdot k + 1 + 2 = 2 \cdot (k + 1) + 1$.

■