DCC638 - Introdução à Lógica Computacional
2023.1

Recursão

Área de Teoria DCC/UFMG

Definições Recursivas e Indução Estrutural

Recursão: Introdução

- Algumas vezes não é fácil definir um objeto explicitamente, mas é relativamente mais fácil defini-lo em termos de si próprio.

  Por exemplo:

  1. Definição dos números naturais em termos de números naturais:
     - 0 é um número natural;
     - o sucessor de um número natural é um número natural.

- A definição de um objeto em termos de si próprio é chamada **definição recursiva**.

- A **recursão** é muito utilizada para definir, por exemplo:

  1. funções,
  2. sequências,
  3. conjuntos, e
  4. algoritmos.

Definição recursiva de funções

- Uma **definição recursiva de uma função** com domínio nos números inteiros não-negativos tem duas partes:

  Definição recursiva de função:

  Passo base: Especifica-se o valor da função em 0.

  Passo recursivo: Especifica-se uma regra para encontrar o valor da função em um inteiro qualquer baseada no valor da função em inteiros menores.

- Lembre-se de que uma função $f(n)$ dos inteiros não-negativos para os reais é equivalente a uma sequência

$$a_0, \quad a_1, \quad a_2, \quad \ldots,$$

  onde a_i é um número real para todo inteiro não-negativo i.

  Logo, definir uma sequência $a_0, a_1, a_2, \ldots$ de números reais de forma recursiva é equivalente a definir uma função recursiva dos inteiros não-negativos para os reais.

Definição recursiva de funções

- $\boxed{\text{Exemplo 1}}$ Seja a função f definida como

$$\begin{cases} f(0) = 3, \\ f(n) = 2f(n-1) + 3, \quad n \geq 1. \end{cases}$$

Encontre $f(1)$, $f(2)$, $f(3)$, $f(4)$.

Solução.

- $f(1) = 2f(0) + 3 = 2 \cdot 3 + 3 = 9$;
- $f(2) = 2f(1) + 3 = 2 \cdot 9 + 3 = 21$;
- $f(3) = 2f(2) + 3 = 2 \cdot 21 + 3 = 45$;
- $f(4) = 2f(3) + 3 = 2 \cdot 45 + 3 = 93$.

Definição recursiva de funções

- $\boxed{\text{Exemplo 2}}$ Encontre uma definição recursiva para a função fatorial $f(n) = n!$, e compute $f(5)$ usando sua definição.

Definição recursiva de funções

- $\boxed{\text{Exemplo 2}}$ Encontre uma definição recursiva para a função fatorial $f(n) = n!$, e compute $f(5)$ usando sua definição.

  Solução. Uma definição recursiva para $f(n) = n!$ é:

$$\begin{cases} f(0) = 1, \\ f(n) = n \cdot f(n-1), \quad n \geq 1 \end{cases}$$

  Podemos então calcular $f(5)$ como:

$$\begin{aligned} f(5) &= 5 \cdot f(4) \\ &= 5 \cdot (4 \cdot f(3)) \\ &= 5 \cdot (4 \cdot (3 \cdot f(2))) \\ &= 5 \cdot (4 \cdot (3 \cdot (2 \cdot f(1)))) \\ &= 5 \cdot (4 \cdot (3 \cdot (2 \cdot (1 \cdot f(0))))) \\ &= 5 \cdot (4 \cdot (3 \cdot (2 \cdot (1 \cdot 1)))) \\ &= 120. \end{aligned}$$

Definição recursiva de funções

- $\boxed{\text{Exemplo 3}}$ Encontre uma definição recursiva para a função $f(n) = a^n$, tendo como domínio os naturais, e compute $f(3)$ usando sua definição.

Definição recursiva de funções

- $\boxed{\text{Exemplo 3}}$ Encontre uma definição recursiva para a função $f(n) = a^n$, tendo como domínio os naturais, e compute $f(3)$ usando sua definição.

  Solução. Uma definição recursiva para $f(n) = a^n$ é:

$$\begin{cases} f(0) = 1, \\ f(n) = a \cdot f(n-1), \quad n \geq 1 \end{cases}$$

  Podemos então calcular $f(3)$ como:

$$\begin{aligned} f(3) &= a \cdot f(2) \\ &= a \cdot (a \cdot f(1)) \\ &= a \cdot (a \cdot (a \cdot f(0))) \\ &= a \cdot (a \cdot (a \cdot 1)) \\ &= a^3. \end{aligned}$$

Definição recursiva de funções

- $\boxed{\text{Exemplo 4}}$ Outros exemplos de definições recursivas:

Somatório:	$\begin{cases} \sum_{i=1}^{1} a_i = a_1 \\ \sum_{i=1}^{n} a_i = \left(\sum_{i=1}^{n-1} a_i\right) + a_n, \quad n \geq 2 \end{cases}$
Produtório:	$\begin{cases} \prod_{i=1}^{1} a_i = a_1 \\ \prod_{i=1}^{n} a_i = \left(\prod_{i=1}^{n-1} a_i\right) \cdot a_n, \quad n \geq 2 \end{cases}$
União:	$\begin{cases} \bigcup_{i=1}^{1} A_i = A_1 \\ \bigcup_{i=1}^{n} A_i = \left(\bigcup_{i=1}^{n-1} A_i\right) \cup A_n, \quad n \geq 2 \end{cases}$
Interseção:	$\begin{cases} \bigcap_{i=1}^{1} A_i = A_1 \\ \bigcap_{i=1}^{n} A_i = \left(\bigcap_{i=1}^{n-1} A_i\right) \cap A_n, \quad n \geq 2 \end{cases}$

Definição recursiva de funções

- $\boxed{\text{Exemplo 5}}$ A sequência de Fibonacci é aquela em que os dois primeiros termos são 1, e cada termo seguinte é a soma dos dois anteriores:

$$0, \quad 1, \quad 1, \quad 2, \quad 3, \quad 5, \quad 8, \quad 13, \quad \ldots$$

Esta seguência pode ser definida recursivamente como:

$$\begin{cases} f(0) = 0, \\ f(1) = 1, \\ f(n) = f(n-1) + f(n-2), \quad \text{para } n \geq 2. \end{cases}$$

Para calcular $f(5)$ podemos fazer:

$$\begin{aligned} f(2) &= f(1) + f(0) = 1 + 0 = 1, \\ f(3) &= f(2) + f(1) = 1 + 1 = 2, \\ f(4) &= f(3) + f(2) = 2 + 1 = 3, \\ f(5) &= f(4) + f(3) = 3 + 2 = 5. \end{aligned}$$

Definição recursiva de funções

- $\boxed{\text{Exemplo 6}}$ A função 91 de McCarthy é a função sobre os inteiros positivos definida como:

$$M(n) = \begin{cases} n - 10, & n > 100, \\ M(M(n+11)), & n \leq 100. \end{cases}$$

Podemos, então, calcular:

$$\begin{aligned} M(99) &= M(M(110)) && \text{(já que } 99 \leq 100\text{)} \\ &= M(100) && \text{(já que } 110 > 100\text{)} \\ &= M(M(111)) && \text{(já que } 100 \leq 100\text{)} \\ &= M(101) && \text{(já que } 111 > 100\text{)} \\ &= 91 && \text{(já que } 101 > 100\text{)} \end{aligned}$$

Esta função retorna 91 para todo inteiro positivo $n \leq 100$, e para inteiros positivo $n > 100$ ela começa em 91 e vai aumentando de 1 em 1.

Definição recursiva de funções

- $\boxed{\text{Exemplo 7}}$ Seja a sequência $\{a_n\}$, $n = 1, 2, 3, \ldots$ definida explicitamente como

$$a_n = n^2 + 3n$$

Encontre uma definição recursiva para esta sequência.

Definição recursiva de funções

- $\boxed{\text{Exemplo 7}}$ Seja a sequência $\{a_n\}$, $n = 1, 2, 3, \ldots$ definida explicitamente como

$$a_n = n^2 + 3n$$

Encontre uma definição recursiva para esta sequência.

Solução.

Primeiro determinamos o passo base da sequência, ou seja, a_1:

$$\begin{aligned} a_1 &= 1^2 + 3 \cdot 1 \\ &= 4. \end{aligned}$$

Agora determinamos o passo recursivo a_{n+1} para $n \geq 1$:

$$\begin{aligned} a_{n+1} &= (n+1)^2 + 3(n+1) && \text{(pela definição explícita da sequência)} \\ &= n^2 + 2n + 1 + 3n + 3 && \text{(expandindo os produtos)} \\ &= (n^2 + 3n) + 2n + 4 && \text{(reagrupando as parcelas)} \\ &= a_n + 2n + 4 && \text{(pela definição explícita da sequência).} \end{aligned}$$

- $\boxed{\text{Exemplo 7}}$ (Continuação)

  Assim obtemos a seguinte definição recursiva para a sequência:

$$\begin{cases} a_1 = 4, \\ a_{n+1} = a_n + 2n + 4, \quad \text{para } n \geq 1. \end{cases}$$

  Note que o passo recursivo da definição acima define os termos $a_2, a_3, a_4 \ldots$ dando uma fórmula para calcular cada termo a_{n+1} para $n \geq 1$.

Definição recursiva de funções

- $\boxed{\text{Exemplo 7}}$ (Continuação)

  Alternativamente, podemos encontrar um passo recursivo que defina $a_2, a_3, a_4 \ldots$ dando uma fórmula para calcular a_n para $n \geq 2$.

  Primeiro determinamos a_{n-1} em função de a_n:

$$\begin{aligned}
a_{n-1} &= (n-1)^2 + 3(n-1) && \text{(pela definição explícita da sequência)}\\
&= n^2 - 2n + 1 + 3n - 3 && \text{(expandindo os produtos)}\\
&= (n^2 + 3n) - 2n - 2 && \text{(reagrupando as parcelas)}\\
&= a_n - 2n - 2 && \text{(pela definição explícita da sequência).}
\end{aligned}$$

  Uma vez determinado que $a_{n-1} = a_n - 2n - 2$, podemos deduzir que $a_n = a_{n-1} + 2n + 2$, o que produz a seguinte definição recursiva:

$$\begin{cases} a_1 = 4, \\ a_n = a_{n-1} + 2n + 2, \quad \text{para } n \geq 2. \end{cases}$$

Definição recursiva de funções

- $\boxed{\text{Exemplo 7}}$ (Continuação)

  Podemos verificar que as duas definições recursivas encontradas são equivalentes à definição explícita para alguns termos da sequência:

Definição explícita: $a_n = n^2 + 3n,\ n \geq 1$	Primeira def. recursiva: $\begin{cases} a_1 = 4, \\ a_{n+1} = a_n + 2n + 4,\ n \geq 1 \end{cases}$	Segunda def. recursiva: $\begin{cases} a_1 = 4, \\ a_n = a_{n-1} + 2n + 2,\ n \geq 2 \end{cases}$
$a_1 = 1^2 + 3 \cdot 1 = 4$	$a_1 = 4$	$a_1 = 4$
$a_2 = 2^2 + 3 \cdot 2 = 10$	$a_2 = 4 + 2 \cdot 1 + 4 = 10$	$a_2 = 4 + 2 \cdot 2 + 2 = 10$
$a_3 = 3^2 + 3 \cdot 3 = 18$	$a_3 = 10 + 2 \cdot 2 + 4 = 18$	$a_3 = 10 + 2 \cdot 3 + 2 = 18$
$a_4 = 4^2 + 3 \cdot 4 = 28$	$a_4 = 18 + 2 \cdot 3 + 4 = 28$	$a_4 = 18 + 2 \cdot 4 + 2 = 28$
$a_5 = 5^2 + 3 \cdot 5 = 40$	$a_5 = 28 + 2 \cdot 4 + 4 = 40$	$a_5 = 28 + 2 \cdot 5 + 2 = 40$
...	...	...

Definição recursiva de conjuntos e estruturas

- Uma **definição recursiva de um conjunto** tem duas partes:

  Definição recursiva de conjunto:

  Passo base: Especifica-se uma coleção inicial de objetos pertencente ao conjunto.

  Passo recursivo: Especificam-se regras para formar novos elementos a partir dos elementos já pertencentes ao conjunto.

- A definição recursiva de conjuntos também depende da seguinte regra, frequentemente implícita:

  Regra de exclusão: elementos que não podem ser gerados a partir da aplicação do passo base e instâncias do passo indutivo não pertencem ao conjunto.

Definição recursiva de conjuntos e estruturas

- $\boxed{\text{Exemplo 8}}$ Seja o conjunto S definido como:

$$\begin{cases} 3 \in S, \\ \text{se } x \in S \text{ e } y \in S, \text{ então } x + y \in S. \end{cases}$$

Então, é verdade que:

- $6 \in S$? Sim, porque $3 \in S$ e $3 + 3 = 6$,
- $9 \in S$? Sim, porque $3 \in S$ e $6 \in S$ e $3 + 6 = 9$,
- $12 \in S$? Sim, porque $3 \in S$ e $9 \in S$ e $3 + 9 = 12$,
- $7 \in S$? Não, pela regra de exclusão.

O conjunto S é o conjunto dos múltiplos positivos de 3.

Definição recursiva de conjuntos e estruturas

- Muitos problemas lidam com **palavras**, ou **strings**, formadas a partir de um **alfabeto**.
- O conjunto Σ^* de **strings** sobre um alfabeto Σ pode ser definido recursivamente como:

  Passo base: $\lambda \in \Sigma^*$ (onde λ representa a string vazia, sem símbolo algum).

  Passo recursivo: Se $w \in \Sigma^*$ e $x \in \Sigma$ então $wx \in \Sigma^*$ (onde wx representa a string formada pelo símbolo x concatenado ao final do prefixo w).

- $\boxed{\text{Exemplo 9}}$ Seja o alfabeto $\Sigma = \{0, 1\}$. Qual o conjunto de strings Σ^* que pode ser formado a partir de Σ?

  Solução.

  Sabemos que:

  - $\lambda \in \Sigma^*$ pelo passo base;
  - $0 \in \Sigma^*$ porque $\lambda \in \Sigma^*$, $0 \in \Sigma$, e podemos juntar $\lambda 0 = 0$;
  - $1 \in \Sigma^*$ porque $\lambda \in \Sigma^*$, $1 \in \Sigma$, e podemos juntar $\lambda 1 = 1$;
  - $00 \in \Sigma^*$ porque $0 \in \Sigma^*$, $0 \in \Sigma$, e podemos juntar 00;
  - $01 \in \Sigma^*$ porque $0 \in \Sigma^*$, $1 \in \Sigma$, e podemos juntar 01;
  - $011 \in \Sigma^*$ porque $01 \in \Sigma^*$, $1 \in \Sigma$, e podemos juntar 011;
  - ...

  De fato, Σ^* é o conjunto de todas as strings binárias.

Definição recursiva de conjuntos e estruturas

- Seja ℓ (*length*) a função que retorna o **comprimento** de uma string, ou seja, para todo $w \in \Sigma^*$, o valor $\ell(w)$ é o número de símbolos em w.

  Podemos definir ℓ recursivamente como:

  Passo base: $\ell(\lambda) = 0$.

  Passo recursivo: $\ell(wx) = \ell(w) + 1$, se $w \in \Sigma^*$ e $x \in \Sigma$.

- $\boxed{\text{Exemplo 10}}$ Podemos calcular o tamanho $\ell(01011)$ como:

$$
\begin{aligned}
\ell(01011) &= \ell(0101) + 1 \\
&= (\ell(010) + 1) + 1 \\
&= ((\ell(01) + 1) + 1) + 1 \\
&= (((\ell(0) + 1) + 1) + 1) + 1 \\
&= ((((\ell(\lambda) + 1) + 1) + 1) + 1) + 1 \\
&= ((((0 + 1) + 1) + 1) + 1) + 1 \\
&= 5
\end{aligned}
$$

Definição recursiva de conjuntos e estruturas

- $\boxed{\text{Exemplo 11}}$ O conjunto das sentenças lógicas bem formadas pode ser definido recursivamente como:

  Passo base: T, F e s são sentenças bem formadas, onde s representa uma proposição lógica.

  Passo recursivo: Se G e H são sentenças bem formadas, então $(\neg G)$, $(G \wedge H)$, $(G \vee H)$, $(G \rightarrow H)$ e $(G \leftrightarrow H)$ são sentenças bem formadas.

- Podemos facilmente verificar que, se p e q são proposições lógicas, então:

  1. F, $(p \wedge T)$, $(p \rightarrow q)$, $((p \rightarrow q) \vee T)$ são sentenças bem formadas.
  2. $\neg pq$, $\wedge q$, TF são sentenças mal-formadas (pela regra da exclusão).

Definição recursiva de árvores

- Para o próximo exemplo, vamos precisar de alguns conceitos úteis.
- Um grafo $G = (V, E)$ é formado por:
  - um conjunto V de **vértices** ou **vértices**,e
  - um conjunto E de **arestas**, em que cada aresta é um par ordenado (v_i, v_j) indicando que os vértices $v_i, v_j \in V$ estão conectados.
- Um **ciclo** em um grafo é um caminho de arestas consecutivas que começa e termina no mesmo vértice.
- Um **vértice interno** está conectado a pelo menos dois outros vértices do grafo.
- Uma **folha** é um vértice conectado a no máximo um outro vértice.

- Por exemplo, no grafo abaixo:

$$\begin{aligned} \mathrm{G} &= \{a, b, c, d, e, f, g, h\} \\ \mathrm{E} &= \{(a, b), (a, d), (b, c), (c, g), \\ &\quad (c, f), (f, h), (f, e), (g, h)\} \end{aligned}$$

  - Vértices são representados por círculos e arestas por linhas conectando vértices.
  - Existe um ciclo começando no vértice c e passando por g, h, f, até voltar em c.
  - Os vértices a, b, c, f, g, h são vértices internos.
  - Os vértice d, e são folhas.

Definição recursiva de árvores

- $\boxed{\text{Exemplo 12}}$ Uma **árvore** é um grafo sem ciclos. Uma **árvore binária completa** é uma árvore em que cada vértice, com exceção das folhas, possui exatamente dois vértices filhos.

  Uma árvore binária completa pode ser definida recursivamente como:

  Passo base: Um vértice isolado é uma árvore binária completa.

  Passo recursivo: Se T_1 e T_2 são árvores binárias completas disjuntas com raízes r_1 e r_2, respectivamente, então pode-se formar uma nova árvore binária completa ao se conectar um vértice r (não presente em T_1 ou T_2, que chamaremos de *raiz*) através de uma aresta a r_1 e outra aresta a r_2.

Definição recursiva de árvores

- $\boxed{\text{Exemplo 12}}$ (Continuação)

  Exemplo de construção recursiva de árvores binárias completas:

Indução estrutural

- Se um conjunto tem uma definição recursiva, é possível demonstrar propriedades dos elementos deste conjunto através de indução.
- A **indução estrutural** é uma maneira de mostrar que se:
  1. os elementos iniciais do conjunto (passo base) satisfazem uma certa propriedade, e
  2. as regras de construção de novos elementos (passo indutivo) preservam esta propriedade,

  então todos os elementos do conjunto satisfazem a propriedade.

Indução estrutural

- Uma **demonstração por indução estrutural** tem duas partes:

  Demonstração por indução estrutural:

  Passo base: Mostra-se que a proposição é válida para todos os elementos especificados no passo base da definição recursiva do conjunto.

  Passo indutivo: Mostra-se que se a proposição é válida para cada um dos elementos usados para se construírem novos elementos do conjunto, então a proposição também é válida para estes novos elementos.

- A **hipótese de indução** é de que a proposição vale para cada um dos elementos usados para se construírem novos elementos do conjunto.

Exemplos de uso de indução estrutural

- $\boxed{\text{Exemplo 13}}$ Seja o conjunto A definido como:

$$\begin{cases} 3 \in A, \\ x, y \in A \rightarrow x + y \in A. \end{cases}$$

Mostre que todos os elementos de A são divisíveis por 3.

Demonstração. Seja $P(x)$ a proposição *"x é divisível por 3"*.

Passo base: O único elemento da base é 3, e $P(3)$ é verdadeiro porque 3 é divisível por 3.

Exemplos de uso de indução estrutural

- $\boxed{\text{Exemplo 13}}$ (Continuação)

Passo indutivo: Suponha que $P(x)$ e $P(y)$ são verdadeiros para dois elementos x e y em A. Ou seja, a H.I. é que os x e y de A são ambos divisíveis por 3.

A regra recursiva diz que posso usar x e y para incluir o elemento $x + y$ em A, logo que mostrar que $P(x + y)$ também é verdadeiro.
Para isto, note que se x e y são divisíveis por 3, então existem $k', k'' \in \mathbb{N}$ tais que $x = 3k'$ e $y = 3k''$. Nesse caso, podemos derivar:

$$x + y = 3k' + 3k'' = 3(k' + k''),$$

de onde concluímos que $x + y$ também é divisível por 3. Isto conclui o passo indutivo.

Como concluímos com sucesso o passo base e o passo indutivo, a demonstração por indução estrutural está concluída. $\square$

Exemplos de uso de indução estrutural

- $\boxed{\text{Exemplo 14}}$ O conjunto das sentenças lógicas bem formadas pode ser definido recursivamente como:

  Passo base: T, F e s são sentenças bem formadas, onde s representa uma proposição lógica.

  Passo recursivo: Se G e H são sentenças bem formadas, então $(\neg G)$, $(G \wedge H)$, $(G \vee H)$, $(G \rightarrow H)$ e $(G \leftrightarrow H)$ são sentenças bem formadas.

  Mostre que em toda sentença bem formada o número de "(" e ")" são iguais.

  Demonstração. Seja $P(E)$ a proposição *"A expressão bem formada E tem um igual número de "(" e de ")""*.

  Passo base: Os elementos da base são T, F, e qualquer proposição lógica, e todos estes elementos não possuem nenhum "(" ou ")".

Exemplos de uso de indução estrutural

- $\boxed{\text{Exemplo 14}}$ (Continuação)

Passo indutivo: Temos que verificar que cada uma das regras de criação de novas sentenças mantém a propriedade de que a nova sentença possui parênteses balanceados.

Supondo como H.I. que $P(G)$ e $P(H)$ sejam verdadeiros para duas sentenças bem formadas G e H, vamos analisar cada regra separadamente:

- Regra "$(\neg G)$ é bem formada": pela H.I., G possui igual número de "("s e ")". Como a regra acrescenta exatamente um "(" e um ")", então $P((\neg G))$ é verdadeiro.
- Regra "$(G \wedge H)$ é bem formada": pela H.I., tanto G quanto H possuem igual número de "("s e ")". Como a regra acrescenta exatamente um "(" e um ")", então $P((G \wedge H))$ é verdadeiro.
- Os casos das regras para $(G \vee H)$, $(G \rightarrow H)$ e $(G \leftrightarrow H)$ são semelhantes.

Exemplos de uso de indução estrutural

- $\boxed{\text{Exemplo 14}}$ (Continuação)

  Tendo analisado todas as regras do caso recursivo, concluímos o passo indutivo e, assim, a demonstração. $\square$

Exemplos de uso de indução estrutural

- $\boxed{\text{Exemplo 15}}$ Seja o conjunto S formado por pares ordenados de números naturais definido recursivamente como:

  Passo base: $(0, 0) \in S$.

  Passo recursivo: Se $(x, y) \in S$ então $(x + 2, y + 3) \in S$ e $(x + 3, y + 2) \in S$.

  Mostre que todo elemento de S satisfaz a propriedade de que a soma de suas coordenadas é divisível por 5.

  Demonstração. Seja $P((x, y))$ a proposição *"$x + y$ é divisível por 5"*.

  Passo base: O único elemento da base é $(0, 0)$, e claramente $P(0, 0)$ é verdadeiro já que $0 + 0$ é divisível por 5.

- $\boxed{\text{Exemplo 15}}$ (Continuação)

Passo indutivo: Temos que verificar que cada uma das regra de criação de novos pares ordenados mantém a propriedade de que o novo par ordenado tem a soma de suas coordenadas divisível por 5. A H.I. é de que $P((x, y))$ é verdadeiro para um $(x, y) \in S$.

Vamos analisar cada regra separadamente.

- Regra $(x+2, y+3) \in S$: a soma das coordenadas deste novo par é $x+2+y+3 = (x+y)+5$. Pela H.I. $(x+y)$ é divisível por 5, logo $(x+y)+5$ também é divisível por 5 e podemos concluir que $P((x+2, y+3))$ é verdadeiro.
- Regra $(x+3, y+2) \in S$: a soma das coordenadas deste novo par é $x+3+y+2 = (x+y)+5$. Pela H.I. $(x+y)$ é divisível por 5, logo $(x+y)+5$ também é divisível por 5 e podemos concluir que $P((x+3, y+2))$ é verdadeiro.

Por termos analisado todas as regras do caso recursivo, o passo indutivo da demonstração está concluído. $\square$

Exemplos de uso de indução estrutural

- $\boxed{\text{Exemplo 16}}$ Neste exemplo demonstramos que a definição recursiva da sequência encontrada em um exemplo anterior está correta.

  Seja sequência $\{a_n\}$ definida explicitamente, para $n \geq 1$, como

  $$a_n = n^2 + 3n,$$

  e seja $\{b_n\}$ a sequência definida recursivamente como

  $$\begin{cases} b_1 = 4, \\ b_n = b_{n-1} + 2n + 2, \quad n \geq 2, \end{cases}$$

  Mostre por indução que as sequências $\{a_n\}$ e $\{b_n\}$ são idênticas.

  Demonstração. Seja $P(b_n)$ a proposição *"$b_n = n^2 + 3n$"* (ou seja, a proposição de que o elemento b_n é igual ao elemento a_n).

  Passo base: Temos $b_1 = 1^2 + 3 \cdot 1 = 4$, o que é verdadeiro pelo passo base da definição recursiva.

Exemplos de uso de indução estrutural

- $\boxed{\text{Exemplo 16}}$ (Continuação)

Passo indutivo: Suponhamos a H.I. de que $P(b_{k-1})$ é verdadeiro para um b_{k-1} arbitrário na sequência, ou seja, que $b_{k-1} = (k-1)^2 + 3(k-1)$.

Queremos mostrar que $P(b_k)$ também é verdadeiro, ou seja, que $b_k = a_k$. Para isto, podemos derivar

$$\begin{aligned}
b_k &= b_{k-1} + 2k + 2 && \text{(pela definição de } \{b_n\}) \\
&= (k-1)^2 + 3(k-1) + 2k + 2 && \text{(pela H.I.)} \\
&= k^2 - 2k + 1 + 3k - 3 + 2k + 2 \\
&= k^2 + 3k,
\end{aligned}$$

de onde concluímos o passo indutivo e, assim, a demonstração. □

Exemplos de uso de indução estrutural

- Para o próximo exemplo, vamos introduzir mais alguns conceitos sobre árvores binárias completas.
- A **altura** $h(T)$ de uma árvore binária completa T é definida recursivamente como:

$$h(T) = \begin{cases} 0, & \text{se o único vértice da árvore binária} \\ & \text{completa } T \text{ é a própria raiz} \\ 1 + \max(h(T_1), h(T_2)), & \text{se a árvore binária completa } T \\ & \text{é formada por uma raiz tendo como} \\ & \text{sub-árvores } T_1 \text{ e } T_2. \end{cases}$$

Exemplos de uso de indução estrutural

- O **número de vértices** $n(T)$ de uma árvore binária completa T é definido recursivamente como:

$$n(T) = \begin{cases} 1, & \text{se o único vértice da árvore binária} \\ & \text{completa } T \text{ é a própria raiz} \\ 1 + n(T_1) + n(T_2), & \text{se a árvore binária completa } T \\ & \text{é formada por uma raiz tendo como} \\ & \text{sub-árvores } T_1 \text{ e } T_2. \end{cases}$$

Exemplos de uso de indução estrutural

- $\boxed{\text{Exemplo 17}}$ Mostre em uma árvore binária completa T, temos

$$n(T) \leq 2^{h(T)+1} - 1 \ .$$

Demonstração. Vamos demonstrar esta desigualdade usando indução estrutural.

Passo base: Para uma árvore binária completa T consistindo apenas num vértice raiz, note que, por definição: $n(T) = 1$ e $h(T) = 0$, logo a desigualdade é satisfeita pois

$$n(T) = 1 \ , \qquad \text{e}$$
$$2^{h(T)+1} - 1 = 2^{0+1} - 1 = 1 \ ,$$

e, portanto,

$$n(T) \leq 2^{h(T)+1} - 1 \ .$$

Exemplos de uso de indução estrutural

- $\boxed{\text{Exemplo 17}}$ (Continuação)

Passo indutivo: A nossa hipótese de indução é que temos

$$n(T_1) \leq 2^{h(T_1)+1} - 1 \,, \qquad \text{e}$$

$$n(T_2) \leq 2^{h(T_2)+1} - 1$$

sempre que T_1 e T_2 foremárvores binárias completas.

Suponha que T é uma árvore binária completa tendo T_1 e T_2 como sub-árvores imediatas.

As fórmulas recursivas de $n(T)$ e $h(T)$ determinam que

$$n(T) = 1 + n(T_1) + n(T_2) \,, \qquad \text{e}$$

$$h(T) = 1 + \max(h(T_1), h(T_2)) \,.$$

Exemplos de uso de indução estrutural

- $\boxed{\text{Exemplo 17}}$ (Continuação)

Assim, podemos computar:

$$
\begin{aligned}
n(T) = & \quad \text{(def. recursiva de } n(T)\text{)} \\
1 + n(T_1) + n(T_2) \leq & \quad \text{(hipótese de indução)} \\
1 + (2^{h(T_1)+1} - 1) + (2^{h(T_2)+1} - 1) \leq & \quad (*) \\
2 \cdot \max(2^{h(T_1)+1}, 2^{h(T_2)+1}) - 1 = & \quad (\max(2^x, 2^y) = 2^{\max(x,y)}) \\
2 \cdot 2^{\max(h(T_1), h(T_2))+1} - 1 = & \quad \text{(def. recursiva de } h(T)\text{)} \\
2 \cdot 2^{h(T)} - 1 = & \quad \text{(man. algébrica)} \\
2^{h(T)+1} - 1 &
\end{aligned}
$$

onde o passo (*) vale porque a soma de dois termos é sempre menor ou igual a duas vezes o maior termo.

□