Problemas de Indução Matemática

Guilherme Augusto Gessele
160007607

Universidade de Brasília, UnB

Abstract

1. Problema 1

Prove a equivalência entre os princípios da Indução Forte (PIF) e da Indução Matemática (PIM).

Para provar tal afirmação, devemos separar a prova em duas partes, onde o PIF $\Rightarrow$ PIM e PIM $\Rightarrow$ PIF.

Demonstração. 1. PIM $\Rightarrow$ PIF

Subproof. Suponhamos que o Princípio da Indução Forte seja falso, ou seja, que $\forall n\,(\forall m, m > n \rightarrow P(m) = V) \Rightarrow P(n) = F$. Por hipóteses, porém, que PIM é válido.

Dessa forma, vamos supor que para alguma propriedade P qualquer:

$$\exists A \subseteq \mathbb{N} : \forall a \in A, P(a) = F$$

Diante disso, é notório que:

$$\forall n, n \in \mathbb{N} : n < min(A) : P(n) = V$$

Uma vez que $min(A)$ será o primeiro número na qual a propriedade irá falhar. Ou seja,

$$\forall k : K \leq n, P(k) = V$$

Preprint submitted to Journal Name *16 de novembro de 2018*

Usaremos, nesses n números o PIM. Com ele, verificamos que, como $P(1) = V$ a $P(n) = V$, por consequência do Conjunto A, temos, por hipótese de Indução, que

$$P(n+1) = P(min(A)) = V$$

Que caracteriza uma contradição, logo o Princípio da Indução Forte é verdadeiro. $\triangleleft$

$\square$

(PIF $\Rightarrow$ PIM)

2. Problema 2

Provar a correteza do Insertion-Sort

Demonstração. Para provar a correteza do Algoritmo Insertion-Sort, é necessário fazermos uma indução estrutural sobre a lista em que pretendemos ordernar.

Case 1. $l = [\,]$; $|\, l \,| = 0$

Subproof. Para o caso em que $l = 0$,

$$\text{InsertionSort}\,(l) = \text{InsertionSort}\,([\,]) = [\,]$$

que é o caso trivial, funcionando como Base de Indução de nossa prova. $\triangleleft$

Case 2. $l = x_1 :: [\,]$, $|\, l \,| = 1$

Subproof. Para esse caso,

$$\text{InsertionSort}\,(l) = \text{InsertionSort}\,(x_1 :: [\,]) = \text{Insert}\,(\text{x}_1, \text{InsertionSort}\,([\,])$$

Pelo caso anterior, temos então que

$$\text{Insert}\,(\ \text{x}_1, \text{InsertionSort}\,(\ [\,]\ ) = \text{Insert}\,(\ \text{x}_1, [\,]\ ) = \text{x}_1 :: [\,]$$

Ou seja, para os casos $l = [\,]$ e $l = x_1 :: [\,]$, o resultado do algoritmo será a própria lista propriamente ordenada. $\triangleleft$

Case 3. $x_2 :: x_1 :: [\,];\ \mid l \mid = 2$

Subproof. Para esse caso,

$$\text{InsertionSort}\,(x_2 :: x_1 :: [\,]) = \text{Insert}\,(x_2,\ x_1 :: [\,])$$

Disso, chegamos a 2 possíveis casos

$$x_2 \leq x_1 \quad \text{ou} \quad x_2 > x_1$$

Para o caso em que $x_2 \leq x_1$ então

$$\text{Insert}\,(x_2, x_1 :: [\,]) = x_2 :: x_1 :: [\,]$$

Para o caso $x_2 > x_1$, obtemos que

$$\text{Insert}\,(x_2,\ x_1 :: [\,]) = x_1 :: \text{Insert}\,(x_2, [\,]) = x_1 :: x_2 :: [\,]$$

$\triangleleft$

Seguindo essa linha de pensamento, realizaremos o restante da prova por meio de Indução Simples, assim, assumiremos que o algoritmo funcionará para uma lista l de comprimento $\mid l \mid = n$.

Case 4. $l = x_{n+1} :: \cdots :: x_1 :: [\,], \quad |l| = n + 1$

Para continuar a prova, chamaremos

$$l' = x_n :: \cdots :: x_1 :: [\,]$$

Dessa forma,

$$l = x_{n+1} :: l'$$

Disso, quando aplicarmos o algoritmo InsertionSort, por definição, teremos:

$$\text{InsertionSort}\,(l) = \text{Insert}\,(x_{n+1}, \quad \text{InsertionSort}\,(\mathrm{l}'))$$

Pela hipótese de Indução,

$$\text{InsertionSort}\,(l') = l''$$

onde

$$l^{''} = x_1^{''} :: \cdots :: x_n^{''} :: [\,]$$

Que nos garante que será uma lista ordenada. Dessa forma,

$$\text{Insert}\left(x_{n+1}, \quad l^{''}\right) = \text{Insert}\left(x_{n+1}, x_1^{''} :: (\cdots :: x_n^{''} :: [\,])\right)$$

Dessa forma, caímos em 2 casos.

O caso trivial é aquele que $x_{n+1} \leq x_1^{''}$, pois o resultado do procedimento anterior seria a lista

$$l^{(3)} = x_{n+1} :: l^{''}$$

Que é uma lista ordenada, pois $l^{''}$ é uma lista ordenada e x_{n+1} é menor ou igual ao primeiro elemento dessa lista. Logo, para esse caso, o Algoritmo Insertion Sort funcionará como o esperado.

O outro caso, seria $x_{n+1} > x_1^{''}$. Nele, o algoritmo retornaria

$$\text{Insert}\left(x_{n+1}, l^{''}\right) = x_1^{''} :: \left(\text{Insert}\left(\mathrm{x_{n+1}, l^{(4)}}\right)\right)$$

Onde, segue a relação

$$l^{''} = x_1^{''} :: l^{(4)}$$

Por $|l^{(4)}| = n - 1$, então o método Insert, pela hipótese de Indução, e pela Base de Indução, retornará uma lista ordenada de comprimento $|l^{(5)}| = n$.

Dessa forma

$$\text{Insert}\left(x_{n+1}, l^{''}\right) = x_1^{''} :: l^{(5)}$$

Dessa forma, igualmente o caso anterior, teremos uma lista ordenada.

Considerando o caso anterior, podemos concluir, por Indução Simples, que o algoritmo InsertionSort funcionará para qualquer lista, não importa seu comprimento e distribuição dos elementos. $\square$