

Tópico 3: Indução e corretude de algoritmos

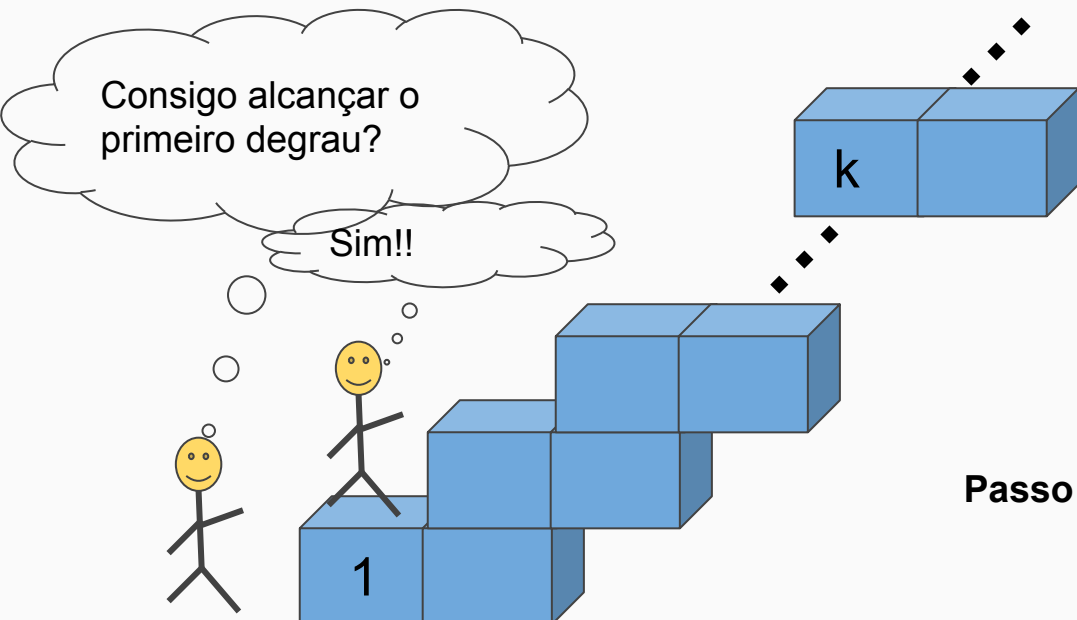
Paradigmas de projeto de algoritmos: conceitos básicos, paradigmas de indução

Sumário:

- Indução
- Corretude de Algoritmos

Indução

Exemplo: Subindo uma escada infinita



Passo básico: Subir primeiro degrau

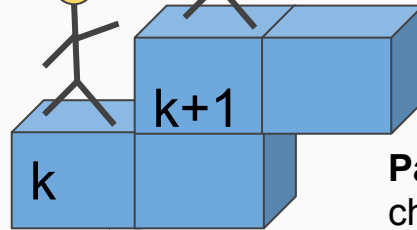


Indução

Se posso chegar a um degrau, consigo chegar ao próximo

Sim!!

Provado por indução !!!



Passo Indutivo: Se consegue chegar a um degrau qualquer, então consegue passar para o **próximo**

Passo básico: Subir primeiro degrau



Indução

Primeiro Princípio da Indução:

Passo básico: $P(1)$ é verdadeiro.

Passo indutivo: $\forall k \in \mathbb{Z}_+, P(k) \rightarrow P(k+1)$

$\left. \begin{array}{l} \text{Passo básico: } P(1) \text{ é verdadeiro.} \\ \text{Passo indutivo: } \forall k \in \mathbb{Z}_+, P(k) \rightarrow P(k+1) \end{array} \right\} \rightarrow P(n) \text{ verdade } \forall n \in \mathbb{Z}_+$

Indução

Exemplo 0: Gerando descendentes

Exemplo 1: Prove que $1+3+5+\dots+(2n-1) = n^2$

Exemplo 2: Prove que $\forall n \in \mathbb{Z}_+, 2^n > n$.

Exemplo 3: Prove que $\forall n \in \mathbb{Z}_+, 2^{2n} - 1$ é divisível por 3.

Exemplo 4: Prove que $\forall n \in \mathbb{Z}_+, n^2 > 3n$ para $n > 3$.

Exemplo 5: Prove que $\forall n \in \mathbb{Z}_+, 2^{n+1} < 3^n$ para $n > 1$.

Indução

Segundo Princípio da Indução (Indução Forte):

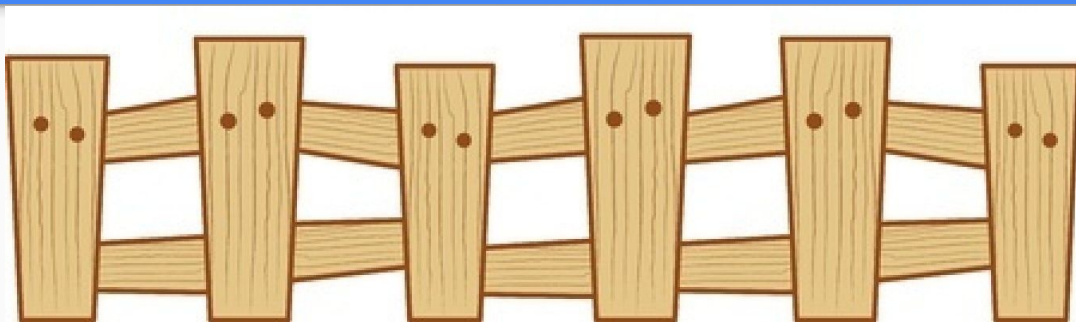
Passo básico: $P(1)$ é verdadeiro.

Passo indutivo: $\forall k \in \mathbb{Z}_+ [(P(r) \text{ com } 0 \leq r \leq k) \rightarrow P(k+1)]$

} $\rightarrow P(n) \text{ verdade } \forall n \in \mathbb{Z}_+$

Indução

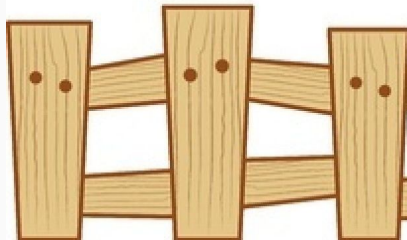
Exemplo 0:



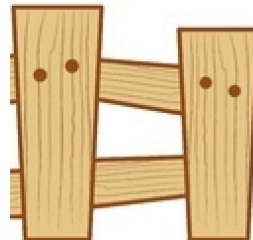
$n=1$ seções = 0



$n=3$ seções = 2



$n=2$ seções = 1



Indução

Exemplo 1: Prove que qualquer quantia em selos maior ou igual a 64 centavos pode ser obtida usando-se apenas selos de 5 e 17 centavos.

Exemplo 2: Prove que $\forall n \geq 3$, a soma dos ângulos internos de um polígono simples fechado com n lados é $(n-2) \cdot 180$.

Corretude de algoritmos

- A **corretude de um algoritmo** indica que ele termina sua execução, retornando saídas corretas para todas as instâncias do problema.
- Vamos provar a **corretude de algoritmos** usando **Indução Matemática**.

Corretude de algoritmos

- **Invariante de laço:**
 - propriedade que é verdadeira cada vez que a condição do laço é avaliada.
 - Propriedade que é verdadeira antes e depois de cada iteração do laço.
- A propriedade de um invariante de laço é satisfeita independente de qual iteração do laço está sendo executada.

Corretude de algoritmos

- **Invariante de laço** pode ser utilizada para determinar a **corretude de algoritmos**.
- Três aspectos precisam ser considerados:
 - **Inicialização**: Um invariante de laço é verdadeiro antes da primeira iteração do laço.
 - **Manutenção**: Se for verdadeiro antes de uma iteração do laço, ele permanece verdadeiro antes da próxima iteração.
 - **Terminação**: A invariante nos dá uma propriedade útil que ajuda a mostrar que o algoritmo está correto quando o laço termina.

Corretude de algoritmos

- Estratégias para verificar tais aspectos:
 - **Estratégia ruim**: Verificar o comportamento após cada iteração.
 - **Estratégia boa**: Indução matemática.
 - **Passo base**: Provar que a hipótese de indução ocorre para os valores de entrada do laço.
 - **Passo Indutivo**: Provar que se a hipótese de indução ocorre após k iterações, ela também é verificada após $k+1$ iterações.
 - Utilize a hipótese de indução para comprovar que o algoritmo está correto ao final do laço.
- A hipótese de indução na **estratégia boa** é a **invariante de laço!!!**

Corretude de algoritmos

Exemplo 1:

Algorithm Muito-Simples

```
a ← c;  
b ← 0;  
while (a > 0)  
    do  a ← a - 1;  
        b ← b + 1;  
return b;
```

- Qual a entrada?
- Qual a saída?
- Qual a propriedade do laço?

Corretude de algoritmos

Exemplo 1:

Algorithm Muito-Simples

```
a ← c;  
b ← 0;  
while (a > 0)  
    do  a ← a - 1;  
        b ← b + 1;  
return b;
```

Verificando a corretude do algoritmo:

Invariante de laço: $a + b = c$

Passo base: Antes do início do laço, temos

```
a ← c;  
b ← 0;
```

Logo, $a + b = c + 0 = c \Rightarrow a + b = c$ Ok!!

Corretude de algoritmos

Exemplo 1:

Algorithm Muito-Simples

```
a ← c;  
b ← 0;  
while (a > 0)  
    do a ← a - 1;  
       b ← b + 1;  
return b;
```

Verificando a corretude do algoritmo:

Invariante de laço: $a + b = c$

Passo Indutivo: Se $a + b = c$ ocorre após k iterações, então $a + b = c$ após $k + 1$ iterações.

Prova:

Após a iteração k , por hipótese de indução, temos $a = c - k$ e $b = k$ com $a + b = c$;

Corretude de algoritmos

Exemplo 1:

Algorithm Muito-Simples

```
a ← c;  
b ← 0;  
while (a > 0)  
    do  a ← a - 1;  
        b ← b + 1;  
return b;
```

Verificando a corretude do algoritmo:

Passo Indutivo: Se $a+b = c$ ocorre após k iterações, então $a+b=c$ após $k+1$ iterações.

Prova:

Na iteração $k+1$, usando a H.I, temos:

$$a \leftarrow a - 1 \Leftrightarrow a = c - k - 1$$

$$b \leftarrow b - 1 \Leftrightarrow b = k + 1$$

$$\text{Logo, } a + b = (c - k - 1) + (k + 1) = c$$

$$\Rightarrow a + b = c \text{ Ok!!}$$

Corretude de algoritmos

Exemplo 1:

Algorithm Muito-Simples

```
a ← c;  
b ← 0;  
while (a > 0)  
    do a ← a - 1;  
       b ← b + 1;  
return b;
```

Verificando a corretude do algoritmo:

Terminação: O algoritmo após o laço deve estar correto.

Prova: A condição do laço é violada quando $a \leftarrow 0$.

Pela propriedade da invariante de laço, temos $b=c$.

Logo,

return b //Retorna valor de c !!

Corretude de algoritmos

- A dificuldade está em se determinar a propriedade de um invariante de laço que permita provar a corretude do algoritmo.

Exemplo 2:

Algorithm Ainda-Muito-Simples

```
a ← 0;  
for i ← 1 to n do;  
    a ← a + 2i;  
    i ← i + 1  
return a;
```

Corretude de algoritmos

Exemplo 2:

Algorithm

Ainda-Muito-Simples

$a \leftarrow 0;$

for $i \leftarrow 0$ to n do;

$a \leftarrow a + 2^i;$

$i \leftarrow i + 1;$

return $a;$

Qual é a invariante de laço ?

Início iteração#0 : $a \leftarrow 0$ e $i \leftarrow 0$

Início iteração#1 : $a \leftarrow 1$ ($0+1$) e $i \leftarrow 1$

Início iteração#2 : $a \leftarrow 3$ ($0+1+2$) e $i \leftarrow 2$

Início iteração#3 : $a \leftarrow 9$ ($0+1+2+4$) e $i \leftarrow 3$

Início iteração#4 : $a \leftarrow 15$ ($0+1+2+4+8$) e $i \leftarrow 4$

Início iteração#5 : $a \leftarrow 31$ ($0+1+2+4+8+16$) e $i \leftarrow 5$

....

Início iteração#k : $a \leftarrow 2^k - 1$ e $i \leftarrow k$ **Invariante de laço!!**

Corretude de algoritmos

Exemplo 2:

Algorithm

Ainda-Muito-Simples

```
a ← 0;
for i ← 0 to n do;
    a ← a + 2i;
    i ← i + 1;
return a;
```

Provando por indução:

Passo base:

Antes do início do laço, temos $a \leftarrow 0$ e $i \leftarrow 0$.

Logo, $a \leftarrow 2^k - 1$ e $i \leftarrow k \iff a \leftarrow 2^0 - 1$ e $i \leftarrow 0$

$\iff a \leftarrow 0$ e $i \leftarrow 0$.

Corretude de algoritmos

Exemplo 2:

Algorithm

Ainda-Muito-Simples

```
a ← 0;
for i ← 0 to n do;
    a ← a + 2i;
    i ← i + 1;
return a;
```

Passo Indutivo: Se o laço invariante é válido no início de k iterações, então será válido no início da iteração $k+1$.

Provando:

No início do laço na iteração k , temos:

$a \leftarrow 2^k - 1$ e $i \leftarrow k$.

Ao executar a iteração k , fazemos:

$a \leftarrow a + 2^i \Leftrightarrow a \leftarrow 2^k - 1 + 2^k \Leftrightarrow a \leftarrow 2 \cdot 2^k - 1 \Leftrightarrow a \leftarrow 2^{k+1} - 1$

$i \leftarrow i + 1 \Leftrightarrow i \leftarrow k + 1$

Assim no início da iteração $k+1$ temos:

$a \leftarrow 2^{k+1} - 1$ e $i \leftarrow k + 1$ ok!!

Corretude de algoritmos

Exemplo 2:

Algorithm
Ainda-Muito-Simples

```
a ← 0;  
for i ← 0 to n do;  
    a ← a + 2i;  
    i ← i + 1;  
return a;
```

Terminação:

Quando a condição do laço é violada, temos no início do laço: $a \leftarrow 2^{n+1} - 1$ e $i \leftarrow n + 1$.

Logo, o valor retornado será $a \leftarrow 2^{n+1} - 1$

Corretude de algoritmos

- Resumindo:
 - Passo 1: Identificar invariante de laço.
 - Passo 2: Provar por indução a invariante de laço.
 - Passo 3: Verificar o término do algoritmo (última iteração).

Corretude de algoritmos

Exemplo 3:

Passo1: Identificar a invariante de laço

Algorithm convert-Bin

```
q ← n;  
i ← 0;  
while q > 0  
do B[i] ← q mod 2;  
   q ← q div 2;  
   i ← i + 1;  
return B;
```

iteração#0: $B=\{\}$, $q \leftarrow 10$, $i \leftarrow 0$ ($n=10$)

iteração#1: $B=\{0\}$, $q \leftarrow 5$, $i \leftarrow 1$ ($n=5 \cdot 2^1 + 0 \cdot 2^0$)

iteração#2: $B=\{0,1\}$, $q \leftarrow 2$, $i \leftarrow 2$ ($n = 2 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0$)

iteração#3: $B=\{0,1,0\}$, $q \leftarrow 1$, $i \leftarrow 3$ ($n = 1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0$)

Iteração#4: $B=\{0,1,0,1\}$, $q \leftarrow 0$, $i \leftarrow 4$ ($n = 0 \cdot 2^4 + 1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0$)

....

iteração#k: $n = q \cdot 2^k + B[k-1] \cdot 2^{k-1} + B[k-2] \cdot 2^{k-2} + \dots + B[1] \cdot 2^1 + B[0] \cdot 2^0$

Invariante de laço!!

Corretude de algoritmos

Exemplo 3:

Algorithm convert-Bin

$q \leftarrow n$;

$i \leftarrow 0$;

while $q > 0$

do $B[i] \leftarrow q \bmod 2$;

$q \leftarrow q \text{ div } 2$;

$i \leftarrow i + 1$;

return B ;

Passo2: Provar por indução a invariante de laço

$$n = q \cdot 2^k + B[k-1] \cdot 2^{k-1} + B[k-2] \cdot 2^{k-2} + \dots + B[1] \cdot 2^1 + B[0] \cdot 2^0$$

Passo básico: Para $q \leftarrow n$ e $i \leftarrow 0$, teremos:

$$n = q \cdot 2^0 + B[-1] \cdot 2^{-1} = q, \text{ pois } B = \{\}.$$

Corretude de algoritmos

Exemplo 3:

```
Algorithm convert-Bin
q ← n;
i ← 0;
while q > 0
do B[i] ← q mod 2;
   q ← q div 2;
   i ← i + 1;
return B;
```

Passo2: Provar por indução a invariante de laço

$$n = q \cdot 2^k + B[k-1] \cdot 2^{k-1} + B[k-2] \cdot 2^{k-2} + \dots + B[1] \cdot 2^1 + B[0] \cdot 2^0$$

Passo de indução: Se a invariante de laço é válida para k iterações, então será válida para $k+1$ iterações.

No início da iteração k , temos:

$$n = q \cdot 2^k + B[k-1] \cdot 2^{k-1} + B[k-2] \cdot 2^{k-2} + \dots + B[1] \cdot 2^1 + B[0] \cdot 2^0$$

Corretude de algoritmos

Exemplo 3:

Algorithm convert-Bin

$q \leftarrow n$;

$i \leftarrow 0$;

while $q > 0$

do $B[i] \leftarrow q \bmod 2$;

$q \leftarrow q \div 2$;

$i \leftarrow i + 1$;

return B;

Passo2: Provar por indução a invariante de laço

Se, por H.I, no início da iteração k , temos:

$$n = q.2^k + B[k-1].2^{k-1} + B[k-2].2^{k-2} + \dots + B[1].2^1 + B[0].2^0$$

Na execução da iteração $i \leftarrow k$, ficamos com:

$$B[k] \leftarrow q \bmod 2,$$

$$q \leftarrow q \div 2 \text{ e } i \leftarrow i + 1 \text{ (} i \leftarrow k \text{)}.$$

Logo,

$$N = (q.2 + B[k]).2^k + B[k-1].2^{k-1} + B[k-2].2^{k-2} + \dots + B[1].2^1 + B[0].2^0$$

Corretude de algoritmos

Exemplo 3:

Passo2: Provar por indução a invariante de laço

Algorithm convert-Bin

```
q ← n;  
i ← 0;  
while q > 0  
do B[i] ← q mod 2;  
   q ← q div 2;  
   i ← i + 1;  
return B;
```

Logo,

$$N = (q \cdot 2 + B[k]) \cdot 2^k + B[k-1] \cdot 2^{k-1} + B[k-2] \cdot 2^{k-2} + \dots + B[1] \cdot 2^1 + B[0] \cdot 2^0$$

$$N = q \cdot 2 \cdot 2^k + B[k] \cdot 2^k + B[k-1] \cdot 2^{k-1} + B[k-2] \cdot 2^{k-2} + \dots + B[1] \cdot 2^1 + B[0] \cdot 2^0$$

$$N = q \cdot 2^{k+1} + B[k] \cdot 2^k + B[k-1] \cdot 2^{k-1} + B[k-2] \cdot 2^{k-2} + \dots + B[1] \cdot 2^1 + B[0] \cdot 2^0$$

Corretude de algoritmos

Exemplo 3:

```
Algorithm convert-Bin
q ← n;
i ← 0;
while q > 0
do B[i] ← q mod 2;
   q ← q div 2;
   i ← i + 1;
return B;
```

Passo3: Verificar o término do algoritmo (última iteração).

No início da última iteração temos $q \leftarrow 0$ e $i \leftarrow i + 1$;

Logo,

return Bin

\Leftrightarrow return $\{B[0], \dots, B[i]\}$

$\Leftrightarrow N = B[i].2^i + B[i-1].2^{i-1} + B[i-2].2^{i-2} + \dots + B[1].2^1 + B[0].2^0$