

Corretude de Algoritmos Recursivos

Prof. André Vignatti

Resumo

- Confiar em algoritmos ao testar e provar a corretude.
- Corretude de algoritmos recursivos são provados diretamente por indução.
- Corretude de algoritmos iterativos são provados usando invariantes de laço e indução.
- Exemplos: números de Fibonacci, máximo, multiplicação.

Corretude

Como saber que um algoritmo funciona?

- Meios de persuasão e retórica (da Grécia antiga, das ciências não exatas)
- Método científico:
 - Empírico - Testes
 - Teórico - Prova de corretude

Testes vs. Prova de Corretude

Teste: testar o algoritmo com amostras de instâncias (todos fazem isso antes de CI165)

Prova de Corretude: provar matematicamente

Testes talvez não encontrem bugs obscuros. Usar somente testes pode ser perigoso.

Provas de corretude também podem conter bugs. O melhor é usar uma combinação de testes e prova de corretude.

Corretude de Algoritmos Recursivos

Para provar a corretude de um algoritmo recursivo:

- Provar por indução no tamanho da entrada
- Base da recursão é a base da indução
- Mostrar que chamadas recursivas sempre são para instâncias menores (geralmente trivial)
- Passo indutivo: assumir que as chamadas recursivas funcionam corretamente, e usar essa suposição para provar que a chamada atual funciona corretamente.

Número de Fibonacci Recursivo

O n -ésimo número de Fibonacci F_n é definido como:

$$F_n = \begin{cases} n & \text{se } n \leq 1 \\ F_{n-1} + F_{n-2} & \text{se } n > 1 \end{cases}$$

Algoritmo $fib(n)$

┌ se $n \leq 1$ então retorna n
└ senão retorna $fib(n-1) + fib(n-2)$

Teorema. Para todo $n \geq 0$, $fib(n)$ devolve F_n .

Demonstração.

Base: para $n = 0$, $fib(n)$ devolve 0 como afirmado. Para $n = 1$, $fib(n)$ devolve 1 como afirmado.

Hipótese: Para $n \geq 2$ e para todo $0 \leq m < n$, $fib(m)$ devolve F_m .

Passo: Queremos provar que $fib(n)$ devolve F_n .

O que $fib(n)$ devolve?

$$fib(n-1) + fib(n-2) \stackrel{(\text{hip. de indução})}{=} F_{n-1} + F_{n-2} \stackrel{(\text{definição})}{=} F_n.$$

□

Máximo Recursivo

Algoritmo *maximo*($A[1..n]$)
┌ se $n \leq 1$ então retorna $A[1]$
└ senão retorna $\max(\text{maximo}(A[1..n-1]), A[n])$

Teorema. Para todo $n \geq 1$, $\text{maximo}(A[1..n])$ devolve $\max\{A[1], A[2], \dots, A[n]\}$.

Demonstração.

Base: para $n = 1$, $\text{maximo}(A[1..n])$ devolve $A[1]$, como afirmado.

Hipótese: Para $n \geq 1$, $\text{maximo}(A[1..n])$ devolve $\max\{A[1], A[2], \dots, A[n]\}$

Passo:

Vamos mostrar que $\text{maximo}(A[1..n+1])$ devolve $\max\{A[1], A[2], \dots, A[n+1]\}$.

O que $\text{maximo}(A[1..n+1])$ devolve?

$$\begin{aligned} & \max(\text{maximo}(A[1..n]), A[n+1]) \\ \stackrel{\text{(hip. de indução)}}{=} & \max(\max\{A[1], A[2], \dots, A[n]\}, A[n+1]) \\ = & \max\{A[1], A[2], \dots, A[n+1]\}. \end{aligned}$$

□

Multiplicação Recursiva

Algoritmo *multiplica*(y, z)
┌ se $z = 0$ então retorna 0
└ senão se z é ímpar então
 ┌ retorna $\text{multiplica}(2y, \lfloor z/2 \rfloor) + y$
 └ senão retorna $\text{multiplica}(2y, \lfloor z/2 \rfloor)$

(Em aula) Faça um exemplo de execução para $y = 3, z = 9$.

Teorema. Para todo $y, z \geq 0$, $\text{multiplica}(y, z)$ devolve yz .

Demonstração. (Indução em z)

Base: para $z = 0$, $\text{multiplica}(y, z)$ devolve 0, como afirmado.

Hipótese: Para $0 \leq q \leq z$, $\text{multiplica}(y, q)$ devolve yq .

Passo: Queremos mostrar que $\text{multiplica}(y, z+1)$ devolve $y(z+1)$.

O que $\text{multiplica}(y, z+1)$ devolve?

Há dois casos, dependendo se $z + 1$ é par ou ímpar.

Se $z + 1$ é ímpar, então $\text{multiplica}(y, z + 1)$ devolve

$$\begin{aligned} \text{multiplica}(2y, \lfloor (z + 1)/2 \rfloor) + y &\stackrel{\text{(hip. de indução)}}{=} 2y \lfloor (z + 1)/2 \rfloor + y \\ &\stackrel{(z \text{ é par})}{=} 2yz/2 + y \\ &= y(z + 1). \end{aligned}$$

Se $z + 1$ é par, então $\text{multiplica}(y, z + 1)$ devolve

$$\begin{aligned} \text{multiplica}(2y, \lfloor (z + 1)/2 \rfloor) &\stackrel{\text{(hip. de indução)}}{=} 2y \lfloor (z + 1)/2 \rfloor \\ &\stackrel{(z \text{ é ímpar})}{=} 2y(z + 1)/2 \\ &= y(z + 1). \end{aligned}$$

□