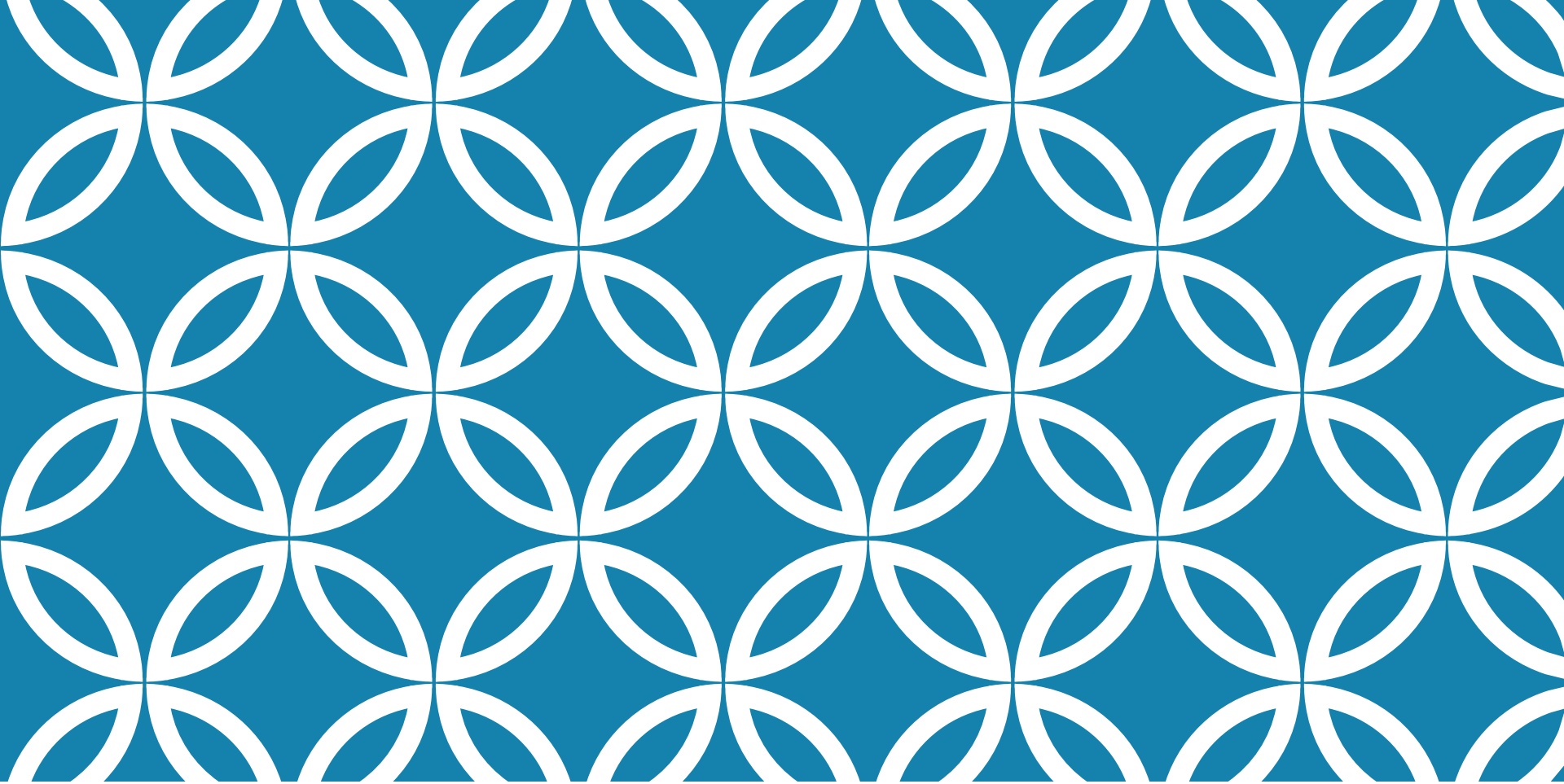

CORRETUDE DE ALGORITMOS RECURSIVOS

Prof. André Vignatti

VISÃO GERAL

- 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?

CORRETUDE

Como saber que um algoritmo funciona?

- Meios de persuasão e retórica (da Grécia antiga, das ciências não exatas)

CORRETUDE

Como saber que um algoritmo funciona?

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

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

TESTE X PROVA DE CORRETUDE

Teste: testar o algoritmo com amostras de instâncias

TESTE X PROVA DE CORRETUDE

Teste: testar o algoritmo com amostras de instâncias

Prova de Corretude: provar matematicamente

TESTE X PROVA DE CORRETUDE

Teste: testar o algoritmo com amostras de instâncias

Prova de Corretude: provar matematicamente

Testes talvez não encontrem bugs obscuros

TESTE X PROVA DE CORRETUDE

Teste: testar o algoritmo com amostras de instâncias

Prova de Corretude: provar matematicamente

Testes talvez não encontrem bugs obscuros

Usar somente testes pode ser perigoso

TESTE X PROVA DE CORRETUDE

Teste: testar o algoritmo com amostras de instâncias

Prova de Corretude: provar matematicamente

Testes talvez não encontrem bugs obscuros

Usar somente testes pode ser perigoso

Provas de corretude também podem conter bugs

TESTE X PROVA DE CORRETUDE

Teste: testar o algoritmo com amostras de instâncias

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:

CORRETUDE DE ALGORITMOS RECURSIVOS

Para provar a corretude de um algoritmo recursivo:

- Provar por indução no tamanho da entrada

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

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)

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.

FIBONACCI

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

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

FIBONACCI

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 $\text{fib}(n-1) + \text{fib}(n-2)$

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

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.

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

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

Demonstração.

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

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

Passo: Queremos provar que $\text{fib}(n)$ devolve F_n.

O que $\text{fib}(n)$ devolve?

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

□

MÁXIMO DE VETOR

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

MÁXIMO DE VETOR

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 $\max\{A[1], A[2], \ldots, A[n]\}$

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

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

Demonstração.

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

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

Demonstração.

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

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

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

Demonstração.

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

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

Passo:

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

O que maximo$(A[1..n+1])$ devolve?

□

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

Demonstração.

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

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

Passo:

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

O que maximo$(A[1..n+1])$ devolve?

$$\max(\text{maximo}(A[1..n]), A[n+1])$$

□

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

Demonstração.

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

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

Passo:

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

O que maximo$(A[1..n+1])$ devolve?

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

□

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

Demonstração.

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

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

Passo:

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

O que maximo$(A[1..n+1])$ devolve?

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

□

MULTIPLICAÇÃO

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

Faça um exemplo de execução para $y = 3, z = 9$.

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

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

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

Demonstração. (Indução em z)

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

Demonstração. (Indução em z)

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

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

Demonstração. (Indução em z)

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

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

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

Demonstração. (Indução em z)

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

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

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

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

Demonstração. (Indução em z)

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

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

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

O que multiplica$(y, z + 1)$ devolve?

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

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

Se $z+1$ é ímpar, então multiplica$(y, z+1)$ devolve

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

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

Se $z+1$ é ímpar, então multiplica$(y, z+1)$ devolve

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

Se $z+1$ é par, então multiplica$(y, z+1)$ devolve

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

□