UNIVERSIDADE
FEDERAL RURAL
DE PERNAMBUCO

Prova de Corretude de Algoritmos

Disciplina: Projeto e Análise de Algoritmos

Alunos: José Bartolomeu Alheiros Dias Neto
Lucas Wagner

Tipos de Algoritmos

Tipos de Algoritmos

Um algoritmo pode ser:

Tipos de Algoritmos

Um algoritmo pode ser:

- Iterativo(com laços for, while, etc);

Tipos de Algoritmos

Um algoritmo pode ser:

- Iterativo(com laços for, while, etc);
- Recursivo(funções que chamam a si próprias).

Fatorial

Fatorial

Recursivo ou Iterativo???

Fatorial

Recursivo ou Iterativo???

- Recursivo;

Fatorial

Recursivo ou Iterativo???

- Recursivo;
- Iterativo.

Fatorial

Recursivo ou Iterativo???

- Recursivo;
- Iterativo.

Prova de Corretude

Se o algoritmo for iterativo, achamos as Invariantes para cada laço e provamos por indução, utilizando as invariantes.

Prova de Corretude

Se o algoritmo for iterativo, achamos as Invariantes para cada laço e provamos por indução, utilizando as invariantes.

Se for recursivo, utilizamos apenas a <u>indução</u>.

Fatorial Recursivo

Fatorial Recursivo

```
Fat(n)
if n == 0
    Fat(n) = 1
else
    Fat(n) = n · Fat(n − 1)
```

Fatorial Recursivo

O que queremos provar:

Fat(n)

if n == 0

    Fat(n) = 1

else

    Fat(n) = n · Fat(n − 1)

Fatorial Recursivo

O que queremos provar:

<u>Proposição de Corretude:</u>

Fat(n)

if n == 0

  Fat(n) = 1

else

  Fat(n) = n · Fat(n − 1)

Fatorial Recursivo

```
Fat(n)
if n == 0
    Fat(n) = 1
else
    Fat(n) = n · Fat(n − 1)
```

O que queremos provar:

<u>Proposição de Corretude:</u>

Fat(n) = n!, n>= 0.

Fatorial Recursivo

Fat(n)

if n == 0

    Fat(n) = 1

else

    Fat(n) = n · Fat(n − 1)

O que queremos provar:

Proposição de Corretude:

Fat(n) = n!, n>= 0.

Caso Base:

Fatorial Recursivo

Fat(n)

if n == 0

    Fat(n) = 1

else

    Fat(n) = n · Fat(n − 1)

O que queremos provar:

<u>Proposição de Corretude:</u>

Fat(n) = n!, n>= 0.

Caso Base:

Fatorial Recursivo

```
Fat(n)
if n == 0
    Fat(n) = 1
else
    Fat(n) = n · Fat(n − 1)
```

O que queremos provar:

<u>Proposição de Corretude:</u>

Fat(n) = n!, n>= 0.

Caso Base:

Para n = 0, nosso algoritmo retorna 1;

Fatorial Recursivo

```
Fat(n)

if n == 0

    Fat(n) = 1

else

    Fat(n) = n · Fat(n − 1)
```

O que queremos provar:

<u>Proposição de Corretude:</u>

Fat(n) = n!, n>= 0.

Caso Base:

Para n = 0, nosso algoritmo retorna 1;

Ou seja: Fat(0) = 1
Mas sabemos, pela definição de fatorial, que 0! = 1;

Fatorial Recursivo

```
Fat(0)
if n == 0
    Fat(n) = 1
else
    Fat(n) = n · Fat(n − 1)
```

O que queremos provar:

<u>Proposição de Corretude:</u>

Fat(n) = n!, n>= 0.

Caso Base:

Para n = 0, nosso algoritmo retorna 1;

Ou seja: Fat(0) = 1

Mas sabemos que, pela definição de fatorial, 0! = 1;

Então Fat(0) = 1 = 0!
E para o caso base nossa proposição está correta.

Fatorial Recursivo

Fat(0)

if n == 0

    Fat(n) = 1

else

    Fat(n) = n · Fat(n − 1)

O que queremos provar:

<u>Proposição de Corretude:</u>

Fat(n) = n!, n>= 0.

Caso Base: ✓

Para n = 0, nosso algoritmo retorna 1;

Ou seja: Fat(0) = 1

Mas sabemos que, pela definição de fatorial, 0! = 1;

Então Fat(0) = 1 = 0!
E para o caso base nossa proposição está correta.

Fatorial Recursivo

```
Fat(n)

if n == 0

    Fat(n) = 1

else

    Fat(n) = n · Fat(n − 1)
```

O que queremos provar:

Proposição de Corretude:

Fat(n) = n!, n>= 0.

Passo Indutivo:

Fatorial Recursivo

```
Fat(n)

if n == 0

    Fat(n) = 1

else

    Fat(n) = n · Fat(n − 1)
```

O que queremos provar:

Proposição de Corretude:

Fat(n) = n!, n>= 0.

Passo Indutivo:

H.I.(Hipótese de Indução): Supomos que para um valor k, k > 0, nossa proposição está correta:

Fatorial Recursivo

```
Fat(n)

if n == 0

    Fat(n) = 1

else

    Fat(n) = n · Fat(n − 1)
```

O que queremos provar:

<u>Proposição de Corretude:</u>

Fat(n) = n!, n>= 0.

Passo Indutivo:

H.I.(Hipótese de Indução): Supomos que para um valor k, k > 0, nossa proposição está correta:

*Fat(k) = k * Fat(k -1) = k!*

Fatorial Recursivo

```
Fat(n)
if n == 0
    Fat(n) = 1
else
    Fat(n) = n · Fat(n − 1)
```

Passo Indutivo(cont…):

Tese: Provamos que para um valor $n = k + 1$ nossa proposição está correta, utilizando a H.I.:

Fatorial Recursivo

Fat(n)

if n == 0

    Fat(n) = 1

else

    Fat(n) = n · Fat(n − 1)

Passo Indutivo(cont…):

Tese: Provamos que para um valor n = k + 1 nossa proposição está correta, utilizando a H.I.:

*Fat(k+1) = k + 1 * Fat(k+ 1 - 1)*

Fatorial Recursivo

Fat(n)

if n == 0

Fat(n) = 1

else

Fat(n) = n · Fat(n − 1)

Passo Indutivo(cont…):

Tese: Provamos que para um valor n = k + 1 nossa proposição está correta, utilizando a H.I.:

*Fat(k+1) = k + 1 * Fat(k+ 1 - 1)*
*(1) Fat(k+1) = k + 1 * Fat(k)*

Fatorial Recursivo

Fat(n)

if n == 0

Fat(n) = 1

else

Fat(n) = n · Fat(n − 1)

Passo Indutivo(cont…):

Tese: Provamos que para um valor n = k + 1 nossa proposição está correta, utilizando a H.I.:

*Fat(k+1) = k + 1 * Fat(k+ 1 - 1)*
*(1) Fat(k+1) = k + 1 * Fat(k)*

Mas, pela H.I., sabemos que:

Fatorial Recursivo

Fat(n)

if n == 0

Fat(n) = 1

else

Fat(n) = n · Fat(n − 1)

Passo Indutivo(cont...):

Tese: Provamos que para um valor n = k + 1 nossa proposição está correta, utilizando a H.I.:

*Fat(k+1) = k + 1 * Fat(k+ 1 - 1)*
*(1) Fat(k+1) = k + 1 * $\boxed{Fat(k)}$ → (2)*

Mas, pela H.I., sabemos que:

*(2) Fat(k) = k * Fat(k -1)*

Fatorial Recursivo

Fat(n)

if n == 0

    Fat(n) = 1

else

    Fat(n) = n · Fat(n − 1)

Passo Indutivo(cont…):

Tese: Provamos que para um valor n = k + 1 nossa proposição está correta, utilizando a H.I.:

*Fat(k+1) = k + 1 * Fat(k+ 1 - 1)*
*(1) Fat(k+1) = k + 1 * Fat(k)* → *(2)*

Mas, pela H.I., sabemos que:

(2) Fat(k) = k!

*Fat(k+1) = (k + 1) * (k!)*

Fatorial Recursivo

```
Fat(n)

if n == 0

    Fat(n) = 1

else

    Fat(n) = n · Fat(n − 1)
```

Passo Indutivo(cont…):

Tese: Provamos que para um valor n = k + 1 nossa proposição está correta, utilizando a H.I.:

*Fat(k+1) = k + 1 * Fat(k+ 1 - 1)*
*(1) Fat(k+1) = k + 1 * $\boxed{Fat(k)}$ → (2)*

Mas, pela H.I., sabemos que:

(2) Fat(k) = k!

*Fat(k+1) = (k + 1) * $\boxed{(k!)}$*

Fat(k+1) = (k + 1)!

Fatorial Recursivo

```
Fat(n)

if n == 0

    Fat(n) = 1

else

    Fat(n) = n · Fat(n − 1)
```

Passo Indutivo(cont…):

Tese: Provamos que para um valor n = k + 1 nossa proposição está correta, utilizando a H.I.:

*Fat(k+1) = k + 1 * Fat(k+ 1 - 1)*
*(1) Fat(k+1) = k + 1 * $\boxed{Fat(k)}$ → (2)*

Mas, pela H.I., sabemos que:

(2) Fat(k) = k!

*Fat(k+1) = (k + 1) * $\boxed{(k!)}$*

Fat(k+1) = (k + 1)! ✓

C.Q.D

Fatorial Recursivo

```
Fat(n)

if n == 0

    Fat(n) = 1

else

    Fat(n) = n · Fat(n − 1)
```

Conclusão:

Nosso algoritmo de Fatorial Recursivo é correto, pois conseguimos provar, usando indução matemática, que para qualquer inteiro k >= 0, o algoritmo é correspondente à função matemática de Fatorial:

Fatorial Recursivo

Fat(n)

if n == 0

    Fat(n) = 1

else

    Fat(n) = n · Fat(n − 1)

Conclusão:

Nosso algoritmo de Fatorial Recursivo é correto, pois conseguimos provar, usando indução matemática, que para qualquer inteiro k >= 0, o algoritmo é correspondente à função matemática de Fatorial:

n! = n*(n-1)*(n-2)...0!, para qualquer inteiro n>=0.

Fatorial Recursivo

Fat(n)

if n == 0

    Fat(n) = 1

else

    Fat(n) = n · Fat(n − 1)

Conclusão:

Nosso algoritmo de Fatorial Recursivo é correto, pois conseguimos provar, usando indução matemática, que para qualquer inteiro k >= 0, o algoritmo é correspondente à função matemática de Fatorial:

n! = n*(n-1)*(n-2)...0!, para qualquer inteiro n>=0.

Exercício: máximo de um vetor com n elementos

Maximo(A,n)

if n≤1 **then**

    return A[1]

else

    return max(Maximo(A,n – 1), A[n])

Exercício: máximo de um vetor com n elementos

```
Maximo(A,n)
if n≤1 then
    return A[1]
else
    return max(Maximo(A,n – 1), A[n])
```

O que queremos provar:

<u>Proposição de Corretude:</u>

Exercício: máximo de um vetor com n elementos

```
Maximo(A,n)

if n≤1 then

    return A[1]

else

    return max(Maximo(A,n – 1), A[n])
```

O que queremos provar:

Proposição de Corretude:

Maximo(A,n) = max{A[1],...,A[n]}, para todo n>=1;

Exercício: máximo de um vetor com n elementos

```
Maximo(A,n)

if n≤1 then

    return A[1]

else

    return max(Maximo(A,n – 1), A[n])
```

O que queremos provar:

Proposição de Corretude:

Maximo(A,n) = max{A[1],...,A[n]}, para todo n>=1;

Caso Base:

Exercício: máximo de um vetor com n elementos

```
Maximo(A,n)
if n≤1 then
    return A[1]
else
    return max(Maximo(A,n – 1), A[n])
```

O que queremos provar:

Proposição de Corretude:

Maximo(A,n) = max{A[1],...,A[n]}, para todo n>=1;

Caso Base:

Fazendo k = 1:

Exercício: máximo de um vetor com n elementos

```
Maximo(A,n)
if n≤1 then
    return A[1]
else
    return max(Maximo(A,n – 1), A[n])
```

O que queremos provar:

Proposição de Corretude:

Maximo(A,n) = max{A[1],...,A[n]}, para todo n>=1;

Caso Base:

Fazendo k = 1:
Maximo(A, 1) = max{A[1]}.

Exercício: máximo de um vetor com n elementos

```
Maximo(A,n)

if n≤1 then

    return A[1]

else

    return max(Maximo(A,n – 1), A[n])
```

O que queremos provar:

Proposição de Corretude:

Maximo(A,n) = max{A[1],...,A[n]}, para todo n>=1;

Caso Base: ✓

Fazendo k = 1:
Maximo(A, 1) = A[1].

Exercício: máximo de um vetor com n elementos

```
Maximo(A,n)
if n≤1 then
    return A[1]
else
    return max(Maximo(A,n – 1), A[n])
```

O que queremos provar:

<u>Proposição de Corretude:</u>

Maximo(A,n) = max{A[1],...,A[n]}, para todo n>=1;

Passo Indutivo:

H.I.(Hipótese de Indução):
Supomos que para um valor k, k > 1:

Exercício: máximo de um vetor com n elementos

```
Maximo(A,n)

if n≤1 then

    return A[1]

else

    return max(Maximo(A,n – 1), A[n])
```

O que queremos provar:

Proposição de Corretude:

Maximo(A,n) = max{A[1],...,A[n]}, para todo n>=1;

Passo Indutivo:

H.I.(Hipótese de Indução):
Supomos que para um valor k, k > 1:

(1) Maximo(A, k) = max{A[1],...,A[k]};

Exercício: máximo de um vetor com n elementos

```
Maximo(A,n)
if n≤1 then
    return A[1]
else
    return max(Maximo(A,n – 1), A[n])
```

Passo Indutivo(cont…):

Exercício: máximo de um vetor com n elementos

```
Maximo(A,n)
if n≤1 then
    return A[1]
else
    return max(Maximo(A,n – 1), A[n])
```

Passo Indutivo(cont…):

Tese:

Exercício: máximo de um vetor com n elementos

Maximo(A,n)

if n≤1 **then**

    return A[1]

else

    return max(Maximo(A,n – 1), A[n])

Passo Indutivo(cont…):

Tese:

Maximo(A, k+1) = max(Maximo(A, k+1-1), A[k+1])

Exercício: máximo de um vetor com n elementos

```
Maximo(A,n)
if n≤1 then
    return A[1]
else
    return max(Maximo(A,n – 1), A[n])
```

Passo Indutivo(cont...):

Tese:

Maximo(A, k+1) = max(Maximo(A, k+1-1), A[k+1])

= max(Maximo(A, k), A[k+1])

Exercício: máximo de um vetor com n elementos

```
Maximo(A,n)
if n≤1 then
    return A[1]
else
    return max(Maximo(A,n – 1), A[n])
```

Passo Indutivo(cont…):

Tese:

Maximo(A, k+1) = max(Maximo(A, k+1-1), A[k+1])

= max($\boxed{\text{Maximo(A, k)}}$, A[k+1])

Exercício: máximo de um vetor com n elementos

```
Maximo(A,n)

if n≤1 then

    return A[1]

else

    return max(Maximo(A,n – 1), A[n])
```

Passo Indutivo(cont…):

Tese:

Maximo(A, k+1) = max(Maximo(A, k+1-1), A[k+1])

= max($\boxed{\text{Maximo(A, k)}}$, A[k+1])

= max($\boxed{\text{max\{A[1],...,A[k]\}}}$, A[k+1])

Exercício: máximo de um vetor com n elementos

```
Maximo(A,n)
if n≤1 then
    return A[1]
else
    return max(Maximo(A,n – 1), A[n])
```

Passo Indutivo(cont…):

Tese:

Maximo(A, k+1) = max(Maximo(A, k+1-1), A[k+1])

= max($\boxed{\text{Maximo(A, k)}}$, A[k+1])

= max($\boxed{\text{max\{A[1],...,A[k]\}}}$, A[k+1])

= max{A[1],...,A[k],A[k+1]}

Exercício: máximo de um vetor com n elementos

```
Maximo(A,n)
if n≤1 then
    return A[1]
else
    return max(Maximo(A,n – 1), A[n])
```

Passo Indutivo(cont...):

Tese:

$$\begin{aligned}\text{Maximo}(A, k+1) &= \max(\text{Maximo}(A, k+1-1), A[k+1]) \\ &= \max(\boxed{\text{Maximo}(A, k)}, A[k+1]) \\ &= \max(\boxed{\max\{A[1],\ldots,A[k]\}}, A[k+1]) \\ &= \boxed{\max\{A[1],\ldots,A[k],A[k+1]\}}\end{aligned}$$

✓

C.Q.D

Exercício: máximo de um vetor com n elementos

```
Maximo(A,n)
if n≤1 then
    return A[1]
else
    return max(Maximo(A,n – 1), A[n])
```

Conclusão:

Exercício: máximo de um vetor com n elementos

```
Maximo(A,n)
if n≤1 then
    return A[1]
else
    return max(Maximo(A,n – 1), A[n])
```

Conclusão:

Nosso algoritmo de Máximo Recursivo é correto, pois conseguimos provar, usando indução matemática, que para qualquer inteiro k >= 1, o algoritmo devolve o máximo elemento de um vetor contendo k valores.

Exercício: máximo de um vetor com n elementos

```
Maximo(A,n)

if n≤1 then

    return A[1]

else

    return max(Maximo(A,n – 1), A[n])
```

Conclusão:

Nosso algoritmo de Máximo Recursivo é correto, pois conseguimos provar, usando indução matemática, que para qualquer inteiro k >= 1, o algoritmo devolve o máximo elemento de um vetor contendo k valores.

Maximo(A,k) = max{A[1],A[2],...,A[k]} para qualquer inteiro k>=1.

Referências Bibliográficas

VIGNATTI, Andre.Corretude de Algoritmos Recursivos.Disponível em: <http://www.inf.ufpr.br/vignatti/courses/ci165/04.pdf>. Acesso em 22 de junho de 2017.

MASSONI, Tiago. FIGUEIREDO, Jorge. Análise e Técnicas de Algoritmos: Corretude. UFCG. Disponível em: <http://www.ebah.com.br/content/ABAAABZ6wAL/02-corretude>. Acesso em 22 de junho de 2017.