



Prova de Corretude de Algoritmos

Disciplina: Projeto e Análise de Algoritmos Alunos: José Bartolomeu Alheiros Dias Neto Lucas Wagner

Um algoritmo pode ser:

Um algoritmo pode ser:

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

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.

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

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

Se for recursivo, utilizamos apenas a indução.

Fat(n)

if n == 0

Fat(n) = 1

else

$$Fat(n) = n \cdot Fat(n - 1)$$

O que queremos provar:

Fat(n)

if
$$n == 0$$

$$Fat(n) = 1$$

else

$$Fat(n) = n \cdot Fat(n - 1)$$

Fat(n)

if n == 0

Fat(n) = 1

else

 $Fat(n) = n \cdot Fat(n - 1)$

O que queremos provar:

Proposição de Corretude:

Fat(n)

if
$$n == 0$$

$$Fat(n) = 1$$

else

$$Fat(n) = n \cdot Fat(n - 1)$$

O que queremos provar:

Proposição de Corretude:

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

Fat(n)

if
$$n == 0$$

$$Fat(n) = 1$$

else

$$Fat(n) = n \cdot Fat(n-1)$$

O que queremos provar:

Proposição de Corretude:

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

Caso Base:

O que queremos provar:

Proposição de Corretude:

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

Fat(n)

if n == 0

Fat(n) = 1

else

 $Fat(n) = n \cdot Fat(n - 1)$

Caso Base:

O que queremos provar:

Proposição de Corretude:

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

Fat(n)

if n == 0

Fat(n) = 1

else

$$Fat(n) = n \cdot Fat(n - 1)$$

Caso Base:

Para n = 0, nosso algoritmo retorna 1;

O que queremos provar:

Proposição de Corretude:

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

Fat(n)

if n == 0

Fat(n) = 1

else

$$Fat(n) = n \cdot Fat(n - 1)$$

Caso Base:

Para n = 0, nosso algoritmo retorna 1;

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

Fat(0)

if n == 0

Fat(n) = 1

else

 $Fat(n) = n \cdot Fat(n - 1)$

O que queremos provar:

Proposição de Corretude:

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.

Fat(0)

if n == 0

Fat(n) = 1

else

$$Fat(n) = n \cdot Fat(n - 1)$$

O que queremos provar:

Proposição de Corretude:

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





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.

Fat(n)

if
$$n == 0$$

$$Fat(n) = 1$$

else

$$Fat(n) = n \cdot Fat(n - 1)$$

O que queremos provar:

Proposição de Corretude:

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

Passo Indutivo:

Fat(n)

if
$$n == 0$$

$$Fat(n) = 1$$

else

$$Fat(n) = n \cdot Fat(n - 1)$$

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:

Fat(n)

if n == 0

Fat(n) = 1

else

$$Fat(n) = n \cdot Fat(n - 1)$$

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:

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

Fat(n)

if n == 0

Fat(n) = 1

else

$$Fat(n) = n \cdot Fat(n - 1)$$

Passo Indutivo(cont...):

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

Fat(n)

if n == 0

Fat(n) = 1

else

$$Fat(n) = n \cdot Fat(n - 1)$$

Passo Indutivo(cont...):

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

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

Fat(n)

if n == 0

Fat(n) = 1

else

$$Fat(n) = n \cdot Fat(n - 1)$$

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

Fat(n)

if n == 0

Fat(n) = 1

else

$$Fat(n) = n \cdot Fat(n - 1)$$

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

Fat(n)

if n == 0

Fat(n) = 1

else

$$Fat(n) = n \cdot Fat(n - 1)$$

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) $Fat(k+1) = k + 1 * Fat(k) \longrightarrow (2)$

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

Fat(n)

if n == 0

Fat(n) = 1

else

$$Fat(n) = n \cdot Fat(n - 1)$$

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) $Fat(k+1) = k + 1 * Fat(k) \longrightarrow (2)$

(2)
$$Fat(k) = k!$$

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

Fat(n)

if n == 0

Fat(n) = 1

else

$$Fat(n) = n \cdot Fat(n-1)$$

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) $Fat(k+1) = k + 1 * Fat(k) \longrightarrow (2)$

(2)
$$Fat(k) = k!$$

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

Fat(n)

if n == 0

Fat(n) = 1

else

$$Fat(n) = n \cdot Fat(n - 1)$$

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) $Fat(k+1) = k + 1 * Fat(k) \longrightarrow (2)$

(2)
$$Fat(k) = k!$$

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

Fat(n)

if
$$n == 0$$

$$Fat(n) = 1$$

else

$$Fat(n) = n \cdot Fat(n - 1)$$

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:

Fat(n)

if n == 0

Fat(n) = 1

else

$$Fat(n) = n \cdot Fat(n - 1)$$

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

$$Fat(n) = n \cdot Fat(n - 1)$$

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.

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

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)

if n≤1 then

return A[1]

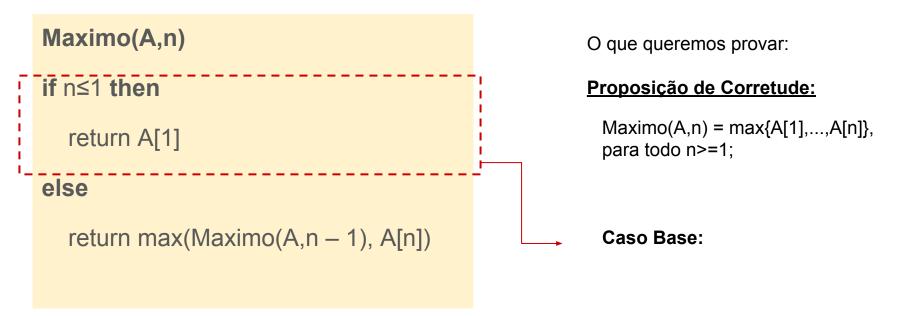
else

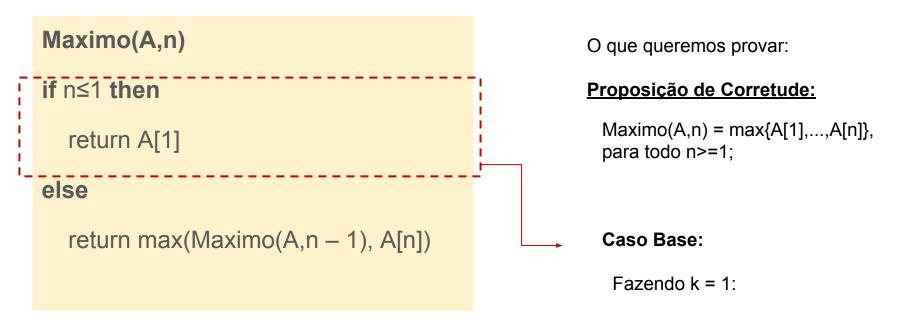
return max(Maximo(A,n-1), A[n])

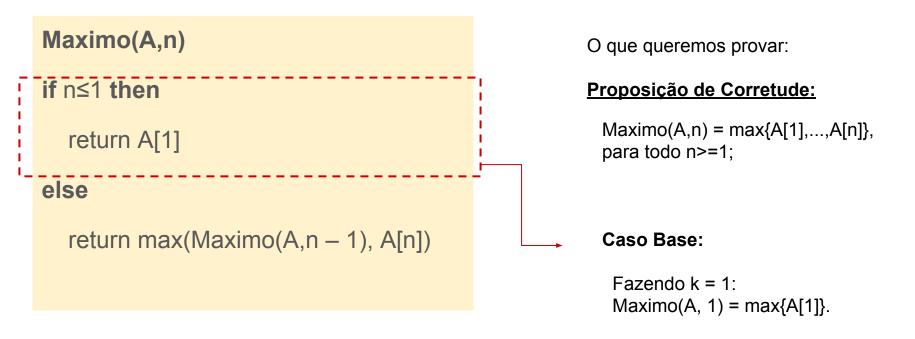
O que queremos provar:

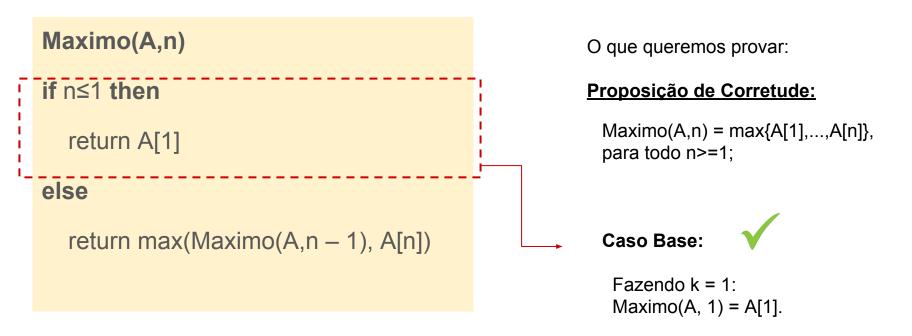
Proposição de Corretude:

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









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:

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

Maximo(A,n)

if n≤1 then

return A[1]

else

return max(Maximo(A, n - 1), A[n])

Passo Indutivo(cont...):

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,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])

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])

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])
```

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

```
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])

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

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

return max(Maximo(A, n - 1), A[n])

Tese: Maximo(A, k+1) = max(Maximo(A, k+1-1), A[k+1]) = max(Maximo(A, k), A[k+1]) = max(max{A[1],...,A[k]}, A[k+1])

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

Passo Indutivo(cont...):

```
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])

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

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



Maximo(A,n)

if n≤1 then

return A[1]

else

return max(Maximo(A,n - 1), A[n])

Conclusão:

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,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.