

[Hoje
Quinta
Sem
Exercício
PL

Corretude de Algoritmos

Lembre a definição de fatorial: $n! = n \cdot (n - 1) \cdot (n - 2) \cdot \dots \cdot 2 \cdot 1$

Corretude de Algoritmos

Lembre a definição de fatorial: $n! = n \cdot (n - 1) \cdot (n - 2) \cdot \dots \cdot 2 \cdot 1$

Exemplo (Calculando o fatorial)

Considere o código abaixo.

Ex: $Fat(1) = 1 = 1!$ ✓
 $Fat(2) = 2 \times Fat(1) = 2 = 2!$ ✓
 $Fat(3) = 3 \times Fat(2) = 3 \times 2 = 6 = 3!$

Fatorial
<i>Entrada: x (inteiro positivo)</i>
Função $Fat(x)$
If $x = 1$
Return 1
End if
Return $x \cdot Fat(x - 1)$
Fim Função

Mostre que para todo $n \geq 1$, $Fat(n)$ retorna $n!$

```
Função Fat(x)
  If x = 1
    Return 1
  End if
  Return x · Fat(x - 1)
Fim Função
```

Ques: Vaz! Fat(n) = n!

Prova p/ indução forte

① Caso Base: ~~$n = 1$. Por causa do "If", $Fat(1) = 1 = 1!$~~

② Passo indutivo: Assume vale p $1, 2, \dots$

```
Função  $Fat(x)$   
  If  $x = 1$   
    Return 1  
  End if  
  Return  $x \cdot Fat(x - 1)$   
Fim Função
```

Caso Base: $n = 1$. Por causa do “**If**”, $Fat(1) = 1 = 1!$

Passo Indutivo. Suponha que a propriedade **valha para n** , ou seja $Fat(n) = n!$. **Queremos** provar que vale para $n + 1$

```
Função  $Fat(x)$   
  If  $x = 1$   
    Return 1  
  End if  
  Return  $x \cdot Fat(x - 1)$   
Fim Função
```

Caso Base: $n = 1$. Por causa do “**If**”, $Fat(1) = 1 = 1!$

Passo Indutivo. Suponha que a propriedade **valha para n** , ou seja $Fat(n) = n!$. **Queremos** provar que vale para $n + 1$

Pelo código, $Fat(n + 1) = (n + 1) \cdot Fat(n)$

```
Função  $Fat(x)$   
  If  $x = 1$   
    Return 1  
  End if  
  Return  $x \cdot Fat(x - 1)$   
Fim Função
```

Caso Base: $n = 1$. Por causa do “**If**”, $Fat(1) = 1 = 1!$

Passo Indutivo. Suponha que a propriedade **valha para n** , ou seja $Fat(n) = n!$. **Queremos** provar que vale para $n + 1$

Pelo código, $Fat(n + 1) = (n + 1) \cdot Fat(n)$

Pela hipótese indutiva $Fat(n) = n!$, então

```
Função Fat( $x$ )  
  If  $x = 1$   
    Return 1  
  End if  
  Return  $x \cdot \text{Fat}(x - 1)$   
Fim Função
```

Caso Base: $n = 1$. Por causa do “**If**”, $\text{Fat}(1) = 1 = 1!$

Passo Indutivo. Suponha que a propriedade **valha para n** , ou seja $\text{Fat}(n) = n!$. **Queremos** provar que vale para $n + 1$

Pelo código, $\text{Fat}(n + 1) = (n + 1) \cdot \text{Fat}(n)$

Pela hipótese indutiva $\text{Fat}(n) = n!$, então

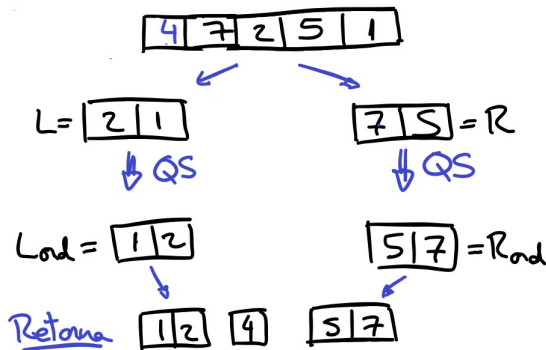
$$\begin{aligned}\text{Fat}(n + 1) &= (n + 1) \cdot \text{Fat}(n) \\ &= (n + 1) \cdot n! \\ &= (n + 1) \cdot n \cdot (n - 1) \cdot \dots \cdot 2 \cdot 1 \quad (\text{Pela definição de } n!) \\ &= (n + 1)!\end{aligned}$$

Quicksort

Quicksort é um dos algoritmos mais populares para ordenar lista de números

Quicksort

Quicksort é um dos algoritmos mais populares para ordenar lista de números



Quicksort

Quicksort(A)

If $A.len = 0$ ou $A.len = 1$

 Return A

Else

L = vetor com números de A **menores** que $A[1]$ # *pivot*

R = vetor com números de A **maiores** que $A[1]$

P = vetor com números de A **iguais** a $A[1]$

$L_{ord} \leftarrow \text{Quicksort}(L)$

$R_{ord} \leftarrow \text{Quicksort}(R)$

 Return concatenação de L_{ord}, P e R_{ord}

End if

Quicksort

Proposição

Quicksort(A) retorna o vetor A ordenado

Proposição

Quicksort(A) retorna o vetor A ordenado

Prova: Por indução forte no tamanho n do vetor A

Proposição

Quicksort(A) retorna o vetor A ordenado

Prova: Por indução forte no tamanho n do vetor A

Caso base: tamanho $n = 0$ ou $n = 1$. Quicksort(A) retorna A , OK

Proposição

Quicksort(A) retorna o vetor A ordenado

Prova: Por indução forte no tamanho n do vetor A

Caso base: tamanho $n = 0$ ou $n = 1$. Quicksort(A) retorna A , OK

Passo indutivo. Considere $n \geq 1$ e **suponha** que Quicksort funcione pra todo vetor de tamanho $\leq n$

Queremos provar que funciona para vetores de tamanho $n + 1$

Quicksort

Considere um vetor A de tamanho $n + 1$

Quicksort

Considere um vetor A de tamanho $n + 1$

Note que os vetores L e R construídos têm tamanho **estritamente** menor que A (ou seja $\leq n$)

Quicksort

Considere um vetor A de tamanho $n + 1$

Note que os vetores L e R construídos têm tamanho **estritamente** menor que A (ou seja $\leq n$)

Então podemos aplicar a **hipótese indutiva** e obter que L_{ord} e R_{ord} estão **ordenados**

Quicksort

Considere um vetor A de tamanho $n + 1$

Note que os vetores L e R construídos têm tamanho **estritamente** menor que A (ou seja $\leq n$)

Então podemos aplicar a **hipótese indutiva** e obter que L_{ord} e R_{ord} estão **ordenados**

Portanto, Quicksort(A) retorna

$$\underbrace{L_{ord}}_{\text{menores que } A[1] \text{ ordenados}} + \underbrace{P}_{\text{iguais a } A[1]} + \underbrace{R_{ord}}_{\text{maiores que } A[1] \text{ ordenados}}$$

Quicksort

Considere um vetor A de tamanho $n + 1$

Note que os vetores L e R construídos têm tamanho **estritamente** menor que A (ou seja $\leq n$)

Então podemos aplicar a **hipótese indutiva** e obter que L_{ord} e R_{ord} estão **ordenados**

Portanto, Quicksort(A) retorna

$$\underbrace{L_{ord}}_{\text{menores que } A[1] \text{ ordenados}} + \underbrace{P}_{\text{iguais a } A[1]} + \underbrace{R_{ord}}_{\text{maiores que } A[1] \text{ ordenados}}$$

\Rightarrow retorna vetor ordenado

Quicksort

Considere um vetor A de tamanho $n + 1$

Note que os vetores L e R construídos têm tamanho **estritamente** menor que A (ou seja $\leq n$)

Então podemos aplicar a **hipótese indutiva** e obter que L_{ord} e R_{ord} estão **ordenados**

Portanto, Quicksort(A) retorna

$$\underbrace{L_{ord}}_{\text{menores que } A[1] \text{ ordenados}} + \underbrace{P}_{\text{iguais a } A[1]} + \underbrace{R_{ord}}_{\text{maiores que } A[1] \text{ ordenados}}$$

\Rightarrow retorna vetor ordenado

Conversor binário

Converter número em sua [representação binária](#)

Conversor binário

Converter número em sua **representação binária**

Saída: vetor $v = v_1, v_2, \dots, v_k$ onde v_1 é o dígito **binário mais significativo**, v_2 o segundo mais significativo, etc.

Conversor binário

Converter número em sua **representação binária**

Saída: vetor $v = v_1, v_2, \dots, v_k$ onde v_1 é o dígito **binário mais significativo**, v_2 o segundo mais significativo, etc.

Ex:

Saída

$n = 6$	$(v_1, v_2, v_3) = (1, 1, 0)$
$n = 8$	$(v_1, v_2, v_3, v_4) = (1, 0, 0, 0)$
$n = 0$	vetor vazio

Conversor binário

Converter número em sua **representação binária**

Saída: vetor $v = v_1, v_2, \dots, v_k$ onde v_1 é o dígito **binário mais significativo**, v_2 o segundo mais significativo, etc.

Ex:

Saída

$$n = 6 \quad (v_1, v_2, v_3) = (1, 1, 0)$$

$$n = 8 \quad (v_1, v_2, v_3, v_4) = (1, 0, 0, 0)$$

$$n = 0 \quad \text{vetor vazio}$$

Q: Qual é a relação entre n e o vetor v ?

Conversor binário

Converter número em sua **representação binária**

Saída: vetor $v = v_1, v_2, \dots, v_k$ onde v_1 é o dígito **binário mais significativo**, v_2 o segundo mais significativo, etc.

Ex:

Saída

$$n = 6 \quad (v_1, v_2, v_3) = (1, 1, 0)$$

$$n = 8 \quad (v_1, v_2, v_3, v_4) = (1, 0, 0, 0)$$

$$n = 0 \quad \text{vetor vazio}$$

Q: Qual é a relação entre n e o vetor v ?

$$n = 2^{k-1}v_1 + 2^{k-2}v_2 + \dots + 2^0v_k$$

Conversor binário

Converte(n)

If $n = 0$

Return vetor vazio

Else if n é par

Vetor $u \leftarrow$ Converte($\frac{n}{2}$)

Return vetor ($u_1, u_2, \dots, u_{u.len}, 0$) [ou seja, u concatenado com 0 no final]

Else if n é impar

Vetor $u \leftarrow$ Converte($\frac{n-1}{2}$)

Return vetor ($u_1, u_2, \dots, u_{u.len}, 1$) [ou seja, u concatenado com 1 no final]

End if

Conversor binário

Converte(n)

If $n = 0$

Return vetor vazio

Else if n é par

Vetor $u \leftarrow$ Converte($\frac{n}{2}$)

Return vetor $(u_1, u_2, \dots, u_{u.len}, 0)$ [ou seja, u concatenado com 0 no final]

Else if n é ímpar

Vetor $u \leftarrow$ Converte($\frac{n-1}{2}$)

Return vetor $(u_1, u_2, \dots, u_{u.len}, 1)$ [ou seja, u concatenado com 1 no final]

End if

Proposição

Para todo n , o vetor v retornado por $\text{Converte}(n)$ é a representação binária de n : (k é o tamanho de v)

$$n = 2^{k-1}v_1 + 2^{k-2}v_2 + \dots + 2^0v_k$$

Prova: Por indução forte em n **Caso base:** $n = 0$. `Converte(0)` retorna lista vazia, correto por definição

Prova: Por indução forte em n **Caso base:** $n = 0$. $\text{Converte}(0)$ retorna lista vazia, correto por definição

Passo indutivo. Considere $n \geq 1$. Suponha que o algoritmo funcione pra todo número $\leq n$

Prova: Por indução forte em n **Caso base:** $n = 0$. $\text{Converte}(0)$ retorna lista vazia, correto por definição

Passo indutivo. Considere $n \geq 1$. Suponha que o algoritmo funcione pra todo número $\leq n$

Precisamos provar que funciona pro número $n + 1$

Conversor binário

Caso 1: $n + 1$ é par. Note que $\frac{n+1}{2}$ é **estritamente menor** que $n + 1$, portanto podemos usar a **hipótese indutiva**

Conversor binário

Caso 1: $n + 1$ é par. Note que $\frac{n+1}{2}$ é **estritamente menor** que $n + 1$, portanto podemos usar a **hipótese indutiva**

Lembre $u = \text{Converte}(\frac{n+1}{2})$. Seja k o tamanho de u

Conversor binário

Caso 1: $n + 1$ é par. Note que $\frac{n+1}{2}$ é **estritamente menor** que $n + 1$, portanto podemos usar a **hipótese indutiva**

Lembre $u = \text{Converte}(\frac{n+1}{2})$. Seja k o tamanho de u

$\text{Converte}(n + 1)$ retorna vetor $(u_1, \dots, u_k, 0)$ com tamanho $k + 1$ que tem valor

$$\begin{aligned} & 2^k u_1 + 2^{k-1} u_2 + \dots + 2^1 u_k + \underbrace{2^0 \cdot 0}_0 \\ &= 2 \cdot \left(2^{k-1} u_1 + 2^{k-2} u_2 + \dots + 2^0 u_k \right) \\ &= 2 \cdot \frac{n + 1}{2} && \text{(pela hipótese indutiva)} \\ &= n + 1 \end{aligned}$$

Caso 2: $n + 1$ é **impar**. Note que $\frac{(n+1)-1}{2}$ é **estritamente menor** que $n + 1$, portanto podemos usar a **hipótese indutiva**

Caso 2: $n + 1$ é **impar**. Note que $\frac{(n+1)-1}{2}$ é **estritamente menor** que $n + 1$, portanto podemos usar a **hipótese indutiva**

Lembre $u = \text{Converte}(\frac{(n+1)-1}{2})$. Seja k o tamanho de u

Conversor binário

Caso 2: $n + 1$ é **ímpar**. Note que $\frac{(n+1)-1}{2}$ é **estritamente menor** que $n + 1$, portanto podemos usar a **hipótese indutiva**

Lembre $u = \text{Converte}(\frac{(n+1)-1}{2})$. Seja k o tamanho de u

$\text{Converte}(n + 1)$ retorna vetor $(u_1, \dots, u_k, 1)$ com tamanho $k + 1$ que tem valor

$$\begin{aligned} & 2^k u_1 + 2^{k-1} u_2 + \dots + 2^1 u_k + 2^0 \cdot 1 \\ &= 2 \cdot \left(2^{k-1} u_1 + 2^{k-2} u_2 + \dots + 2^0 u_k \right) + 1 \\ &= 2 \cdot \frac{(n + 1) - 1}{2} + 1 && \text{(pela hipótese indutiva)} \\ &= n + 1 \end{aligned}$$

Exercício 2: Prove por indução em $n \geq 0$ que para toda entrada (a, n) o algoritmo abaixo retorna a^n

```
procedure power(a: nonzero real number, n: nonnegative integer)  
if  $n = 0$  then return 1  
else return  $a \cdot \textit{power}(a, n - 1)$ 
```

Exercício

Exercício 3: (Avaliação de polinômio)

a) Prove por indução em $n \geq 0$ que para toda entrada (A, x, n) o algoritmo abaixo retorna $A[0] + A[1] \cdot x + A[2] \cdot x^2 + \dots + A[n] \cdot x^n$

```
Algo(Vetor de números  $A$ , número real  $x$ , inteiro  $n$ )
```

```
If  $n = 0$ 
```

```
    Return  $A[0]$ 
```

```
Else
```

```
     $B = (A[1], A[2], \dots, A[n])$     # remove  $A[0]$ 
```

```
     $tmp = \mathbf{Algo}(B, x, n - 1)$ 
```

```
    Return  $(tmp \cdot x) + A[0]$ 
```

```
End if
```

b) Prove por indução que para toda entrada (A, x, n) o número de operações aritméticas ($+$ e \cdot) realizadas é menor ou igual a $2n$