

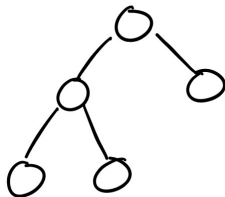
Exemplos

Definição (Informal)

Uma árvore binária enraizada é

(i) um conjunto vazio ou

(ii) um nó denominado raiz com uma subárvore a esquerda e outra a direita



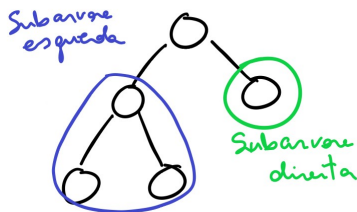
Exemplos

Definição (Informal)

Uma *árvore binária enraizada* é

(i) um conjunto vazio ou

(ii) um nó denominado raiz com uma subárvore a esquerda e outra a direita



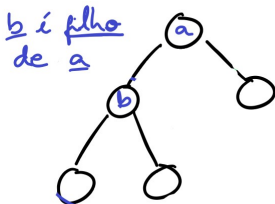
Exemplos

Definição (Informal)

Uma *árvore binária enraizada* é

(i) um conjunto vazio ou

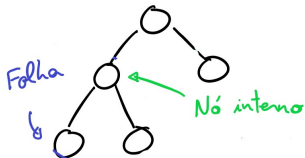
(ii) um nó denominado raiz com uma subárvore a esquerda e outra a direita



Definição (Informal)

Dada uma árvore binária enraizada:

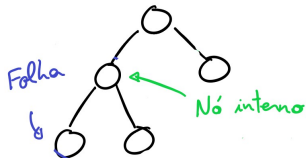
- Uma **folha** é um nó com 0 filhos
- Um **nó interno** é um nó com pelo menos um filho



Definição (Informal)

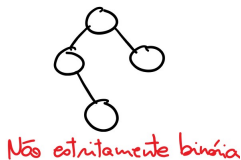
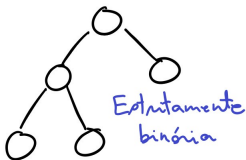
Dada uma árvore binária enraizada:

- Uma **folha** é um nó com 0 filhos
- Um **nó interno** é um nó com pelo menos um filho



Definição

Uma árvore é dita **estritamente binária** se todo nó possui 0 ou 2 filhos



Exemplos

Exemplo

Prove por *indução no número de nós internos* que para toda árvore estritamente binária T temos

$$\text{folhas}(T) - \text{internos}(T) = 1,$$

onde

$\text{folhas}(T)$ = número de folhas

$\text{internos}(T)$ = número de nós internos

Exemplo

Prove por *indução no número de nós internos* que para toda árvore estritamente binária T temos

$$\text{folhas}(T) - \text{internos}(T) = 1,$$

onde

$$\text{folhas}(T) = \text{número de folhas}$$

$$\text{internos}(T) = \text{número de nós internos}$$

Prova:

Caso base: Árvore com $n = 0$ nós internos.

Exemplos

Exemplo

Prove por *indução no número de nós internos* que para toda árvore estritamente binária T temos

$$\text{folhas}(T) - \text{internos}(T) = 1,$$

onde

$$\text{folhas}(T) = \text{número de folhas}$$

$$\text{internos}(T) = \text{número de nós internos}$$

Prova:

Caso base: Árvore com $n = 0$ nós internos. Tem que ser **só uma folha**.

Verificamos $1 - 0 = 1$

Exemplo

Por *indução no número de nós internos*, toda árvore estritamente binária T

$$\text{folhas}(T) = \text{internos}(T) + 1.$$

Passo indutivo: Suponha que **vale** para toda árvore com $0, 1, \dots, n$ nós internos. **Queremos** provar pra $n + 1$

Exemplo

Por *indução no número de nós internos*, toda árvore estritamente binária T

$$\text{folhas}(T) = \text{internos}(T) + 1.$$

Passo indutivo: Suponha que **vale** para toda árvore com $0, 1, \dots, n$ nós internos. **Queremos** provar pra $n + 1$

Considere uma árvore com $n + 1$ nós internos

Exemplo

Por *indução no número de nós internos*, toda árvore estritamente binária T

$$\text{folhas}(T) = \text{internos}(T) + 1.$$

Passo indutivo: Suponha que **vale** para toda árvore com $0, 1, \dots, n$ nós internos. **Queremos** provar pra $n + 1$

Considere uma árvore com $n + 1$ nós internos

Sub-árvores esquerda e direita tem **estritamente menos nós** que a árvore toda (então $\leq n$ nós)

Exemplo

Por *indução no número de nós internos*, toda árvore estritamente binária T

$$\text{folhas}(T) = \text{internos}(T) + 1.$$

Passo indutivo: Suponha que **vale** para toda árvore com $0, 1, \dots, n$ nós internos. **Queremos** provar pra $n + 1$

Considere uma árvore com $n + 1$ nós internos

Sub-árvores esquerda e direita tem **estritamente menos nós** que a árvore toda (então $\leq n$ nós)

Portanto podemos aplicar a **hipotese indutiva** às sub-árvores:

Exemplo

Por *indução no número de nós internos*, toda árvore estritamente binária T

$$\text{folhas}(T) = \text{internos}(T) + 1.$$

Passo indutivo: Suponha que **vale** para toda árvore com $0, 1, \dots, n$ nós internos. **Queremos** provar pra $n + 1$

Considere uma árvore com $n + 1$ nós internos

Sub-árvores esquerda e direita tem **estritamente menos nós** que a árvore toda (então $\leq n$ nós)

Portanto podemos aplicar a **hipotese indutiva** às sub-árvores:

$$\text{folhas}(E) = \text{internos}(E) + 1 \quad \text{e} \quad \text{folhas}(D) = \text{internos}(D) + 1$$

Exemplo

Por *indução no número de nós internos*, toda árvore estritamente binária T

$$\text{folhas}(T) = \text{internos}(T) + 1.$$

$$\text{folhas}(E) = \text{internos}(E) + 1 \quad \text{e} \quad \text{folhas}(D) = \text{internos}(D) + 1$$

Exemplo

Por *indução no número de nós internos*, toda árvore estritamente binária T

$$\text{folhas}(T) = \text{internos}(T) + 1.$$

$$\text{folhas}(E) = \text{internos}(E) + 1 \quad \text{e} \quad \text{folhas}(D) = \text{internos}(D) + 1$$

Porém toda folha de T vem ou de E ou de D :

$$\text{folhas}(T) = \text{folhas}(E) + \text{folhas}(D)$$

Exemplo

Por *indução no número de nós internos*, toda árvore estritamente binária T

$$\text{folhas}(T) = \text{internos}(T) + 1.$$

$$\text{folhas}(E) = \text{internos}(E) + 1 \quad \text{e} \quad \text{folhas}(D) = \text{internos}(D) + 1$$

Porém toda folha de T vem ou de E ou de D :

$$\begin{aligned} \text{folhas}(T) &= \text{folhas}(E) + \text{folhas}(D) \\ &= \text{internos}(E) + 1 + \text{internos}(D) + 1 \quad (\text{hipotese indutiva}) \end{aligned}$$

Exemplo

Por *indução no número de nós internos*, toda árvore estritamente binária T

$$\text{folhas}(T) = \text{internos}(T) + 1.$$

$$\text{folhas}(E) = \text{internos}(E) + 1 \quad \text{e} \quad \text{folhas}(D) = \text{internos}(D) + 1$$

Porém toda folha de T vem ou de E ou de D :

$$\begin{aligned} \text{folhas}(T) &= \text{folhas}(E) + \text{folhas}(D) \\ &= \text{internos}(E) + 1 + \text{internos}(D) + 1 \quad (\text{hipotese indutiva}) \\ &= \text{internos}(T) + 1 \end{aligned}$$

Exemplo inducao

Definição

A **altura** de uma árvore binária enraizada é definida de forma recursiva:

- A árvore vazia tem altura 0
- Se a árvore não é vazia, sua altura é igual a

$$\max\{\text{altura subárvore esquerda}, \text{altura subárvore direita}\} + 1.$$

Exemplo inducao

Proposição

Para toda árvore binária enraizada temos

$$\text{número de nós} \leq 2^{\text{altura}} - 1.$$

Exemplo inducao

Proposição

Para toda árvore binária enraizada temos

$$\text{número de nós} \leq 2^{\text{altura}} - 1.$$

Prova: Por indução forte no número de nós n

Caso base: Árvore com $n = 0$ nós. Como temos 0 nós e a altura é 1, verificamos

$$0 \leq 2^0 - 1.$$

Exemplo inducao

Proposição

Para toda árvore binária enraizada temos

$$\text{número de nós} \leq 2^{\text{altura}} - 1.$$

Prova: Por indução forte no número de nós n

Caso base: Árvore com $n = 0$ nós. Como temos 0 nós e a altura é 1, verificamos

$$0 \leq 2^0 - 1.$$

Passo indutivo. Suponha que seja **verdade** pra toda árvore com $0, 1, \dots, n$ nós

Exemplo inducao

Proposição

Para toda árvore binária enraizada temos

$$\text{número de nós} \leq 2^{\text{altura}} - 1.$$

Prova: Por indução forte no número de nós n

Caso base: Árvore com $n = 0$ nós. Como temos 0 nós e a altura é 1, verificamos

$$0 \leq 2^0 - 1.$$

Passo indutivo. Suponha que seja **verdade** pra toda árvore com $0, 1, \dots, n$ nós

Queremos provar pra árvore com $n + 1$ nós

Exemplo inducao

Exemplo inducao

Considere uma árvore com $n + 1$ nós.

Exemplo inducao

Considere uma árvore com $n + 1$ nós.

$$\# \text{ nós} = \# \text{ nós sub-árvore esquerda} + \# \text{ nós sub-árvore direita} + 1$$

Exemplo inducao

Considere uma árvore com $n + 1$ nós.

$$\# \text{ nós} = \# \text{ nós sub-árvore esquerda} + \# \text{ nós sub-árvore direita} + 1$$

Sub-árvores esquerda e direita tem **estritamente menos nós** que a árvore toda (então $\leq n$ nós)

Exemplo inducao

Considere uma árvore com $n + 1$ nós.

$$\# \text{ nós} = \# \text{ nós sub-árvore esquerda} + \# \text{ nós sub-árvore direita} + 1$$

Sub-árvores esquerda e direita tem **estritamente menos nós** que a árvore toda (então $\leq n$ nós)

Portanto podemos aplicar a **hipotese indutiva** às sub-arvores:

Exemplo inducao

Considere uma árvore com $n + 1$ nós.

$$\# \text{ nós} = \# \text{ nós sub-árvore esquerda} + \# \text{ nós sub-árvore direita} + 1$$

Sub-árvores esquerda e direita tem **estritamente menos nós** que a árvore toda (então $\leq n$ nós)

Portanto podemos aplicar a **hipotese indutiva** às sub-árvores:

$$\# \text{ nós} = \underbrace{\# \text{ nós sub-árvore esquerda}}_{\leq 2^{\text{altura sub-árvore esquerda}} - 1} + \underbrace{\# \text{ nós sub-árvore direita}}_{\leq 2^{\text{altura sub-árvore direita}} - 1} + 1$$

Exemplo inducao

Considere uma árvore com $n + 1$ nós.

$$\# \text{ nós} = \# \text{ nós sub-árvore esquerda} + \# \text{ nós sub-árvore direita} + 1$$

Sub-árvores esquerda e direita tem **estritamente menos nós** que a árvore toda (então $\leq n$ nós)

Portanto podemos aplicar a **hipotese indutiva** às sub-arvores:

$$\begin{aligned} \# \text{ nós} &= \underbrace{\# \text{ nós sub-árvore esquerda}}_{\leq 2^{\text{altura sub-árvore esquerda}} - 1} + \underbrace{\# \text{ nós sub-árvore direita}}_{\leq 2^{\text{altura sub-árvore direita}} - 1} + 1 \\ &\leq 2^{\text{altura sub-árvore esquerda}} + 2^{\text{altura sub-árvore direita}} - 1 \end{aligned}$$

Exemplo inducao

Considere uma árvore com $n + 1$ nós.

$$\# \text{ nós} = \# \text{ nós sub-árvore esquerda} + \# \text{ nós sub-árvore direita} + 1$$

Sub-árvores esquerda e direita tem **estritamente menos nós** que a árvore toda (então $\leq n$ nós)

Portanto podemos aplicar a **hipotese indutiva** às sub-arvores:

$$\begin{aligned} \# \text{ nós} &= \underbrace{\# \text{ nós sub-árvore esquerda}}_{\leq 2^{\text{altura sub-árvore esquerda}} - 1} + \underbrace{\# \text{ nós sub-árvore direita}}_{\leq 2^{\text{altura sub-árvore direita}} - 1} + 1 \\ &\leq 2^{\text{altura sub-árvore esquerda}} + 2^{\text{altura sub-árvore direita}} - 1 \\ &\leq 2 \cdot 2^{\max\{\text{alt esq}, \text{alt dir}\}} - 1 \end{aligned}$$

Exemplo inducao

Considere uma árvore com $n + 1$ nós.

$$\# \text{ nós} = \# \text{ nós sub-árvore esquerda} + \# \text{ nós sub-árvore direita} + 1$$

Sub-árvores esquerda e direita tem **estritamente menos nós** que a árvore toda (então $\leq n$ nós)

Portanto podemos aplicar a **hipotese indutiva** às sub-árvores:

$$\begin{aligned} \# \text{ nós} &= \underbrace{\# \text{ nós sub-árvore esquerda}}_{\leq 2^{\text{altura sub-árvore esquerda}} - 1} + \underbrace{\# \text{ nós sub-árvore direita}}_{\leq 2^{\text{altura sub-árvore direita}} - 1} + 1 \\ &\leq 2^{\text{altura sub-árvore esquerda}} + 2^{\text{altura sub-árvore direita}} - 1 \\ &\leq 2 \cdot 2^{\max\{\text{alt esq}, \text{alt dir}\}} - 1 \\ &= 2^{\max\{\text{alt esq}, \text{alt dir}\} + 1} - 1 \end{aligned}$$

Exemplo inducao

Considere uma árvore com $n + 1$ nós.

$$\# \text{ nós} = \# \text{ nós sub-árvore esquerda} + \# \text{ nós sub-árvore direita} + 1$$

Sub-árvores esquerda e direita tem **estritamente menos nós** que a árvore toda (então $\leq n$ nós)

Portanto podemos aplicar a **hipotese indutiva** às sub-arvores:

$$\begin{aligned} \# \text{ nós} &= \underbrace{\# \text{ nós sub-árvore esquerda}}_{\leq 2^{\text{altura sub-árvore esquerda}} - 1} + \underbrace{\# \text{ nós sub-árvore direita}}_{\leq 2^{\text{altura sub-árvore direita}} - 1} + 1 \\ &\leq 2^{\text{altura sub-árvore esquerda}} + 2^{\text{altura sub-árvore direita}} - 1 \\ &\leq 2 \cdot 2^{\max\{\text{alt esq}, \text{alt dir}\}} - 1 \\ &= 2^{\max\{\text{alt esq}, \text{alt dir}\} + 1} - 1 \\ &= 2^{\text{altura árvore toda}} - 1. \end{aligned}$$

Exercício 1:

Seja T uma árvore estritamente binária (vide definições abaixo) com n folhas. Definimos o *potencial* de uma folha como 2 e o potencial de um nó interno como o produto do potencial de seus filhos.

- Desenhe um árvore estritamente binária com seis folhas e encontre o potencial de sua raiz.
- Utilize indução na altura da árvore para mostrar que o potencial da raiz de uma árvore estritamente binária de altura h é no máximo 2^{2^h} .

Corretude de Algoritmos

$$3! = 3 \times 2 \times 1 = 6$$

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)

Ex: $Fat(1) = 1 = 1!$
 $Fat(2) = 2 \times Fat(1)$
 $\tau = 2 \times 1 =$

Considere o código abaixo.

Fatorial
<i>Entrada: x (inteiro positivo)</i>
Função $Fat(x)$
If $x = 1$ 2
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  $x \cdot Fat(x - 1)$   
Fim Função
```

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

```
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 $n + 1$

```
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

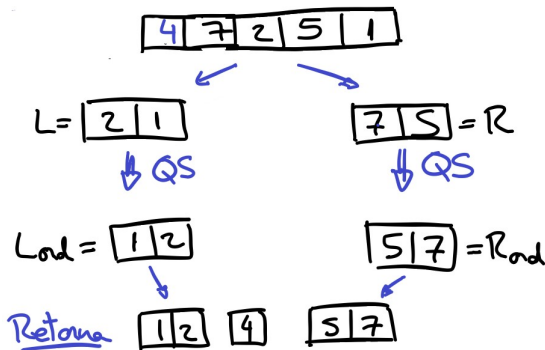
$$\begin{aligned}\text{Fat}(n + 1) &= (n + 1) \cdot \text{Fat}(n) \\ &= (n + 1) \cdot n! \\ &= (n + 1) \cdot n \cdot (n - 1) \cdot \dots \cdot 2 \cdot 1 \quad (\text{Pela definição de } n!) \\ &= (n + 1)!\end{aligned}$$

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(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

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

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

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

Conversor binário

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 $2n$