Estruturas Discretas

Marco Molinaro

Indução Forte

Indução Forte

Indução forte

- **(Caso base)** Que *Propriedade* vale para n_0
- **(Passo indutivo)** Assume que propriedade vale para **todos os números** $\{n_0, n_0 + 1, \ldots, n\}$, mostra que vale para n + 1

Indução Forte

Indução forte

- **(Caso base)** Que *Propriedade* vale para n_0
- **(Passo indutivo)** Assume que propriedade vale para **todos os números** $\{n_0, n_0 + 1, \ldots, n\}$, mostra que vale para n + 1

- **Indução forte** faz tudo que a indução fraca faz

Indução Forte

Indução forte

- **(Caso base)** Que *Propriedade* vale para n_0
- **(Passo indutivo)** Assume que propriedade vale para **todos os números** $\{n_0, n_0 + 1, \ldots, n\}$, mostra que vale para n + 1

- **Indução forte** faz tudo que a indução fraca faz
- Porém pode precisar de múltiplos casos base, **cuidado**

Fibonacci: $f_1 = 1$, $f_2 = 1$, e $f_n = f_{n-1} + f_{n-2}$ para todo $n \geq 3$

Proposição

Para todo $n \geq 1$, $f_n \leq 2^n$

$f_1=1$, $f_2=1$, $f_3=2$, $f_4=3$, $f_5=5$, $f_6=8$, $f_7=13$

Prova p/ ind. forte:

① Caso base: DEPOIS

② Passo ind: Assume que vale p/ $f_1, f_2, \ldots f_n$. Quer:

vale p/ f_{n+1}, i.e $f_{n+1} \leq 2^{n+1}$

$$f_{n+1} = f_n + f_{n-1}$$

$$\overset{HI}{\leq} 2^n + 2^{n-1} \leq 2^{n+1}$$

$$\ldots$$

$$\leq 2^{n+2}$$

Fibonacci: $f_1 = 1$, $f_2 = 1$, e $f_n = f_{n-1} + f_{n-2}$ para todo $n \geq 3$

Proposição

Para todo $n \geq 1$, $f_n \leq 2^n$

Prova: Por indução **forte**

Fibonacci: $f_1 = 1$, $f_2 = 1$, e $f_n = f_{n-1} + f_{n-2}$ para todo $n \geq 3$

Proposição

Para todo $n \geq 1$, $f_n \leq 2^n$

Prova: Por indução **forte**
Casos base: $n = 1$ **e** $n = 2$**.** É verdade que $f_1 \leq 2^1$ e $f_2 \leq 2^2$, OK
(veremos depois porque 2 casos base)

Fibonacci: $f_1 = 1$, $f_2 = 1$, e $f_n = f_{n-1} + f_{n-2}$ para todo $n \geq 3$

Proposição

Para todo $n \geq 1$, $f_n \leq 2^n$

Prova: Por indução **forte**
Casos base: $n = 1$ **e** $n = 2$**.** É verdade que $f_1 \leq 2^1$ e $f_2 \leq 2^2$, OK
(veremos depois porque 2 casos base)

Passo Indutivo. Pegue $n \geq 2$. Assuma que propriedade vale para todos até f_n, ou seja

$$f_1 \leq 2^1,\ f_2 \leq 2^2,\ \ldots, f_n \leq 2^n.$$

Fibonacci: $f_1 = 1$, $f_2 = 1$, e $f_n = f_{n-1} + f_{n-2}$ para todo $n \geq 3$

Proposição

Para todo $n \geq 1$, $f_n \leq 2^n$

Prova: Por indução **forte**
Casos base: $n = 1$ **e** $n = 2$**.** É verdade que $f_1 \leq 2^1$ e $f_2 \leq 2^2$, OK
(veremos depois porque 2 casos base)

Passo Indutivo. Pegue $n \geq 2$. Assuma que propriedade vale para todos até f_n, ou seja

$$f_1 \leq 2^1,\ f_2 \leq 2^2,\ \ldots, f_n \leq 2^n.$$

Queremos provar que vale pra f_{n+1}, ou seja, $f_{n+1} \leq 2^{n+1}$.

Fibonacci: $f_1 = 1$, $f_2 = 1$, e $f_n = f_{n-1} + f_{n-2}$ para todo $n \geq 3$

Proposição

Para todo $n \geq 1$, $f_n \leq 2^n$

Prova: Por indução **forte**
Casos base: $n = 1$ **e** $n = 2$**.** É verdade que $f_1 \leq 2^1$ e $f_2 \leq 2^2$, OK
(veremos depois porque 2 casos base)

Passo Indutivo. Pegue $n \geq 2$. Assuma que propriedade vale para todos até f_n, ou seja

$$f_1 \leq 2^1,\ f_2 \leq 2^2,\ \ldots, f_n \leq 2^n.$$

Queremos provar que vale pra f_{n+1}, ou seja, $f_{n+1} \leq 2^{n+1}$.

Relacionando f_{n+1} (queremos) com os anteriores $f_1, f_2, \ldots, f_n$ (sabemos):

Fibonacci: $f_1 = 1$, $f_2 = 1$, e $f_n = f_{n-1} + f_{n-2}$ para todo $n \geq 3$

Proposição

Para todo $n \geq 1$, $f_n \leq 2^n$

Prova: Por indução **forte**
Casos base: $n = 1$ **e** $n = 2$**.** É verdade que $f_1 \leq 2^1$ e $f_2 \leq 2^2$, OK
(veremos depois porque 2 casos base)

Passo Indutivo. Pegue $n \geq 2$. Assuma que propriedade vale para todos até f_n, ou seja

$$f_1 \leq 2^1,\ f_2 \leq 2^2,\ \ldots, f_n \leq 2^n.$$

Queremos provar que vale pra f_{n+1}, ou seja, $f_{n+1} \leq 2^{n+1}$.

Relacionando f_{n+1} (queremos) com os anteriores $f_1, f_2, \ldots, f_n$ (sabemos):

$$f_{n+1} = f_n + f_{n-1} \overset{sabemos}{\leq} 2^n + 2^{n-1} \leq 2 \cdot 2^n = 2^{n+1}$$

Fibonacci: $f_1 = 1$, $f_2 = 1$, e $f_n = f_{n-1} + f_{n-2}$ para todo $n \geq 3$

Proposição

Para todo $n \geq 1$, $f_n \leq 2^n$

Prova: Por indução **forte**
Casos base: $n = 1$ **e** $n = 2$**.** É verdade que $f_1 \leq 2^1$ e $f_2 \leq 2^2$, OK
(veremos depois porque 2 casos base)

Passo Indutivo. Pegue $n \geq 2$. Assuma que propriedade vale para todos até f_n, ou seja

$$f_1 \leq 2^1,\ f_2 \leq 2^2,\ \ldots, f_n \leq 2^n.$$

Queremos provar que vale pra f_{n+1}, ou seja, $f_{n+1} \leq 2^{n+1}$.

Relacionando f_{n+1} (queremos) com os anteriores $f_1, f_2, \ldots, f_n$ (sabemos):

$$f_{n+1} = f_n + f_{n-1} \overset{sabemos}{\leq} 2^n + 2^{n-1} \leq 2 \cdot 2^n = 2^{n+1}$$

Portanto $f_{n+} \leq 2^{n+1}$ como queríamos

Fibonacci: $f_1 = 1$, $f_2 = 1$, e $f_n = f_{n-1} + f_{n-2}$ para todo $n \geq 3$

Proposição

Para todo $n \geq 1$, $f_n \leq 2^n$

Casos base: $n = 1$ **e** $n = 2$**.** É verdade que $f_1 \leq 2^1$ e $f_2 \leq 2^2$, OK

Passo Indutivo. Pegue $n \geq 2$. Assuma que propriedade vale f_n

$$f_1 \leq 2^1,\ f_2 \leq 2^2,\ \ldots, f_n \leq 2^n.$$

Queremos provar que vale pra f_{n+1}, ou seja, $f_{n+1} \leq 2^{n+1}$.

Relacionando f_{n+1} (queremos) com os anteriores $f_1, f_2, \ldots, f_n$ (sabemos):

$$f_{n+1} = f_n + f_{n-1} \overset{sabemos}{\leq} 2^n + 2^{n-1} \leq 2 \cdot 2^n = 2^{n+1}$$

Pq dois casos base:

Fibonacci: $f_1 = 1$, $f_2 = 1$, e $f_n = f_{n-1} + f_{n-2}$ para todo $n \geq 3$

Proposição

Para todo $n \geq 1$, $f_n \leq 2^n$

Casos base: $n = 1$ **e** $n = 2$**.** É verdade que $f_1 \leq 2^1$ e $f_2 \leq 2^2$, OK

Passo Indutivo. Pegue $n \geq 2$. Assuma que propriedade vale f_n

$$f_1 \leq 2^1,\ f_2 \leq 2^2,\ \ldots, f_n \leq 2^n.$$

Queremos provar que vale pra f_{n+1}, ou seja, $f_{n+1} \leq 2^{n+1}$.

Relacionando f_{n+1} (queremos) com os anteriores $f_1, f_2, \ldots, f_n$ (sabemos):

$$f_{n+1} = f_n + f_{n-1} \overset{sabemos}{\leq} 2^n + 2^{n-1} \leq 2 \cdot 2^n = 2^{n+1}$$

Pq dois casos base: O problema é que tentamos referenciar "2 pra trás" quando usamos $f_{n+1} = f_n + f_{n-1}$, mas isso não é possivel quando $n + 1 = 2$

$$n - 1 = 0 \ \text{ e } \ f_0 \text{ não está definido}$$

Fibonacci: $f_1 = 1$, $f_2 = 1$, e $f_n = f_{n-1} + f_{n-2}$ para todo $n \geq 3$

Proposição

Para todo $n \geq 1$, $f_n \leq 2^n$

Casos base: $n = 1$ **e** $n = 2$**.** É verdade que $f_1 \leq 2^1$ e $f_2 \leq 2^2$, OK

Passo Indutivo. Pegue $n \geq 2$. Assuma que propriedade vale f_n

$$f_1 \leq 2^1,\ f_2 \leq 2^2,\ \ldots, f_n \leq 2^n.$$

Queremos provar que vale pra f_{n+1}, ou seja, $f_{n+1} \leq 2^{n+1}$.

Relacionando f_{n+1} (queremos) com os anteriores $f_1, f_2, \ldots, f_n$ (sabemos):

$$f_{n+1} = f_n + f_{n-1} \overset{sabemos}{\leq} 2^n + 2^{n-1} \leq 2 \cdot 2^n = 2^{n+1}$$

Pq dois casos base: O problema é que tentamos referenciar "2 pra trás" quando usamos $f_{n+1} = f_n + f_{n-1}$, mas isso não é possivel quando $n + 1 = 2$

$$n - 1 = 0 \quad \text{e} \quad f_0 \text{ não está definido}$$

Portanto, temos que provar para f_2 "na mão"

Algoritmo eficiente para exponenciação

Considere o seguinte algoritmo **pow**(x, n) que computa x^n (pra n natural):

Algoritmo **pow**(x, n):

If $n = 1$, **Return** x

If n par
    $temp \leftarrow$ **pow**$(x, \frac{n}{2})$
    Return $temp * temp$
Else if n ímpar
    $temp \leftarrow$ **pow**$(x, \frac{n-1}{2})$
    Return $temp * temp * x$
End if

Proposição

*Seja $T(x, n)$ o número total de multiplicações que o algoritmo **pow**(x, n) faz. Então para todo $n \geq 1$, $T(x, n) \leq 2 \log_2 n$.*

Algoritmo **pow**(x, n):

If $n = 1$, **Return** x

If n par
    $temp \leftarrow$ **pow**$(x, \frac{n}{2})$
    Return $temp * temp$
Else if n ímpar
    $temp \leftarrow$ **pow**$(x, \frac{n-1}{2})$
    Return $temp * temp * x$
End if

Proposição

$T(x, n)$ é número total de multiplicações. Para $n \geq 1$, $T(x, n) \leq 2 \log_2 n$

Prova: Por indução forte em n

```
Algoritmo pow(x, n):

If n = 1, Return x

If n par
        temp ← pow(x, n/2)
        Return temp * temp
Else if n ímpar
        temp ← pow(x, (n−1)/2)
        Return temp * temp * x
End if
```

Proposição

$T(x, n)$ é número total de multiplicações. Para $n \geq 1$, $T(x, n) \leq 2 \log_2 n$

Prova: Por indução forte em n

Caso base: $n = 1$, temos $T(x, 1) = 0$, OK

```
Algoritmo pow(x, n):
If n = 1, Return x
If n par
        temp ← pow(x, n/2)
        Return temp * temp
Else if n ímpar
        temp ← pow(x, (n−1)/2)
        Return temp * temp * x
End if
```

Proposição

$T(x, n)$ é número total de multiplicações. Para $n \geq 1$, $T(x, n) \leq 2 \log_2 n$

Passo Indutivo. Assuma que propriedade vale pra
$T(x, \mathbf{1}), T(x, \mathbf{2}), \ldots, T(x, \mathbf{n})$

Queremos provar q vale pra $T(x, \mathbf{n+1})$, ou seja, $T(x, \mathbf{n+1}) \leq 2 \log_2(n+1)$

Caso 1: $n+1$ é par

$T(x, n+1) =$

```
Algoritmo pow(x, n):
If n = 1, Return x
If n par
        temp ← pow(x, n/2)
        Return temp * temp
Else if n ímpar
        temp ← pow(x, (n−1)/2)
        Return temp * temp * x
End if
```

Proposição

$T(x, n)$ é número total de multiplicações. Para $n \geq 1$, $T(x, n) \leq 2 \log_2 n$

Passo Indutivo. Assuma que propriedade vale pra
$T(x, \mathbf{1}), T(x, \mathbf{2}), \ldots, T(x, \mathbf{n})$

Queremos provar q vale pra $T(x, \mathbf{n+1})$, ou seja, $T(x, \mathbf{n+1}) \leq 2 \log_2(n+1)$

Caso 1: $n+1$ é par

$$T(x, n+1) = T(x, \tfrac{n+1}{2}) + 1$$

```
Algoritmo pow(x, n):
If n = 1, Return x
If n par
        temp ← pow(x, n/2)
        Return temp * temp
Else if n ímpar
        temp ← pow(x, (n−1)/2)
        Return temp * temp * x
End if
```

Proposição

$T(x,n)$ é número total de multiplicações. Para $n \geq 1$, $T(x,n) \leq 2\log_2 n$

Passo Indutivo. Assuma que propriedade vale pra $T(x,\mathbf{1}), T(x,\mathbf{2}), \ldots, T(x,\mathbf{n})$

Queremos provar q vale pra $T(x,\mathbf{n+1})$, ou seja, $T(x,\mathbf{n+1}) \leq 2\log_2(n+1)$

Caso 1: $n+1$ é par

$$T(x,n+1) = T(x,\tfrac{n+1}{2})+1 \overset{sabemos}{\leq} 2\log_2(\tfrac{n+1}{2})+1$$

```
Algoritmo pow(x, n):
If n = 1, Return x
If n par
        temp ← pow(x, n/2)
        Return temp * temp
Else if n ímpar
        temp ← pow(x, (n−1)/2)
        Return temp * temp * x
End if
```

Proposição

$T(x,n)$ é número total de multiplicações. Para $n \geq 1$, $T(x,n) \leq 2\log_2 n$

Passo Indutivo. Assuma que propriedade vale pra $T(x,\mathbf{1}), T(x,\mathbf{2}), \ldots, T(x,\mathbf{n})$

Queremos provar q vale pra $T(x,\mathbf{n+1})$, ou seja, $T(x,\mathbf{n+1}) \leq 2\log_2(n+1)$

Caso 1: $n+1$ é par

$$T(x,n+1) = T(x,\tfrac{n+1}{2})+1 \overset{sabemos}{\leq} 2\log_2(\tfrac{n+1}{2})+1 = 2(\log_2(n+1)-1)+1$$

```
Algoritmo pow(x, n):
If n = 1, Return x
If n par
        temp ← pow(x, n/2)
        Return temp * temp
Else if n ímpar
        temp ← pow(x, (n−1)/2)
        Return temp * temp * x
End if
```

Proposição

$T(x,n)$ é número total de multiplicações. Para $n \geq 1$, $T(x,n) \leq 2\log_2 n$

Passo Indutivo. Assuma que propriedade vale pra $T(x,\mathbf{1}), T(x,\mathbf{2}), \ldots, T(x,\mathbf{n})$

Queremos provar q vale pra $T(x,\mathbf{n+1})$, ou seja, $T(x,\mathbf{n+1}) \leq 2\log_2(n+1)$

Caso 1: $n+1$ é par

$$T(x,n+1) = T(x,\tfrac{n+1}{2})+1 \overset{sabemos}{\leq} 2\log_2(\tfrac{n+1}{2})+1 = 2(\log_2(n+1)-1)+1 \leq 2\log_2(n+1)$$

Caso 2: $n+1$ é impar

$T(x,n+1) =$

```
Algoritmo pow(x, n):
If n = 1, Return x
If n par
        temp ← pow(x, n/2)
        Return temp * temp
Else if n ímpar
        temp ← pow(x, (n−1)/2)
        Return temp * temp * x
End if
```

Proposição

$T(x,n)$ é número total de multiplicações. Para $n \geq 1$, $T(x,n) \leq 2\log_2 n$

Passo Indutivo. Assuma que propriedade vale pra $T(x,\mathbf{1}), T(x,\mathbf{2}), \ldots, T(x,\mathbf{n})$

Queremos provar q vale pra $T(x,\mathbf{n+1})$, ou seja, $T(x,\mathbf{n+1}) \leq 2\log_2(n+1)$

Caso 1: $n+1$ é par

$$T(x,n+1) = T(x,\tfrac{n+1}{2})+1 \overset{sabemos}{\leq} 2\log_2(\tfrac{n+1}{2})+1 = 2(\log_2(n+1)-1)+1 \leq 2\log_2(n+1)$$

Caso 2: $n+1$ é impar

$$T(x,n+1) = T(x,\tfrac{n}{2})+2$$

```
Algoritmo pow(x, n):
If n = 1, Return x
If n par
        temp ← pow(x, n/2)
        Return temp * temp
Else if n ímpar
        temp ← pow(x, (n−1)/2)
        Return temp * temp * x
End if
```

Proposição

$T(x,n)$ é número total de multiplicações. Para $n \geq 1$, $T(x,n) \leq 2\log_2 n$

Passo Indutivo. Assuma que propriedade vale pra
$T(x,\mathbf{1}), T(x,\mathbf{2}), \ldots, T(x,\mathbf{n})$

Queremos provar q vale pra $T(x,\mathbf{n+1})$, ou seja, $T(x,\mathbf{n+1}) \leq 2\log_2(n+1)$

Caso 1: $n+1$ é par

$$T(x,n+1) = T(x,\tfrac{n+1}{2})+1 \overset{sabemos}{\leq} 2\log_2(\tfrac{n+1}{2})+1 = 2(\log_2(n+1)-1)+1 \leq 2\log_2(n+1)$$

Caso 2: $n+1$ é impar

$$T(x,n+1) = T(x,\tfrac{n}{2})+2 \overset{sabemos}{\leq} 2\log_2(\tfrac{n}{2})+2$$

```
Algoritmo pow(x, n):
If n = 1, Return x
If n par
        temp ← pow(x, n/2)
        Return temp * temp
Else if n ímpar
        temp ← pow(x, (n−1)/2)
        Return temp * temp * x
End if
```

Proposição

$T(x,n)$ é número total de multiplicações. Para $n \geq 1$, $T(x,n) \leq 2\log_2 n$

Passo Indutivo. Assuma que propriedade vale pra $T(x,\mathbf{1}), T(x,\mathbf{2}), \ldots, T(x,\mathbf{n})$

Queremos provar q vale pra $T(x,\mathbf{n+1})$, ou seja, $T(x,\mathbf{n+1}) \leq 2\log_2(n+1)$

Caso 1: $n+1$ é par

$$T(x,n+1) = T(x,\tfrac{n+1}{2})+1 \overset{sabemos}{\leq} 2\log_2(\tfrac{n+1}{2})+1 = 2(\log_2(n+1)-1)+1 \leq 2\log_2(n+1)$$

Caso 2: $n+1$ é impar

$$T(x,n+1) = T(x,\tfrac{n}{2})+2 \overset{sabemos}{\leq} 2\log_2(\tfrac{n}{2})+2 = 2(\log_2 n - 1)+2$$

```
Algoritmo pow(x, n):
If n = 1, Return x
If n par
        temp ← pow(x, n/2)
        Return temp * temp
Else if n ímpar
        temp ← pow(x, (n−1)/2)
        Return temp * temp * x
End if
```

Proposição

$T(x,n)$ é número total de multiplicações. Para $n \geq 1$, $T(x,n) \leq 2\log_2 n$

Passo Indutivo. Assuma que propriedade vale pra $T(x,\mathbf{1}), T(x,\mathbf{2}), \ldots, T(x,\mathbf{n})$

Queremos provar q vale pra $T(x,\mathbf{n+1})$, ou seja, $T(x,\mathbf{n+1}) \leq 2\log_2(n+1)$

Caso 1: $n+1$ é par

$$T(x,n+1) = T(x,\tfrac{n+1}{2})+1 \overset{sabemos}{\leq} 2\log_2(\tfrac{n+1}{2})+1 = 2(\log_2(n+1)-1)+1 \leq 2\log_2(n+1)$$

Caso 2: $n+1$ é impar

$$T(x,n+1) = T(x,\tfrac{n}{2})+2 \overset{sabemos}{\leq} 2\log_2(\tfrac{n}{2})+2 = 2(\log_2 n - 1)+2 = 2\log_2 n$$

```
Algoritmo pow(x, n):
If n = 1, Return x
If n par
        temp ← pow(x, n/2)
        Return temp * temp
Else if n ímpar
        temp ← pow(x, (n−1)/2)
        Return temp * temp * x
End if
```

Proposição

$T(x,n)$ é número total de multiplicações. Para $n \geq 1$, $T(x,n) \leq 2\log_2 n$

Passo Indutivo. Assuma que propriedade vale pra $T(x,\mathbf{1}), T(x,\mathbf{2}), \ldots, T(x,\mathbf{n})$

Queremos provar q vale pra $T(x,\mathbf{n+1})$, ou seja, $T(x,\mathbf{n+1}) \leq 2\log_2(n+1)$

Caso 1: $n+1$ é par

$$T(x,n+1) = T(x,\tfrac{n+1}{2})+1 \overset{sabemos}{\leq} 2\log_2(\tfrac{n+1}{2})+1 = 2(\log_2(n+1)-1)+1 \leq 2\log_2(n+1)$$

Caso 2: $n+1$ é impar

$$T(x,n+1) = T(x,\tfrac{n}{2})+2 \overset{sabemos}{\leq} 2\log_2(\tfrac{n}{2})+2 = 2(\log_2 n-1)+2 = 2\log_2 n \leq 2\log_2(n+1)$$

Exercícios

Exercício 1: Considere novamente a sequencia de Fibonacci: $f_1 = 1$, $f_2 = 1$, e $f_n = f_{n-1} + f_{n-2}$ para todo $n \geq 3$

Prove por indução forte que para todo $n \geq 1$

$$f_1 + f_2 + \ldots + f_n = f_{n+2} - 1$$

Exercícios

Exercício 2: Prove por indução **forte** que todo valor maior ou igual a \$8 unidades pode ser pago exatamente utilizando apenas essas notas de \$3 e \$5

Verifique que você provou todos os casos base necessários

Exercicios

Exercício 3: Considere o seguinte procedimento

Prog2(int n)
If n $<$ 3
    Imprima(‘Oi’) 4 vezes
    Return
Else
    Prog2(n-1)
    Prog2(n-2)
End if

Seja $T(n)$ o número de vezes que a palavra “OI” é impressa quando Prog2 é chamado com parâmetro n

Prove por indução (forte) que $T(n) \leq 4 \cdot (7/4)^n$ pra todo $n \geq 1$