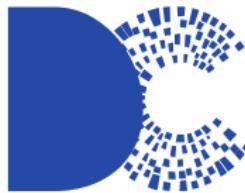




UNIVERSIDADE
FEDERAL DE
SERGIPE



DEPARTAMENTO
DE COMPUTAÇÃO

Força bruta e indução matemática

Projeto e Análise de Algoritmos

Bruno Prado

Departamento de Computação / UFS

Introdução

- ▶ O que é força bruta?
 - ▶ É um método direto para resolução de um problema

Introdução

- ▶ O que é força bruta?
 - ▶ É um método direto para resolução de um problema
 - ▶ Geralmente utiliza as técnicas mais simples

Introdução

- ▶ O que é força bruta?
 - ▶ É um método direto para resolução de um problema
 - ▶ Geralmente utiliza as técnicas mais simples
 - ▶ Baseado na própria definição do problema

Introdução

- ▶ O que é indução matemática?
 - ▶ É um método de prova matemática

Introdução

- ▶ O que é indução matemática?
 - ▶ É um método de prova matemática
 - ▶ Estabelece que uma proposição é verdadeira para o conjunto dos números naturais

Introdução

- ▶ O que é indução matemática?
 - ▶ É um método de prova matemática
 - ▶ Estabelece que uma proposição é verdadeira para o conjunto dos números naturais
 - ▶ Intuição: uma sequência de dominós caindo

Força bruta

- ▶ Problema de avaliação da expressão a^n

- ▶ Definição: $f(a, n) = \overbrace{a \times \dots \times a}^n$
- ▶ Com a constante e $n \in \mathbb{N}$

```
1 // Padrão de tipos por tamanho
2 #include <stdint.h>
3 // Exponenciação por força bruta
4 int64_t exp_fb(int32_t a, uint32_t n) {
5     // r = 1
6     int64_t r = 1;
7     // n iterações
8     for(uint32_t i = 0; i < n; i++)
9         // a * ... * a
10        r = r * a;
11    // r = a^n
12    return r;
13 }
```

Força bruta

- ▶ Problema de avaliação da expressão a^n

- ▶ Definição: $f(a, n) = \overbrace{a \times \dots \times a}^n$
- ▶ Com a constante e $n \in \mathbb{N}$

```
1 // Padrão de tipos por tamanho
2 #include <stdint.h>
3 // Exponenciação por força bruta
4 int64_t exp_fb(int32_t a, uint32_t n) {
5     // r = 1
6     int64_t r = 1;
7     // n iterações
8     for(uint32_t i = 0; i < n; i++)
9         // a * ... * a
10        r = r * a;
11    // r = a^n
12    return r;
13 }
```

Espaço $\Theta(1)$ e tempo $\Theta(n)$

Força bruta

- ▶ Problema de avaliação da expressão a^n
 - ▶ Aplicando a estratégia de exponenciação binária

$$\begin{aligned} \text{▶ Definição: } f(a, n) = & \begin{cases} 1 & n = 0 \\ a & n = 1 \\ a \times f(a^2, \frac{n-1}{2}) & n > 1 \text{ (ímpar)} \\ f(a^2, \frac{n}{2}) & n > 1 \text{ (par)} \end{cases} \\ \text{▶ Com } a \text{ constante e } n \in \mathbb{N} \end{aligned}$$

Força bruta

- ▶ Problema de avaliação da expressão a^n
 - ▶ Aplicando a estratégia de exponenciação binária

```
1 // Padrão de tipos por tamanho
2 #include <stdint.h>
3 // Exponenciação binária recursiva
4 int64_t exp_bin_r(int32_t a, uint32_t n) {
5     // Casos base
6     if(n == 0) return 1;
7     else if(n == 1) return a;
8     // n é ímpar
9     else if(n % 2) return a * exp_bin_r(a * a, (n - 1)
10         / 2);
11     // n é par
12     else return exp_bin_r(a * a, n / 2);
}
```

Força bruta

- ▶ Problema de avaliação da expressão a^n
 - ▶ Aplicando a estratégia de exponenciação binária

```
1 // Padrão de tipos por tamanho
2 #include <stdint.h>
3 // Exponenciação binária recursiva
4 int64_t exp_bin_r(int32_t a, uint32_t n) {
5     // Casos base
6     if(n == 0) return 1;
7     else if(n == 1) return a;
8     // n é ímpar
9     else if(n % 2) return a * exp_bin_r(a * a, (n - 1)
10         / 2);
11     // n é par
12     else return exp_bin_r(a * a, n / 2);
}
```

Espaço e tempo $O(\log_2 n)$

Força bruta

- ▶ Problema de avaliação da expressão a^n
 - ▶ Aplicando a estratégia de exponenciação binária

```
1 // Padrão de tipos por tamanho
2 #include <stdint.h>
3 // Exponenciação binária iterativa
4 int64_t exp_bin_i(int32_t a, uint32_t n) {
5     // r = 1
6     int64_t r = 1;
7     // Iterando enquanto n > 0
8     while(n > 0) {
9         // Ajuste se n for ímpar
10        if(n % 2) { r = r * a; n = n - 1; }
11        //  $a^{2m} = a^m * a^m$ 
12        a = a * a; n = n / 2;
13    }
14    // r =  $a^n$ 
15    return r;
16 }
```

Força bruta

- ▶ Problema de avaliação da expressão a^n
 - ▶ Aplicando a estratégia de exponenciação binária

```
1 // Padrão de tipos por tamanho
2 #include <stdint.h>
3 // Exponenciação binária iterativa
4 int64_t exp_bin_i(int32_t a, uint32_t n) {
5     // r = 1
6     int64_t r = 1;
7     // Iterando enquanto n > 0
8     while(n > 0) {
9         // Ajuste se n for ímpar
10        if(n % 2) { r = r * a; n = n - 1; }
11        // a^2m = a^m * a^m
12        a = a * a; n = n / 2;
13    }
14    // r = a^n
15    return r;
16 }
```

Espaço $\Theta(1)$ e tempo $O(\log_2 n)$

Força bruta

- ▶ Por que e quando usar força bruta?
 - ▶ Soluções fáceis e simples de implementar

Força bruta

- ▶ Por que e quando usar força bruta?
 - ▶ Soluções fáceis e simples de implementar
 - ▶ Conjunto de entrada de tamanho muito pequeno

Força bruta

- ▶ Por que e quando usar força bruta?
 - ▶ Soluções fáceis e simples de implementar
 - ▶ Conjunto de entrada de tamanho muito pequeno
 - ▶ Propósito educacional e entendimento do problema

Indução

- ▶ Estruturação do método de prova
 - ▶ Caso base
 - ▶ Utiliza valores iniciais, geralmente os menores possíveis
 - ▶ Para $n \geq 0$, o caso base é $P(0)$
 - ▶ Demonstra seu funcionamento básico

Indução

- ▶ Estruturação do método de prova
 - ▶ Caso base
 - ▶ Utiliza valores iniciais, geralmente os menores possíveis
 - ▶ Para $n \geq 0$, o caso base é $P(0)$
 - ▶ Demonstra seu funcionamento básico

$$P(n) = 0 + 1 + \dots + n \stackrel{?}{=} \frac{n \times (n + 1)}{2}$$

Indução

- ▶ Estruturação do método de prova
 - ▶ Caso base
 - ▶ Utiliza valores iniciais, geralmente os menores possíveis
 - ▶ Para $n \geq 0$, o caso base é $P(0)$
 - ▶ Demonstra seu funcionamento básico

$$P(n) = 0 + 1 + \dots + n \stackrel{?}{=} \frac{n \times (n + 1)}{2}$$

$$P(0) = 0 \stackrel{?}{=} P(0) = \frac{0 \times (0 + 1)}{2}$$

Indução

- ▶ Estruturação do método de prova
 - ▶ Caso base
 - ▶ Utiliza valores iniciais, geralmente os menores possíveis
 - ▶ Para $n \geq 0$, o caso base é $P(0)$
 - ▶ Demonstra seu funcionamento básico

$$P(n) = 0 + 1 + \dots + n \stackrel{?}{=} \frac{n \times (n + 1)}{2}$$

$$\begin{aligned} P(0) &= 0 \stackrel{?}{=} P(0) = \frac{0 \times (0 + 1)}{2} \\ 0 &\stackrel{V}{=} 0 \end{aligned}$$

Indução

- ▶ Estruturação do método de prova
 - ▶ Passo indutivo
 - ▶ Utiliza-se um valor $k > 0$ arbitrário como hipótese

$$P(n) = 0 + 1 + \dots + n \stackrel{?}{=} \frac{n \times (n + 1)}{2}$$

Indução

- ▶ Estruturação do método de prova
 - ▶ Passo indutivo
 - ▶ Utiliza-se um valor $k > 0$ arbitrário como hipótese

$$P(n) = 0 + 1 + \dots + n \stackrel{?}{=} \frac{n \times (n + 1)}{2}$$

$$P(k) = 0 + 1 + \dots + k \stackrel{?}{=} P(k) = \frac{k \times (k + 1)}{2}$$

Indução

- ▶ Estruturação do método de prova
 - ▶ Passo indutivo
 - ▶ Utiliza-se um valor $k > 0$ arbitrário como hipótese

$$P(n) = 0 + 1 + \dots + n \stackrel{?}{=} \frac{n \times (n + 1)}{2}$$

$$\begin{aligned} P(k) &= 0 + 1 + \dots + k \stackrel{?}{=} P(k) = \frac{k \times (k + 1)}{2} \\ 0 + 1 + \dots + k &\stackrel{?}{=} \frac{k \times (k + 1)}{2} \end{aligned}$$

Indução

- ▶ Estruturação do método de prova
 - ▶ Passo indutivo
 - ▶ Assumimos a hipótese como verdadeira
 - ▶ Aplica-se o valor $k + 1$ confirmar a tese

$$P(k+1) = 0 + 1 + \dots + k + (k+1) \stackrel{?}{=} P(k+1)$$

Indução

- ▶ Estruturação do método de prova
 - ▶ Passo indutivo
 - ▶ Assumimos a hipótese como verdadeira
 - ▶ Aplica-se o valor $k + 1$ confirmar a tese

$$\begin{aligned} P(k+1) &= 0 + 1 + \dots + k + (k+1) & \stackrel{?}{=} P(k+1) \\ P(k) + (k+1) &\stackrel{?}{=} \frac{(k+1)^2 + (k+1)}{2} \end{aligned}$$

Indução

- ▶ Estruturação do método de prova
 - ▶ Passo indutivo
 - ▶ Assumimos a hipótese como verdadeira
 - ▶ Aplica-se o valor $k + 1$ confirmar a tese

$$\begin{aligned} P(k+1) &= 0 + 1 + \dots + k + (k+1) & \stackrel{?}{=} & P(k+1) \\ P(k) + (k+1) & & \stackrel{?}{=} & \frac{(k+1)^2 + (k+1)}{2} \\ \frac{k \times (k+1)}{2} + (k+1) & & \stackrel{?}{=} & \frac{(k+1)^2 + (k+1)}{2} \end{aligned}$$

Indução

- ▶ Estruturação do método de prova
 - ▶ Passo indutivo
 - ▶ Assumimos a hipótese como verdadeira
 - ▶ Aplica-se o valor $k + 1$ confirmar a tese

$$\begin{aligned} P(k+1) = 0 + 1 + \dots + k + (k+1) &\stackrel{?}{=} P(k+1) \\ P(k) + (k+1) &\stackrel{?}{=} \frac{(k+1)^2 + (k+1)}{2} \\ \frac{k \times (k+1)}{2} + (k+1) &\stackrel{?}{=} \frac{(k+1)^2 + (k+1)}{2} \\ \frac{k^2 + k + k + 1 + (k+1)}{2} &\stackrel{?}{=} \frac{(k+1)^2 + (k+1)}{2} \end{aligned}$$

Indução

- ▶ Estruturação do método de prova

- ▶ Passo indutivo

- ▶ Assumimos a hipótese como verdadeira
- ▶ Aplica-se o valor $k + 1$ confirmar a tese

$$P(k+1) = 0 + 1 + \dots + k + (k+1) \stackrel{?}{=} P(k+1)$$

$$P(k) + (k+1) \stackrel{?}{=} \frac{(k+1)^2 + (k+1)}{2}$$

$$\frac{k \times (k+1)}{2} + (k+1) \stackrel{?}{=} \frac{(k+1)^2 + (k+1)}{2}$$

$$\frac{k^2 + k + k + 1 + (k+1)}{2} \stackrel{?}{=} \frac{(k+1)^2 + (k+1)}{2}$$

$$\frac{(k+1)^2 + (k+1)}{2} \stackrel{V}{=} \frac{(k+1)^2 + (k+1)}{2}$$

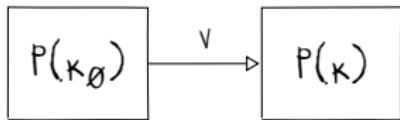
Indução

- ▶ Estruturação do método de prova
 - ▶ Caso base: $P(k_0)$
 - ▶ Hipótese: $P(k)$
 - ▶ Tese: $P(k + 1)$

$$\boxed{P(k_0)}$$

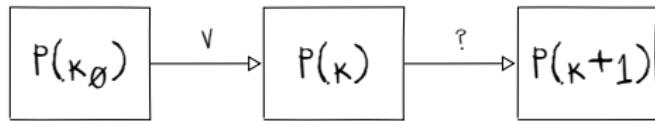
Indução

- ▶ Estruturação do método de prova
 - ▶ Caso base: $P(k_0)$
 - ▶ Hipótese: $P(k)$
 - ▶ Tese: $P(k + 1)$



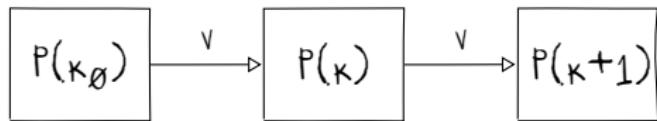
Indução

- ▶ Estruturação do método de prova
 - ▶ Caso base: $P(k_0)$
 - ▶ Hipótese: $P(k)$
 - ▶ Tese: $P(k + 1)$



Indução

- ▶ Estruturação do método de prova
 - ▶ Caso base: $P(k_0)$
 - ▶ Hipótese: $P(k)$
 - ▶ Tese: $P(k + 1)$



$\forall n \geq k_0$ a proposição $P(k)$ é verdadeira

Indução

- ▶ Definições de prova por indução
 - ▶ Fraca
 - ▶ Somente o passo k é considerado na hipótese
 - ▶ $P(k_0) \wedge \forall k \geq k_0 (P(k) \rightarrow P(k + 1))$
 - ▶ É preciso demonstrar que o caso base $P(k_0)$ e a hipótese indutiva $P(k) \rightarrow P(k + 1)$ são verdadeiras

Indução

- ▶ Definições de prova por indução
 - ▶ Fraca
 - ▶ Somente o passo k é considerado na hipótese
 - ▶ $P(k_0) \wedge \forall k \geq k_0 (P(k) \rightarrow P(k + 1))$
 - ▶ É preciso demonstrar que o caso base $P(k_0)$ e a hipótese indutiva $P(k) \rightarrow P(k + 1)$ são verdadeiras
 - ▶ Forte
 - ▶ Os passos anteriores até k são utilizados na hipótese
 - ▶ $P(k_0) \wedge P(k_1) \wedge \dots \wedge P(k) \rightarrow P(k + 1)$
 - ▶ Além do caso base, todas as proposições menores ou iguais à k são verdadeiras

Indução

► Definições de prova por indução

► Fraca

- Somente o passo k é considerado na hipótese
- $P(k_0) \wedge \forall k \geq k_0 (P(k) \rightarrow P(k+1))$
- É preciso demonstrar que o caso base $P(k_0)$ e a hipótese indutiva $P(k) \rightarrow P(k+1)$ são verdadeiras

► Forte

- Os passos anteriores até k são utilizados na hipótese
- $P(k_0) \wedge P(k_1) \wedge \dots \wedge P(k) \rightarrow P(k+1)$
- Além do caso base, todas as proposições menores ou iguais à k são verdadeiras

Ambas as definições são logicamente equivalentes

Exemplo

- ▶ Prove que $\forall n \geq 0 \rightarrow 3^n > n^2$ é verdadeira
 - ▶ Caso base
 - ▶ $n = 0$ temos $3^0 > 0^2 \rightarrow 1 > 0$ é verdadeira
 - ▶ $n = 1$ temos $3^1 > 1^2 \rightarrow 3 > 1$ é verdadeira

Exemplo

- ▶ Prove que $\forall n \geq 0 \rightarrow 3^n > n^2$ é verdadeira
 - ▶ Caso base
 - ▶ $n = 0$ temos $3^0 > 0^2 \rightarrow 1 > 0$ é verdadeira
 - ▶ $n = 1$ temos $3^1 > 1^2 \rightarrow 3 > 1$ é verdadeira
 - ▶ Passo indutivo
 - ▶ Para um valor de $k > 1$ é obtida a hipótese $3^k > k^2$
 - ▶ Assume-se que a hipótese é verdadeira
 - ▶ Demonstrar que a tese $3^{k+1} > (k + 1)^2$ é verdadeira

Exemplo

- ▶ Prove que $\forall n \geq 0 \rightarrow 3^n > n^2$ é verdadeira
 - ▶ Hipótese: $3^k > k^2$
 - ▶ Tese: $3^{k+1} > (k + 1)^2$
 - ▶ Multiplicando ambos os lados da hipótese por 3

$$\begin{aligned}3^k &> k^2 \\3 \times 3^k &> 3k^2 \\3^{k+1} &> 3k^2\end{aligned}$$

Exemplo

- ▶ Prove que $\forall n \geq 0 \rightarrow 3^n > n^2$ é verdadeira
 - ▶ Se $k > 1$, temos que
$$3k^2 = k^2 + k^2 + k^2 > k^2 + 2k + 1 = (k + 1)^2$$

Exemplo

- ▶ Prove que $\forall n \geq 0 \rightarrow 3^n > n^2$ é verdadeira
 - ▶ Se $k > 1$, temos que
$$3k^2 = k^2 + k^2 + k^2 > k^2 + 2k + 1 = (k + 1)^2$$
 - ▶ Pela propriedade de transitividade
$$3^{k+1} > 3k^2 > (k + 1)^2 \rightarrow 3k^2 > (k + 1)^2$$

Exemplo

- ▶ Prove que $\forall n \geq 0 \rightarrow 3^n > n^2$ é verdadeira
 - ▶ Se $k > 1$, temos que
$$3k^2 = k^2 + k^2 + k^2 > k^2 + 2k + 1 = (k + 1)^2$$
 - ▶ Pela propriedade de transitividade
$$3^{k+1} > 3k^2 > (k + 1)^2 \rightarrow 3k^2 > (k + 1)^2$$
 - ▶ Precisa demonstrar que para $n > 1$ que a proposição
$$3n^2 > (n + 1)^2$$
 é verdadeira

Exemplo

- ▶ Prove que $\forall n \geq 1 \rightarrow 3n^2 > (n + 1)^2$ é verdadeira

- ▶ Caso base: $n = 2 \rightarrow 3 \times 2^2 \stackrel{?}{>} (2 + 1)^2 \rightarrow 12 \stackrel{v}{>} 9$

- ▶ Hipótese: $\forall k > 2 \rightarrow 3k^2 > (k + 1)^2$

- ▶ Tese: $3(k + 1)^2 > [(k + 1) + 1]^2$

$$3(k + 1)^2 \stackrel{?}{>} [(k + 1) + 1]^2$$

$$3k^2 + 6k + 3 \stackrel{?}{>} k^2 + 4k + 4$$

$$3k^2 + 6k + 3 \stackrel{?}{>} (k + 1)^2 + 2k + 3$$

Exemplo

- ▶ Prove que $\forall n \geq 1 \rightarrow 3n^2 > (n + 1)^2$ é verdadeira

- ▶ Caso base: $n = 2 \rightarrow 3 \times 2^2 \stackrel{?}{>} (2 + 1)^2 \rightarrow 12 \stackrel{v}{>} 9$

- ▶ Hipótese: $\forall k > 2 \rightarrow 3k^2 > (k + 1)^2$

- ▶ Tese: $3(k + 1)^2 > [(k + 1) + 1]^2$

$$3(k + 1)^2 \stackrel{?}{>} [(k + 1) + 1]^2$$

$$3k^2 + 6k + 3 \stackrel{?}{>} k^2 + 4k + 4$$

$$3k^2 + 6k + 3 \stackrel{?}{>} (k + 1)^2 + 2k + 3$$

Assumindo a hipótese $3k^2 > (k + 1)^2$, é preciso demonstrar
que $6k + 3 \geq 2k + 3$

$$6k + 3 \stackrel{?}{\geq} 2k + 3$$

Exemplo

► Prove que $\forall n \geq 1 \rightarrow 3n^2 > (n + 1)^2$ é verdadeira

► Caso base: $n = 2 \rightarrow 3 \times 2^2 > (2 + 1)^2 \rightarrow 12 > 9$

► Hipótese: $\forall k > 2 \rightarrow 3k^2 > (k + 1)^2$

► Tese: $3(k + 1)^2 > [(k + 1) + 1]^2$

$$3(k + 1)^2 \stackrel{?}{>} [(k + 1) + 1]^2$$

$$3k^2 + 6k + 3 \stackrel{?}{>} k^2 + 4k + 4$$

$$3k^2 + 6k + 3 \stackrel{?}{>} (k + 1)^2 + 2k + 3$$

Assumindo a hipótese $3k^2 > (k + 1)^2$, é preciso demonstrar
que $6k + 3 \geq 2k + 3$

$$6k + 3 \stackrel{?}{\geq} 2k + 3$$

$$3k \stackrel{V}{\geq} k$$

Exemplo

- ▶ Prove que $\forall n \geq 0 \rightarrow 3^n > n^2$ é verdadeira
 - ▶ Foi demonstrado que $3n^2 > (n + 1)^2$ para $n > 1$

Exemplo

- ▶ Prove que $\forall n \geq 0 \rightarrow 3^n > n^2$ é verdadeira
 - ▶ Foi demonstrado que $3n^2 > (n + 1)^2$ para $n > 1$
 - ▶ Com a hipótese $3^k > k^2 \rightarrow 3^{k+1} > 3k^2$ e aplicando a propriedade de transitividade
 $3^{k+1} > 3k^2 > (k + 1)^2 \rightarrow 3^{k+1} > (k + 1)^2$ para $n > 1$

Exemplo

- ▶ Prove que $\forall n \geq 0 \rightarrow 3^n > n^2$ é verdadeira
 - ▶ Foi demonstrado que $3n^2 > (n + 1)^2$ para $n > 1$
 - ▶ Com a hipótese $3^k > k^2 \rightarrow 3^{k+1} > 3k^2$ e aplicando a propriedade de transitividade
 $3^{k+1} > 3k^2 > (k + 1)^2 \rightarrow 3^{k+1} > (k + 1)^2$ para $n > 1$

A proposição $\forall n \geq 0 \rightarrow 3^n > n^2$ é verdadeira

Exemplo

- ▶ Prove por indução a corretude da implementação recursiva do algoritmo de busca binária

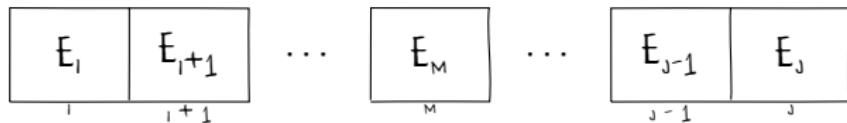
```
1 // Padrão de tipos por tamanho
2 #include <stdint.h>
3 // Busca binária recursiva
4 int32_t busca(int32_t x, int32_t* V, int32_t i, int32_t
   j) {
5     // Pivô
6     int32_t m = (i + j) / 2;
7     // Caso base 1
8     if(i > j) return -1;
9     // Caso base 2
10    else if(V[m] == x) return m;
11    // Metade inferior
12    else if(V[m] > x) return busca(x, V, i, m - 1);
13    // Metade superior
14    else return busca(x, V, m + 1, j);
15 }
```

Exemplo

- ▶ Prove por indução a corretude da implementação recursiva do algoritmo de busca binária
 - ▶ Casos base
 - ▶ $P(x, V, i, j) = -1$ se $i > j$
 - ▶ $P(x, V, i, j) = m$ se $i = j$ e $V[m = \lfloor (i + j) \div 2 \rfloor] = x$, onde m é o índice do elemento x

Exemplo

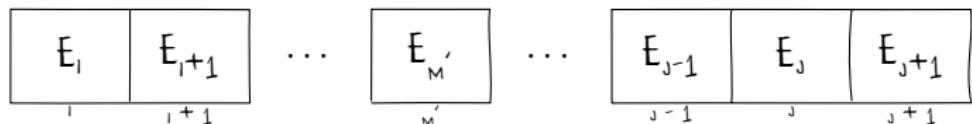
- ▶ Prove por indução a corretude da implementação recursiva do algoritmo de busca binária
 - ▶ Casos base
 - ▶ $P(x, V, i, j) = -1$ se $i > j$
 - ▶ $P(x, V, i, j) = m$ se $i = j$ e $V[m] = \lfloor (i + j) \div 2 \rfloor = x$, onde m é o índice do elemento x
 - ▶ Hipótese



- ▶ $V[m] = x$, é retornado seu índice $m = \lfloor (i + j) \div 2 \rfloor$
- ▶ $V[m] > x$, x pode estar em $E_i \leq \dots \leq E_{m-1}$
- ▶ $V[m] < x$, x pode estar em $E_{m+1} \leq \dots \leq E_j$

Exemplo

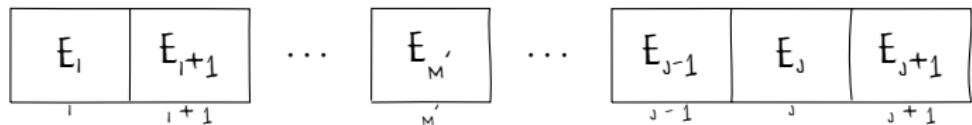
- ▶ Prove por indução a corretude da implementação recursiva do algoritmo de busca binária
 - ▶ Tese



Exemplo

- ▶ Prove por indução a corretude da implementação recursiva do algoritmo de busca binária

- ▶ Tese

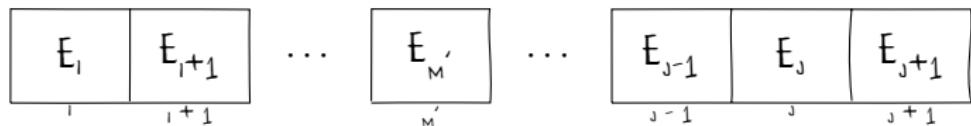


$$V[m'] = x \rightarrow V\left[\left\lfloor \frac{(i+j+1)}{2} \right\rfloor\right] = V\left[\left\lfloor m + \frac{1}{2} \right\rfloor\right] = x$$

Exemplo

- ▶ Prove por indução a corretude da implementação recursiva do algoritmo de busca binária

- ▶ Tese

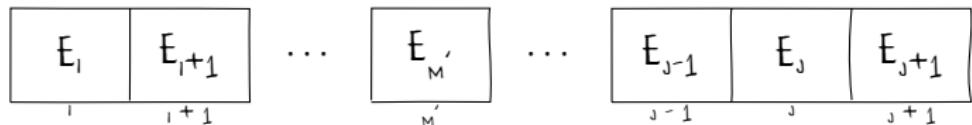


$$V[m'] > x \rightarrow E_l \leq \dots \leq E_{m'-1}, \text{ onde } E_{m'-1} < E_{m'}$$

Exemplo

- ▶ Prove por indução a corretude da implementação recursiva do algoritmo de busca binária

- ▶ Tese



$V[m'] < x \rightarrow E_{m'+1} \leq \dots \leq E_{j+1}$, onde $E_{m'+1} > E_{m'}$

Exercícios

- ▶ Prove por indução matemática que as proposições são verdadeiras, detalhando os casos bases e os passos indutivos
 - ▶ $1^2 + 2^2 + \dots + n^2 = \frac{n(n+1)(2n+1)}{6}$
 - ▶ $2^n < n!$, $n \geq 4$
 - ▶ $n^3 + 2n$ é divisível por 3