Correção de Algoritmos

Objetivos

- Aprender a demonstrar a correção de um algoritmo iterativo (ñ recursivo)
- Aprender o que é invariante de laço e como demonstrá-lo

Correção de Algoritmos

- Um algoritmo está <u>correto</u> qndo devolve uma resposta correta para qualquer instância válida do problema
  - Ele está <u>incorreto</u> quando devolve uma resposta errada para <u>alguma</u> instância
- Como saber se o seu algoritmo está correto?

  R: demostrando!

Correção de Algoritmos Iterativos

→ Algoritmos iterativos possuem laços de repetição

Def. Uma *inveriante de laço* é uma afirmação (predicado) que é verdadeira no início de qualquer iteração do laço.

- No geral, formalizamos como "antes da t-ésima iteração começar, vale que ..." e a afirmação envolve variáveis importantes p/ o laço.

  - Notação: $P(t)$, $Q(t)$, $R(t)$

- São úteis quando nos permitem concluir algo importante após o término do laço.

Como provar que uma frase é uma invariante?

- Se $P(t)$ é a invariante, basta provar que $P(1), P(2), P(3), \dots P(k), P(k+1)$ são válidas, onde k é o número de iterações realizadas
- $P(k)$ diz que a invariante é válida no início da última iteração, $P(k+1)$ nos diz que é válida qndo o laço termina.
- Como demonstrar isso?

Como provar que uma frase é uma invariante?

- Se $P(t)$ é a invariante, basta provar que $P(1), P(2), P(3), \ldots P(k), P(k+1)$ são válidas, onde k é o número de iterações realizadas

- $P(k)$ diz que a invariante é válida no início da última iteração, $P(k+1)$ nos diz que é válida qndo o laço termina.

- Como demonstrar isso?

  - Indução!

1) Prove $P(1)$
2) Prove que $P(t) \Rightarrow P(t+1)\ \forall\ t \geq 1$

Assuma que a invariante é verdade no início da t-ésima iteração e conclua que ela continua válida no início da $(t+1)$-ésima

Como provar que uma frase é uma invariante?

- Se $P(t)$ é a invariante, basta provar que $P(1), P(2), P(3), \ldots$ $P(k), P(k+1)$ são válidas, onde k é o número de iterações realizadas

- $P(k)$ diz que a invariante é válida no início da última iteração, $P(k+1)$ nos diz que é válida qndo o laço termina.

- Como demonstrar isso?

Não será necessário saber quem é k

- Indução!

1) Prove $P(1)$

2) Prove que $P(t) \Rightarrow P(t+1) \; \forall \, t \geq 1$

Assuma que a invariante é verdade no início da t-ésima iteração e conclua que ela continua válida no início da $(t+1)$-ésima

Um problema Simples

Problema: Busca em Dados Ordenados

Entrada: (A, n, k), onde $A[1..n]$ é um vetor contendo n números inteiros em ordem crescente, i.e., $A[i] < A[i+1]$ para todo $1 \leq i < n$, e $k \in \mathbb{N}$

Saída: Posição i da primeira ocorrência de k em A, i.e., $A[i] = k$, ou -1, caso $k \notin A$

$A =$

1	2	3	4	5	6	7	8	9	10
-6	-1	3	7	10	27	35	37	52	100

Busca Linear

```
BuscaLinear (A, n, k)
    i = 1
    Enquanto  i ≤ n  e  A[i] ≠ k
        i = i + 1
    Se  i ≤ n
        devolve  i
    devolve -1
```

Invariante ???

Busca Linear

```
BuscaLinear (A, n, k)
    i = 1
    Enquanto  i ≤ n  e  A[i] ≠ k
        i = i + 1
    Se  i ≤ n
        devolve  i
    devolve  -1
```

Invariante ???

$P(t)$ = "antes da t-ésima iteração começar, $i = t$"

Busca Linear

```
BuscaLinear (A, n, k)
    i = 1
    Enquanto i ≤ n e A[i] ≠ k
        i = i + 1
    Se i ≤ n
        devolve i
    devolve -1
```

Invariante ???

$P(t)$ = "antes da t-ésima iteração começar, $i = t$"

Verdade! Mas não me ajuda a argumentar que BuscaLinear está correto

Busca Linear

```
BuscaLinear (A, n, k)
    i = 1
    Enquanto i ≤ n e A[i] ≠ k
        i = i + 1
    Se i ≤ n
        devolve i
    devolve -1
```

Invariante ???

$Q(t)$ = "antes da t-ésima iteração começar, $k \notin A[1..i-1]$"

Busca Linear

```
BuscaLinear (A, n, k)
    i = 1
    Enquanto i ≤ n e A[i] ≠ k
        i = i + 1
    Se i ≤ n
        devolve i
    devolve -1
```

Invariante ???

$Q(t)$ = " antes da t-ésima iteração começar, $k \notin A[1..i-1]$ "

↳ Verdade! Útil p/ argumentar sobre a correção do algoritmo

Busca Linear

```
BuscaLinear (A, n, k)
    i = 1
    Enquanto i ≤ n e A[i] ≠ k
        i = i + 1
    Se i ≤ n
        devolve i
    devolve -1
```

Consigo garantir que o laço termina?

Invariante ???

$Q(t)$ = "antes da t-ésima iteração começar, $k \notin A[1..i-1]$"

↳ Verdade! Útil p/ argumentar sobre a correção do algoritmo

Busca Linear

```
BuscaLinear (A, n, k)
1    i = 1
2    Enquanto i ≤ n e A[i] ≠ k
3        i = i + 1
4    Se i ≤ n
5        devolve i
6    devolve -1
```

Invariante ???

útil p/ argumentar sobre o valor retornado

$R(t)$ = " antes da t-ésima iteração começar, $i = t$ e $k \notin A[1..i-1]$ "

laço termina: Qdo $t = n+1 \Rightarrow i = n+1$ e o condicional da linha 2 dá falso!

Teo. O Algoritmo Buscalinear resolve o problema Busca em Dados Ordenados.

Teo. O Algoritmo BuscaLinear resolve o problema Busca em Dados Ordenados.

Demonstração.

- O algoritmo inicializa a variável $i=1$ (linha 1)
- Na sequência, o algoritmo executa o laço da linha 2. Pela invariante $R(t)$, sabemos que o laço executa no máximo n vezes, pois $R(n+1)$ implica que $i=n+1$, o que faz com que a condicional do laço seja Falsa

```
BuscaLinear (A, n, k)
1   i = 1
2   Enquanto i ≤ n
          e A[i] ≠ k
3       i = i + 1
4   Se i ≤ n
5       devolve i
6   devolve -1
```

- Quando o laço da linha 2 terminar, o condicional da linha 4 é executado. Se $i \leq n$, isso implica que o laço de linha 2 falhou devido ao fato de $A[i]=k$. O algoritmo prossegue retornando i corretamente

Teo. O Algoritmo Buscalinear resolve o problema Busca em Dados Ordenados.

Demonstração.

- Caso $i > n$, o teste da linha 4 falha e o algoritmo retorna -1
- Note que se $i > n$, temos que $i = n+1$
- Pela invariante $R(n+1)$, sabemos que $k \notin [1..n+1-1] = [1..n]$
- Assim, o algoritmo retornou -1 corretamente

$\square$

```
BuscaLinear (A, n, k)
1   i = 1
2   Enquanto i ≤ n
            e A[i] ≠ k
3       i = i + 1
4   Se i ≤ n
5       devolve i
6   devolve -1
```

→ Na demonstração anterior, usamos a invariante sem demonstrá-la. Para nossa demonstração estar correta, precisamos demonstrar que a invariante está correta!

$R(t) =$ "antes da t-ésima iteração começar,

$i = t$ e $k \in A[1..i-1]$ "

→ Na demonstração anterior, usamos a invariante sem demonstrá-la. Para nossa demonstração estar correta, precisamos demonstrar que a invariante está correta!

$R(t)$ = " antes da t-ésima iteração começar,

$i = t$ e $k \in A[1..\cancel{t}-1]$ "

① propague a variável livre sempre que for possível

② Ela é fixa dentro da iteração

$R(t) =$ " antes da t-ésima iteração começar, $i = t$ e $k \in A[1..t-1]$ "

Base $(t=1)$

Antes da primeira iteração começar, temos que $i=1$ (linha 1) e $k \in A[1..t-1] = A[1..0] = \emptyset$. Assim, temos que abase é válida.

Passo $[P(t) \Rightarrow P(t+1)]$

Suponha que $P(t)$ seja verdadeiro, ou seja, antes do início da t-ésima iteração, temos que $i = t$ e $k \notin A[1..t-1]$. Assumindo que a iteração t acontece, temos que o teste da linha 2 da verdadeiro, consequentemente

$$t = i \leq n \quad e \quad k \neq A[i] = A[t]$$

Como $k \notin A[1..t-1]$ e $k \neq A[t]$, temos que $k \notin A[1..t]$. Escrevendo de outra forma

$$k \notin A[1..(t+1)-1] \quad *$$

- Na sequência o laço executa a linha 3, fazendo

$$i = t + 1 \qquad (**)$$

- Após executar a linha 3, a t-ésima iteração finaliza
- Por $(*)$ e $(**)$, temos que $P(t+1)$ é válida

$\square$

```
BuscaLinear(A, n, k)
1   i = 1
2   Enquanto i ≤ n
          e A[i] ≠ k
3       i = i + 1
4   Se i ≤ n
5       devolve i
6   devolve -1
```

Estratégia Usada Pelo Maycon

- Pense na "melhor" invariante $P(t)$ possível
- Prove a correção do algoritmo usando $P(t)$
- Prove a correção de $P(t)$

Busca Binária

```
BuscaBinária (A, n, k)
    esq = 1
    dir = n
    Enquanto esq < dir
        meio = ⌊(esq + dir)/2⌋
        se k > A[meio]
            esq = meio + 1
        Senão
            dir = meio
    Se A[esq] == k
        devolve esq
    devolve -1
```

Busca Binária

```
BuscaBinária (A, n, k)
    esq = 1
    dir = n
    Enquanto esq < dir
        meio = ⌊(esq + dir)/2⌋
        se k > A[meio]
            esq = meio + 1
        senão
            dir = meio
    Se A[esq] == k
        devolve esq
    devolve -1
```

$P(t)$ = "antes da t-ésima iteração, vale que

$|A[esq..dir]| \leq \left\lceil \frac{n}{2^{t-1}} \right\rceil$

$1 \leq esq \leq dir \leq n$

$k \notin A[1..esq-1]$

$k \notin A[dir+1..n]$ "

Teo. O Algoritmo Busca Binaria resolve o problema Busca em Dados Ordenados

Demo.

- O algoritmo inicializa as variáveis $esq = 1$ (linha 1) e $dir = n$ (linha 2)
- Na sequência, ele executa o laço da linha 3. Pela invariante $P(t)$, temos que o laço encerra-se em no máximo $\lceil \lg n \rceil + 1$ iterações, pois qndo $t = \lceil \lg n \rceil + 1$ temos

$$\frac{n}{2^{t-1}} = \frac{n}{2^{\lceil \lg n \rceil + 1 - 1}} = \frac{n}{2^{\lceil \lg n \rceil}}$$

$$\leq \frac{n}{2^{\lg n}} = 1$$

Assim, a invariante $P(t)$ diz que

$$|A[esq..dir]| = dir - esq + 1 \leq \left\lceil \frac{n}{2^{t-1}} \right\rceil$$

$$\leq 1 \Rightarrow dir - esq \leq 0$$

```
BuscaBinária (A, n, k)
1   esq = 1
2   dir = n
3   Enquanto esq < dir
4       meio = ⌊(esq + dir)/2⌋
5       se k > A[meio]
6           esq = meio + 1
7       senão
8           dir = meio
9   Se A[esq] == k
10      devolve esq
11  devolve -1
```

$P(t) =$ "antes da t-ésima iteração, vale que

$|A[esq..dir]| \leq \left\lceil \frac{n}{2^{t-1}} \right\rceil$

$1 \leq esq \leq dir \leq n$

$k \notin A[1..esq-1]$

$k \notin A[dir+1..n]$ "

Teo. O Algoritmo BuscaBinaria resolve o problema Busca em Dados Ordenados.

Demo.

- Quando o laço é encerrado, temos que $P(\ell)$ é verdade, onde $\ell-1$ é o número de iteraçõs em que o corpo do laço foi executado
- Como $P(k)$ vale, temos

$$|A[esq..dir]| \leq \left\lceil \frac{n}{2^{\ell-1}} \right\rceil$$
$$1 \leq esq \leq dir \leq n$$
$$k \notin A[1..esq-1]$$
$$k \notin A[dir+1..n]$$

Ⓐ

- Como o laço terminou, temos $esq \geq dir$. Juntando com Ⓐ, concluímos que

$$esq = dir$$

Ⓑ

```
BuscaBinária (A, n, k)
   esq = 1
   dir = n
   Enquanto esq < dir
       meio = ⌊(esq + dir)/2⌋
       se k > A[meio]
           esq = meio + 1
       Senão
           dir = meio
   Se A[esq] == k
       devolve esq
   devolve -1
```

$P(t)$ = "antes da t-ésima iteração, vale que

$$|A[esq..dir]| \leq \left\lceil \frac{n}{2^{t-1}} \right\rceil$$
$$1 \leq esq \leq dir \leq n$$
$$k \notin A[1..esq-1]$$
$$k \notin A[dir+1..n]$$

"

Teo. O Algoritmo BuscaBinaria resolve o problema Busca em Dados Ordenados

Demo.

- Se o teste da linha 9 da verdadeiro, então $A[esq] == k$ e o algoritmo retorna k corretamente (linha 10)
- Se o teste da linha 9 da falso, então

$$A[esq] \neq k \qquad \text{C}$$

- Juntando A e B temos

$$k \notin A[1..esq-1] \qquad \text{D}$$
$$k \notin A[esq+1..n]$$

- Juntando C e D, temos

$$k \notin A[1..n]$$

- Assim, o algoritmo devolve -1 corretamente na linha 11 $\square$

BuscaBinária (A, n, k)

```
1   esq = 1
2   dir = n
3   Enquanto esq < dir
4       meio = ⌊(esq + dir)/2⌋
5       se k > A[meio]
6           esq = meio + 1
7       Senão
8           dir = meio
9   Se A[esq] == k
10      devolve esq
11  devolve -1
```

$P(t)$ = "antes da t-ésima iteração, vale que

$$|A[esq..dir]| \leq \left\lceil \frac{n}{2^{t-1}} \right\rceil$$
$$1 \leq esq \leq dir \leq n$$
$$k \notin A[1..esq-1]$$
$$k \notin A[dir+1..n]$$
"

Agora, vamos provar a invariante da busca binária

$P(t) =$ "antes da t-ésima iteração, vale que

$$|A[esq..dir]| \leq \left\lceil \frac{n}{2^{t-1}} \right\rceil$$

$$1 \leq esq \leq dir \leq n$$

$$k \notin A[1..esq-1]$$

$$k \notin A[dir+1..n]$$ "

Demo.

- Prova por indução em t.

Base $(t=1)$

Antes da primeira iteração começar, temos que $esq=1$ e $dir=n$.

Assim
$$n = |A[esq..dir]| \leq \left\lceil \frac{n}{2^0} \right\rceil = n$$

$$1 \leq esq = 1 \leq n = dir \leq n$$

$$k \notin A[1..esq-1] = A[1..0] = \emptyset$$

$$k \notin A[dir+1..n] = A[n+1..n] = \emptyset$$

Passo $[P(t) \Rightarrow P(t+1)]$

- Sejam e e d os valores das variáveis esq e dir, respectivamente, antes do início da t-ésima iteração.
- Suponha que $P(t)$ seja verdade. Assim

$$1 \leq e \leq d \leq n \quad (1)$$

$$k \notin A[1..e-1] \quad (2)$$

$$k \notin A[d+1..n] \quad (3)$$

$$|A[e..d]| \leq \left\lceil \frac{n}{2^{t-1}} \right\rceil \quad (4)$$

- Assumindo que a t-ésima iteração ocorre, temos que o teste da linha 3 falha e consequentemente

$$e < d \quad (5)$$

- Na linha 4, fazemos

$$meio = \lfloor (e+d)/2 \rfloor \quad (6)$$

```
BuscaBinária (A, n, k)
1   esq = 1
2   dir = n
3   Enquanto esq < dir
4       meio = ⌊(esq + dir)/2⌋
5       se k > A[meio]
6           esq = meio + 1
7       Senão
8           dir = meio
9   Se A[esq] == k
10      devolve esq
11  devolve -1
```

$P(t)$ = "antes da t-ésima iteração, vale que

$$|A[esq..dir]| \leq \left\lceil \frac{n}{2^{t-1}} \right\rceil$$

$$1 \leq esq \leq dir \leq n$$

$$k \notin A[1..esq-1]$$

$$k \notin A[dir+1..n]$$ "

• Note que $meio = \lfloor (e+d)/2 \rfloor \le \frac{e+d}{2} < \frac{2d}{2} = d$ e

$\lfloor \frac{x}{2} \rfloor \geq \frac{x-1}{2}$

$$meio = \lfloor (e+d)/2 \rfloor \geq \frac{e+d-1}{2} \geq \frac{2e-1}{2} = e - \frac{1}{2}$$

• Assim $e \le meio < d$ (7)

• Na sequência, o algoritmo executa o teste da linha 5 e termina a iteração na linha 6 ou 8 dependendo do resultado do teste. O restante da prova é dividido em dois casos, a depender do fato de $k > A[meio]$ ou não

```
BuscaBinária (A, n, k)
1  esq = 1
2  dir = n
3  Enquanto esq < dir
4      meio = ⌊(esq + dir)/2⌋
5      se k > A[meio]
6          esq = meio + 1
7      senão
8          dir = meio
9  Se A[esq] == k
10     devolve esq
11 devolve -1
```

$P(t)$ = "antes da t-ésima iteração, vale que

$$|A[esq..dir]| \le \lceil \frac{n}{2^{t-1}} \rceil$$

$$1 \le esq \le dir \le n$$

$$k \notin A[1..esq-1]$$

$$k \notin A[dir+1..n]$$ "

- Caso 1: $k > A[meio]$

- Então o teste da linha 5 dá verdadeiro, o algoritmo executa a linha 6, fazendo $esq = meio + 1 = \lfloor (e+d)/2 \rfloor + 1$ e termina a iteração

- Note que

$$|A[esq..dir]| = |A[\lfloor (e+d)/2 \rfloor + 1 .. d]|$$

$$= d - \left(\left\lfloor \frac{(e+d)}{2} \right\rfloor + 1 \right) + 1$$

$$= d - \left\lfloor \frac{(e+d)}{2} \right\rfloor \leqslant d - \frac{e+d-1}{2}$$

$$= \frac{d-e-1}{2} < \frac{\boxed{d-e+1}}{2} \quad |A[e..d]|$$

Por (4)

$$\leqslant \frac{\left\lceil \frac{n}{2^{t-1}} \right\rceil}{2} \leqslant \left\lceil \frac{\left\lceil \frac{n}{2^{t-1}} \right\rceil}{2} \right\rceil = \left\lceil \frac{n}{2^t} \right\rceil$$

$$= \left\lceil \frac{n}{2^{(t+1)-1}} \right\rceil$$

```
BuscaBinária (A, n, k)
1  esq = 1
2  dir = n
3  Enquanto esq < dir
4      meio = ⌊(esq + dir)/2⌋
5      se k > A[meio]
6          esq = meio + 1
7      Senão
8          dir = meio
9  Se A[esq] == k
10     devolve esq
11 devolve -1
```

$P(t+1)$ = "antes da $(t+1)$-ésima iteração, vale que

$|A[esq..dir]| \leqslant \left\lceil \frac{n}{2^{t+1-1}} \right\rceil$ ✓

$1 \leqslant esq \leqslant dir \leqslant n$

$k \notin A[1..esq-1]$

$k \notin A[dir+1..n]$"

- Como $esq = meio + 1$, por ① e ⑦, temos

$$2 \overset{①}{\leq} e + 1 \overset{⑦}{\leq} meio + 1 = esq$$

- Como $esq = meio + 1 < d + 1$, temos que $esq \leq d = dir$. Por ②, sabemos que $d \leq n$.

- Assim

$$1 \leq esq \leq dir \leq n$$

```
BuscaBinária (A, n, k)
1   esq = 1
2   dir = n
3   Enquanto esq < dir
4       meio = ⌊(esq + dir)/2⌋
5       se k > A[meio]
6           esq = meio + 1
7       Senão
8           dir = meio
9   Se A[esq] == k
10      devolve esq
11  devolve -1
```

$P(t+1) =$ "antes da $(t+1)$-ésima iteração, vale que

$|A[esq..dir]| \leq \left\lceil \frac{n}{2^{t+1-1}} \right\rceil$ ✓

$1 \leq esq \leq dir \leq n$ ✓

$k \notin A[1..esq-1]$

$k \notin A[dir+1..n]$ "

- Como $k > A[meio]$ e A é um vetor ordenado, temos que $k > A[1..meio]$. Portanto

$$k \notin A[1..meio] = A[1..esq-1]$$

```
BuscaBinária (A, n, k)
    esq = 1
    dir = n
    Enquanto esq < dir
        meio = ⌊(esq + dir)/2⌋
        se k > A[meio]
            esq = meio + 1
        Senão
            dir = meio
    Se A[esq] == k
        devolve esq
    devolve -1
```

$P(t+1)$ = "antes da $(t+1)$-ésima iteração, vale que

$$|A[esq..dir]| \leq \left\lceil \frac{n}{2^{t+1-1}} \right\rceil$$

$$1 \leq esq \leq dir \leq n$$

$$k \notin A[1..esq-1]$$

$$k \notin A[dir+1..n]$$"

- Note que no final de iteração $dir = d$
- Por ④

$$k \notin A[d+1..n] = A[dir+1..n]$$

```
BuscaBinária (A, n, k)
    esq = 1
    dir = n
    Enquanto esq < dir
        meio = ⌊(esq + dir) / 2⌋
        se k > A[meio]
            esq = meio + 1
        Senão
            dir = meio
    Se A[esq] == k
        devolve esq
    devolve -1
```

$P(t+1) =$ "antes da $(t+1)$-ésima iteração, vale que

$$|A[esq..dir]| \leq \left\lceil \frac{n}{2^{t+1-1}} \right\rceil \checkmark$$

$$1 \leq esq \leq dir \leq n \checkmark$$

$$k \notin A[1..esq-1] \checkmark$$

$$k \notin A[dir+1..n] \text{"} \checkmark$$

- caso 2: $k \leq A[meio]$

- Então o bloco else é executado. Assim, a linha 8 faz

$$dir = meio$$

$\boxed{\text{Exercício}}$

```
BuscaBinária(A, n, k)
1   esq = 1
2   dir = n
3   Enquanto esq < dir
4       meio = ⌊(esq + dir)/2⌋
5       se k > A[meio]
6           esq = meio + 1
7       Senão
8           dir = meio
9   Se A[esq] == k
10      devolve esq
11  devolve -1
```

$P(t+1)$ = "antes da $(t+1)$-ésima iteração, vale que

$$|A[esq..dir]| \leq \left\lceil \frac{n}{2^{t+1-1}} \right\rceil$$
$$1 \leq esq \leq dir \leq n$$
$$k \notin A[1..esq-1]$$
$$k \notin A[dir+1..n]$$
"