Aula 02 - Correção de algoritmos

Objetivos

- aprender a demonstrar a correção de um algoritmo iterativo
- aprender o que é uma invariante de laço e como demonstrá-la.

Correção de algoritmos

→ Um algoritmo está <u>correto</u> qndo devolve uma resposta correta para <u>qualquer</u> 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: demonstrando!

Correção de algoritmos iterativos

- Algoritmos iterativos possuem laços de repetição

<u>Definição</u>: Uma <u>invariante de laço</u> é uma afirmação que é verdadeira no início de qualquer iteração do laço

- Em geral começam com "antes da t-ésima iteração começar, vale que ..." e envolvem variáveis importantes para 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?

- Por definição, basta provar que $P(1), P(2), \ldots, P(T+1)$ são verdadeiras onde T é a quantidade de iterações realizadas.

- Por que provar até $P(T+1)$ e não $P(T)$?
  - ↳ $P(T)$ diz que algo é válido no início da última iteração, mas no decorrer dela as coisas podem mudar.
  - ↳ $P(T+1)$ diz algo no início de uma iteração cujo corpo não é executado, ou seja, algo que é verdadeiro qndo o laço acaba.

Como provar que uma afirmação é uma invariante de laço?

- Por indução!

  → Prove $P(1)$

  → Prove que $P(t) \Rightarrow P(t+1)$ para todo $t \geq 1$

  ↳ Assuma que $P(t)$ é verdade e conclua que $P(t+1)$ também é

- O valor de T não é necessário nos passos acima
  - ↳ Um vez que provamos que P é invariante, usaremos $P(T+1)$ para dizer algo útil sobre o laço.

Um Problema Simples

Problema Busca em Dados ordenados

Entrada: $\langle A, n, K \rangle$, onde $A[1..n]$ é um vetor contendo n elementos cujas chaves estão em ordem crescente, i.e., $A[i].chave < A[i+1].chave$ para todo $1 \leq i < n$, e K é um número.

Saída: Posição i do elemento de A cuja chave é K, se este existe, ou -1, caso contrário.

A =

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

Busca Linear

BUSCALINEAR(A, n, k)

```
1 i = 1
2 enquanto i ≤ n e A[i].chave ≠ k faça
3     i = i + 1
4 se i ≤ n [redacted] então
5     devolve i
6 devolve −1
```

Busca Linear resolve o problema?

invariantes?!

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

$Q(t)$ = "antes da t-ésima iteração começar, $i = t$ e os elementos de $A[1..i-1]$ já foram acessados"

$R(t)$ = "antes da t-ésima iteração começar, $i = t$ e K já foi comparado a $i-1$ chaves"

$S(t)$ = "antes da t-ésima iteração começar, $i = t$ e $K \notin A[1..i-1]$" ← Parece boa

Teorema. O algoritmo BuscaLinear resolve o problema de Busca em dados ordenados.

Demonstração.
Para demonstrar esse resultado, vamos dividir a prova em dois casos, a depender se $K \in A$ ou não.

Primeiro suponha que $K \in A$. Note que os únicos pontos de retorno do algoritmo ocorrem nas linhas 5 e 6. Assim, o algoritmo atinge a linha 4. Isso implica que o laço da linha 2 terminou, ou seja, $i > n$ ou $A[i].chave = K$. Se $i > n$, como o laço incrementa i em 1 unidade em cada iteração, temos que $i = n+1$. Por $S(n+1)$, temos que $K \notin A[1..i-1] = A[1..n]$, o que é uma contradição. Assim, o teste da linha 2 deve ter falhado pois $A[i].chave = i$ e, consequentemente, $i \leq n$. Portanto o teste da linha 4 dá verdadeiro e o algoritmo retorna i corretamente.

Agora, suponha que $K \notin A$. Note que os únicos pontos de retorno ocorrem nas linhas 5 e 6. Assim, o algoritmo atinge a linha 4. Isso significa que o laço da linha 2 terminou, o que implica que $i > n$ ou $A[i].chave = K$. Como $K \notin A$, temos que $i > n$. Como o laço da linha 2 incrementa i em uma unidade em cada iteração, temos que $i = n+1$. Assim, o teste da linha 4 falha e o algoritmo retorna -1 corretamente. □

nesta demonstração não vamos dividir a prova em dois casos. Vamos assumir uma entrada arbitrária e deixar que a invariante nos ajude a distinguir se estamos em uma instância $K \in A$ ou $K \notin A$.

Demonstração 2

Quando o laço da linha 2 terminar temos que $i > n$ ou $A[i].chave = K$. Se $A[i].chave = K$, então $i \leq n$. Assim, o teste da linha 4 dá verdadeiro e o algoritmo retorna i corretamente. Se $A[i].chave \neq K$, então $i > n$. Como o laço da linha 2 incrementa a variável i em 1 a cada iteração, temos que $i = n+1$. Assim, o teste da linha 4 falha e o algoritmo retorna -1 na linha 6. Agora, vamos mostrar que neste caso $K \notin A$. Por $S(i) = S(n+1)$, temos que $K \notin A[1..i-1] = A[1..n]$. Portanto, o algoritmo está correto ■

Agora, vamos provar que a invariante $S(t)$ está correta.

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

Base. ($t=1$)

Antes da primeira iteração começar, temos que $i = 1$ e $K \notin A[1..0] = \emptyset$.

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

Suponha $P(t)$ seja verdade, ou seja, antes da t-ésima iteração começar $i = t$ e $K \notin [1..t-1]$. Como $P(t+1)$ ocorre, temos que o teste da linha 2 é satisfeito, o que implica que $t \leq n$ e $A[t].chave \neq K$. Assim, $K \notin A[1..t]$. Na linha 3 fazemos $i = t+1$ e o laço encerra-se. Note que no início da iteração $t+1$ temos que $i = t+1$ e $K \notin A[1..t] = A[1..i-1]$. Assim, o resultado segue. □

Receita de Bolo para provar a correção

① Pense na "melhor" invariante $P(t)$ possível.

② Prove a correção do algoritmo usando $P(t)$.

③ Prove a correção do algoritmo.

Busca Binária

```
BuscaBinaria(A, n, k)
1  esq = 1
2  dir = n
3  enquanto esq < dir faça
4      meio = ⌊(esq + dir)/2⌋
5      se k > A[meio].chave então
6          esq = meio + 1
7      senão
8          dir = meio
9  se A[esq].chave == k então
10     devolve esq
11 devolve −1
```

$P(t)$ = "antes da t-ésima iteração, vale que
$$1 \leq e \leq d \leq n$$
$$k \notin A[1..e-1]$$
$$k \notin A[d+1..n]\text{"}$$

Teorema: O algoritmo BuscaBinária resolve o problema da Busca em dados ordenados.

Demonstração.

Quando o laço da linha 3 é encerrado, temos que $P(t)$ é verdade, onde $t-1$ é o número de iterações do laço 3. Assim

$$1 \leq esq \leq dir \leq n$$
$$k \notin A[1..esq-1] \qquad (A)$$
$$k \notin A[dir+1..n]$$

Quando esse laço termina, temos $esq \geq dir$. Como (A) também vale, concluímos que

$$esq = dir. \qquad (B)$$

Se o teste da linha 9 dá verdadeiro, então $A[esq].chave = k$ e o algoritmo retorna esq corretamente (linha 10). Se o teste da linha 9 falha, então

$$A[esq].chave \neq k, \qquad (C)$$

e o algoritmo devolve -1.

Vamos agora argumentar que $k \notin A$ e que portanto o algoritmo devolveu o valor correto. Por (A), temos que $k \notin A[1..esq-1]$. Por (A) e (B), temos que $k \notin A[1..dir+1] = A[1..esq+1]$. Por (C), temos que $k \notin A[esq]$. Assim, $k \notin A$. ∎

Agora, vamos provar a invariante da busca binária.

$$P(t) = \text{" antes da } t\text{-ésima iteração, vale que } \begin{array}{l} 1 \leq e \leq d \leq n \\ K \notin A[1..e-1] \\ K \notin A[d+1..n] \text{ "} \end{array}$$

<u>Demonstração</u>

A prova segue por indução em t.

<u>Base $(t=1)$</u>

Antes da primeira iteração começar, temos que $esq = 1$, $dir = n$, $A[1..esq-1] = A[1..0] = \emptyset$, $A[dir..n] = A[n+1..n] = \emptyset$. Portanto, temos que $P(1)$ vale.

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

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

$1 \leq e \leq d \leq n$ ①

$K \notin A[1..e-1]$ ②

$K \notin A[d+1..n]$ ③

Como a t-ésima iteração foi executada, temos que

$e < t$ ④

Ao executarmos a linha 4, temos

$m = \lfloor (e+d)/2 \rfloor$ ⑤

Note que

$$m = \lfloor (e+d)/2 \rfloor \leq \frac{e+d}{2} < d, \quad e$$

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

que implica em

$m \geq e$.

Assim,

$e \leq m < d$ ⑤

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 dividida em dois casos a depender do fato de $K > A[m]$.chave ou não.

Primeiro, suponha que $K > A[m]$.chave. Então, o algoritmo executa a linha 6 e finaliza a iteração. Neste caso temos

$esq = m + 1$

$dir = d$ ⑥

Note que se $K > A[m]$.chave, temos que

$K \notin A[1..m] = A[1..esq-1]$,

devido ao fato de A ser um vetor ordenado e por ⑥.

Por 3 e 6, temos

$k \notin A[d+1..n] = A[dir+1..n]$ (8)

Por 5 e 6, $esq - 1 = m < d = dir$, que implica em

$esq \leq dir$ (9)

Por 1 e 6,

$dir = d \leq n.$ (10)

Por 1 e 5,

$1 \leq e \leq m = esq - 1 < esq$ (11)

Por 7, 8, 9, 10 e 11, temos que o resultado segue.

Agora, suponha que $k \leq A[meio].chave$. Então...

(o restante da prova fica de exercício)

∎