Teoria da Computação

Guilherme Henrique de Souza Nakahata

Universidade Estadual do Paraná - Unespar

17 de Abril de 2024

- Um algoritmo está correto quando envolve uma resposta correta para qualquer instância;
- Incorreto quando devolve uma resposta errada para alguma instância;
- Como verificar se um algoritmo está correto?

- Recursivo:
  - Condição verdadeira no início de cada chamada recursiva e permanece verdadeira após cada chamada;
  - Garante a corretude do algoritmo, mantendo a condição em cada chamada recursiva;
  - Exemplos: Busca em árvores, profundidade, algoritmos de divisão e conquista, algoritmos de árvores de decisão...
- Iterativo:
  - Condição verdadeira no início de cada iteração do laço e permanece verdadeira após cada iteração;
  - Garante a corretude do algoritmo, mantendo a condição em cada passo do laço;
  - Exemplos: Busca linear, ordenação por inserção, iterações em estruturas de dados...

Invariante de laço recursão

- Uma função é recursiva se chama a si mesma;
- Técnica muito poderosa de solução de problemas;
- Reduzir a instância em instância menores;
- Ajuda a resolver a instância original;
- Exemplo:

Algorithm 1 Busca Linear Recursiva

```
procedure BuscaLinearRecursiva(A, n, x)
    if n == 0 then
        return -1
    end if
    if A[n] == x then
        return n
    end if
    return BuscaLinearRecursiva(A, n − 1, x)
end procedure
```

Invariante de laço recursão

- Queremos provar:
  - P(n) = "BuscaLinearRecursiva(A,n,x) devolve -1 se $x \notin A[1...n]$ e devolve i se A[i] = x para qualquer n";
  - **Indução**;
  - Provar para P(0);
  - Para provar P(n), primeiro suponha que P(k) vale, para $0 <= k < n$ e então use isso para provar P(n).

Algorithm 2 Busca Linear Recursiva

```
procedure BuscaLinearRecursiva(A, n, x)
    if n == 0 then
        return -1
    end if
    if A[n] == x then
        return n
    end if
    return BuscaLinearRecursiva(A, n − 1, x)
end procedure
```

Invariante de laço recursão

- Queremos provar:
  - Caso base: n = 0;
  - O algoritmo diretamente devolve -1;
  - Resposta esperada;
  - A [1...0] é vazio e não contém x;
  - Logo, P(0) vale;
- Hipótese: P(k) vale, para $0 <= k < n$.

Algorithm 3 Busca Linear Recursiva

```
procedure BuscaLinearRecursiva(A, n, x)
    if n == 0 then
        return -1
    end if
    if A[n] == x then
        return n
    end if
    return BuscaLinearRecursiva(A, n − 1, x)
end procedure
```

Invariante de laço recursão

- Queremos provar:
  - Quando o algoritmo recebe n $>$ 0, ele verifica se A[n] $=$ x (linha 3);
  - Se forem iguais, devolve n;
  - P(n) vale nesse caso;
  - Se **A[n] != x**, devolve a mesma chamada (BuscaLinearRecursiva(A,**n-1**,x));
  - A chamada recursiva é sobre um valor menor que **n**;
  - Vale a hipótese, que a chamada é menor que n;
  - Hipótese: P(k) vale, para $0 <= k < n$;
  - Por hipótese, essa chamada detecta se $x \in A[1...n-1]$;
  - Como A[n] != x, concluímos que a chamada funciona;
  - C.Q.D.

Algorithm 4 Busca Linear Recursiva

```
procedure BuscaLinearRecursiva(A, n, x)
    if n == 0 then
        return -1
    end if
    if A[n] == x then
        return n
    end if
    return BuscaLinearRecursiva(A, n − 1, x)
end procedure
```

Invariante de laço

- Invariante de laço Iterativo:
  - Uma afirmação verdadeira;
  - Início de qualquer iteração do laço;
- Começam com "antes da t-ésima iteração começar, vale..." ;
- Envolvem variáveis importantes para o laço;
- Notação: P(t);
- Concluir algo importante após o término do laço;

Invariante de laço

- Como provar que uma frase é uma invariante?
- Por definição:
  - Basta provar que P(1), P(2), ..., P(T), P(T+1) são verdadeiras;
  - T $\rightarrow$ quantidade de iterações realizadas;
- Por que P(T+1)?
- P(T) válido no início da última iteração;
- P(T+1) válido no início de uma iteração que não existe.

Invariante de laço

- Como provar que uma frase é uma invariante?
- Por definição:
  - Basta provar que P(1), P(2), ..., P(T), P(T+1) são verdadeiras;
  - T $\rightarrow$ quantidade de iterações realizadas;
- Por que P(T+1)?
- P(T) válido no início da última iteração;
- P(T+1) válido no início de uma iteração que não existe (fim do laço).

Invariante de laço

- Indução!
  - Prove P(1);
  - Considere $t \geq 1$ e suponha que P(t) vale;
  - Prove P(t+1);

  - Provamos que P é invariante;
  - Usamos P(T+1) para dizer algo útil ao fim do laço.

Busca Linear

- P(t) = "Antes da t-ésima iteração começar, i = t e os elementos de A[1...1-1] já foram acessados";
- R(t) = "Antes da t-ésima iteração começar, i = t e k já foi comparada com i-1 chaves de A".

Algorithm 5 Busca Linear

```
procedure BUSCA LINEAR(A, n, k)
    i ← 1
    while i ≤ n and A[i].chave ≠ k do
        i ← i + 1
    end while
    if i ≤ n and A[i].chave = k then
        return i                    ▷ Elemento encontrado na posição i
    end if
    return −1                       ▷ Elemento não encontrado
end procedure
```

Invariante de laço

- Como provar?
  - "Existe um elemento com chave k no vetor se e somente se o algoritmo devolve um índice que contém tal elemento";
  - De forma equivalente:
    - Se existe um elemento com chave k no vetor, então o algoritmo devolve um índice do vetor e se não existe um elemento com chave k no vetor, então o algoritmo não devolve um índice do vetor.

Invariante de laço

- Suponha a seguinte invariante:
  - P(t) = "Antes da t-ésima iteração começar, vale que i = t e o vetor A[1...i-1] não contém um elemento com chave k";
- Vamos supor:
  - I_1 = Se não existe um elemento com chave K em A, então a busca não devolve um índice válido;
  - O laço pode terminar porque i > n, nesse caso ele executou n vezes e P(n+1) nos diz que o vetor A[1...n] não contém elementos com chave k;
  - Então I_1 é verdadeira;

Algorithm 6 Busca Linear

```
procedure BUSCA LINEAR(A, n, k)
    i ← 1
    while i ≤ n and A[i].chave ≠ k do
        i ← i + 1
    end while
    if i ≤ n and A[i].chave = k then
        return i                          ▷ Elemento encontrado na posição i
    end if
    return −1                             ▷ Elemento não encontrado
end procedure
```

Invariante de laço

- $I_2 =$ Se existe elemento com chave K em A, então a busca devolve seu índice";
- O laço pode terminar porque $i <= n$ e A[i].chave = k;
- Existe um elemento com chave K, então I_2 é verdadeira.

Algorithm 7 Busca Linear

```
procedure BUSCA LINEAR(A, n, k)
    i ← 1
    while i ≤ n and A[i].chave ≠ k do
        i ← i + 1
    end while
    if i ≤ n and A[i].chave = k then
        return i                          ▷ Elemento encontrado na posição i
    end if
    return −1                             ▷ Elemento não encontrado
end procedure
```

Invariante de laço

- Vamos provar que:
  - P(t) = "Antes da t-ésima iteração começar, vale que i = t e o vetor A[1...i-1] não contém um elemento com chave K";
  - Por indução no n° t de vezes que o teste de laço executa;
- Caso base: t = 1. O teste executou uma única vez;
  - i = 1;
  - A[1...0];
  - Vazio;
  - Não contém um elemento com chave K;
  - Logo, P(1) vale.

Invariante de laço

- Passo de indução: $t >= 1$. Suponha que P(t) vale e vamos provar P(t+1);
  - Como o t-ésimo teste foi executado, pois o laço começou, sabemos que A[t].chave != k (i = t já que P(t) vale);
  - Juntando com o fato de A[1...t-1] não ter elementos com chave K;
  - Sabemos que A[1...t] não tem elemento com chave K;
  - O laço incrementa i, fazendo i = t + 1;
  - P(t+1);

Algorithm 8 Busca Linear

```
procedure BUSCA LINEAR(A, n, k)
    i ← 1
    while i ≤ n and A[i].chave ≠ k do
        i ← i + 1
    end while
    if i ≤ n and A[i].chave = k then
        return i                          ▷ Elemento encontrado na posição i
    end if
    return −1                             ▷ Elemento não encontrado
end procedure
```

Invariante de laço

- Término: O laço termina quando i $>$ n ou quando A[i].chave é igual a k. Se o laço terminar porque i $>$ n, então todos os elementos de A[1...n] foram verificados e nenhum deles possui a chave de busca k, o que confirma a invariante. Se o laço terminar porque A[i].chave é igual a k, então o algoritmo retorna i, indicando que o elemento foi encontrado.

Algorithm 9 Busca Linear

```
procedure BUSCA LINEAR(A, n, k)
    i ← 1
    while i ≤ n and A[i].chave ≠ k do
        i ← i + 1
    end while
    if i ≤ n and A[i].chave = k then
        return i                                ▷ Elemento encontrado na posição i
    end if
    return −1                                   ▷ Elemento não encontrado
end procedure
```

Invariante de laço

- Inicialização;
- Manutenção;
- Término;

Invariante de laço

- Exercícios:

Bibliografia Básica

- LEWIS, H. R.; PAPADIMITRIOU, C. H. **Elementos de Teoria da Computação**. 2 ed. Porto Alegre: Bookman, 2000.
- VIEIRA, N. J. **Introdução aos Fundamentos da Computação**. Editora Pioneira Thomson Learning, 2006.
- DIVERIO, T. A.; MENEZES, P. B. **Teoria da Computação: Máquinas Universais e Computabilidade**. Série Livros Didáticos Número 5, Instituto de Informática da UFRGS, Editora Sagra Luzzato, 1 ed. 1999.

Obrigado!
Dúvidas?

Guilherme Henrique de Souza Nakahata

guilhermenakahata@gmail.com

https://github.com/GuilhermeNakahata/UNESPAR-2024