Prova de corretude do InsertionSort

Atílio Luiz

2 de abril de 2024

```
Insertion-Sort(A, n)
1   para j = 2 até n faça
2       chave = A[j]
3       i = j − 1
4       enquanto i ≥ 1 e A[i] > chave faça
5           A[i + 1] = A[i]
6           i = i − 1
7       A[i + 1] = chave
```

Figura 1: Pseudocódigo do Algoritmo InsertionSort.

A fim de provar formalmente a corretude do InsertioSort, vou apresentar duas invariantes de laço e provar a validade de cada uma delas.

> **Invariante de Laço 1.** *No início de cada iteração i do laço* ***enquanto****, com $0 \leq i \leq j-1$, os elementos do subarray $A[i+2...j]$ estão em ordem crescente, são todos maiores que o valor na variável* ***chave****, e são os elementos que estavam originalmente no intervalo de $i+1$ até $j-1$ do vetor A. Além disso, os elementos no subarray $A[1...i]$ estão em ordem crescente, são todos menores que os elementos que estão no subarray $A[i+2...j]$, e são os elementos que estavam originalmente no intervalo de 1 até i do vetor A.*

Demonstração. A prova se dá por **indução regressiva** na variável contadora i do laço **enquanto**.

Caso Básico (Inicialização): No início da primeira iteração do laço **enquanto**, temos $i = j - 1$. A variável **chave** guarda o valor $A[j]$. Como $i = j-1$, o subarray $A[i+2...j]$ é vazio. Logo, as afirmações do invariante sobre esse subarray são todas válidas por vacuidade. Os elementos no subarray $A[1...j-1]$ estão ordenados em ordem crescente e são os valores que estavam originalmente no intervalo de índices de 1 a $j-1$ do vetor A. Além disso, como o subarray $A[i+2...j]$ é vazio, todos os elementos de $A[1...j-1]$ são menores que os elementos de $A[i+2...j]$ por vacuidade. Logo, o invariante de laço é verdadeiro no início da primeira iteração.

Passo Indutivo (Manutenção): Suponha que o invariante de laço seja verdadeiro no início da iteração $i = k$, com $k \leq j - 1$, e que entramos no laço **enquanto** nesta iteração (pois o algoritmo não terminou). Vamos provar que o invariante continuará verdadeiro no início da próxima iteração, que será quando $i = k - 1$. Para isso, vamos executar o corpo do laço **enquanto** descrevendo o que acontece nesta iteração.

Como o invariante de laço é verdadeiro no início da iteração $i = k$, temos que os elementos do subarray $A[k+2...j]$ estão em ordem crescente, são todos maiores que o valor na variável **chave**, e são os elementos que estavam originalmente no intervalo de $k+1$ até $j-1$ do vetor A. Além disso, os elementos no subarray $A[1...k]$ estão em ordem crescente, são todos menores que os elementos que estão no subarray $A[k+2...j]$, e são os elementos que estavam originalmente no intervalo de 1 até k do vetor A.

Como entramos no laço, sabemos que $k \geq 1$ e $A[k] >$ **chave** (essa é a condição de entrada do laço **enquanto**). Assim, ao executar a linha 5 do algoritmo, o elemento $A[k+1]$ recebe o valor do elemento $A[k]$, fazendo o subarray $A[k+1...j]$ conter todos os elementos maiores que a variável **chave**, que estavam originalmente no intervalo de k a $j-1$ do vetor A. Note que os elementos no subarray $A[1...k-1]$ continuam em ordem crescente (pois não foram alterados), são todos menores que os elementos que estão no subarray $A[k+1...j]$, e são os elementos que estavam originalmente no intervalo de 1 até $k-1$ do vetor A. Portanto, ao executar a linha 6 do algoritmo, a variável i passa a ter o valor $i = k-1$ e a invariante de laço é restabelecida (é verdadeira) no início da próxima iteração.

Término: O laço **enquanto** termina em uma das seguintes ocasiões: ou quando $i = 0$; ou quando $A[i] \leq chave$ para algum $1 \leq i \leq j-1$. Então temos dois casos a analisar.

SubCaso 1. Vou começar com o caso $i = 0$. Neste caso, o invariante de laço diz que os elementos do subarray $A[2...j]$ estão em ordem crescente, são todos maiores que o valor na variável **chave**, e são os elementos que estavam originalmente no intervalo de 1 até $j-1$ do vetor A. Além disso, o subarray $A[1...0]$ é vazio. Isso significa que todos os elementos que estavam no intervalo de 1 a $j-1$ são maiores que o valor na variável **chave** e que todos eles foram deslocados uma posição para a direita. Logo, temos a seguinte configuração final neste caso: **chave** $< A[2] \leq A[3] \leq \cdots \leq A[j]$.

SubCaso 2. $A[i] \leq chave$ para algum $1 \leq i \leq j-1$. Neste caso, o invariante de laço diz que os elementos do subarray $A[i+2...j]$ estão em ordem crescente, são todos maiores que o valor na variável **chave**, e são os elementos que estavam originalmente no intervalo de $i+1$ até $j-1$ do vetor A. Além disso, os elementos no subarray $A[1...i]$ estão em ordem crescente, são todos menores que os elementos que estão no subarray $A[i+2...j]$, e são os elementos que estavam originalmente no intervalo de 1 até i do vetor A. Tudo isso implica, que o slot $A[i+1]$ está livre para receber um novo valor que, no nosso algoritmo, será o valor da variável **chave**.

Portanto, a partir dos dois subcasos analisados, concluímos que na última iteração do laço enquanto, temos que os elementos dos subarrays $A[1...i]$ e $A[i+2...j]$ respeitam a seguinte relação:

$$A[1] \leq \cdots \leq A[i] \leq \textbf{chave} < A[i+2] \leq \cdots \leq A[j]$$

□

Usando a Invariante de Laço 1, provamos a Invariante de Laço 2, que é descrita a seguir.

Invariante de Laço 2. *Imediatamente antes de cada iteração do laço* ***para****, o subvetor $A[1...j-1]$ consiste dos elementos originalmente em $A[1...j-1]$, ordenados em ordem crescente.*

Demonstração. A prova se dá por indução na variável contadora j do laço **para**.

Caso Básico (Inicialização): No início da primeira iteração do laço **para**, temos $j = 2$. Neste caso, o subvetor $A[1...j-1]$ contém apenas um elemento, que é o $A[1]$. Note que $A[1]$ é o primeiro elemento do vetor original e está ordenado. Então, a invariante vale antes da primeira iteração.

Passo indutivo (Manutenção): Suponha que a invariante vale no início de alguma iteração j, com $j \geq 2$. Nessa iteração, ao executar a linha 2, temos **chave** $= A[j]$. Pelo Invariante de Laço 1, após executar as linhas 3 a 6 do algoritmo, temos que existe algum índice $i+1$, com $1 \leq i+1 \leq j$, tal que o subarray $A[1...i]$ contém os elementos menores ou iguais a **chave** e o subarray $A[i+2...j]$ contém os elementos maiores que **chave**, que estavam originalmente no subarray $A[1...j-1]$, e ambos estão ordenados.

Após o laço enquanto, na linha 7 do algoritmo, $A[i+1]$ recebe o valor de **chave**. Assim, o subarray $A[1...j]$ consiste dos elementos originalmente em $A[1...j]$, mas em ordem crescente. Incrementando j para a próxima iteração do laço **para** preserva a invariante de laço.

Término: Finalmente, examinamos o que acontece quando o laço **para** termina. O laço **para** termina quando $j = n+1$. Substituindo $n+1$ por j no enunciado da invariante de laço, temos que: o subarray $A[1...n]$ consiste dos elementos originalmente em $A[1...n]$, em ordem crescente. Assim, o array inteiro está ordenado e o algoritmo é correto. □