Projeto e Análise de Algoritmos

Flávio L. C. de Moura

27 de janeiro de 2022

1 Ordenação por inserção

```
INSERTION-SORT(A)
1  for j = 2 to A.length
2      key = A[j]
3      // Insert A[j] into the sorted sequence A[1 .. j − 1].
4      i = j − 1
5      while i > 0 and A[i] > key
6          A[i + 1] = A[i]
7          i = i − 1
8      A[i + 1] = key
```

1.1 Correção:

Ao final da execução de Insertion-Sort(A) teremos um vetor que é uma permutação do vetor original A, e que está ordenado. Isto será provado por meio da seguinte invariante de laço:

> Antes de cada iteração do laço **for** o subvetor A[1..j-1] é uma permutação do subvetor original A[1..j-1] que está ordenado.

1. Prova:

   A prova é por indução no número de iterações do laço **for**:

   (a) Base da indução (Inicialização): Antes da primeira iteração do laço **for**, temos que j = 2 (condição necessária para iniciar o laço), e portanto a invariante é trivial porque o subvetor unitário A[1] está ordenado por definição.

(b) Passo indutivo (Manutenção): Considere a k-ésima iteração. Temos como hipótese que "Antes de cada iteração do laço **for** o subvetor A[1..k-1] é uma permutação do subvetor original A[1..k-1] que está ordenado."Assim, na k-ésima iteração, o laço **while** vai deslocar cada elemento maior do que A[k], i.e. key, uma posição para a direita até encontrar a posição correta onde o elemento A[k] deve ser inserido.

i. Informalmente estamos dizendo que o laço **while** encontra a posição correta para inserir A[j] que está armazenado na key. Precisamos então construir uma invariante para o laço **while**.

> Antes de cada iteração do laço **while** o subvetor A[i..j] possui elementos que são maiores ou iguais a key.

Prova: Inicialização: Antes da primeira iteração do **while**, estamos assumindo as condições para que o laço seja executado pela primeira vez, ou seja, temos que i = k-1, A[j] = key e A[i] > key, e portanto a invariante vale.
Manutenção: Por hipótese temos que o subvetor A[i+1..j] possui elementos que são maiores ou iguais a key. Durante uma iteração do laço, o elemento A[i] é copiado na posição A[i+1] e portanto a invariante continua valendo.
Finalização: Ao final da execução do laço, temos que i é, de fato, a posição correta para inserir o elemento A[k] já que todos os elementos do subvetor A[i+1..j] são maiores ou iguais a key. É importante observar que a inserção do elemento A[k] na posição i não elimina nenhum elemento do vetor original porque o elemento que está na posição i foi copiado para a posição i+1, se o laço **while** foi executado, ou ele é o próprio elemento armazenado em key, quando o laço não é executado.

(c) Conclusão (Terminação): Ao final da execução do laço **for**, temos j = n+1, e portanto a invariante corresponde a dizer que o vetor A[1..n] obtido ao final da execução do algoritmo está ordenado, e é uma permutação do vetor original A[1..n].