Avaliação Continuada 02 (Provando corretude de algoritmos iterativos)

Questão 1

Questão 1.2.1 do livro Algoritmos em C do professor Paulo Feofiloff.

Um **invariante de laço** é uma relação entre os valores das variáveis que vale no início de cada iteração do processo iterativo. Os invariantes explicam o funcionamento do processo iterativo e permitem provar, por indução, que ele tem o efeito desejado. Considere, por exemplo, a função **Max** descrita abaixo.

```
  Algoritmo: Max(A, n)
    Entrada: um vetor A com n inteiros, cujos índice vão de 1 a n
    Saída: o valor de um elemento máximo do vetor A[1..n]

      maximum = A[1]
      for j = 2 to n
          if maximum < A[j]
              maximum = A[j]
      return maximum
```

O processo iterativo controlado pelo laço **for** tem o seguinte invariante:

"No início de cada iteração do laço for (imediatamente antes da comparação de j com n), a variável maximum guarda o valor de um elemento máximo do subvetor A[1 ... j-1]".

Mostre que o invariante da função **Max** vale no início da primeira iteração do laço **for**. Suponha que o invariante vale no início de uma iteração qualquer e mostre que ele vale no início da iteração seguinte. Suponha que o invariante vale no início da última iteração e deduza daí que a função devolve um elemento máximo do vetor A[1...n].

Questão 2

[CORMEN et al.] Questão 2-2 (Corretude do Bubblesort). Bubblesort é um algoritmo de ordenação popular, mas ineficiente. Ele funciona fazendo varreduras pelo vetor, trocando repetidamente elementos adjacentes que estiverem fora de ordem.

```
  Algoritmo: BubbleSort(A)
    Entrada: um vetor de inteiros A
    Saída: vetor A ordenado em ordem crescente

1.    for i = 1 to A.length-1
2.      for j = A.length downto i+1
3.        if A[j] < A[j-1]
4.          exchange A[j] with A[j-1]
```

Responda os itens abaixo:

A. Seja A' o vetor resultante após a plicação do algoritmo $\mathbf{BubbleSort(A)}$. Para provar que o BubbleSort está correto, nós precisamos provar que ele termina e que

$$A'[1] \leq A'[2] \leq \cdots \leq A'[n].$$

onde $n = A.length$. A fim de mostrar que BubbleSort de fato ordena o array dado como entrada, o que mais precisamos provar?

B. Enuncie com precisão um invariante de laço para o laço **for** das linhas 2 a 4 e prove que esse invariante de laço é válido. Sua prova deve usar a estrutura da prova do invariante de laço em 3 etapas que foi apresentada em sala de aula (Caso Base, Passo Indutivo e Término).

C. Usando a condição de término do invariante de laço demonstrado na parte (B) acima, enuncie um invariante de laço para o laço for das linhas 1 a 4 que permita provar a desigualdade (2.3). Sua prova deve empregar a estrutura da prova do invariante de laço apresentada em sala de aula.

D. Qual é o tempo de execução do pior caso de bubblesort? Como ele se compara com o tempo de de execução da ordenação por inserção?

Respostas

Legendas

- $..=$: Indica um intervalo de valores até o valor indicado, incluindo o próprio valor. Ex.: $1..=3=[1,2,3]$
- $..$: Indica um intervalo de valores, excluindo o limite. Ex.: $1..3=[1,2]$

Q1

Considere $P(j)$ como os valores de j para cada iteração do laço.

Caso base: Para $P(2)$, temos que $j = 2$. No início do laço, $maximum$ é igual a $A[1]$ e, no fim do laço, $maximum$ pode manter seu valor ao ser igual a $A[2]$. Dada a definição da invariante para esse laço, temos que ela está correta, dado que no início do laço, $maximum$ guarda o valor de $A[1]$ que está dentro do intervalo do subvetor $A[1.. = 1]$.

Hipótese de indução: Dado um k arbitrário, sendo que $n > k \geq 2$, suponha que a invariante seja verdadeira para $P(k)$.

Passo indutivo: Se $P(k)$ é válido, então a invariante está correta para o subvetor $A[1.. = k-1]$. Nesse caso, existem dois casos a serem considerados para $P(k+1)$:

- Se $A[k] > maximum$ na iteração $P(k)$, então $maximum$ guarda o valor de $A[k]$ e, no início do laço na iteração $P(k+1)$, maximum possuí um valor que está dentro do subvetor $A[1.. = k]$.
- Se $A[k] \leq maximum$ na iteração $P(k)$, então $maximum$ não mudará seu valor guardado e, no início do laço na iteração $P(k+1)$, maximum possuir um valor que ainda está dentro do subvetor $A[1.. = k-1]$, que, por sua vez, está dentro do subvetor $A[1.. = k]$.

Portanto, a invariante é válida para todas as iterações do laço.

Q2

A.

A' deve ter os mesmos elementos de A, sem adicionar coisa alguma.

B.

Invariante: $A[j]$ é o menor dentre os valores de $A[j.. = A.length]$

Prova por indução fraca:

Caso base: Para $j = A.length$, temos que $A[j]$ é o menor dentre os valores de $A[j.. = A.length]$, dado que existe apenas um valor, o próprio $A[j]$.

Hipótese de indução: Suponha que para um j arbitrário, dentro dos limites do laço, a invariante seja verdadeira para $A[j]$.

Passo indutivo: Para $j - 1$, existem dois casos a serem considerados:

- Se $A[j] < A[j - 1]$, então $A[j]$ é o menor dentre os valores de $A[j - 1.. = A.length]$, que, por sua vez, também está dentro de $A[j.. = A.length]$.
- Se $A[j] \geq A[j - 1]$, então $A[j - 1]$ é o menor dentre os valores de $A[j - 1.. = A.length]$.

Portanto, a invariante é válida para todas as iterações do laço.

C.

Invariante: Para o início de cada iteração, $A[1..i]$ sempre terá $i - 1$ elementos ordenados.

Prova por indução fraca:

Caso base: Para $i = 1$, temos que $A[1]$ é o maior dentre os valores de $A[1.. = 1]$, dado que existem apenas um valor, $A[1]$.

Hipótese de indução: Suponha que para um i arbitrário, dentro dos limites do laço, a invariante seja verdadeira para $A[i]$. Então $A[1..i]$ possui $i - 1$ elementos ordenados.

Passo indutivo: Para $i + 1$, no início do laço, considerando a invariante provada no **item B**, existem dois casos a serem considerados:

- Se $A[i]$ não estava ordenado, ele foi ordenado no fim da última iteração.
- Senão, tudo estava ordenado.

Portanto, a invariante é válida para todas as iterações do laço.

D.

O primeiro laço será executado $n - 1$ vezes e o segundo laço será executado $n - i - 1$ vezes para cada iteração do primeiro laço, ou seja, $(n(n - 1))/2$.

Então temos o polinômio $(n^2 - n)/2$ que, por sua vez, é menor ou igual a Cn^2, sendo C uma constante.

Portanto, $\Theta(n^2)$, o mesmo do Insertion Sort.

José Mykael Alves Nogueira