Provas de Correção de Algoritmos e Análise de Complexidade Assintótica

Ana Paula Tomás

Desenho e Análise de Algoritmos 2017/18

Setembro 2017

Exemplo: Posição do máximo

Problema

Escrever uma função $\textsc{Posmax}(v, k, n)$ *para determinar o índice da primeira ocorrência do elemento máximo no segmento* $v[k], v[k+1], \ldots, v[n]$*, do vetor* v*. Admitir que* $k \leq n$ *e que esse segmento está dentro dos limites do vetor.*

Resposta 1

$\textsc{Posmax}(v, k, n)$

$\quad pmax \leftarrow k;$

$\quad$ Para $i \leftarrow k+1$ até n fazer

$\quad\quad$ Se $v[i] > v[pmax]$ então

$\quad\quad\quad pmax \leftarrow i;$

$\quad$ retorna $pmax;$

Exemplo: Posição do máximo

Problema

Escrever uma função $\textsc{Posmax}(v, k, n)$ *para determinar o índice da primeira ocorrência do elemento máximo no segmento* $v[k], v[k+1], \ldots, v[n]$, *do vetor* v.

Resposta 2 (equivalente a Resp 1 segundo a semântica dos ciclos **Para** e **Enquanto**)

```
POSMAX(v, k, n)
    pmax ← k;
    i ← k + 1;
    Enquanto i ≤ n fazer
        Se v[i] > v[pmax] então
            pmax ← i;
        i ← i + 1;
    retorna pmax;
```

Exemplo: Posição do máximo

```
POSMAX(v, k, n)
1.    pmax ← k;
2.    i ← k + 1;
3.    Enquanto i ≤ n fazer
4.        Se v[i] > v[pmax] então
5.            pmax ← i;
6.        i ← i + 1;
7.    retorna pmax;
```

Invariante de ciclo:

Quando a condição de paragem do ciclo está a ser testada para um dado valor de i (na linha 3), o valor de pmax é o índice da primeira ocorrência do máximo de $v[k], v[k+1], \ldots, v[i-1]$*, sendo isto verdade para todo valor de i tal que* $k+1 \leq i \leq n+1$*.*

Exemplo: Posição do máximo

```
PosMax(v, k, n)
    pmax ← k;
    i ← k + 1;
    Enquanto i ≤ n fazer
        Se v[i] > v[pmax] então
            pmax ← i;
        i ← i + 1;
    retorna pmax;
```

Esqueleto da prova do invariante (por indução matemática):
Se provarmos (i) e (ii), concluimos (pelo Princípio de Indução) que o invariante se verifica em todas as iterações do ciclo.

- (i) **Caso de base:** o invariante verifica-se no início do ciclo
- (ii) **Hereditariedade:** o invariante é preservado em cada iteração do ciclo (se verifica numa iteração então verifica-se na seguinte).

Exemplo: Posição do máximo

$\text{POSMAX}(v, k, n)$

1. $pmax \leftarrow k$;
2. $i \leftarrow k + 1$;
3. Enquanto $i \leq n$ fazer
4.   Se $v[i] > v[pmax]$ então
5. $pmax \leftarrow i$;
6. $i \leftarrow i + 1$;
7. retorna $pmax$;

Propriedade (**invariante de ciclo**): Quando a condição de paragem do ciclo está a ser testada para um dado valor de i (na linha 3), o valor de $pmax$ é o índice da primeira ocorrência do máximo de $v[k], v[k+1], \ldots, v[i-1]$, sendo tal verdade para todo valor de i tal que $k+1 \leq i \leq n+1$.

Prova (por indução):

- (i) Caso de base:

  No ínicio do ciclo, $pmax = k$ e $i = k+1$, sendo verdade que $pmax$ guarda o índice da primeira ocorrência do máximo da sequência $v[k], v[k+1], \ldots, v[i-1]$, já que esta sequência se reduz a $v[k]$.

Exemplo: Posição do máximo

Prova (por indução):

- (ii) Hereditariedade:
  - Suponhamos, como hipótese de indução, que a condição se verifica para um certo valor de i_0, com $i_0 < n$.
    Então, será efetuada uma nova iteração (execução do bloco 4–6).
  - Pela hipótese de indução sobre o estado de *pmax* e i, concluimos que quando executamos o bloco 4-5, o valor de *pmax* será:
    - preservado se $v[i_0] \leq \max(v[k], \ldots, v[i_0 - 1])$
    - alterado para i_0 se $v[i_0] > \max(v[k], \ldots, v[i_0 - 1])$.
  - Portanto, se *pmax* tinha o indíce da primeira ocorrência do máximo de $v[k], \ldots, v[i_0 - 1]$ passará a ter o indíce da primeira ocorrência do máximo de $v[k], \ldots, v[i_0 - 1], v[i_0]$.
  - Na instrução 6, o valor de i passa a ser $i_0 + 1$, e a seguir a condição de paragem do ciclo (linha 3) volta a ser testada. Do que vimos acima, concluimos que a condição enunciada sobre o estado das variáveis se verifica para $i = i_0 + 1$ se se verificar para $i = i_0$.

Exemplo: Posição do máximo

Prova (por indução):

- (ii) Hereditariedade:
  - Suponhamos, como hipótese de indução, que a condição se verifica para um certo valor de i_0, com $i_0 < n$.
    Então, será efetuada uma nova iteração (execução do bloco 4–6).
  - Pela hipótese de indução sobre o estado de *pmax* e i, concluimos que quando executamos o bloco 4-5, o valor de *pmax* será:
    - preservado se $v[i_0] \leq \max(v[k], \ldots, v[i_0 - 1])$
    - alterado para i_0 se $v[i_0] > \max(v[k], \ldots, v[i_0 - 1])$.
  - Portanto, se *pmax* tinha o indíce da primeira ocorrência do máximo de $v[k], \ldots, v[i_0 - 1]$ passará a ter o indíce da primeira ocorrência do máximo de $v[k], \ldots, v[i_0 - 1], v[i_0]$.
  - Na instrução 6, o valor de i passa a ser $i_0 + 1$, e a seguir a condição de paragem do ciclo (linha 3) volta a ser testada. Do que vimos acima, concluimos que a condição enunciada sobre o estado das variáveis se verifica para $i = i_0 + 1$ se se verificar para $i = i_0$.

Exemplo: Posição do máximo

Prova (por indução):

- (ii) Hereditariedade:
    - Suponhamos, como hipótese de indução, que a condição se verifica para um certo valor de i_0, com $i_0 < n$.
      Então, será efetuada uma nova iteração (execução do bloco 4–6).
    - Pela hipótese de indução sobre o estado de *pmax* e i, concluimos que quando executamos o bloco 4-5, o valor de *pmax* será:
        - preservado se $v[i_0] \leq \max(v[k], \ldots, v[i_0 - 1])$
        - alterado para i_0 se $v[i_0] > \max(v[k], \ldots, v[i_0 - 1])$.
    - Portanto, se *pmax* tinha o indíce da primeira ocorrência do máximo de $v[k], \ldots, v[i_0 - 1]$ passará a ter o indíce da primeira ocorrência do máximo de $v[k], \ldots, v[i_0 - 1], v[i_0]$.
    - Na instrução 6, o valor de i passa a ser $i_0 + 1$, e a seguir a condição de paragem do ciclo (linha 3) volta a ser testada. Do que vimos acima, concluimos que a condição enunciada sobre o estado das variáveis se verifica para $i = i_0 + 1$ se se verificar para $i = i_0$.

Exemplo: Posição do máximo

Prova (por indução):

- (ii) Hereditariedade:
  - Suponhamos, como hipótese de indução, que a condição se verifica para um certo valor de i_0, com $i_0 < n$.
    Então, será efetuada uma nova iteração (execução do bloco 4–6).
  - Pela hipótese de indução sobre o estado de *pmax* e i, concluimos que quando executamos o bloco 4-5, o valor de *pmax* será:
    - preservado se $v[i_0] \leq \max(v[k], \ldots, v[i_0 - 1])$
    - alterado para i_0 se $v[i_0] > \max(v[k], \ldots, v[i_0 - 1])$.
  - Portanto, se *pmax* tinha o indíce da primeira ocorrência do máximo de $v[k], \ldots, v[i_0 - 1]$ passará a ter o indíce da primeira ocorrência do máximo de $v[k], \ldots, v[i_0 - 1], v[i_0]$.
  - Na instrução 6, o valor de i passa a ser $i_0 + 1$, e a seguir a condição de paragem do ciclo (linha 3) volta a ser testada. Do que vimos acima, concluimos que a condição enunciada sobre o estado das variáveis se verifica para $i = i_0 + 1$ se se verificar para $i = i_0$.

Exemplo: Posição do máximo

Conclusão

Pelo princípio de indução, podemos concluir que a propriedade se verifica para todo o valor de $i \geq k+1$.

Mas, o ciclo termina quando $i = n+1$ e a seguir executa instrução 7

Sendo $i = n+1$, a propriedade diz que o valor de *pmax* é o índice da primeira ocorrência do máximo da sequência $v[k], v[k+1], \ldots, v[n]$.
A seguir, executa a instrução 7 (sai da função e retorna o valor *pmax*).
Portanto, o valor que a função retorna está correto.

Ordenação por seleção (selection sort)

Problema

Ordenar as primeiras n posições do vetor v por **ordem decrescente**, supondo que são indexadas a partir de 1.

```
SelectionSort(v, n)
Para cada k ← 1 até n − 1 fazer
    j ← PosMax(v, k, n);
    Se j ≠ k então
        aux ← v[k];
        v[k] ← v[j];
        v[j] ← aux;
```

Invariante de ciclo: Para todo $i \geq 1$, imediatamente após a i-ésima iteração do ciclo "Para" (e incremento de k), o valor da variável k é $i+1$, o vetor v contém exatamente os mesmos elementos que tinha antes da entrada no ciclo, embora possam estar em posições distintas, $v[1] \geq v[2] \geq \ldots \geq v[i]$ e, se $i < n$ então $v[i] \geq \max(v[i+1], \ldots, v[n])$.

Ordenação por inserção (insertion sort)

Ordenar $v[1], \ldots, v[n]$ por **ordem decrescente**, dados v e n.

$\textsc{InsertionSort}(v, n)$

$k \leftarrow 2;$
Enquanto $k \leq n$ fazer
    $x \leftarrow v[k]; \;\; j \leftarrow k - 1;$
    Enquanto $(j \geq 1 \wedge v[j] < x)$ fazer
        $v[j+1] \leftarrow v[j]; \;\; j \leftarrow j - 1;$
    $v[j+1] \leftarrow x; \;\; k \leftarrow k + 1;$

- Este algoritmo segue uma **abordagem incremental**: se $v^0[1], \ldots, v^0[n]$ denotar o estado inicial de v então, após a iteração i, em $v[1], \ldots, v[i+1]$ terá os valores $v^0[1], \ldots, v^0[i+1]$ mas por ordem decrescente.
- **Invariante de ciclo:** à entrada da i-ésima iteração do ciclo "Enquanto $k \leq n$", as primeiras i posições de v estão ordenadas e têm os valores de $v^0[1], \ldots, v^0[i]$ (por ordem decrescente) e as posições $v[i+1], \ldots, v[n]$ mantêm os valores originais (não foram analisadas). O valor de k é $i + 1$.

Ordenação por inserção (insertion sort)

Para mostrar esse invariante pode-se começar por analisar o segundo ciclo e provar que:

- **Invariante do ciclo "Enquanto ($j \geq 1 \wedge v[j] < x$)" para k fixo:**
  Supondo que $v^k[1], \ldots, v^k[n]$ é o estado inicial de v à entrada do ciclo (4–5) e que $v^k[1] \geq v^k[2] \geq \ldots \geq v^k[k-1]$, então os valores de k, x e de $v[k+1], \ldots, v[n]$ não são alterados no ciclo, sendo $x = v^k[k]$ e quando se vai testar a condição do ciclo pela m-ésima vez, tem-se:
  - o valor de j é $k - m$ e a posição $v[j+1]$ está "livre" (i.e., o seu valor está em x se $m = 1$ ou já foi copiado para $v[j+2]$ se $m > 1$)
  - $v[p] = v^k[p-1] > x$, para $j + 2 \leq p \leq k$,
  - $v[p] = v^k[p]$, para $1 \leq p \leq j$

$v^k[1]$	$v^k[2]$	$\cdots$	$v^k[j]$	****	$v^k[j+1]$	$\cdots$	$v^k[k-1]$

- **Conclusão:** Se, após o ciclo 4–5, inserir x na posição $j+1$ (linha 6), terá $v[1] \geq \cdots \geq v[j] \geq v[j+1] > v[j+2] \geq \cdots \geq v[k]$ (antes de incrementar k) e tais posições contêm os valores $v^0[1], \ldots, v^0[k-1], v^0[k]$ ordenados se à entrada do ciclo 3–6, as $k-1$ primeiras posições de v contiverem os $v^0[1], \ldots, v^0[k-1]$ ordenados.

Ordenação por inserção (insertion sort)

Complexidade temporal assintótica
Caraterização da ordem de grandeza de $T(n)$.

$\textsc{InsertionSort}(v, n)$	
$k \leftarrow 2;$	c_1: atribuição de valor a variável
Enquanto $k \leq n$ fazer	c_2: teste e transferências de controlo
$x \leftarrow v[k];\ \ j \leftarrow k-1;$	c_3: acesso à memória, cálculos, atribuição
Enquanto $(j \geq 1 \wedge v[j] < x)$ fazer	c_4: teste e transferência de controlo
$v[j+1] \leftarrow v[j];\quad j \leftarrow j-1;$	c_5: acesso à memória, cálculos, atribuição
$v[j+1] \leftarrow x;\quad k \leftarrow k+1;$	c_6: acesso à memória, cálculos, atribuição

c_1, c_2, c_3, c_4, c_5 constantes positivas.

$T(n)$ **no pior caso** e $T(n)$ **no melhor caso**

- O que carateriza as instâncias no pior caso e no melhor caso?
- Qual é a expressão de $T(n)$ em cada um desses casos?

Ordenação por inserção (insertion sort)

- Pior caso: v está ordenado por ordem crescente e os valores são todos distintos, ou seja, $v^0[1] < v^0[2] < \cdots < v^0[n]$.

  Para k fixo, a condição de paragem do ciclo "enquanto" (linha 4) é testada k vezes e a instrução 5 é executada $k-1$ vezes.

$$T(n) \leq c_1 + c_2 n + (c_3 + c_6)(n-1) + c_4 \sum_{k=2}^{n} k + c_5 \sum_{k=2}^{n} (k-1)$$

- Melhor caso: v está ordenado por ordem decrescente, $v^0[1] \geq \cdots \geq v^0[n]$.

  Para k fixo, a condição de paragem do ciclo "enquanto" (linha 4) é testada **uma vez** e a instrução 5 é executada **zero** vezes.

$$T(n) \geq c_1 + c_2 n + (c_3 + c_6)(n-1) + c_4 \sum_{k=2}^{n} 1 + c_5 \sum_{k=2}^{n} 0$$

Para a expressão no pior caso, notar que: $\sum_{k=2}^{n}(k-1) = 1 + 2 + \cdots + (n-1) = \sum_{k=1}^{n-1} k$.

Ordenação por inserção (insertion sort)

Recordar que $\sum_{k=a}^{b} k = \frac{b+a}{2}(b-a+1)$ e $\sum_{k=a}^{b} 1 = (b-a+1)$, se $b \geq a$. Caso contrário, o valor é 0.

$$c_1+c_2n+(c_3+c_6+c_4)(n-1) \leq T(n) \leq c_1+c_2n+(c_3+c_6)(n-1)+c_4\frac{n+2}{2}(n-1)+c_5\frac{n}{2}(n-1)$$

Tomando $c' = \min(c_1, c_2, \ldots, c_6)$ e $c'' = \max(c_1, c_2, \ldots, c_6)$, podemos escrever

$$4c'n - 2c' \leq T(n) \leq c'' + c''n + 2c''(n-1) + c''(n+1)(n-1)$$

Ou seja,

$$4c'n - 2c' \leq T(n) \leq c''n^2 + 3c''n - 2c''$$

e concluir que, para $n \geq 1$, se tem $2c'n \leq T(n) \leq 4c''n^2$ pois

$$4c'n - 2c'n \leq 4c'n - 2c' \leq T(n)$$

$$T(n) \leq c''n^2 + 3c''n - 2c'' \leq c''n^2 + 3c''n \leq c''n^2 + 3c''n^2 \leq 4c''n^2$$

Concluimos que o tempo de execução do algoritmo InsertionSort para instâncias de tamanho n satisfaz: $T(n) \in \Omega(n)$ e $T(n) \in O(n^2)$. Das expressões de $T(n)$, concluimos também que, no pior caso, $T(n) \in \Theta(n^2)$. No melhor caso, $T(n) \in \Theta(n)$.

Ordens de grandeza O, Θ e Ω

$O(g(n))$, $\Omega(g(n))$, e $\Theta(g(n))$ designam **conjuntos de funções**: todas as funções $f : \mathbb{N} \rightarrow \mathbb{R}_0^+$ que se relacionam com uma dada função $g : \mathbb{N} \rightarrow \mathbb{R}_0^+$ da forma indicada na definição correspondente.

$$O(g(n)) = \{f(n) \mid \text{existem } c > 0 \text{ e } n_0 > 0 \text{ tais que } f(n) \leq cg(n) \text{ para todo } n \geq n_0\}$$

$$\Omega(g(n)) = \{f(n) \mid \text{existem } c > 0 \text{ e } n_0 > 0 \text{ tais que } f(n) \geq cg(n) \text{ para todo } n \geq n_0\}$$

$$\Theta(g(n)) = \{f(n) \mid \text{existem } c_1 > 0,\ c_2 > 0 \text{ e } n_0 > 0 \text{ tais que } c_1 g(n) \leq f(n) \leq c_2 g(n) \text{ para todo } n \geq n_0\}$$

- $f(n) \in O(g(n))$ sse $f(n)$ é majorada por $cg(n)$ para alguma constante $c \in \mathbb{R}^+$, a partir de uma certa ordem (definida por n_0, com $n_0 \in \mathbb{Z}^+$ fixo).
- $f(n) \in \Omega(g(n))$ sse $f(n)$ é minorada por $cg(n)$ para alguma constante $c \in \mathbb{R}^+$, a partir de uma certa ordem (definida por n_0, com $n_0 \in \mathbb{Z}^+$ fixo).
- $f(n) \in \Theta(g(n))$ sse $f(n)$ é majorada por $c_2 g(n)$ e minorada por $c_1 g(n)$ para algum $c_1 \in \mathbb{R}^+$ e algum $c_2 \in \mathbb{R}^+ > 0$, a partir de uma certa ordem (definida por n_0, com $n_0 \in \mathbb{Z}^+$ fixo).