Exercícios - Análise de Algoritmos Iterativos

Prof. André Vignatti

Exercício 1. Expresse (**informalmente**, como visto em sala) a função $n^3/1000 - 100n^2 - 100n + 3$ em termos da notação Θ.

Exercício 2. Considere ordenar n números armazenados num vetor A encontrando primeiramente o menor elemento de A e troque-o com o elemento em $A[1]$. Em seguida, encontre o segundo menor elemento de A, e troque-o com $A[2]$. Continue usando a mesma ideia para os primeiro $n-1$ elementos de A.

(a) Escreva um pseudo-código para este algoritmo (que é conhecido como `Selection Sort`).

(b) Porque é preciso executar somente para os $n-1$ primeiro elementos, ao invés de n?

(c) Dê o tempo de execução de melhor-caso e pior-caso do `Selection Sort` na notação Θ.

Exercício 3. Considere o pseudo-código do Insertion Sort visto em aula. No início da linha 5 do pseudo-código, valem os seguintes invariantes:

(I2) $A[1..i]$ e $A[i+2..j]$ contém os elementos de $A[1..j]$ <u>antes</u> de entrar no laço que começa na linha 5.

(I3) $A[1..i]$ e $A[i+2..j]$ são crescentes.

(I4) $A[1..i] \leq A[i+2..j]$

(I5) $A[i+2..j] >$ chave.

Assim:

(a) Prove que todos os invariantes I2, I3, I4 e I5 estão corretos.

(b) Prove que a corretude dos invariantes I2 a I5 juntamente com a condição de parada na linha 5 e a atribuição na linha 7 implicam no invariante I1 visto em aula.

Exercício 4. Considere o seguinte algoritmo, que converte um número em sua representação binária:

Entrada: inteiro n
Saída: vetor b com a representação binária de n

```
início
    t ← n
    k ← −1
    enquanto t > 0 faça
        k ← k + 1
        b[k] ← t mod 2
        t ← t div 2
    retorna b
```

Considere o seguinte invariante: "Ao entrar no laço 4-7, o inteiro m representado pelo subvetor $b[0..k]$ é tal que $n = t \cdot 2^{k+1} + m$".

(a) Porque o invariante descrito é útil para mostrar a corretude do algoritmo acima?

(b) Prove que o invariante descrito está correto.

Exercício 5. Prove a corretude do seguinte algoritmo para encontrar o maior valor em um vetor $A[1..n]$:

Entrada: vetor $A[1..n]$
Saída: o maior elemento de A

```
início
    m ← A[1]
    para i ← 2 até n faça
        se A[i] > m então m ← A[i]
    retorna m
```

Exercício 6. Prove a corretude do seguinte algoritmo (conhecido como Bubblesort) para ordenar um vetor $A[1..n]$:

Entrada: vetor $A[1..n]$
Saída: o vetor A ordenado

```
início
    para i ← 1 até n − 1 faça
        para j ← 1 até n − i faça
            se A[j] > A[j + 1] então
                Troque A[j] com A[j + 1]
    retorna A
```

Exercício 7. Considere o seguinte problema de busca:

Entrada: Uma seqüência de n números $A = (a_1, a_2, \ldots, a_n)$ e um valor v.
Saída: Um índice i tal que $v = A[i]$ ou o valor especial `NIL` se v não aparece em A.

(a) Escreva um pseudo-código para a busca linear, que varre a seqüência procurando por v.

(b) Usando uma **invariante de laço**, prove que seu algoritmo é correto. Certifique-se que o invariante de laço cumpre as três propriedades necessárias (veja notas de aula).

(c) Quantos elementos da entrada devem ser verificados em média, assumindo que o elemento a ser buscado é igualmente provável de estar em qualquer posição do vetor?

(d) E com relação ao pior-caso?

(e) Qual é o tempo de execução da busca linear no caso médio e no pior caso usando a notação Θ.