Projeto e Análise de Algoritmos

Prof. Flávio L. C. de Moura*

Exercícios para a Prova 1

1. Sejam $f(n)$, $g(n)$ e $h(n)$ funções não-negativas tais que $f(n) = O(h(n))$ e $g(n) = O(h(n))$. Prove, utilizando as definições de notação assintótica, que $f(n) + g(n) = O(h(n))$.

   Solução Sabemos que $f(n) = O(h(n))$, i.e. existem constantes positivas c_1 e n_1 tais que $f(n) \leq c_1.h(n)$, para todo $n \geq n_1$. Analogamente, de $g(n) = O(h(n))$, i.e. existem constantes positivas c_2 e n_2 tais que $g(n) \leq c_2.h(n)$, para todo $n \geq n_2$. Queremos mostrar que $f(n) + g(n) = O(h(n))$, i.e. existem constantes positivas c e n_0 tais que $f(n) + g(n) \leq c.h(n)$, para todo $n \geq n_0$. Somando as desigualdades acima, temos $f(n) + g(n) \leq (c_1 + c_2).h(n)$, para todo $n \geq \max n_1, n_2$. Logo, basta tomar $c = c_1 + c_2$ e $n_0 = \max n_1, n_2$.

2. Sejam $f(n)$ e $g(n)$ funções não-negativas tais que $g(n) = O(f(n))$. Prove, utilizando as definições de notação assintótica, que $f(n) + g(n) = \Theta(f(n))$.

3. O pseudocódigo a seguir:

*flaviomoura@unb.br

Algorithm 1: BinarySearch($A[1..n], low, high, key$)

```
if high < low then
    return -1;
end
mid = ⌊(high + low)/2⌋;
if key > A[mid] then
    return BinarySearch(A, mid + 1, high, key);
end
else
    if key < A[mid] then
        return BinarySearch(A, low, mid − 1, key);
    end
    else
        return mid;
    end
end
```

(a) Faça a análise da complexidade do melhor caso para este algoritmo.

Solução. No melhor caso, o elemento procurado está na posição central, e portanto não teremos chamadas recursivas na execução do algoritmo. Logo $T_b(n) = \Theta(1)$.

(b) Faça a análise da complexidade do pior caso para este algoritmo.

Solução. No pior caso, o elemento procurado não se encontra no vetor. Neste caso, a complexidade é dada pela recursão $T_w(n) = T_w(n/2) + \Theta(1)$
que tem solução $T_w(n) = O(\lg(n))$ pelo TM.

(c) A correção deste algoritmo pode ser estabelecida em duas etapas. A primeira dela consiste em provar que se a chave key não ocorre no vetor $A[1..n]$, então BinarySearch($A[1..n], 1, n, key$) retorna o valor -1. Prove o lema a seguir:

Seja $A[1..n]$ um vetor ordenado de inteiros distintos. Mostre que se a chave key não ocorre em $A[1..n]$, então BinarySearch($A[1..n], 1, n, key$) retorna o valor -1.

4. Mostre como podemos ordenar n inteiros contidos no intervalo de 0 a $n^3 - 1$ em tempo linear, ou seja, em tempo $O(n)$.