Projeto e Análise de Algoritmos (2025-1)

Flávio L. C. de Moura*

13 de maio de 2025

Exercícios para a Prova 1

Teorema 0.1. *Sejam $a \geq 1$ e $b > 1$ constantes, $f(n)$ uma função assintoticamente positiva, e $T(n)$ definida nos inteiros não-negativos pela recorrência:*

$$T(n) = a.T(n/b) + f(n)$$

onde n/b deve ser interpretado como $\lfloor n/b \rfloor$ ou $\lceil n/b \rceil$. Então $T(n)$ tem as seguintes cotas assintóticas:

1. *Se $f(n) = O(n^{\log_b a - \epsilon})$ para alguma constante $\epsilon > 0$, então $T(n) = \Theta(n^{\log_b a})$.*

2. *Se $f(n) = \Theta(n^{\log_b a})$, então $T(n) = \Theta(n^{\log_b a}. \lg n)$.*

3. *Se $f(n) = \Omega(n^{\log_b a + \epsilon})$ para alguma constante $\epsilon > 0$, e se $a.f(n/b) \leq c.f(n)$ para alguma constante $c < 1$, então para todo n suficientemente grande, temos que $T(n) = \Theta(f(n))$.*

---

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 $f(n) + g(n) = O(h(n))$.

2. Sejam $f(n), g(n)$ e $h(n)$ funções dos inteiros não-negativos nos reais positivos. Mostre que se $f(n) = O(g(n))$ e $g(n) = O(h(n))$ então $f(n) = O(h(n))$.

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

*flaviomoura@unb.br

4. Prove que $\sum_{i=1}^{n} i^k = \Theta(n^{k+1})$ para qualquer inteiro $k \geq 0$ fixado.

5. O pseudocódigo a seguir:

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

```
1  if high < low then
2  |  return -1;
3  end
4  mid = ⌊(high + low)/2⌋;
5  if key > A[mid] then
6  |  return BinarySearch(A, mid + 1, high, key);
7  end
8  else
9  |  if key < A[mid] then
10 |  |  return BinarySearch(A, low, mid − 1, key);
11 |  end
12 |  else
13 |  |  return mid;
14 |  end
15 end
```

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

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

(c) A correção deste algoritmo pode ser estabelecida em duas etapas.

i. 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 a seguinte afirmação:

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.

ii. A segunda etapa consiste em provar que se a chave key ocorre no vetor $A[1..n]$, então BinarySearch($A[1..n], 1, n, key$) retorna uma posição válida do vetor, digamos k, tal que $A[k] = key$. Prove a seguinte afirmação:

Seja $A[1..n]$ um vetor ordenado de inteiros distintos. Mostre que se a chave key ocorre no vetor $A[1..n]$, então BinarySearch($A[1..n], 1, n, key$) retorna o valor $1 \leq k \leq n$, tal que $A[k] = key$.

6. Considere o problema dos elementos únicos, que checa se todos os n elementos de um vetor de tamanho são distintos:

Algorithm 2: EUnicos($A[0..n-1]$)

```
1 for i = 0 to n − 2 do
2 |   for j = i + 1 to n − 1 do
3 |   |   if A[i] = A[j] then
4 |   |   |   return false
5 |   |   end
6 |   end
7 end
8 return true
```

Faça a análise assintótica deste algoritmo, e justifique sua resposta.

7. Considere um algoritmo recursivo hipotético que resolve uma classe de problemas da seguinte forma:

   (a) A instância do problema de tamanho n recebida como entrada é dividida em 4 subproblemas de tamanho $n/2$, que são por sua vez resolvidas recursivamente;

   (b) As soluções dos 4 subproblemas são combinadas em tempo $n^2 . \lg n$, para que seja possível formar uma solução do problema original.

   Responda os itens a seguir:

   (a) Escreva a recorrência que modela a complexidade deste algoritmo.

   (b) Determine a complexidade assintótica deste algoritmo hipotético.