Projeto e Análise de Algoritmos (2023-2)

Primeira avaliação

Prof. Flávio L. C. de Moura

30 de outubro de 2023

1. O pseudocódigo para o algoritmo recursivo de busca binária em um vetor ordenado de inteiros com n elementos é dado a seguir:

```
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
```

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

   (a) (2.5 pontos) Faça a análise da complexidade do melhor caso para este algoritmo.

   (b) (2.5 pontos) Faça a análise da complexidade do pior caso para este algoritmo.

   (c) (2.5 pontos) 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.

   Dica: Indução (forte) sobre o tamanho n do vetor.

2. (2.5 pontos)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 $f(n) + g(n) = O(h(n))$.

1(a) No melhor caso, o elemento procurado está na posição central do vetor A. Depois de duas comparações (linhas 5 e 9), o algoritmo para e retorna mid. Assim, no melhor caso, o custo $T_{bb}^{b}(n)$ da busca binária em um vetor com n elementos é constante, ou seja, $T_{bb}^{b}(n) = \theta(1)$.

1(b) No pior caso, o elemento procurado não está no vetor A, ou está na última posição possível na busca. A análise pode ser modelada pela seguinte recorrência:

$$T_{bb}^{w}(n) = T_{bb}^{w}(n/2) + \theta(1).$$

Pelo TM, $f(n) = \theta(1) = \theta(n^{\log_2 1})$

e pelo caso 2, $T_{bb}^{w}(n) = \theta(n^{\log_2 1} \cdot \lg n) = \theta(\lg n)$.

1(c) Suponha que Key não ocorre em $A[1..n]$. Temos 2 casos:

① Key > A[mid] (linha 5): Neste caso, temos por hipótese de indução que BinarySearch(A, mid+1, n, Key) retorna -1, e portanto o BinarySearch(A, 1, n, Key) retorna -1 como desejado.

② Key < A[mid] (linha 9): Neste caso, temos por hipótese de indução que BinarySearch(A, 1, mid-1, Key) retorna -1, e portanto o BinarySearch(A, 1, n, Key) retorna -1 como desejado.

② Por hipótese, temos que $f(n) = O(h(n))$ e $g(n) = O(h(n))$. No primeiro caso, existem constantes positivas c_1 e n_1 tais que $f(n) \leq c_1 \cdot h(n)$, $\forall n \geq n_1$. No segundo caso, existem constantes positivas c_2 e n_2 tais que $g(n) \leq c_2 \cdot h(n)$, $\forall n \geq n_2$. Somando as duas desigualdades para $n \geq \max(n_1, n_2)$, temos

$$f(n) + g(n) \leq (c_1 + c_2) \cdot h(n), \quad \forall n \geq \max(n_1, n_2)$$

ou seja, $f(n) + g(n) = O(h(n))$. ▨