Universidade Federal de Pernambuco
Centro de Informática
Bach. em Ciência da Computação

IF672cc Algoritmos e Estr. de Dados 2017.2
Prof. Paulo Fonseca

Primeira Prova — 28 de Setembro de 2017

- Esta prova tem 04 questões.
- A duração da prova é de 02h00min.

■ **QUESTÃO 1** Considere o problema de encontrar o elemento de ordem k de um vetor dado $V = (v_0, \ldots, v_{n-1})$. O elemento de ordem $k = 0$ é o menor elemento, o elemento de ordem $k = 1$ é o segundo menor elemento, e assim sucessivamente, até o maior elemento quando $k = n - 1$. O Algoritmo Quicksort *visto em aula* pode ser adaptado para esse problema da seguinte maneira: a cada passo, escolha o pivô e particione o trecho correspondente, retornando a posição final p do pivô. Se $p = k$, então retorne o pivô. Se $p < k$ então continue a busca no trecho à direita do pivô. Se $k < p$, então continue a busca à esquerda do pivô.

a) Ilustre a execução do algoritmo acima para $k = 3$ sobre o vetor

$$V = (12, 15, 4, 0, 11, 1, 2, 9, 7, 10),$$

exibindo o vetor V após cada chamada à função *partition*, e indicando o valor retornado. Suponha que o elemento escolhido para pivô é o elemento mais à esquerda do trecho a ser particionado. ▷ *1,5 pt*

b) Qual o custo assintótico do pior caso desse algoritmo? Justifique sucintamente (máx. 05 linhas). ▷ *0,5 pt*

c) Supondo que a função *partition* sempre produz partições perfeitamente equilibradas, com o pivô terminando bem no meio do trecho particionado, qual o custo assintótico do pior caso do algoritmo? Justifique sucintamente (máx. 05 linhas). ▷ *0,5 pt*

■ **QUESTÃO 2** Considere uma árvore AVL aumentada na qual cada nó N, além da sua chave $N.val$ e do seu fator de balanço $N.bf$, possui um campo $N.rank$ que indica qual a ordem da sua chave dentre os valores na sub-árvore enraizada em N, ou equivalentemente, quantos nós existem na sub-árvore à esquerda de N. As inserções nessa AVL são feitas quase da mesma forma que numa AVL 'normal', com as seguintes modificações. Uma nova folha é sempre inserida com $rank = 0$; quando fazemos uma inserção à esquerda de um nó N, seu $rank$ aumenta uma unidade; e a cada rotação (à esquerda ou à direita), o $rank$ de um nó precisa ser atualizado, o que pode ser feito em tempo constante.

a) Ilustre a AVL aumentada resultante da inserção dos itens do vetor V da Questão 1.a, na mesma ordem do vetor. Para cada nó, exiba o seu valor e o seu $rank$. ▷ *1,5 pt*

Essa árvore AVL aumentada também pode ser utilizada para encontrar o elemento de ordem k, como discutido na Questão 1. Para tal, basta fazer a busca na AVL baseada no $rank$. Iniciando pela raiz ($N = root$), se $N.rank = k$, retorna $N.val$; se $k < N.rank$, busca o elemento de ordem k à esquerda de

N; se $N.rank < k$, procura o elemento de ordem $k' = k - (N.rank + 1)$ à direita de N.

b) Ilustre a procura pelo elemento de ordem $k = 6$ na árvore resultante da letra (a), indicando quais os nós visitados. ▷ *0,5 pt*

c) Indique o custo assintótico no pior caso para procurar o elemento de ordem k numa AVL aumentada de n elementos *já construída*. Justifique sucintamente (máx. 05 linhas). ▷ *0,5 pt*

■ **QUESTÃO 3** Outra possível alternativa para o problema de encontrar o elemento de ordem k de um vetor $V = (v_0, \ldots, v_{n-1})$, discutido na Questão 1, consiste em, primeiro, construir uma min-heap binária a partir do vetor V de modo *offline*, e em seguida realizar $k+1$ extrações sucessivas, retornando o último valor extraído.

Considere a execução desse algoritmo sobre a entrada da Questão 1.a ($k = 3$), e represente

a) O processo de construção da min-heap. ▷ *1,0 pt*

b) As extrações sucessivas. ▷ *1,0 pt*

c) Indique a complexidade assintótica do pior caso desse algoritmo, justificando sucintamente (máx. 05 linhas) ▷ *0,5 pt*

■ **QUESTÃO 4** Responda os itens abaixo considerando o algoritmo a seguir.

a) Ilustre a execução do Algoritmo R sobre a entrada

$head \to \backslash, 0, 15, 1, 5, 10, 14, 6, 9, 13, 4;\ v = 6.$

Exiba a lista encabeçada por *shead* ao final do algoritmo, assim como o valor retornado. ▷ *0,5 pt*

b) Descreva '*o que*' (e não '*como*') o Algoritmo R calcula. (máx. 03 linhas) ▷ *1,0 pt*

c) Qual a complexidade assintótica do Algoritmo R, no pior caso, para uma lista de entrada de n elementos? Justifique sucintamente (máx. 05 linhas). ▷ *1,0 pt*

Algoritmo *R*
Entrada *head*: ptr. para uma lista encadeada de inteiros (com sentinela);
v: um inteiro
Saída ???

```
1  shead ← new_list ()   ▷ retorna ptr para nó sentinela de nova lista
2  cur ← head→next
3  enquanto cur ≠ ⊥ faça:
4      scur ← shead
5      enquanto scur→next ≠ ⊥ e
       cur→val > scur→next→val faça
6          scur ← scur→next
7      fim faça
8      list_insert(scur, cur→val)
9      cur ← cur→next
10 fim faça
11 r ← 0
12 scur ← shead→next
13 enquanto scur ≠ ⊥ faça
14     se scur→val = v então
15         devolva r
16     fim se
17     r ← r + 1
18     scur ← scur→next
19 fim faça
20 devolva -1
fim
```

■ ■ ■