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

**IF672ec Algoritmos e Estr. de Dados 2019.1**
Prof. Paulo Fonseca

## Primeira Prova — 22 de Abril de 2019 (Descobrimento do Brasil)

- Esta prova tem 04 questões.
- A duração da prova é de 02h00min.
- Pode responder a lápis porém de forma clara e legível.
- Se possível, responder em ordem usando uma face da folha de respostas para cada questão.
- Devolva apenas a folha com as respostas devidamente identificada com seu nome.

### QUESTÃO 1 (2,5pt)

Responda sobre o algoritmo a seguir.

a) Ilustre a execução do algoritmo sobre a lista

$$\backslash \to 6 \to 8 \to 1 \to 3 \to 7 \to 5 \to 2 \to 4$$

exibindo a lista após cada chamada à função *func* (linha 4 do Alg. *enigma*)
▷ *0,5 pt*

b) O que o algoritmo faz? (máx. 02 linhas)
▷ *0,5 pt*

c) Explique brevemente qual o *melhor caso* e o qual o *pior caso* do algoritmo em função do tamanho da lista de entrada $n$, e forneça a complexidade assintótica exata de cada um deles. (máx. 08 linhas)
▷ *1,5 pt*

**Algoritmo** *enigma*
**Entrada** *head*: ptr p/ cabeça de lista com sentinela
*tail*: ptr p/ último elto. da lista
**Saída** ???

```
1  se head = tail or head→next = tail então
2      devolva
3  fim se
4  hp ← func(head, tail)
5  enigma(head, hp)
6  enigma(hp→next, tail)
fim
```

**Algoritmo** *func*
**Entrada** *head*, *tail*
**Saída** ???

```
1  pv ← tail→val
2  lp ← head
3  cur ← head
4  enquanto cur→next ≠ tail faça
5      se cur→next→val < pv então
6          tmp ← lp→next→val
7          lp→next→val ← cur→next→val
8          cur→next→val ← tmp
9          lp ← lp→next
10     fim se
11     cur ← cur→next
12 fim faça
13 tail→val ← lp→next→val
14 lp→next→val ← pv
15 devolva lp
fim
```

### QUESTÃO 2 (1,5pt)

Um *array associativo* é um TAD usado para armazenar uma coleção de valores, cada um associado a uma chave numérica única no intervalo $0, \ldots, s-1$. A operação $Insert(k, v)$ insere o valor $v$ associado à chave $k$.

Considere a implementação do TAD como uma *hashtable* na qual são armazenados os pares chave-valor $(k, v)$, usando apenas o $k$

como entrada para a função de dispersão. Supondo uma tabela de tamanho $m = 11$ com *double hashing* e funções de dispersão

$$h_0(k) = k \bmod m \quad \text{e} \quad h_1(k) = (2k-1) \bmod 7,$$

ilustre a sequência de inserções abaixo, exibindo a tabela após cada operação.

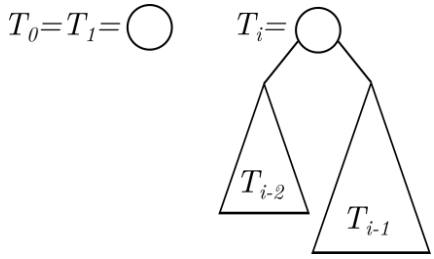
1. $Insert(14, R)$
2. $Insert(54, L)$
3. $Insert(76, S)$
4. $Insert(22, B)$
5. $Insert(91, I)$
6. $Insert(77, A)$.

## QUESTÃO 3 (4,0pt)

Considere uma sequência de BSTs $T_0, T_1, T_2, \ldots$ A árvore $T_i$ possui $N_i$ nós com valores $1, \ldots, N_i$, sendo

$$N_i = \begin{cases} 1, \text{ se } i = 0 \text{ ou } i = 1 \\ 1 + N_{i-1} + N_{i-2}, \text{ se } i > 1. \end{cases}$$

A estrutura da árvore $T_i$ está definida também recursivamente da seguinte forma




a) Forneça a enumeração de $T_4$ em pré-ordem. ▷ *0,5 pt*

b) Forneça uma expressão para a altura $H_i$ da árvore $T_i$, para $i \geq 0$. ▷ *0,5 pt*

c) Cada árvore $T_i$ é uma AVL? Justifique sucintamente. (máx. 4 linhas) ▷ *0,5 pt*

d) Ilustre a inserção do valor 7.5 (sete e meio) em $T_4$. Caso tenha respondido afirmativamente o item anterior, considere a inserção como numa AVL e ilustre as rotações eventualmente necessárias. ▷ *1,5 pt*

e) O Prof. Sogupa acha que descobriu uma propriedade interessante das BSTs. Suponha que uma busca por uma chave $k$ percorra um caminho da raiz até uma folha. Considere três conjuntos de nós: o conjunto $A$ é composto de todos os nós *à esquerda* do caminho da busca, o conjunto $B$ é formado pelos nós no caminho da busca, e o conjunto $C$ é formado por todos os nós *à direita* do caminho de busca. Então para todo $a \in A, b \in B, c \in C$ temos $a \leq b \leq c$. Você acha que o Prof. está certo? Se sim, justifique sucintamente (máx. 05 linhas). Se não, forneça um contra-exemplo da propriedade através da menor BST possível definida como no enunciado, isto é, desenhe $T_i$ e indique o valor da chave $k$, e os valores $a$, $b$ e $c$ para os quais a propriedade não se verifica. ▷ *1,0 pt*

## QUESTÃO 4 (2,0pt)

A extração do elemento de uma posição arbitrária $i$ de uma heap binária pode ser implementada de maneira semelhante à extração do elemento de maior prioridade. Primeiro, permuta-se o elemento da posição $i$ com o último elemento, e logo desconsidera-se a última posição. Em seguida é necessário fazer o novo elemento da posição $i$ mover-se até o nível adequado. A diferença é que agora pode ser necessário subir ou descer.

Ilustre a extração dos elementos 25 e 8 da max-heap

$$H = (30, 25, 29, 14, 20, 28, 13, 7, 8, 19, 6, 27, 2)$$

exibindo-a na forma de árvore após cada remoção.

■ ■ ■