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

IF672cc Algoritmos e Estr. de Dados 2022.2
Prof. Paulo Fonseca

Primeira Prova — 02 de Março de 2023

QUESTÃO 1 (2,0pt)

Considere o algoritmo a seguir.

Algoritmo *enigma*
Entrada *head* cabeça de lista de inteiros com sentinela; s inteiro

```
1  fst ← head
2  enquanto fst→next→next ≠ ⊥ faça
3      a ← fst→next→val
4      snd ← fst→next
5      enquanto snd→next ≠ ⊥ faça
6          b ← snd→next→val
7          se a + b = s então
8              print(a, b)
9          fim se
10         snd ← snd→next
11     fim faça
12     fst ← fst→next
13 fim faça
fim
```

a) Ilustre a execução do algoritmo sobre a entrada constituída pela lista contendo os números $(0,1,2,3,4,5)$ e $s = 5$, reproduzindo o que ele imprime.

b) Descreva brevemente '***o que***' esse algoritmo faz (e não 'como').

c) Qual o custo assintótico do algoritmo no pior caso em função do tamanho n da lista de entrada? Justifique sucintamente.

QUESTÃO 2 (2,0pt)

Classifique cada afirmação a seguir como Verdadeira (V) ou Falsa (F). ***Cada alternativa errada anula uma correta.***

a) O MergeSort é um algoritmo do tipo ***dividir para conquistar***.

b) O MergeSort tem complexidade de ***tempo*** $\Theta(n)$ no ***melhor*** caso.

c) O MergeSort tem complexidade de ***espaço*** $\Theta(n)$ no ***pior*** caso.

d) Considere a execução do MergeSort sobre a entrada $V = (3,4,7,6,2,1,0,5)$. Imediatamente após a ***terceira*** execução da função ***merge***, o vetor estará na forma $V = (3,4,6,7,1,2,0,5)$.

e) O QuickSort tem complexidade de ***tempo*** $\Theta(n \lg n)$ no ***pior*** caso.

f) O pior caso do QuickSort ocorre sempre que o vetor de entrada já está ordenado.

g) Se usarmos como entrada do QuickSort (versão vista em aula) o vetor $V = (3,4,7,6,2,1,0,5)$, e o pivô escolhido for o elemento mais ***à esquerda*** do trecho a ser particionado, então, após a ***primeira*** partição, o vetor estará na forma $V = (2,0,1,3,6,7,4,5)$

h) O QuickSort é um algoritmo ***estável***, ou seja, preserva a ordem relativa de elementos considerados "iguais" da entrada.

i) A busca binária é um algoritmo que pode ser usado em qualquer vetor de entrada e tem complexidade de tempo $O(\lg n)$ no pior caso.

j) O melhor caso da busca binária ocorre quando buscamos um valor que não pertence ao vetor de entrada.

QUESTÃO 3 (2,0pt)

Considere uma ***hashtable*** fechada com tamanho m ***primo***, com resolução de colisões por ***sondagem linear*** e função de dispersão

$$h(k,i) = (h_0(k) + i) \bmod m,$$

onde

$$h_0(k) = (ak + b) \bmod m.$$

Imediatamente ***antes*** da inserção de um novo valor k, deve ser feita a verificação se o fator de carga atingiu o limite $\alpha \geq 0.5$. Caso isso seja verdade, deve ser realizado um ***rehashing*** antes da inserção de k. O novo tamanho da tabela deve ser o menor primo $m' > 2m$.

Ilustre a inserção dos valores

$$7,\ 13$$

na tabela de tamanho $m = 5$ abaixo, considerando $a = 7, b = 5$.

$$T = \begin{array}{|c|}\hline 10 \\ \hline \\ \hline \\ \hline 2 \\ \hline \end{array}$$

QUESTÃO 4 (2,0pt)

a) Represente a árvore AVL cuja enumeração em pós-ordem é

$$4, 2, 8, 12, 10, 6, 16, 18, 14.$$

b) Represente a inserção do valor 7 na árvore do item (a), ilustrando todas as rotações necessárias.

QUESTÃO 5 (2,0pt)

a) Ilustre a construção ***offline*** de uma ***min-heap*** a partir do array

$$H = (O, M, T, I, R, O, G, L, A)$$

Considere a comparação dos caracteres em ordem alfabética $A \leq B \leq \cdots \leq Z$.

b) Ilustre a inserção do caractere B na heap resultante do item (a).

■ ■ ■