CSI 103 – Algoritmos e Estruturas de Dados I

2ª Lista de Exercícios

Prof. Mateus Ferreira Satler

DECSI – ICEA - UFOP

Obs. 1: para todos os exercícios de **codificação (criação de código na linguagem C)** a seguir, utilize a modularização em C, ou seja, crie arquivos separados **.h** e **.c**. Crie 1 projeto separado para cada exercício (não faça mais de um exercício dentro do mesmo código).

Obs. 2: os exercícios **3, 4, 5, 6, 7, 8, 9, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23** e **24** **NÃO TRATAM DE GERAÇÃO DE CÓDIGO NA LINGUAGEM C**. Resolva os exercícios à mão em uma folha ou usando qualquer editor de texto de sua preferência. Gere **1 arquivo PDF PARA CADA EXERCÍCIO EM SEPARADO (NÃO COLOQUE TODOS OS EXERCÍCIOS EM UM MESMO PDF).** Os envios que estiverem em desacordo com esse padrão **SERÃO DESCONSIDERADOS.**

1. Reescreva os algoritmos *BubbleSort*, *SelectionSort* e *InsertionSort* vistos para que as chaves se ordenem de forma **decrescente**.

2. Escreva um procedimento que receba um vetor com registros que contém uma chave e imprima **ORDENADO** se o vetor estiver em ordem crescente. Caso contrário, imprima **NAO ORDENADO**.

3. Mostre o passo a passo da ordenação dos vetores abaixo utilizando o algoritmo *ShellSort* com os incrementos especificados:

a) 81 94 11 96 12 35 17 95 28 58 41 75 15 – Incrementos (valores de **h**) = {1, 3, 5}

b) 17 25 49 12 18 23 45 38 53 42 27 13 11 28 10 14 – Incrementos (valores de **h**) = {1, 2, 4}

4. Mostre um exemplo que demonstra que o *ShellSort* é instável para incrementos h={1,2}.

5. Mostre o passo a passo da ordenação dos vetores abaixo utilizando o algoritmo *MergeSort*:

a) 81 94 11 96 12 35 17 95 28 58 41 75 15

b) 17 25 49 12 18 23 45 38 53 42 27 13 11 28 10 14

6. Dada a sequência de números: 3 4 9 2 5 8 2 1 7 4 6 2 9 8 5 1, ordene-a em ordem não decrescente segundo o algoritmo *MergeSort*, apresentando a sequência obtida após cada passo do algoritmo.

7. Mostre o passo a passo da ordenação dos vetores abaixo utilizando o algoritmo *QuickSort*:

a) 81 94 11 96 12 35 17 95 28 58 41 75 15

b) 17 25 49 12 18 23 45 38 53 42 27 13 11 28 10 14

8. Refaça o exercício anterior, considerando que o pivô é escolhido através da mediana de três elementos do vetor. Mostre o passo a passo e a escolha dos pivôs.

9. Dada a sequência de números: 3 4 9 2 5 8 2 1 7 4 6 2 9 8 5 1, ordene-a em ordem não decrescente segundo o algoritmo *QuickSort*, apresentando a sequência obtida após cada passo do algoritmo.

10. Escreva um algoritmo para determina se uma árvore binária é:

a) Completa
b) Balanceada
c) Perfeitamente Balanceada

11. Escreva uma função que conta o número de nós **folha** de uma árvore binária.

12. Projete uma estrutura do tipo Árvore Binária de Busca para armazenar produtos de uma loja de conveniência com os seguintes campos: **código do produto** (inteiro) e **preço** (real). A árvore será ordenada pelo código dos produtos.

a) Escreva um procedimento **`adicionar`**, que pede ao usuário para digitar as informações via teclado e posteriormente cria e insere um nó na árvore binária, indexando o produto pelo código.

b) Escreva um procedimento **`preco`**, que busca na árvore binária pelo código do produto desejado e imprime na tela seu preço correspondente, no seguinte formato: **`O PRODUTO 132 CUSTA R$ 3.79.`** Se o produto não existir, imprima na tela: **`PRODUTO 642 NÃO ENCONTRADO.`**

c) Escreva uma função remover, que pede ao usuário para informar um código de produto e remove esse produto da árvore, caso ele exista e imprima na tela: **`O PRODUTO 132 FOI REMOVIDO COM SUCESSO.`** Se o produto não existir, imprima na tela: **`PRODUTO 642 NÃO ENCONTRADO.`**

d) Escreva uma função que retorne o valor **máximo** em uma árvore binária de busca. Faça também uma função que retorne o valor **mínimo** dessa árvore binária de busca. Por último, faça um pequeno programa em **C** para testar sua função.

13. Escreva uma função que retorne o valor **máximo** em uma árvore binária de busca. Faça também uma função que retorne o valor **mínimo** dessa árvore binária de busca. Por último, faça um pequeno programa em **C** para testar sua função.

14. Insira os números abaixo na ordem que são apresentados numa árvore binária de busca balanceada (AVL). Mostre todos os passos.

`20; 30; 25; 84; 56; 12; 1; 69; 78`

15. Considerando a árvore obtida no exercício anterior, mostre o passo-a-passo de uma busca realizada para os valores **69** e **81**.

16. Ainda considerando a árvore gerada no exercício **14.**, remova os números abaixo na ordem que são apresentados. Mostre todos os passos.

`25; 20; 1; 30; 78; 56; 12; 84; 69`

17. Insira os números abaixo na ordem que são apresentados numa árvore vermelho-preto esquerdista. Mostre todos os passos.

`41; 38; 31; 12; 19; 8`

18. Considere a árvore rubro-negra caída para a esquerda cujo percurso em nível (em largura) é: 67 – 51 – 87 – 23 – 53 – 82 – 90 – 17 – 31 – 52 – 60 – 16 – 21. Liste as chaves em nós rubros em ordem crescente. Lembre-se da regra: para cada nó, todos os caminhos desse nó para os nós folhas descendentes contém **o mesmo número de nós pretos**.

19. Mostre, passo a passo, o resultado de inserir as chaves “ATO”, “ATOR”, “ATRIZ”, “ATROCIDADE”, “ANTIGO”, “ALGORITMO”, “ANTIGA”, “AMÉRICA”, “AMERICANO”, “ALGOL”, “AMIGO”, “AMIZADE”, “AMIGA”, “ALISTAMENTO”, “ALISTAR” em uma árvore PATRICIA inicialmente vazia.

20. Dada a árvore do exercício anterior, mostre o passo a passo para buscar as chaves “AMIZADE”, “ANTIGA”, “AMIGOS”.

21. Considerando ainda a árvores Patricia resultante do exercício **19.**, remova as chaves “AMIGO”, “ANTIGA”, “ATROCIDADE”, “ALISTAR”, “ATO”. Mostre o passo a passo.

22. O vetor **`161 41 101 141 71 91 31 21 81 17 16`** é um Heap? Desenhe a árvore e justifique sua resposta.

23. Ordene as seguintes sequências de número em ordem não decrescente, utilizando o algoritmo **HeapSort**, apresentado a sequência dos números e explicando cada passo do algoritmo.

a) 3 4 9 2 5 1 8
b) 3 4 9 2 5 8 2 1 7 4 6 2 9 8 5 1

24. A função **HeapSort** produz um rearranjo **estável** do vetor, ou seja, preserva a ordem relativa de elementos de mesmo valor? Dê um exemplo para justificar sua resposta.