Disciplina: BC1424 - Algoritmos e Estruturas de Dados I
Prof[a]. Letícia Rodrigues Bueno

Lista de Exercícios

1. Aplique o algoritmo de ordenação por inserção sobre a sequência 13 50 43 95 59 72 41 3 e mostre passo-a-passo a ordenação sendo efetuada.

2. Escreva uma versão recursiva do algoritmo de ordenação por inserção.

3. Aplique o algoritmo de ordenação por seleção sobre a sequência 73 57 49 87 99 42 51 7 e mostre passo-a-passo a ordenação sendo efetuada.

4. Escreva uma versão recursiva do algoritmo de ordenação por seleção.

5. Aplique o algoritmo de ordenação por bolha sobre a sequência 83 46 12 18 27 87 32 11 e mostre passo-a-passo a ordenação sendo efetuada.

6. Um algoritmo de ordenação é **estável** se preserva ordem relativa de itens com valores idênticos. Responda: ordenação por inserção, seleção e bolha são estáveis? Exemplifique. Para cada um deles que não for estável, é possível modificá-lo para se tornar estável?

7. Modifique os três algoritmos de ordenação (inserção, seleção e bolha) para fazer ordenação decrescente.

8. Aplique o *Mergesort* sobre a sequência 43 79 63 85 29 62 51 38 e mostre passo-a-passo a ordenação sendo efetuada.

9. *Mergesort* é estável? Exemplifique. Se não é estável, como podemos modificá-lo para ser estável?

10. Modifique o algoritmo recursivo *Mergesort* visto em aula para ordenar um vetor dado em ordem decrescente.

11. A sequência 33 32 28 31 26 29 25 30 27 é um heap?

12. A sequência 33 32 28 31 29 26 25 30 27 é um heap?

13. Aplique o *Heapsort* sobre a sequência 33 39 60 95 28 66 70 78 e mostre passo-a-passo a ordenação sendo efetuada.

14. Um vetor ordenado em ordem decrescente é um *heap* máximo? Explique.

15. *Heapsort* é estável? Exemplifique. Se não é estável, como podemos modificá-lo para ser estável?

16. Escreva uma função que decida se um vetor $V[1..m]$ é um *heap* máximo.

17. Modifique o *heapsort* para fazer ordenação decrescente. Dica: utilize um *heap* mínimo.

18. Escreva uma função que decida se um vetor $V[1 \ldots n]$ é um *heap* máximo.

19. O que você acha de ordenarmos um vetor em ordem decrescente com o objetivo de transformá-lo em um *heap*?

20. Aplique o *Quicksort* não-aleatório sobre a sequência 98 73 68 12 57 61 31 48 e mostre passo-a-passo a ordenação sendo efetuada.

21. Aplique o *Quicksort* não-aleatório sobre a sequência 10 10 10 10 10 10 10 10 e mostre passo-a-passo a ordenação sendo efetuada.

22. *Quicksort* não-aleatório é estável? Exemplifique. Se não é estável, como podemos modificá-lo para ser estável?

23. Modifique o *Quicksort* não-aleatório para fazer ordenação decrescente.

24. Modifique o *Quicksort* não-aleatório visto em sala de aula para uma versão aleatória (conforme discutido em sala de aula).

25. Qual é a principal propriedade de uma árvore binária de busca?

26. Escreva os algoritmos recursivos para percorrer uma árvore binária de busca na forma: in-ordem, pré-ordem e pós-ordem.

27. Escreva um algoritmo recursivo para calcular a altura de uma árvore binária. Dica: utilize um dos algoritmos do exercício anterior.

28. Escreva um algoritmo recursivo que imprima as chaves de uma árvore binária de busca em ordem crescente.

29. Escreva um algoritmo recursivo que procure uma chave x em uma árvore binária de busca. No máximo quantas comparações o algoritmo fará se a árvore é degenerada? No máximo quantas comparações o algoritmo fará se a árvore é balanceada? Justifique.

30. Descrever um algoritmo para remover uma dada chave de uma árvore binária de busca. A complexidade do algoritmo deve ser da ordem da altura da árvore.

Bibliografia Utilizada

CORMEN, T. H.; LEISERSON, C. E.; RIVEST, R. L. e STEIN, C. *Introduction to Algorithms*, 3^a edição, MIT Press, 2009.

SZWARCFITER, J. L. e MARKENZON, L. Estruturas de Dados e seus Algoritmos, LTC, 1994.

ZIVIANI, N. Projeto de Algoritmos: com implementações em Java e C++, 1a edição, Cengage Learning, 2009.