Lista de Exercícios

1. Qual é a principal propriedade de uma árvore binária de busca?
2. Escreva os algoritmos recursivos para percorrer uma árvore binária de busca na forma: in-ordem, pré-ordem e pós-ordem.
3. Escreva um algoritmo recursivo para calcular a altura de uma árvore binária. Dica: utilize um dos algoritmos do exercício anterior.
4. Escreva um algoritmo recursivo que imprima as chaves de uma árvore binária de busca em ordem crescente.
5. 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.
6. 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.
7. Quais são as propriedades de uma árvore AVL?
8. Descreva o algoritmo de inserção em uma árvore AVL, justificando porque é válido efetuar o balanceamento da árvore.
9. Descreva o algoritmo de remoção em uma árvore AVL, justificando porque é válido efetuar o balanceamento da árvore.
10. Mostrar que a rotação dupla esquerda pode ser obtida por uma rotação direita seguida por uma rotação esquerda.
11. Mostrar que a rotação dupla direita pode ser obtida por uma rotação esquerda seguida por uma rotação direita.
12. Quais são as propriedades de uma árvore rubro-negra?

13. Desenhar a árvore rubro-negra obtida pela sequência de inserções das chaves 19, 18, 16, 15, 17, 2, 6 nessa ordem.

14. Desenhar uma árvore rubro-negra que contém dois caminhos da raiz às folhas tal que um caminho é o dobro do tamanho do outro.

15. Dê um exemplo de inserção em árvore rubro-negra cuja recoloração dos nós se propaga até a raiz.

16. Provar ou dar contra-exemplo: Seja uma árvore rubro-negra cuja raiz possui a cor rubra. Se esta for alterada para negra, a árvore mantém-se rubro-negra.

17. Compare árvores AVL e árvores rubro-negras.

18. Prove ou dê contra-exemplo:

    (a) Toda árvore rubro-negra é AVL;

    (b) Toda árvore AVL é completa;

    (c) Toda árvore rubro-negra é completa.

19. No máximo quantas comparações são necessárias para determinar se uma chave x está em uma árvore *trie*? Justifique.

20. Descreva o algoritmo de inserção em uma árvore *trie*.

21. Descreva o algoritmo de remoção em uma árvore *trie*.

22. Descreva o algoritmo de busca em uma árvore *trie*.

23. A partir de um prefixo p fornecido, escreva um algoritmo que imprima até 10 sugestões de chaves válidas (isto é, contidas na árvore *trie*) que tenham p como prefixo. Forneça as palavras em ordem alfabética.

24. Para um conjunto de n chaves x formada pelos n primeiros múltiplos de 7, quantas colisões são obtidas para as funções *hash* abaixo?

    (a) $x \pmod 7$

    (b) $x \pmod{14}$

    (c) $x \pmod 5$

25. Quais características são desejáveis em uma função *hash*?

26. Responder se é certo ou errado:
O fator de carga de qualquer tabela de dispersão é no máximo 1. Justifique.

27. Dada uma tabela *hash* onde as colisões são resolvidas usando encadeamento exterior, considere a implementação por: (a) listas encadeadas; (b) árvore binária de busca. Discuta vantagens e desvantagens das duas implementações.

28. Descreva algumas formas de imprimir as chaves de uma tabela *hash* na ordem crescente.

29. Cite algumas limitações de uma tabela *hash*.

30. Descreva algoritmos de busca, inserção e remoção em uma tabela *hash* com tratamento de colisões por encadeamento exterior.

31. Descreva algoritmos de busca e inserção em tabela *hash* com tratamento de colisões por encadeamento interior, supondo não-existência de exclusões.

32. Descreva algoritmos de busca e inserção em tabela *hash* com tratamento de colisões por endereçamento aberto, supondo não-existência de exclusões.

33. Descreva algoritmos de busca, inserção e remoção em tabela *hash* com tratamento de colisões por endereçamento aberto, supondo que cada compartimento possa estar nos estados vazio, ocupado ou liberado.

34. Compare árvores binárias de busca e *hashing*, destacando vantagens, desvantagens e situações em que cada técnica é mais apropriada.

35. Compare busca sequencial, busca binária e *hashing* destacando vantagens, desvantagens e situações em que cada técnica é mais apropriada.

36. Desenhar a árvore de Huffman para o seguinte conjunto de chaves e frequências, respectivamente na primeira e segunda linhas da tabela:

s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8
1	6	2	1	1	9	2	3

37. A árvore fornecida pelo algoritmo de Huffman é única, ou seja, o algoritmo sempre fornecerá a mesma árvore?

38. Em que situações algoritmos *union-find* são utilizados e quais as operações fundamentais nesta estrutura?

39. Discuta duas formas de implementação de algoritmos *union-find*, destacando vantagens e desvantagens.

40. Explique a heurística de união ponderada.

41. Explique a heurística de união por ordenação e a heurística de compressão de caminho. Discuta os detalhes de implementação relacionados.

42. Para algoritmos *union-find*, escreva o pseudocódigo para **makeSet**(**x**), **union**(**x**, **y**) e **findSet**(**x**) usando a representação de lista ligada e a heurística de união ponderada. Suponha que cada objeto x tenha um atributo *rep* apontando para o representante do conjunto que contém x, e que cada conjunto S tem atributos *inicio*, fim e *tamanho* (que é igual ao comprimento da lista).

43. Para algoritmos *union-find*, escreva o pseudocódigo para **makeSet**(**x**), **union**(**x**, **y**) e **findSet**(**x**) usando a representação por florestas de conjuntos disjuntos e as heurísticas de união por ordenação e compressão de caminho.

44. Explique o problema de árvore geradora mínima, dê exemplos e aplicações relacionadas.

45. Explique o funcionamento do algoritmo de Kruskal para o problema da árvore geradora mínima. Justifique porque o algoritmo sempre funciona. Ilustre com um exemplo.

46. A árvore fornecida pelo algoritmo de Kruskal é única, ou seja, o algoritmo sempre fornecerá a mesma árvore geradora mínima?

Bibliografia Utilizada

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.