Lista de Exercícios
"Árvores, AVL, Hash"

Universidade Tecnológica Federal do Paraná (UTFPR), campus Apucarana
Curso de Engenharia de Computação
Disciplina de Estrutura de Dados - ED62A - 1°Semestre 2019
Prof. Dr. Rafael Gomes Mantovani

Exercício 1. Suponha que um conjunto dinâmico S seja representado por uma tabela de endereços diretos T de comprimento m. Descreva um procedimento que determine o elemento máximo de S. Qual é o desempenho do pior caso do seu procedimento?

Exercício 2. Sugira como implementar uma tabela de endereços diretos na qual as chaves de elementos armazenados não precisem ser distintas e os elementos possam ter dados satélites. Todas as três operações de dicionário (`INSERT`, `DELETE` e `SEARCH`) devem ser executadas no tempo O(1). Não esqueça que `DELETE` adota como argumento um ponteiro para um objeto a ser eliminado, não uma chave).

Exercício 3. Demonstre o que acontece quando inserimos as chaves {5, 28, 19, 15, 20, 33, 12, 17, 10} em uma tabela de espalhamento com colisões resolvidas por encadeamento com nove posições e a função hash $h(k) = k \, mod \, 9$.

Exercício 4. Um professor apresenta a hipótese de que podemos obter ganhos substanciais de desempenho modificando o esquema de encadeamento para manter cada lista em sequência ordenada. Como a modificação do professor afeta o tempo de execução para pesquisas mal-sucedidas, bem sucedidas, inserções e eliminações?

Exercício 5. Considere uma tabela de espalhamento de tamanho m = 1000, e uma função hash correspondente $h(k) = \lfloor((k * A) \, mod \, 1)\rfloor$, para A = ($\sqrt{5}$-1)/2. Calcule as localizações para as quais são mapeadas as chaves {61, 62, 63, 64, 65}.

Exercício 6. Desenhe o conteúdo da tabela hash resultante da inserção de registros com as chaves {Q, U, E, S, T, A, O, F, C, I, L}, nesta ordem, em uma tabela inicialmente vazia de tamanho 13 (treze), usando endereçamento aberto com hashing linear para a escolha de localizações alternativas. Use a função hash $h(k) = k \, mod \, 13$, para a k-ésima letra do alfabeto.

Exercício 7. Quais as características de uma boa função hash?

Exercício 8. Um dos métodos utilizados para organizar dados é pelo uso de tabelas hash.

a) Em que situações a tabela hash deve ser utilizada?

b) Descreva dois mecanismos diferentes para resolver o problema de colisões de várias chaves em uma mesma posição de memória. Quais são as vantagens e desvantagens de cada mecanismo?

Exercício 9. Trace árvores de busca binária de alturas 2, 3, 4, 5 e 6 para o conjunto de chaves {1, 4, 5, 10, 16, 17, 21}.

Exercício 10. Qual é a principal propriedade de uma árvore binária de pesquisa?

Exercício 11. Dê um algoritmo não recursivo que execute um percurso em árvore em ordem. Sugestão: usar uma pilha como estrutura de dados auxiliar.

Exercício 12. Suponha que temos números entre 1 e 1000 em uma árvore de busca binária e queremos procurar o elemento 363. Qual das seguintes sequências não poderia ser a sequência de nós examinados?

a) 2, 252, 401, 398, 330, 344, 397, 363

b) 924, 220, 911, 244, 898, 258, 362, 363

c) 925, 202, 911, 240, 912, 245, 363

d) 2, 399, 387, 219, 266, 382, 381, 278, 363

e) 935, 278, 347, 621, 299, 392, 358, 363

Exercício 13. Escreva versões recursivas para Tree-Minimum e Tree-Maximum.

Exercício 14. Dê uma versão recursiva do procedimento Tree-Insert.

Exercício 15. Como podemos ordenar um conjunto de n elementos por meio de uma árvore binária de busca? Faça um programa que resolva este problema.

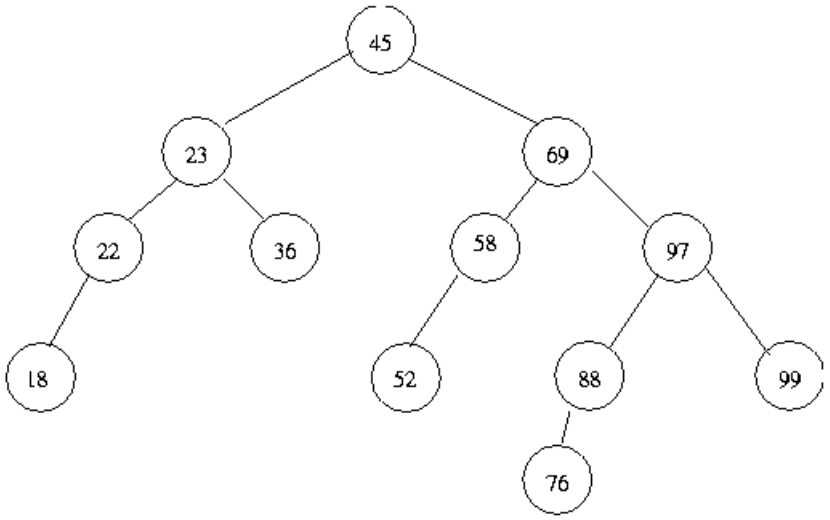
Exercício 16. A operação de remoção em uma árvore binária é "comutativa" no sentido de que eliminar x e depois y de uma árvore de busca resulta na mesma árvore que eliminar y e depois x? Mostre porque ou dê um contra-exemplo.

Exercício 17. Quantas árvores binárias de pesquisa diferentes podem armazenar as chaves { 1, 2, 3, 4}?

Exercício 18. Implemente rotinas para realizar as operações de rotações simples em AVLs, e rotinas de rotação dupla usando estas funções de rotações simples. Fazer para ambos os casos: direita e esquerda.

Exercício 19. Insira em uma árvore AVL as chaves apresentadas em cada item, na ordem em que aparecem. Desenhe a árvore resultante da inserção, desenhando uma nova árvore toda vez que ocorrer uma rotação. Indique também qual rotação foi realizada.

a) 30, 40, 24, 58, 48, 26, 11, 13, 14



b) 20, 15, 25, 10, 30, 24, 17, 12, 5, 3

c) 40, 30, 50, 45, 55, 52

d) 20, 15, 25, 12, 17, 24, 30, 10, 14, 13

e) 20, 15, 25, 12, 17, 30, 26

Exercício 20. Dada uma árvore binária T, proponha um algoritmo que determine se T é uma árvore AVL (lembrando-se das restrições de altura e de árvore de busca).

Exercício 21. Mostre os estados inicial, intermediários e final após as remoções das chaves = {36, 23, 5, 99, 88, 76}, em sequencia, da arvore AVL mostrada abaixo.

Exercício 22. Aplique os percursos preordem(), emordem() e posordem() à arvore inicial do exercício anterior.

Exercício 23. Considere duas listas ordenadas de números. Determine se cada elemento da lista menor está presente também na lista maior. Pode assumir que não existem duplicações em nenhuma das duas listas