Lista de exercícios

Algoritmos e Estrutura de Dados
Pós-graduação em Ciência da Computação
Departamento de Ciência da Computação – Universidade Federal de Juiz de Fora

1) Em uma tabela hash com 100 entradas, as colisões são resolvidas usando listas encadeadas. Para reduzir o tempo de pesquisa, decidiu-se que cada lista seria organizada como uma árvore binária de pesquisa. A função utilizada é h(k) = k mod 100. Infelizmente, as chaves inseridas seguem o padrão $k_i = 50i$, onde k_i corresponde à i-ésima chave inserida.

a) Mostre a situação da tabela após a inserção de k_i, com i = 1,2,••,13. (Faça desenho)

b) Depois que 1000 chaves são inseridas de acordo com o padrão acima, inicia-se a inserção de chaves escolhidas de forma randômica (isto é, não seguem o padrão das chaves já inseridas). Assim responda: Qual é a ordem do pior caso (isto é, o maior número de comparações) para inserir uma chave?

2) Faça um algoritmo de inserção em uma tabela hashing que insere uma chave k em uma tabela T de inteiros utilizando a abordagem de re-hashing para tratar colisões. Lembre-se de caminhar de forma circular na tabela e de garantir que todas as posições da tabela são testadas uma única vez.

3) Considere o seguinte conjunto de 15 valores S = { 16384, 1, 17, 3, 33, 513, 8193, 1025, 65, 5, 129, 2049, 9, 257, 4097}. Crie uma função de hash **perfeita** para inserir estes valores em uma tabela de tamanho 15.

4) Exercício 10.3 (livro do Jayme – na edição antiga é capítulo 8)

5) Exercício 10.6 (livro do Jayme – na edição antiga é capítulo 8)

6) Exercício 10.13 (livro do Jayme – na edição antiga é capítulo 8)

7) Exercício 10.19 (livro do Jayme – na edição antiga é capítulo 8)

8) Uma forma de caminhar pelos registros de tabelas sem a utilização de *hashing* é através a utilização de um campo chamado do **elo.** O elo é um campo do registro que indica qual o próximo registro da sequência. Neste tipo de abordagem, é comum a utilização de um descritor para indicar qual o primeiro campo da tabela. A figura ao lado ilustra a utilização do campo de descritor e da tabela com elos. Ao caminhar pelos campos de elos, verifique que você estará caminhando pelos registros de forma ordenada (por número). Crie uma função reordena(T, d) para **reordenar fisicamente** uma tabela T = {chave, elo} a partir do seu elo, dado um descritor *d* inteiro que aponta para o índice de início da tabela.

início → 2

NÚMERO	Demais campos...	Elo
25		7
12		6
71		5
56		3
93		0
17		1
42		4

9) Outra forma, equivalente à acima, mas em alguns casos mais adequada, de caminhar pelos registros de tabelas sem a utilização de *hashing* é através da utilização de uma tabela adicional chamada de **VIO (vetor indireto de ordenação).** O VIO é uma tabela que contém um campo que indica qual o próximo registro da sequência. Diferente da abordagem de elo, o VIO pode ser armazenado em local diferente da tabela original e é possível ter diferentes VIOs para a mesma

Chave 1	Chave 2
5	25
7	12
9	71
10	56
56	93
70	17
88	42

VIO
2
6
1
7
4
3
5

tabela (um para cada ordenação que se deseja). Ainda, na abordagem de VIO não é necessária a utilização de um descritor para indicar o primeiro registro. O valor do primeiro registro do VIO indica a posição do segundo registro da sequência. O valor do segundo registro do VIO indica a posição do terceiro registro da sequência e assim por diante. A figura ao lado ilustra a utilização do VIO para caminhar na tabela em relação à chave 1. Considerando esta abordagem, faça o que se pede:

a. Crie uma função reordena(V, VIO) para **reordenar fisicamente** um vetor V de chaves a partir seu VIO.

b. Suponha que você dispõe de um arquivo ordenado fisicamente pela chave 1 e por VIO pela chave 2. Por conveniência operacional, você decidiu que seria melhor que ele estivesse ordenado fisicamente pela chave 2 e por VIO pela chave 1. Proponha um procedimento reordena(Tabela T[], int VIO[]) que faça essa troca, sem que seja necessário classificar novamente o arquivo. Considere struct Tabela { int chave1, chave2}.