Pesquisa e Classificação de Dados
Lista 5 (Hashing, Árvore B e variações)

Prof. Ricardo Oliveira

Esta lista **não** vale nota e **não** deve ser entregue, mas apenas utilizada como material de apoio para estudo. Naturalmente, você pode tirar eventuais dúvidas com o professor.

Exercícios marcados com (B) são básicos e essenciais para a matéria. Exercícios marcados com (C) são complementares. Recomenda-se fortemente a resolver todos os exercícios.

1. (B) Considere uma tabela *hash* de tamanho $M = 8$ com endereçamento fechado por lista ligada, cuja função *hash* é $h(x) = x\%8$ (onde % denota o resto da divisão). Insira as chaves a seguir e apresente as listas resultantes: 2, 7, 8, 17, 18, 19, 20, 23, 24, 37, 38, 40, 43, 47, 52, 60, 67, 85, 94 e 98.
2. (B) Considere uma tabela *hash* de tamanho $M = 8$ com endereçamento fechado por lista ligada, cuja função *hash* é $h(x) = x\%8$ (onde % denota o resto da divisão). Insira as chaves a seguir e apresente as listas resultantes: 76, 36, 68, 84, 20, 28, 4, 60, 45, 52 e 12.
3. (C) Considere uma tabela *hash* de tamanho $M = 3$ com endereçamento fechado por árvore AVL, cuja função *hash* é $h(x) = (2x)\%3$. Insira as seguintes chaves na ordem dada, apresentando as árvores resultantes: 87, 88, 80, 38, 5, 58, 53, 41, 57, 10, 22 e 42.
4. (C) Considere uma tabela *hash* de tamanho $M = 4$ com endereçamento fechado por lista ligada, cuja função *hash* é $h(x) = (x^x + 3x - 2)\%4$. Insira as seguintes chaves, apresentando as listas resultantes: 1, 2, 3, 4, 5, 6, 7, 8 e 9.
5. (B) Considere uma tabela *hash* de tamanho $M = 11$ com endereçamento aberto com as seguintes funções: $h_1(x) = x\%11$; $h_i(x) = (h_{i-1}(x) + 1)\%M$ para $i \in \{2..11\}$. Insira, nesta ordem, as chaves 14, 30, 12, 25, 21, 19, 15, 36, 11, 17 e 22. Apresente a tabela *hash* resultante.
6. (B) Considere uma tabela *hash* de tamanho $M = 5$ com endereçamento aberto e as seguintes funções:
   $h_1(x) = x\%2$
   $h_2(x) = x\%2 + 3$
   $h_3(x) = 2$ se x é primo, 3 caso contrário
   $h_i(x) = (h_{i-1}(x) + 1)\%5$ para $i \geq 4$.
   Insira, nesta ordem, as chaves 15, 41, 30, 11 e 17, e apresente a tabela *hash* resultante.
7. (B) Considere que "c" mapeia uma letra maiúscula de $\{A..Z\}$ para $\{0..25\}$, isto é, "A" $= 0$, "B" $= 1, ...,$ "Z" $= 25$. Seja h uma função *hash* para strings com letras maiúsculas dada por $(\sum_{i=0}^{|s|-1} "s_i")\%5$. Compute a função para as strings A, BA, COC, DIA e ZZZ.

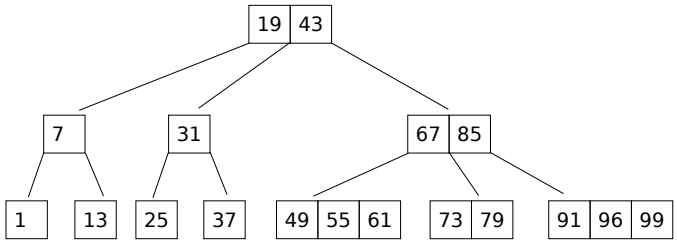
8. (B) Indique as colisões ocorridas no exercício anterior.
9. (C) Considere uma função *hash* para strings com letras maiúsculas, dada por $h(s) = (\sum_{i=0}^{|s|-1} "s_i" \times 26^i)\%256$. Compute a função para as strings ANA, POP, TOLEDO, UTFPR, HASH, CABO, AGUA, PIKACHU, BORUTO, SHINGEKINOKYOJIN e PARALELEPIPEDO (considere implementar a função em uma linguagem de programação).
10. (C) Indique as colisões ocorridas no exercício anterior.
11. (B) Seja D a ordem/grau máximo de uma árvore B. Em função de D, determine seu grau mínimo, o número máximo de chaves que um nodo pode ter e o número mínimo de chaves que um nodo deve ter (exceto raiz).
12. (B) Determine o grau mínimo, número máximo de chaves e número mínimo de chaves por nodo (exceto raiz) de uma árvore B de ordem:

    (a) $D = 5$
    (b) $D = 7$
    (c) $D = 42$
    (d) $D = 43$
    (e) $D = 44$
    (f) $D = 1024$
    (g) $D = 4$
    (h) $D = 3$
    (i) $D = 2$

13. (C) Uma árvore B de ordem $D = 2$ tem exatamente uma chave por nó (não vazio) e, portanto, seria equivalente a uma BST balanceada (como a AVL ou a Rubro-Negra). Mostre que não é possível *manter* uma árvore B de ordem $D = 2$. Dica: estude a operação *split* neste caso.
14. (B) Considere a seguinte estrutura para representar um nodo de uma árvore B:

    ```
    #define D <ordem_da_arvore>
    struct nodo {
        struct nodo *filho[D];
        int chave[D-1];
    };
    ```

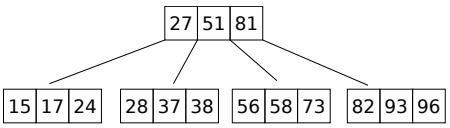
    Determine qual deve a ordem da árvore B em um sistema de arquivos cujas páginas têm 4Kb = 4096 bytes. Considere que um *int* ocupa 4 bytes e que um ponteiro ocupa 8 bytes.
15. (B) Refaça o exercício anterior considerando a inclusão do vetor `void *conteudo[D-1]` na estrutura.
16. (B) Considere a seguinte árvore B de ordem $D = 4$:



Insira, na ordem dada, as chaves abaixo. Para cada inserção, indique o número de leituras e escritas em disco realizadas pela operação (assuma que a raiz está em disco e que apenas uma página pode estar em memória principal a qualquer momento). Ao final, apresente a árvore B resultante.

(a) 76
(b) 58
(c) 94
(d) 82
(e) 95
(f) 98

17. (B) Considere a seguinte árvore B de ordem $D = 5$:



Remova, em ordem, as seguintes chaves: 81, 58, 51, 17, 15. Se uma rotação for necessária, a faça com o irmão de maior tamanho e, se ambos os irmãos têm o mesmo tamanho, modifique o irmão esquerdo (isto é, faça a rotação RR). Se um *merge* for necessário, o faça também com o irmão esquerdo, se existir. Ao final, apresente a árvore B resultante.

18. (C) Apresente um algoritmo que, dada uma árvore B, crie uma árvore B+ de mesma ordem com o mesmo conjunto de chaves.
19. (B) Em uma árvore B+ de ordem $D = 4$ inicialmente vazia, insira, em ordem, todos os inteiros entre 1 e 15, inclusive. Apresente a árvore resultante.
20. (B) Considere a árvore B+ obtida no exercício anterior. Indique quantas leituras são necessárias para cada consulta a seguir:
    (a) Buscar o conteúdo da chave 8;
    (b) Buscar o conteúdo da chave 12;
    (c) Buscar o conteúdo da chave 7;
    (d) Buscar o conteúdo de todas as chaves entre 4 e 9, inclusive.
    (e) Buscar o conteúdo de todas as chaves entre 5 e 6, inclusive.
    (f) Buscar o conteúdo de todas as chaves entre 1 e 15, inclusive.
21. (B) Considerando a mesma árvore, remova, nesta ordem, as chaves 12, 13, 10, 14, 9, 8 e 15. Apresente a árvore resultante.