MAC5710 - Estrutura de Dados e suas Aplicações
Primeiro semestre de 2009

Lista 7

1. Mostre qual é a árvore de busca ótima para o conjunto $\{1, 2, 3, 4, 5\}$, considerando

$$p_1 = \frac{1}{3}, \quad p_2 = \frac{1}{7}, \quad p_3 = \frac{1}{4}, \quad p_4 = \frac{1}{42}, \quad p_5 = \frac{1}{4}.$$

   Suponha que não ocorrerão buscas mal-sucedidas (ou seja, ignore os q_j's).

2. Considere o conjunto $\{1, 2, 3\}$ com as seguintes probabilidades de acesso

$$q_0 = \frac{1}{20}, \quad p_1 = \frac{1}{5}, \quad q_1 = \frac{1}{10}, \quad p_2 = \frac{2}{5}, \quad q_2 = \frac{1}{20}, \quad p_3 = \frac{3}{20}, \quad q_3 = \frac{1}{20}.$$

   Lembre-se que, para $1 \leq i < 3$, q_i é a probabilidade de ocorrer uma busca mal-sucedida por uma chave entre i e $i + 1$, enquanto que q_0 (q_n) é a probabilidade de uma busca mal-sucedida por uma chave menor (maior) que 1 (3).

   (a) Qual é o número esperado de comparações **com chaves** para cada uma das 5 possíveis árvores de busca com os três elementos 1, 2 e 3?

   (b) Qual é a árvore ótima (ou quais são)?

3. Escreva um algoritmo MontaABB(r, n) que recebe a tabela r construída pelo algoritmo ABBÓtima(p, n) visto em aula e constrói uma ABB ótima para as chaves de 1 a n e para as probabilidades dadas por $p[1\,..\,n]$.

4. Reescreva a recorrência vista em aula para o valor de uma ABB ótima levando em conta as probabilidades $q_0, \ldots, q_n$ das buscas mal-sucedidas. Reescreva o algoritmo ABBÓtima(p, n) para determinar esse valor, adicionando o vetor q à lista de parâmetros. Reescreva também, se necessário, o algoritmo MontaABB(r, n) para esse caso mais geral do problema.

5. Prove que a altura de uma B-árvore de parâmetro $t = 4$ é no máximo $\log_4 \frac{n+1}{2}$, onde n é o número de chaves da árvore, e no mínimo $\log_8(n + 1) - 1$.

6. Deduza uma delimitação superior justa para o número de **nós** de uma B-árvore com parâmetro t e altura h.

7. Mostre como fica a B-árvore para $t = 3$ que resulta da inserção das chaves

$$G \;\; S \;\; Q \;\; K \;\; C \;\; L \;\; H \;\; T \;\; V \;\; W \;\; M \;\; R \;\; N \;\; P \;\; A \;\; B \;\; X \;\; Y \;\; D \;\; Z \;\; E \;\; F \;\; U.$$

   Desenhe apenas as árvores imediatamente após algum nó se dividir.

8. Escreva a função MínimoBarv(r) que devolve (q, i) onde $x = k_i(q)$ é a chave mínima da B-árvore de raiz r. Não esqueça de chamar adequadamente a função DiskRead!

9. Escreva a função SucessorBarv(r, q, i) que determina um par (p, j) que identifica a chave sucessora da chave $x = k_i(q)$ na B-árvore cuja raiz é r. Assuma que cada nó, além dos campos usados em aula, tem também o campo *pai*, como os nós de uma ABB. Não esqueça de chamar adequadamente a função DiskRead.

10. Escreva uma rotina RemovaMaxBarv(r) que remove a maior chave armazenada na B-árvore de raiz r. Suponha que $n(r) \geq t$.

11. Escreva a sua versão da rotina RemovaBarv(r, k) seguindo a descrição dada a seguir, que segue a linha da rotina de inserção em B-árvore dada em aula.

    Para tanto, escreva uma rotina auxiliar RemovaNãoMagro(q, x), que remove a chave x de uma subárvore de raiz q, onde q tem pelo menos t chaves.

    Sua rotina RemovaBarv(r, k) deve usar a rotina RemovaNãoMagro(q, x).

    Para a rotina RemovaNãoMagro(q, x), considere os seguintes casos:

    (a) se x está no nó q e o nó q é uma folha, remova x de q e pronto;

    (b) se x está no nó q mas este não é uma folha, então considere os três seguintes casos: se o filho y de q que precede x tem pelo menos t chaves, encontre o máximo x' da subárvore y e o remova (use a rotina do exercício anterior), substitua k por k' em q; se o filho z de q que sucede x tem pelo menos t chaves, faça o equivalente do outro lado; se ambos y e z têm $t-1$ chaves, junte x e as chaves de z em y, removendo x e o apontador para z de q, e recursivamente remova x de y;

    (c) se x não aparece no nó interno q, seja $p_i(q)$ a raiz da subárvore que deve conter x, caso ele apareça na subárvore enraizada em q; se $p_i(q)$ tem apenas $t-1$ chaves, então ou ele tem um irmão com pelo menos t chaves, ou seus dois irmãos têm exatamente $t-1$ chaves; no primeiro caso, repasse uma chave do irmão com pelo menos t chaves para $p_i(q)$ e prossiga; no segundo caso, junte $p_i(q)$ a um de seus vizinhos, fazendo os ajustes necessários na árvore, e prossiga na busca pela chave x a ser removida.