Árvores B

Remoção

B-Tree-Set

B-Tree-Delete(r,k) remoção da chave k da sub-árvore com raiz x.

1. Se a chave k está no nó x e x é uma folha, exclua a chave k de x (pelos procedimentos anteriores garantido que a folha tem pelo menos t chaves).
2. Se a chave k está no nó x e x é um nó interno, faça:
   1. Caso contrário, se ambos y e z possuem apenas t-1 chaves, faça a junção de k e todas as chaves de z em y, de forma que x perde tanto a chave como o ponteiro para z, e y agora contém 2\*t-1 chaves. Então libere z e delete recursivamente k de x.
3. Se a chave k não está presente no nó interno x, determine a raiz ci[x] da sub-árvore apropriada que deve conter k (se k estiver presente na árvore).
   1. Se ci[x] contém apenas t-1 chaves, mas tem um irmão imediato com pelo menos t chaves, dê para ci[x] uma chave extra movendo uma chave de x para ci[x], movendo uma chave do irmão imediato de ci[x] à esquerda ou à direita, e movendo o ponteiro do filho apropriado do irmão para o nó ci[x]. (Se o ci[x] for ficar com menos que t-1, tira do irmão imediato um e trás para ci[x] para se poder deletar a chave que se quer, chamando recursivamente B-Tree-Delete(c0[x],k)).
   2. Se c1[x] e ambos os irmãos imediatos de c1[x] contêm t-1 chaves, (não se pode chamar recursivamente em um nó que contenha o mínimo) faça a junção de c1[x] com um de seus irmãos (caso dois sejam mínimos). Isso implicará em mover uma chave de x para o novo nó fundido (que se tornará a chave mediana para aquele nó, criando um novo pai). Só assim poderá ser chamada a recursão para deleção.
   3. Se a
4. Se a chave k é a raiz e ela for folha. Excluir a raiz desalocando-a.

B-Tree-Delete-From-Root(T, k){

r <- raiz[T];

se (n[r] = 0) retorna;

senão B-Tree-Delete(r,k);

se (n[r] = 0 E (! Leave[r]){

raiz[T] <- c1[r]; c1 sendo o primeiro índice do vetor.

desaloca(r);

}

}

Obs: Cuidado quando os nós internos se tornam folhas!

Árvores B+

-Os nós internos armazenam apenas índices (ponteiros de filhos e chaves).

-As folhas armazenam os registros de dados (conectadas da esquerda para direita, permitindo acesso seqüencial ordenado mais eficiente).

-Blocagem menor na folha (altura menor).

-Fator de blocagem é maior.

-Âncora da primeira folha não aparece no índice, ou seja, somente os posteriores à da segunda incluindo ela, se temos 2, 10, 24, etc. Só o 24 aparece, e 2 e 10 aparecem como primeira folha porém não entram no índice.

Adaptações dos algoritmos:

* Busca: tem sempre que descer às folhas.
* Inserção:
  + Split:
    - Mediana de uma folha é COPIADA para o pai.
    - Mediana de um nó interno é MOVIDO para o pai.
* Remoção: nas folhas
  + Se k (chave a ser removida) ocorrer em um nó interno, o valor deve ser substituído pelo valor predecessor nas folhas.

Árvores B\*:

* Propostas por Knuth em 1973.
* Preenchimento mínimo de 2/3 (mais precisamente (2\*t-1/3)
* Split postergado até que dois nós irmãos (imediatos) estejam cheios