Kushal R. Bhandari

CS 302 001

Prof. Adriana Wise

B-TREES

DIAGRAM OF B-TREE DELETION:

1. Initial Tree
2. D deleted: CASE I
3. M deleted: CASE 2A
4. G deleted: CASE 2C
5. E deleted: CASE 3B
6. B deleted: CASE 3A

DESCRIPTION OF B-TREE DELETION:

1. If the key k is in node x and x is a leaf, delete the key k from x.
2. If the key k is in node x and x is an internal node, do the following.
   1. If the child y that precedes k in node x has at least t keys, then find the predecessor k0 of k in the sub-tree rooted at y. Recursively delete k0, and replace k by k0 in x. (We can find k0 and delete it in a single downward pass.)
   2. If y has fewer than t keys, then, symmetrically, examine the child z that follows k in node x. If z has at least t keys, then find the successor k0 of k in the subtree rooted at z. Recursively delete k0, and replace k by k0 in x. (We can find k0 and delete it in a single downward pass.)
   3. Otherwise, if both y and z have only t-1 keys, merge k and all of z into y, so that x loses both k and the pointer to z, and y now contains 2t-1 keys. Then free z and recursively delete k from y.
3. If the key k is not present in internal node x, determine the root x.c(i) of the appropriate subtree that must contain k, if k is in the tree at all. If x.c(i) has only t-1 keys, execute step 3a or 3b as necessary to guarantee that we descend to a node containing at least t keys. Then finish by recursing on the appropriate child of x.
   1. If x.c(i) has only t-1 keys but has an immediate sibling with at least t keys, give x.c(i) an extra key by moving a key from x down into x.c(i), moving a key from x.c(i) ’s immediate left or right sibling up into x, and moving the appropriate child pointer from the sibling into x.c(i).
   2. If x.c(i) and both of x.c(i)’s immediate siblings have t-1 keys, merge x.c(i) with one sibling, which involves moving a key from x down into the new merged node to become the median key for that node.
4. Since most of the keys in a B-tree are in the leaves, deletion operations are most often used to delete keys from leaves. The recursive delete procedure then acts in one downward pass through the tree, without having to back up. When deleting a key in an internal node, however, the procedure makes a downward pass through the tree but may have to return to the node from which the key was deleted to replace the key with its predecessor or successor (cases 2a and 2b).

PSEUDO CODE

K → key to be deleted key to be deleted

A → key to be deleted node containing the key

B-Tree-Delete-Key (A, K)

**IF** not leaf[A] **THEN**

B ← Preceding-Child(A) Preceding-Child(A)

C ← Preceding-Child(A) Successor-Child(A)

**IF** n[B] > t − 1 **THEN**

K' ← Preceding-Child(A) Find-Predecessor-Key (K, A)

Move-Key (K', B, A)

Move-Key (K, A, C)

B-Tree-Delete-Key (K, C)

**ELSE IF** n[C] > t − 1 **THEN**

K' ← Preceding-Child(A) Find-Successor-Key (K, A)

Move-Key (K', C, A)

Move-Key (K, A, B)

B-Tree-Delete-Key (K, B)

**ELSE**

Move-Key (K, A, B)

Merge-Nodes (B, C)

B-Tree-Delete-Key (K, B)

**ELSE** (leaf node)

B ← Preceding-Child(A) Preceding-Child(A)

C ← Preceding-Child(A) Successor-Child(A)

P ← Preceding-Child(A) root(A)

Q ← Preceding-Child(A) Root-Key(A)

**IF** n[A] > t − 1 **THEN** Remove-Key (K, A)

**ELSE IF** n[B] > t − 1 **THEN**

K' ← Preceding-Child(A) Find-Predecessor-Key (P, Q)

Move-Key (K', B, P)

K' ← Preceding-Child(A) Find-Successor-Key (P, Q)

Move-Key (K', P, A)

B-Tree-Delete-Key (K, A)

**ELSE IF** n[P] > t − 1 **THEN**

K' ← Preceding-Child(A) Find-Successor-Key (P, Q)

Move-Key (K', C, P)

K' ← Preceding-Child(A) Find-Predecessor-Key (P, Q)

Move-Key (K', P, A)

B-Tree-Delete-Key (K, A)

**ELSE**

R ← Preceding-Child(A) Find-Sibling(P)

P' ← Preceding-Child(A) root(P)

**IF** n[P'] = t − 1 **THEN**

Merge-Nodes (P', P)

Merge-Nodes (P, R)

B-Tree-Delete-Key (K, A)

**ELSE**

Move-Key (Q, P, A)

B-Tree-Delete-Key (K, A)

Citation

Introduction to Algorithms 3rd Edition by Clifford Stein, Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest

“Deletion in B Tree,” Tree. [Online]. Available: https://scanftree.com/Data\_Structure/deletion-in-b-tree.