CMSC420: 2-3 Trees

Problems with AVL Trees

* Have to store height, compare subtrees heights for balance
* Rotations are expensive, Especially RL and LR
* We can do better while maintaining O(log n): 2-3 Trees, B-Trees, RB-Trees

Properties of 2-3 Trees

* Perfectly Balanced
  + All nodes have either 0 children or the max they support
  + All leaf nodes are at the same depth
  + That is, for all elements n in T, B(n) = 0 (For a suitable definition of B)

How do we Achieve Perfect Balance?

* Define 2-nodes and 3-nodes
  + 2-nodes have 2 children
  + 3-nodes have 3 children

Node Types

* A 2-node is a BST node
* A 3-node has 2 keys and 3 children nodes where the left child is smaller than the left key the middle child is in between the 2 keys and the right child is larger then the 2 keys.

Shape, polygon

Description automatically generated

* A working definition of balance for a 3-node. H(a) = H(b) = H(c)

A key Distinction

* BSTs grow downwards, but 2-3 trees grow upwards.

Insertion Abstracted

1. Search to find the appropriate leaf for this element
2. Is it a 2-node?
   1. Yes, Add the new element here, making it a 3-node, and terminate
   2. No, continue to the next step
3. Temporarily create a 4-node with 3 keys: K1 < K2 < K3
4. Is this the root?
   1. Yes, create a new root 2-node with K2 and children K1 (with children a and b) and K3 (with children c and d); Terminate;
   2. No, create K1 and K3  as in step a, Add K2 to P; go step 2.

Shape, polygon

Description automatically generated

Keeping Trees Shorter

* We tend to create more 2-nodes, making the tree taller.
  + More 2-nodes => O(log2n)
  + More 3-nodes => O(log3n)
    - We need to reduce the amount of 2-nodes in the tree\*\*\*\*\*\*

Key Rotation

CORE IDEA: We might have siblings who are 2-nodes, we can expand.

Key Rotation Abstraction

* First assign each node an age, increasing with keys
* Prefer to rotate towards older siblings, starting with closest in age
  + Try younger siblings (closest first) if no available older ones

Diagram

Description automatically generatedDiagram

Description automatically generated

* First, assign each node with an age, increasing with keys
  + 17,17 is older than 8 and younget than 20,25
* Perfer to rotate towards older siblings, starting with closest in age
  + Trey younger siblings (closest first) if no available older ones

Key Rotation Abstrated (sibling is a 2-node

1)A picture containing diagram

Description automatically generated2)A picture containing diagram

Description automatically generated

3)A picture containing boat, clock, street, old

Description automatically generated 4)A picture containing diagram

Description automatically generated

Deletion from a 2-3 Tree

* Inner node deletion will become leaf deletions.
* Deleting a key from a leaf 3-node is easy.
  + [K1,K2] => [K1]
* Cases to consider when deleting a node in a 2-3 tree with other elements.

Diagram

Description automatically generated

* Cases B, D, F (after deleting b)

Diagram

Description automatically generated

* Cases C and E (after deleting b)

Diagram

Description automatically generated

* Case A (deleting b)

A picture containing diagram

Description automatically generated

Propagating Deleted Nodes

* + There are three possible cases
    - Case A (corresponding to cases A, B, and C)

Diagram

Description automatically generated🡺🡺A picture containing diagram

Description automatically generated

* + - Case B (deleting b)

Shape

Description automatically generated🡺🡺Diagram, shape

Description automatically generated

* + - Case C (deleting c)

A picture containing shape

Description automatically generated🡺🡺Diagram, shape

Description automatically generated

Additional Wrinkle (deleting d)

Diagram

Description automatically generated🡺🡺Shape

Description automatically generated

* Always merge right when given the option
  + This will be important when we cover B-Trees

Deleting Interior Keys

* + Like with BSTs, 2-3 trees replace a removed interior item with it is in-order successor
  + If the successor is still an interior node, we continue with its successor
  + This will ultimately result in reaching a leaf node

Deleting is Expensive

* Deleting a single item can cause a cascade of deletions
* Many implementations use Mark-and-Sweep, both for 2-3 trees and AVL trees
  + - Delete less often
    - May be able to combine deletions for efficiency gains