Comp 282

Lecture 15

2-3 Trees

Introduction
Properties

• They are NOT Binary trees!!
• Nodes have exactly, either
  – Two children, or
  – Three children. (hence not binary)
• A 2-3 Tree containing n nodes has a maximum height of $\log(n+1)$ [regardless of what order items were inserted or removed.
• Therefore appears perfectly balanced.
Definition

• A tree $T$ is a 2-3 tree if its topology conforms to either:

  - Where $T_l$, $T_m$, and $T_r$ are also 2-3 trees.

  Or...

  - Where $T_l < I_a < T_r$

• Where $T_l$, $T_m$, and $T_r$ are also 2-3 trees.
Commandments

• After the completion of any insertion or deletion task a 2-3 Tree must **ALWAYS** adhere to the following two rules:
  – All leaf nodes **must** have all null children and **must** exist at the same level in the tree as all other leaf nodes.
  – Non-leaf nodes cannot have null children.
    • 2-Nodes **MUST** have exactly 1 item and 2 children subtrees which are not null.
    • 2-Nodes **MUST** have exactly 2 items and 3 children subtrees which are not null.
Traversals

- We can perform in-order (and even pre-order or post-order traversals in nearly the same fashion...
  - In-order traversal pseudo-code
    - Recursively traverse the left subtree $T_l$.
    - Visit the current node's left item $I_a$.
    - If the middle subtree is not empty then
      - Recursively traverse the middle subtree $T_m$
      - Visit the current node's right item $I_b$
    - Recursively traverse the right subtree $T_r$. 

“3” nodes
Insertion

• Similar to BSTs we first locate the proper node to be the parent of the new item.
  – BSTs then attach a new leaf containing the inserted item to this node.
  – 2-3 trees add the inserted item to the node found instead.
    • great if the node was a “2” node;
    • not so good if it was already a “3” node (overfull “4”-ish node)
Example

• Adding 39

“2” node becomes a “3” node
More complicated: “3” nodes

- Adding 38:
• We conceptually add a third element by pushing the middle element up to the parent node, split the node into two siblings and redistribute references…
Propagation

• Pushing the item up to the parent may overfill the parent node.
• This bloated node will also be resolved by pushing its middle item up one level and splitting the node into two siblings.
• This may continue all the way up the tree to the root.
• Or it may stop early when the node pushed up doesn't cause an overfull parental node.
• Insertion of 1 into the following tree…

• No parent exists to push 30 up to…
The root is special

• If the root node is the node that has become overfull then we increase the height of the tree...

• A new “2” node is formed as the root node
Effects of Splitting Root

• Height of the tree has been increased by one.
• The tree is still perfectly balanced.
• All leaves are on the same level (as is always true for 2-3 trees.)