CPSC 223
Algorithms & Data Abstract Structures

Lecture 23:
2-3-4 Trees (cont.)

Today …
• 2-3-4 Trees (cont.) [Ch 12: 651-670]
2-3-4 Trees

- A 2-3-4 Tree is a 2-3 Tree that also allows 4-nodes
  - That is, nodes with 3 items and 4 children
    - More efficient insertion
    - Split 3-Item Nodes *as you search* for the leaf
    - As opposed to bottom up in a 2-3 tree
    - Guarantees we insert into a leaf node with an empty slot
    - Works because a node has 3 items (consider root case)

**Insert 25**

```
<table>
<thead>
<tr>
<th>20</th>
<th>30</th>
<th>40</th>
<th>60</th>
<th>70</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

**Split**

```
<table>
<thead>
<tr>
<th>20</th>
<th>40</th>
</tr>
</thead>
<tbody>
<tr>
<td>30</td>
<td>50</td>
</tr>
<tr>
<td>60</td>
<td>70</td>
</tr>
</tbody>
</table>
```
2-3-4 Tree Insertion

• Insert *new item* at *lowest level* in the tree

  – 1-Item nodes becomes 2-Item nodes

    ![Insert c](image1)

  – 2-Item nodes becomes a 3-Item nodes

    ![Insert b](image2)

  – *Can’t insert into a 3-Item node …*

2-3-4 Tree Insertion

• Since we can’t insert into 3-Item nodes …

  – As we search for the leaf, we *split* 3-Item nodes

  – This guarantees room for the insert

![Insert h](image3)

Split into 2 nodes (create 1 new node)
Move middle-element up
2-3-4 Tree Insertion

- The root is a special case …
  - For the root case, we can’t move the middle element up

\[
\text{insert } h
\]

Split into 3 nodes (add 2)
Leave middle-element in root

2-3-4 Insertion

- Example:
  - Insert the following nodes into a 2-3-4 tree

\[
10, 30, 60, 20, 40, 50, 70, 80, 15, 90, 100, 85
\]
2-3-4 Removal

- Removal refresher …
  - Find node to remove
  - Find inorder successor (in a leaf node)
  - Copy and remove inorder successor

Also more efficient removal

- Transform nodes with only 1 item into nodes with 2 or 3 items during top-down traversal
- Guarantees that the leaf node has more than one item
- This means that removal does not require restructuring the tree bottom up
- A number of cases to consider …

Case 0

- If the only node in the tree is the root …
  - Remove the item from the root (adjust remaining items)
  - If the root node has no more items, remove the root node

```
root  a b c

remove b  root  a c
```

```
root  a c

remove a  root  c
```

```
root  c

remove c  root  NULL
```
Removal – Case 1 (“transfer”)

- When remove encounters a 1-item node (not the root)
  - If the node has an adjacent sibling with > 1 item
  - We “transfer” (or “rotate”) an item into current node …

![Diagram of Case 1](image1)

Removal – Case 2 (“fusion”)

- When remove encounters a 1-item node (not the root)
  - If the node’s adjacent siblings also have 1 item
  - And the parent has at least 2 items
    - Parent may have fewer than 2 items only if the root!
  - We take an item from the parent and absorb a sibling …

![Diagram of Case 2](image2)
Removal – Case 3 (the root case)

- When remove encounters a 1-item node (not the root)
  
  - If the parent is the root and the root has 1 item
  
  - And the sibling only has 1 item (how many siblings possible?)
  
  - Then the node, parent, and sibling are fused

![Diagram of node removal](image)

Nearest Sibling (Textbook)

Instead of “adjacent” sibling

- Which could be the sibling to the immediate left or immediate right

The textbook uses “nearest” sibling such that:

- A node’s nearest sibling is its immediate left sibling

- Unless the node is a left child … in which case its nearest sibling is its immediate right sibling
2-3-4 Removal

• Example:
  – From the 2-3-4 tree of the previous example, remove 70, 30, 80, 50

Encoding Red Black Trees as 2-3-4 Trees

• Red Black trees can be used to encode 2-3-4 Trees
  – Can lead to more efficient storage … since nodes in a Red Black tree are binary (have two children)
  – This can also simplify (and explain) rotations and color changes
  – The basic idea (see textbook for details):
    • All parent-child edges in a 2-3-4 tree remain black
    • 3-Nodes and 4-Nodes converted to binary nodes with red edges