CSE 2320 Notes 11: Rooted Trees

(CLRS 10.4, 12.1-12.3, 14.1, 13.2)

11.A. Trees

Representing Trees (main memory, disk devices in CSE 3330)

Binary tree

<table>
<thead>
<tr>
<th>Mandatory</th>
<th>Optional</th>
</tr>
</thead>
<tbody>
<tr>
<td>Left</td>
<td>Parent</td>
</tr>
<tr>
<td>Right</td>
<td>Key</td>
</tr>
<tr>
<td>Parent</td>
<td>Data</td>
</tr>
<tr>
<td>Key</td>
<td>Subtree Size</td>
</tr>
</tbody>
</table>

Rooted tree with linked siblings

<table>
<thead>
<tr>
<th>Mandatory</th>
<th>Optional</th>
</tr>
</thead>
<tbody>
<tr>
<td>First Child</td>
<td>Last Child</td>
</tr>
<tr>
<td>Right Sibling</td>
<td>Left Sibling</td>
</tr>
<tr>
<td>Parent</td>
<td>Parent</td>
</tr>
<tr>
<td>Key</td>
<td>Key</td>
</tr>
<tr>
<td>Data</td>
<td>Data</td>
</tr>
<tr>
<td>Subtree Size</td>
<td>Subtree Size</td>
</tr>
</tbody>
</table>
11.B. Binary Tree Traversals (review)

1<sup>st</sup> Visit – Preorder
recTraversal(Node h)
{
    if (h!=null)
    {
        recTraversal(h.l);
        recTraversal(h.r);
    }
}

Preorder
D B C E H F A I G

2<sup>nd</sup> Visit – Inorder
Inorder
C B H E D A G I F

3<sup>rd</sup> Visit – Postorder
Postorder
C H E B G I A F D

11.C. Binary Search Trees

Basic property – Go left for smaller keys. Go right for larger keys. (Use of sentinel)

Which traversal lists the keys in ascending order?
Operations: (see http://ranger.uta.edu/~weems/NOTES2320/REDBLACKC/RB.c )

1. Search (searchR)

2. Minimum / maximum in tree

3. Successor/predecessor of a node

4. Insert (unbalanced in http://ranger.uta.edu/~weems/NOTES2320/REDBLACKC/RB.loadAndGo.c )

5. Deletion of key and associated data is contained in:
   
a. Leaf

b. Node with one child

c. Node with two children
   
   1. Find node’s successor (convention)
   
   2. Move key and data (but not pointer values) from successor node to node of deletion.
   
   3. Successor has either
   
      a. Zero children – leaf is removed (5.a)

      b. One child (right) – point around successor node to remove (5.b)

   May also use tombstones and periodically recycle garbage.

Implementing operations 6. and 7. efficiently requires maintaining subtree sizes “incrementally”.

Rank of a key X that appears in tree = number of nodes with keys ≤ X.

\[
\text{Number of nodes on search path to } X \text{ with keys } \leq \text{key in given node} + \\
\text{Sizes of their left subtrees}
\]

6. Rank of a key (invSelectR).

7. Finds key with a given rank (selectR) - This is the same as flattening tree into an ordered array and then subscripting (or using inorder traversal).

Time for operations?
11.D. ROTATIONS

Technique for rebalancing in balanced binary search tree schemes. Takes $\Theta(1)$ time.

11.E. INSERTION AT ROOT: rotates all edges on the insertion path:
http://ranger.uta.edu/~weems/NOTES2320/bst.step.html