Balanced Search Trees

• Motivation
  o Usually it is the case that a binary search tree grows evenly
  o However, worst-case performance of many operations using binary search trees is linear (contains, add, remove)
  o Worst-case: tree is a linked list
  o Example: insert 1,2,3,4,5,6
  o We need a way to guarantee that the tree remains balanced as new items are inserted

• Two-three trees
  o Definition: A 2-3 search tree is a tree that is either
    ▪ empty
    ▪ A 2-node, which contains one data item (call it x). It may or may have two links, a left link to a 2-3 search tree with all of whose items are "less than" x (according to compareTo), and a right link to a 2-3 search tree with items greater than x
    ▪ A 3-node, with two data items (call them x and y). It may or may not have three links, a left link to a 2-3 search tree with data items less than x, a middle link to a 2-3 search tree with data items between x and y, items, and a right link to a 2-3 search tree with data items greater than y.
  o Example:

![Diagram of a 2-3 tree](image)

  o By this definition, all leaves in a 2-3 tree are at the same depth (the tree is perfectly balanced)
  o What is the maximum height of a 2-3 tree?
  o What is the minimum height of a 2-3 tree?
• Searching a 2-3 tree

Does tree $T$ contain a data item?

```java
node = root
while node != null
    if node is a 2-node
        if item equals data(node)
            return true
        else if item less than data(node)
            node = leftChild(node)
        else
            node = rightChild(node)
    else // node is a 3-node
        if item equals leftData(node) or item equals rightData(node)
            return true
        else if item less than leftData(node)
            node = leftChild(node)
        else if item less than rightData(node)
            node = middleChild(node)
        else
            node = rightChild(node)
    return false
```

• Inserting into a 2-3 tree

- Always modify a leaf (and if necessary, some of the leaf's ancestors)
- If leaf is a 2-node, change it to a 3-node
- If leaf is a 3-node, change it to a temporary 4-node. Then there are 3 cases:
- Note that if the root is a 3-node and is modified, the height of the tree is increased by 1

- **Example insertion sequence**

Text, p. 430

- **Exercise**: Starting with an empty tree, insert 5, 3, 7, 2, 4, 11, 1, 0, 12, 6, 9, 8, 10, 13, 14
- **Implementation**

  Code will be posted

- **Deleting from a 2-3 tree**

  Not covered in Sedgewick and Wayne text

  - Sometimes involves temporary 1-nodes
    - A node with no data, and only one child
  - Delete from a leaf:
    - If the leaf is a 3-node, change it to a 2-node
    - If the leaf is a 2-node, change to a temporary 1-node (to be dealt with below)
  - Delete from interior node
    - Similar to BST deletion: replace deleted item with the next largest item in the tree (which **always is a leaf**)
    - Then, change the leaf node as above

- **Removing 1-nodes**

  There are many cases

  1. The 1-node has a 2-node parent and a 2-node sibling

     ![Diagram](attachment:image.png)

  2. The 1-node has a 2-node parent and a 3-node sibling

     ![Diagram](attachment:image.png)

  3. A 3-node parent and a 2-node sibling

     ![Diagram](attachment:image.png)
4. 3-node parent and a 3-node sibling

- Example sequence

Remove 0, 1, 3, 7, 6, 4, 9, 10
Remove 0

Remove 1, 3, 7

