2-3 Trees

- Leaf nodes contain data item and all leaves are at the same level.
- Each internal nonleaf node:
  - Has 2 or 3 children.
  - Two search values:
    - Largest item in left subtree.
    - Largest item in middle subtree.
- Smallest 2-3 tree:
  - Only one leaf node.
- Balanced and ordered.
height of tree

- n data items

- all nodes have two children
  \[ \text{height} = \lceil \log_2(n+1) \rceil - 1 \]

- all nodes have three children
  \[ \text{height} = \lfloor \log_3(2n+1) \rfloor - 1 \]

Time:
- For a general tree, \( O(\text{height}) \)
  \[ \log_3(2n+1) - 1 \leq \text{height} \leq \lfloor \log_2(n+1) \rfloor - 1 \]

Time to search 1 key: \( O(\text{height}) \)

Time to list all elements in sorted order: \( O(n) \).
Insertions in 2-3 trees

- Insert new leaf in its appropriate place.
- Repeat until all nonleaf nodes have 2 or 3 children
  - If there is a node with 4 children, split the parent into two parent nodes with 2 children each.
  - If the root node is split, add a new root.
- Adjust the search values along insertion paths.

Example

Insert 5

\[\begin{array}{c}
\text{5} \\
\text{5} \\
\end{array}\]

Insert 21

\[\begin{array}{c}
\text{5} \\
\text{5} \\
\text{21} \\
\end{array}\]

Insert 8

\[\begin{array}{c}
\text{5} \\
\text{5} \\
\text{8} \\
\text{21} \\
\end{array}\]

Insert 63

\[\begin{array}{c}
\text{5} \\
\text{5} \\
\text{8} \\
\text{21} \\
\text{63} \\
\end{array}\]

Insert 69

\[\begin{array}{c}
\text{5} \\
\text{5} \\
\text{8} \\
\text{21} \\
\text{63} \\
\text{69} \\
\end{array}\]
Insert 32

\[
\begin{array}{c}
\text{5} & \text{8} & \text{21} & \text{32} & \text{63} & \text{69} \\
\end{array}
\]

\[
\begin{array}{c}
\text{5} & \text{8} & \text{21} & \text{32} & \text{63} & \text{69} \\
\end{array}
\]

Insert 7, 19, 25

\[
\begin{array}{c}
\text{5} & \text{8} & \text{19} & \text{21} & \text{25} & \text{32} & \text{63} & \text{69} \\
\end{array}
\]

\[
\begin{array}{c}
\text{5} & \text{8} & \text{19} & \text{21} & \text{25} & \text{32} & \text{63} & \text{69} \\
\end{array}
\]

Time to insert is \(O(\text{height})\)
Deletions in 2-3 trees

Delete \( x \) from the tree; \( p(x) \): parent of \( x \).

- If \( x \) is the root: delete \( x \)
- If \( p(x) \) has 3 children, delete \( x \)
- If \( p(x) \) has 2 children (one is \( x \) and the other child is \( y \))
  - If \( p(x) \) is the root
    
    \[
    \begin{array}{c}
    p(x) \\
    \text{(}a\text{)} \\
    \text{ } \\
    \text{(}c\text{)} \\
    \end{array}
    \Rightarrow \begin{array}{c}
    3
    \end{array}
    
  - If \( p(x) \) is not the root node. Let \( l \) be the left sibling and let \( r \) be the right sibling of \( p(x) \); (note \( l \) or \( r \) may not exist)

\[\begin{array}{c}
\text{3 children}
\end{array}\]
\[\begin{array}{c}
l \text{ has} \\
\text{3 children}
\end{array}\]
\[\begin{array}{c}
l \text{ has} \\
3 \text{ children}
\end{array}\]
\[\begin{array}{c}
\text{Replace } x \text{ by the 3rd child}
\end{array}\]
l has two children a b x s  

Steps:
- remove x
- combine p(x) with l
- make s a child of l
- rename p(x) to l
- recursively remove x
Example

delete 47

delete 63

10:69
Example

```
32:69
  /    \
10:32  47:69
   /    \
5:8   21:32  36:47  63:69

   /    \
5 8 10 21 32 36 47 63 69
```

```
32:69
  /    \
10:32  47:69
   /    \
5:8   21:32  36:63

   /    \
5 8 10 21 32 36 63 69
```