Lecture 10: March 15, 2001

Today: 2-3 Trees

• Successor
• Insert
• Delete

in $O(\log n)$ time.

What is a 2-3 tree?

• elements only in the leaves, the internal nodes hold maximum of their subtrees
• elements in increasing order from left to right
• every internal nodes has 2 or 3 children
• all leaves have equal depth

More rigorous structure than BST.

An example

What is the depth of a 2-3 tree?

• a 2-3-tree with depth $h$ contains (as a subgraph) a complete binary tree of depth $h \Rightarrow$ has $\geq 2^h$ leaves
• therefore, any 2-3-tree of depth $\geq \log(n + 1)$ has $\geq n + 1$ leaves
• ...so, any such tree with $\leq n$ leaves has depth $< \log(n + 1)$

Depth is $O(\log n)$ - can perform operations efficiently.
How to find a successor?

To find a successor for \( k \):

- start from the root
- find the first child with \( \text{max} \geq k \) (if it does not exist, successor undefined since \( k \) is larger than all elements)
- search the child’s subtree

The general philosophy as for BST’s. Time \( O(\log n) \).

How to insert?

To insert a node \( x \):

- find a successor \( y \) of \( \text{key}[x] \) (if does not exist, find a predecessor and perform symmetric operations)
- attach \( x \) to the parent \( p \) of \( y \) and hope \( p \) has still \( \leq 3 \) children
- if the hope fails (i.e., \( p \) has 4 children now):
  - create a new node \( p' \) and attach 2 children of \( p \)
    to it (so now \( p \) has only two children)
  - insert (recursively) \( p' \) to \( p \)'s parent; if \( p \) is an orphan, create its parent first

Does not violate the “equal depth” constraint
\( \Rightarrow \) still 2-3-tree!

Insertion example

\[
\begin{array}{c}
\begin{array}{c}
\text{1} \\
\text{2} \\
\text{3} \\
\text{5} \\
\text{6} \\
\text{7} \\
\text{8} \\
\text{4}
\end{array}
\end{array}
\quad \rightarrow 
\begin{array}{c}
\begin{array}{c}
\text{1} \\
\text{2} \\
\text{3} \\
\text{5} \\
\text{6} \\
\text{7} \\
\text{8} \\
\text{4}
\end{array}
\end{array}
\begin{array}{c}
\begin{array}{c}
\text{1} \\
\text{2} \\
\text{3} \\
\text{5} \\
\text{6} \\
\text{7} \\
\text{8} \\
\text{4}
\end{array}
\end{array}
\]