2-3 trees: a dictionary for general data

Assume data items are totally ordered (\(<, >, =\))

Assume \(n\) items in the dictionary

Structure: a tree

- Data stored only at leaves (no duplicates)
- All leaves at the same level, in sorted order
- Each internal node:
  - has either 2 or 3 children
  - has a “guide”: the maximum data item in its subtree

Height of tree is \(O(\log n)\)
Example
Search(x): use guides

Insert(x): Search for x, and if it should belong under p:

add x as a child of p (if not already present)

if p now has 4 children:
  • split p into two two nodes, \( p_1 \) and \( p_2 \), each with two children
  • process p’s parent in the same way
  • Special case: no parent — create new root, increasing height of tree by 1

Also need to update “guides” — easy

Time = \( O(\text{height}) = O(\log n) \)
Case when $p$ ends up with 4 children
Delete($x$): Search for $x$, and if found under $p$:

remove $x$

if $p$ now only has one child:

• if $p$ is the root: delete $p$ (height decreases by 1)

• if one of $p$’s siblings has 3 children: borrow one

• if none of $p$’s siblings has 3 children:
  ◦ one sibling $q$ must have 2 children
  ◦ give $p$’s only child to $q$
  ◦ delete $p$
  ◦ process $p$’s parent
Easy case: borrow from sibling
Harder case: give away only child
2-3 trees: summary

Assume $n$ items in dictionary

Running time for lookup, insert, delete:

$O(\log n)$ comparisons, plus $O(\log n)$ overhead

Space: $O(n)$ pointers