Augmenting 2-3 trees

Examples

Store # of items in subtree at each internal node

Queries:

• What is the $k$th smallest item?
• How many items are $\leq x$?
Items may be marked with an attribute, say, “active”/“inactive”

Store a count of active items in subtree at each internal node

Queries:

• What is the $k$th smallest active item?
• How many active items are $\leq x$?
• Attribute flipping...

• Operation $Flip(x, y)$ flips all attribute bits of items in the range

• Assume attributes are bits

• Store an XOR-bit at each internal node
  ◦ “effective” value of the attribute is the XOR of all bits on path from root to leaf

• To perform $Flip(x, y)$:
  ◦ trace paths $e, f$ to $x, y$
  ◦ flip bits at $x, y$, and all roots of “internal” subtrees
Example:
Join($T_1, T_2$) joins two 2-3 trees in time $O(\log n)$
Assume $\max(T_1) < \min(T_2)$
Assume $T_i$ has height $h_i$ for $i = 1, 2$
Case 1: $h_1 = h_2$
Case 2: $h_1 < h_2$

- Attach $v$ as the left-most child of $p$
- If $p$ now has 4 children, we split $p$, and proceed up the tree as in $Insert$
- Time: $O(h_2 - h_1) = O(\log n)$

Case 3: $h_1 > h_2$ — similar
\[ \text{Split}(T, x) \implies (T_1 \leq x, T_2 \geq x) \]
Observations:

- Initially: at most 2 trees of any given height — except there may be 3 of height 0
- General fact: joining two or three trees of height $\leq h$ yields a tree of height $\leq h + 1$
- During the join process, we maintain the invariant that there are 2 trees of any given height, except there may be three of the smallest height
If the distinct heights of the trees to merge are $h_1 > h_2 > \cdots > h_k$, then the total cost is $O(t)$, where

$$t \leq (h_1 - h_2 + 1) + (h_2 - h_3 + 1) + \cdots + (h_{k-1} - h_k + 1)$$

$$= h_1 - h_k + k - 1$$

$$\leq 2h,$$

where $h$ is the height of the original tree.

Conclusion: total time for Split is $O(\log n)$
Mergeable Priority Queues

Operations:
- Insert
- Delete Min
- Merge two queues

Using heaps:
- need to re-build — time $O(n)$

Using 2-3 trees:
- Can support all 3 operations in time $O(\log n)$
Mergeable Priority Queues using 2-3 trees

- Same tree structure as ordinary 2-3 trees
- Items stored at leaves, but
  - duplicates allowed
  - values not in any particular order
- Internal nodes contain “min values” as guides
- Insert: just make a new leaf (anywhere), and update guides
- Delete Min: follow guides to find min, delete, and update guides
- Merge: use Join procedure, and update guides
This data structure supports Insert, Delete Min, and Merge, in time $O(\log n)$

It does not directly support Search and Delete

Implementation notes for both heaps and 2-3 trees:

- data structure stores pointers to objects
- objects may contain a “hook” into the data structure