2-3-4 Tree

Anwar Mamat
**2-3-4 Tree**

- Self-balancing tree
- Every internal node has either two, three, or four child nodes.
  - A 2-node has one data element, and if internal has two child nodes;
  - A 3-node has two data elements, and if internal has three child nodes;
  - A 4-node has three data elements, and if internal has four child nodes.
2-3-4 Tree Properties

• Every node (leaf or internal) is a 2-node, 3-node or a 4-node, and holds one, two, or three data elements, respectively.
• All leaves are at the same depth (the bottom level).
• All data is kept in sorted order.
• Tree height.
  • Worst case: \( \log N \) [all 2-nodes]
  • Best case: \( \log_4 N = \frac{1}{2} \log N \) [all 4-nodes]
• Between 10 and 20 for 1 million nodes.
• Between 15 and 30 for 1 billion nodes.
• Guaranteed logarithmic performance for both search and insert.
2-3-4 Tree Insertion

• If the current node is a 4-node:
  • Remove and save the middle value to get a 3-node.
  • Split the remaining 3-node up into a pair of 2-nodes (the now missing middle value is handled in the next step).
  • If this is the root node (which thus has no parent):
    • the middle value becomes the new root 2-node and the tree height increases by 1. Ascend into the root.
    • Otherwise, push the middle value up into the parent node. Ascend into the parent node.

• Find the child whose interval contains the value to be inserted.
• If that child is a leaf, insert the value into the child node and finish.
  • Otherwise, descend into the child and repeat from step 1
2-3-4 Tree Example: Insertion
2-3-4 Tree Example
2-3-4 Tree Example

Insert 2
2-3-4 Tree Example

Insert 2
2-3-4 Tree Example

Insert 25
2-3-4 Tree Example

Insert 25
2-3-4 Tree Example

Insert 6
2-3-4 Tree Example

Insert 6
2-3-4 Tree Example

Insert 14
2-3-4 Tree Example

Insert 14
2-3-4 Tree Example

Insert 28
2-3-4 Tree Example

Insert 28
2-3-4 Tree Example

Insert 17
2-3-4 Tree Example

Insert 7
2-3-4 Tree Example

Insert 7
2-3-4 Tree Example

Insert 7
2-3-4 Tree Example

Insert 52

1 12 8 2 25 6 14 28 17 7 52 16 48 68 3 26 29 53 55 45
2-3-4 Tree Example

Insert 52
2-3-4 Tree Example

Insert 16
2-3-4 Tree Example

Insert 16
2-3-4 Tree Example

Insert 48
2-3-4 Tree Example

Insert 48
2-3-4 Tree Example

Insert 68
2-3-4 Tree Example

Insert 68
2-3-4 Tree Example

Insert 3, 26
2-3-4 Tree Example

Insert 3, 26
2-3-4 Tree Example

Insert 55
2-3-4 Tree Example

Insert 55
2-3-4 Tree Example

Insert 45
2-3-4 Tree Example

Insert 45
2-3-4 Tree: Delete

- Leaf:
  - Just delete the key
  - Make sure that a leaf is not empty after deleting a key

Delete 2
2-3-4 Tree: Delete

- Leaf:
  - When key deletion would create an empty leaf, borrow a key from leaf 's immediate siblings (i.e. to the left and then right) and try to.

```
delete 4:

3,5

1,2  4  6

rotate

2,5

1  3,4  6
```
2-3-4 Tree: Delete

- Leaf:
  - If siblings are 2-nodes (no immediate sibling from which to borrow a key), steal a key from our parent by doing the opposite of a split.

Delete 6
2-3-4 Tree: Delete

- What if parent is a 2-node (one key)?

```
delete 7:
```

```
  4
 /  \
2   6
 /|   |
1 3  5 7
```

merge

```
  4
/  \
2   \
/    |
1 3  5,6,7
```

```
2-3-4 Tree: Delete

- What if parent is a 2-node (one key)?
  - Steal from siblings (parent’s)
  - Merge

delete 7:

```
2
  /  \
2   6
  / \  /
1   3 5 7
```

merge

```
2,4,6
  /  \
1   3 5 7
```

merge

```
2,4
  /  \
1   3
```

```
2-3-4 Tree: Delete

- What if parent is a 2-node (one key)?
  - Steal from siblings (parent’s)
  - Merge

delete 9:

```
6
/    \
2,4   8
\   /  \\
1 3 5 7 9
```

rotate
```
2
/  \\
1 3
```

merge
```
2
/  \\
1 3
```
```
4
/  \\
5 7 9
```
```
4
/  \\
5 7,8,9
```
2-3-4 Tree: Delete

- Internal Node:
  - Delete the predecessor, and swap it with the node to be deleted.

Delete 5: first delete 4, then swap 4 for 5.
2-3-4 Tree: Delete

- Internal Node:
  - Delete the predecessor, and swap it with the node to be deleted.
  - Key to deleted may move.

Delete 2: first delete 1, then swap 1 for 2.
2-3-4 Tree Example: Delete

Delete 3, 17, 55
2-3-4 Tree Example: Delete

Delete 1: borrow from siblings (rotate)
2-3-4 Tree Example: Delete

Delete 1
2-3-4 Tree Example: Delete

Delete 52: borrow from sibling
2-3-4 Tree Example: Delete

Delete 52: borrow from sibling
2-3-4 Tree Example: Delete

Delete 48: borrow from parent
2-3-4 Tree Example: Delete

Delete 2: borrow from parent, and parent
Delete 2: borrow from parent, and parent
2-3-4 Tree Example: Delete

Delete 14: delete 12, swap 12 for 14
2-3-4 Tree Example: Delete

Delete 14: delete 12, swap 12 for 14
2-3-4 Tree Example: Delete

Delete 25: delete 16, swap 16 for 25
2-3-4 Tree Example: Delete

Delete 25: delete 16, swap 16 for 25
Represent 2-3-4 tree as a BST

• Use "internal" red edges for 3- and 4- nodes.
• Require that 3-nodes be left-leaning.
Represent 2-3-4 tree as a BST

• Elementary BST search works
• Easy-to-maintain 1-1 correspondence with 2-3-4 trees
• Trees therefore have perfect black-link balance