2-3-4 tree

rooted tree

- perfectly balanced

(for each node, both subtrees have same height)

- each node has

  1 element & 2 subtrees

  2 elements & 3 subtrees
3 elements + 4 subtrees

(subtrees are empty at leaves)
subtrees store elements that are $< \text{ than everything in node}$

a) $< \text{ all in node or }$

b) $> \text{ in the gaps between node elements}$
contains: compare against root element, recurse in appropriate subtree

# nodes doubles each level & each node contains ≥ 1 elements

=> height ≤ \( \log n \) = \( O(\log n) \)
Insert: search for leaf where element would go

Add element to existing leaf

2 x 4
New element could be the fourth in least an overflow.

1 9

2 4 6 8
Solution: Promote middle element to parent, & split node.
If overflow was the root, just make a new root with middle element.
Remove: Always end up removing an element of a leaf.

\[ \times 34 \]
If lead was a 2-node, you underflow!
1-node was no element!
If root, just remove it & make one child the new root.
If adjacent sibling has $\geq 2$ elements, move closest element to parent & demote parent element separating you two.
If no adjacent sibling with \( \geq 2 \) elements...
Demote separating parent element & merge with sibling.
Red-black trees.
Emulates a 2-3-4 tree.
Represent each x-node with a couple BST nodes.
2-node

\[ \begin{array}{c}
\text{3} \\
\end{array} \quad \Rightarrow \quad \begin{array}{c}
\text{3}
\end{array} \]

3-node

\[ \begin{array}{c}
\text{2} \\
\text{4}
\end{array} \quad \Rightarrow \quad \begin{array}{c}
\text{2} \\
\text{4}
\end{array} \]

red

(child could be on other side)
black node is the leader

extra links from rep. of x-node point
to reps of children

x-nodes
Properties of red-black binary search tree

1) Every node is red or black
2) Root is black
3) Red nodes have only black children
4) Every path from any node \( v \) to null has
the same A black nodes (2-3-4 tree is perfectly balanced).

Every tree with those properties reps exactly one 2-3-4 tree.

Properties usually taken as definition.
Depth of red-black tree $\leq 2 \lg n$

contains same as always: $O(\log n)$
Insert
Add new leaf.
Make it red to be safe.
If parent is black:
Done!
If no parent:

Color it black:

\[ \text{Done!} \]

How we deal with a red node with red parent \( p \).
Two cases:

Case 1:

A misshapen 4-node rep. (p's sibling is black)
Case 1a:
If \( v \) is a left child.

Single rotation. \( p + g \)

Color \( p \) black \( + \) green.
Case 16.

Double rotation so $v$ has children $p$ and $w$. 
Case 2: p has red sibling!
Looks like an overflowed x-node!

Promote g's element to split by
- color g red
- color p black
- color w black

[Diagram]

- Node g
  - Child p
  - Child w
May need to keep going if \( g \) has red parent.

(color \( g \) black if \( g \) is root)

Insert does \( \leq 1 \) single or double rotation.
Remove:

Comes down to removing node $v$ with $\leq 1$ child.

If $v$ is red, it is a leaf. Just remove it.

Done.

If $v$ is black...
If $v$ has a red child $c$, we have a 3-node rep.

Just make $c$ black to rep a 2-node.
Otherwise, $r$ is lone rep. of a 2-node, that is about to underflow.

Treat situation as. We have a black node $u$ somewhere with 1 child. We want
To remove it, child must be black too.

If \( v \) is root, just delete it, and make child the new root.