Asymptotic Analysis

2-3 R/B AVL

Find \( O(\log n) \) \( O(\log n) \) \( O(\log n) \)

Add \( O(\log n) \) \( O(\log n) \) \( O(\log n) \)

Remove \( O(\log n) \) \( O(\log n) \) \( O(\log n) \)

So why use 2-3 tree?
- because on non-asymptotic differences

\[
\begin{array}{c}
2 \\
4 \\
8 \\
16 \\
\end{array}
\]

Worst case number of comparisons to find an element in a tree on \( n \) elements comes from this tree.

The height of this tree is \( \log_2 n \)

The number of comparisons is \( 2 \) per level.

So the number of comparisons is \( 2 \log_2 n \)

Which is double an AVL.

However, fewer adds and removes require a rebalance.

What if the number of elements per node is increased?
- Answer 1: Read becomes slower, write faster
- Answer 2: B-tree
B-tree is a multi-way search tree where
a B-tree of order m has the following properties:
- root has k-1 elements and k children
  where \(2 \leq k \leq m\), or is empty
- internal nodes have k-1 elements and \(k\) children where \(\lceil m/2 \rceil \leq k \leq m\)
- leaves have k-1 elements where \(\lceil m/2 \rceil \leq k \leq m\)
- tree is full

B-trees are useful when storing data that is too large to fit into memory.
When the order is set such that one page of virtual memory fits
in a node, HD access is limited.

Problem: in order to read pages of memory efficiently, nodes must use arrays.
This means that a B-tree could be half empty, this wastes memory
and slows operations.

Solution: B+ tree - minimum number of child is \(\lceil 2(n-1)/3 \rceil\) instead of \(\lceil m/2 \rceil\)

Problem: sequential access requires alternate reading from parent and child.
Solution: B+ tree - all elements are in leaves of the tree and each leaf has a pointer
to the next leaf in the tree.