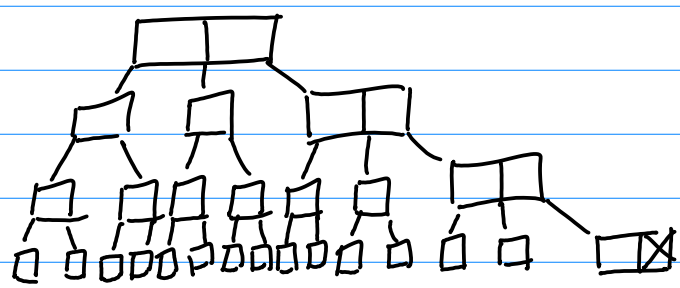
CPSC 220 11/4

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

2
4
8
16



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?
- answer1: read becomes slower, write faster
- answer2: B-tree

B-tree multi-way search tree where
a B-tree of order m has the following properties
- root has k-1 elements and k children
  where $2 <= k <= m$, or is empty
- internal nodes have k-1 elements and
  k children where $\lceil m/2 \rceil <= k <= m$
- leaves have k-1 elements where
  $\lceil m/2 \rceil <= k <= m$
- 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 $\lceil (2m-1)/3 \rceil$ 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