Colin Bradley

Design of a B-Tree Data Structure

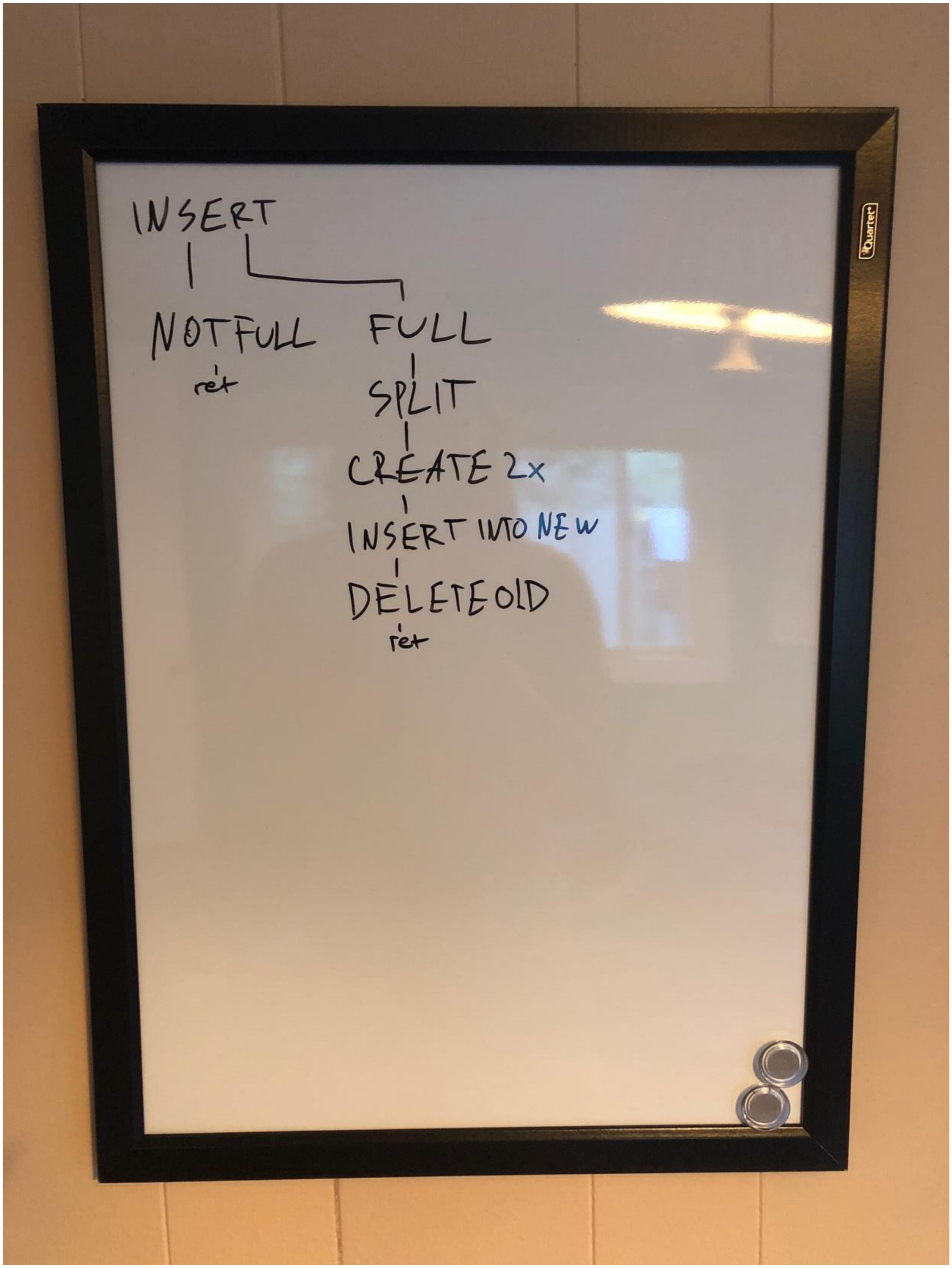
A node must contain:

m number of keys
keys must be sorted
depth of the tree, useful for checking if leaves are all on the correct level
is_leaf boolean value
keys.length = m -1
min_keys = ceil(m/2)

Required Functions:

- create new node
- check if keys are in ascending order
- sort keys (bubble sort? something not too complicated)
- move keys around to obey invariants
- delete nodes
- insert key into full and not-full node
-
- remove key from node
- find node with key
- ensure nodes are not overfilled (over m)
- ensure nodes are not underfilled (under ceil(m/2))
- merge two nodes
- split single node into two
- promote keys to parent node

Colin Bradley


INSERT
NOTFULL
ret
FULL
SPLIT
CREATE 2x
INSERT INTO NEW
DELETE OLD
ret