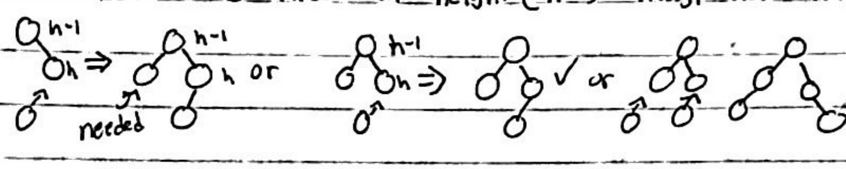
Nafis Abeer EC330 HW6

1a) Prove that for any AVL search tree, the number of only children is at most $n/2$, if the tree has n nodes.

- In order to maintain AVL search tree properties, any left subtrees from any given ~~root~~ node can differ in height by at most 1 from that nodes right subtree.
- That means for any given node at height (h), in order for it to acquire an only child, the remainder of the nodes at height (h) must also acquire children and every node at height (h-1) must not have only children.



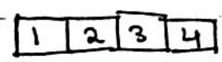
- In any scenario, the only children of an AVL search tree can only exist in the leaves of the tree, with the level above the height of the tree having been completely filled, causing there to be at least twice as many nodes as only children. Every only child also counts as a node and has a parent that is not an only child.

1b) B-trees have numbers stored in ascending order. Trees are split based on initial bias as well. Because of various reasons as such, the order of insertion matters for This reason, the statement is false. Counter example:

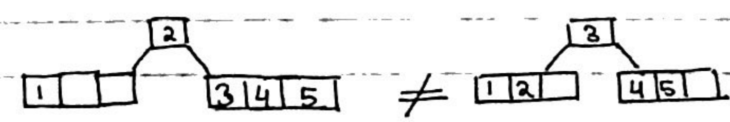
m=4

insert 1, 2, 3, 4, 5

↳ 1 → [1|2] → [1|2|3]



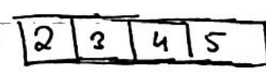
split down the middle we will go w/ right bias



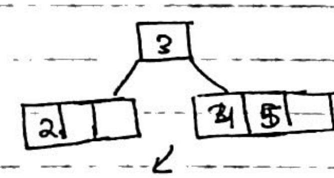
m=4

insert 4 3 2 5 1

→ 4 → [3|4] → [2|3|4]

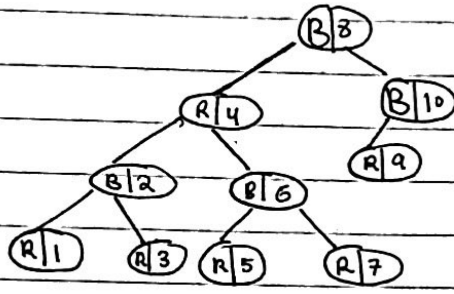


split required, will go right bias again,



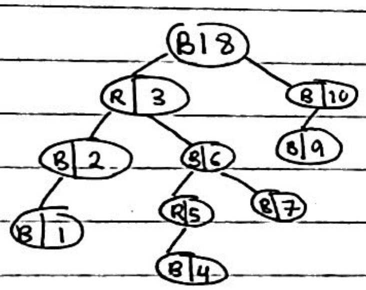
2a) R B R R R B R B R B in-order traversal # 1-10

1) Every node either Red or black
2) The root is Black
3) Every leaf (NIL) is black
4) if a node is Red both its children are Black
5) All simple path down to a leaf from each node contains the same # of Black nodes



Inorder-Tree-walk (x)
if x ≠ NIL
    Inorder-Tree-walk (x.left)
    Print x.key
    Inorder-Tree-walk (x.right)
return

b) B B B R B B R B B B



Post-order-Tree-walk (x)
if x ≠ NIL
    Post-order-Tree-walk (x.Left)
    Post-order-Tree-walk (x.right)
    Print x.key
return

OR