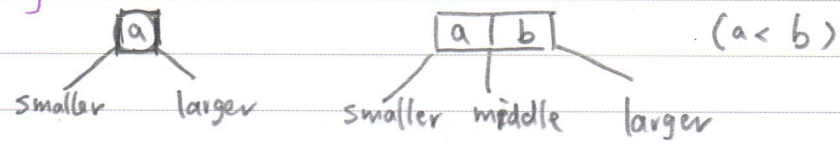


* remove in AVL tree
rotate (similarly) , and go up

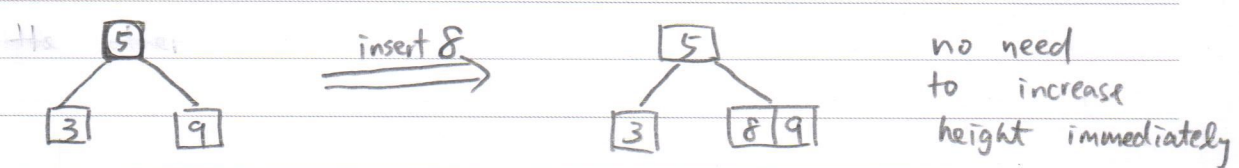
* AVL : strict & rebalance "almost immediately"
after adding a new node
what if we can temporarily
add a new element without increasing height

* (2,3) tree

① each node can contain flexibility 1 or 2 elements

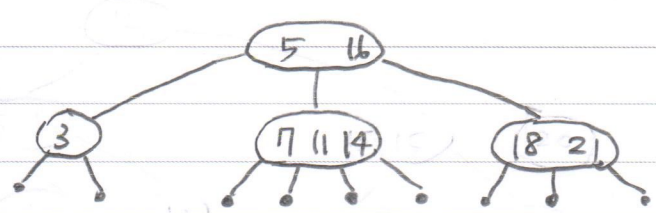
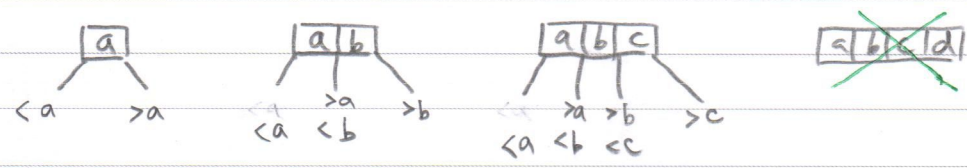


i.e. 2 or 3 children

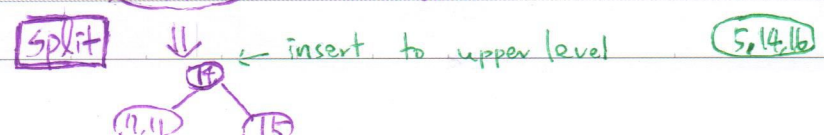


② all leaf nodes are of the same height (balanced)

* (2,3,4) tree ((2,4) tree)



- Insert 4 : (3,4)
- Insert 19 : (18, 19, 21) or Insert 25 : (18, 21, 25)
- Insert 15 : (7, 11, 14, 15) overflow



* 2-3-4 tree

- nodes w/ 2 or 3 or 4 children
(1 or 2 or 3 keys)
- all leaves of the same depth

h levels, at most a 4-way tree

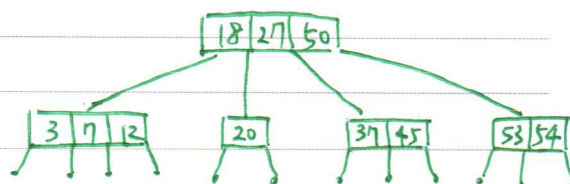
$$n \leq \sum_{i=0}^h 4^i$$

at least a binary tree

$$n \geq \sum_{i=0}^h 2^i$$

$$\Rightarrow h = O(\log n)$$

- multi-way search property



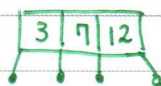
* insert

- search for the leaf to be inserted $O(h)$

- 2-node : just insert
- 3-node : just insert
- 4-node : split because of "overflow"

— to keep same-depth property

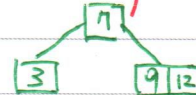
9 →



⇒ overflow



⇒ create new mid point



insert to upper level

- # splits : at most h

$$\Rightarrow \text{insertion} = \begin{cases} O(h) \\ + O(h) \end{cases} = O(\log n)$$

* removal

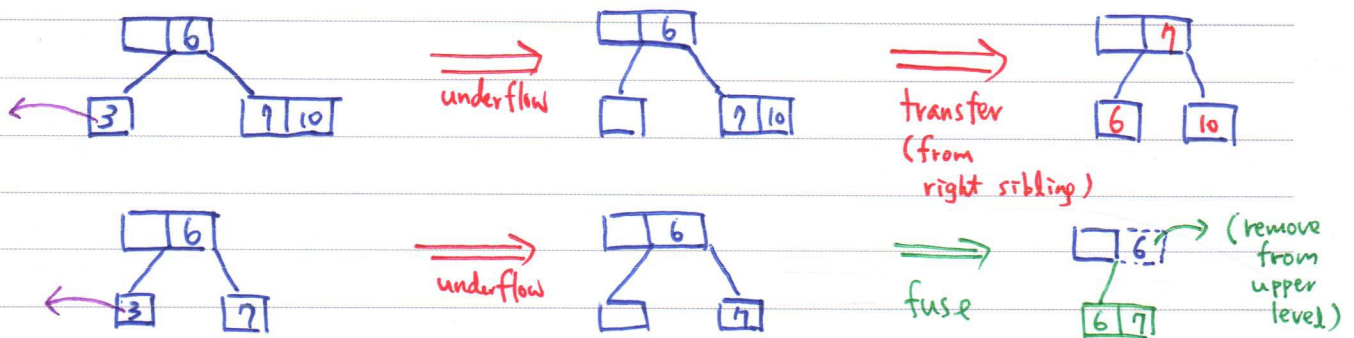
- ① remove a leaf node always
— to remove subtrees "easily"

non-leaf remove = (replace w/ succeeding)
+ remove leaf

- ② 4-node : just remove
3-node : just remove
2-node : "underflow"

Ⓐ borrow from sibling to get depth same (transfer)
else

Ⓑ fuse parent with sibling to avoid "empty node"

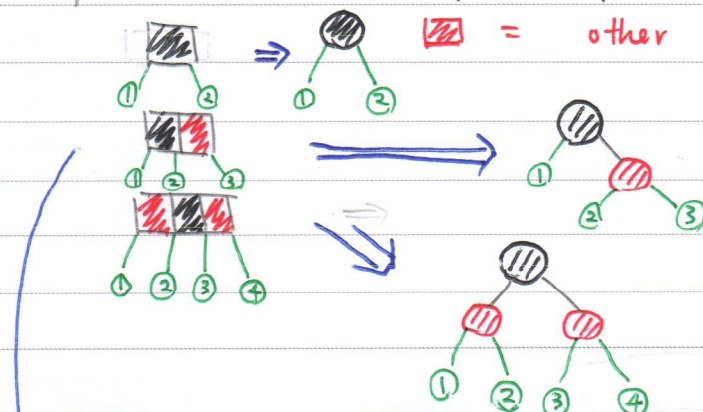


transfer : $O(1)$ + search for sibling + transfer
fuse : $O(h)$ (similar to split) } remove at most $O(h) = O(\lg n)$ [often faster]

* final words about 2-3-4 tree
easy to understand, very hard to implement

next: red-black tree \equiv 2-3-4 tree
"harder" to understand, "easier" to implement

* Simple idea = representative key of a 2-3-4 node
 = other keys



decompose
2-3-4 node
to
nodes in BST

or another way of coloring


2012

Subject :


No. :

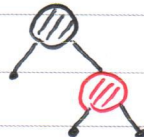
10-8


Date :/...../.....


* insert to 2-node
 \equiv add  to

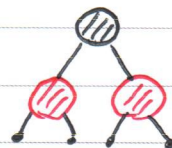


insert to 3-node
 \equiv add  to




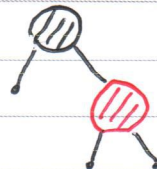
if left of 

remove from 4-node
 \equiv remove  from



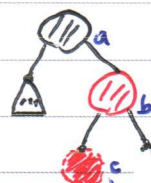
for 

remove from 3-node
 \equiv remove  from

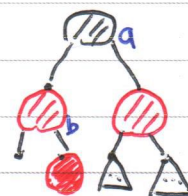


for 

* insert  to right of



or





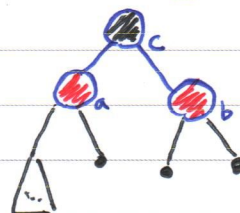
red-red conflict

(i.e. no valid 2-3-4 equivalence)

restructure

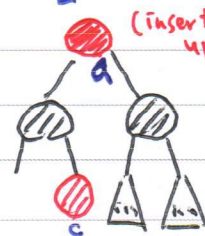
split \equiv recolor

insert
 $=$ add  to leaf
 $+$ restructure
 (if RR in "3-node")
 or recolor 
 (if RR in "4-node")



$O(1)$

sibling of
 b is black
 [or empty]



$O(h)$

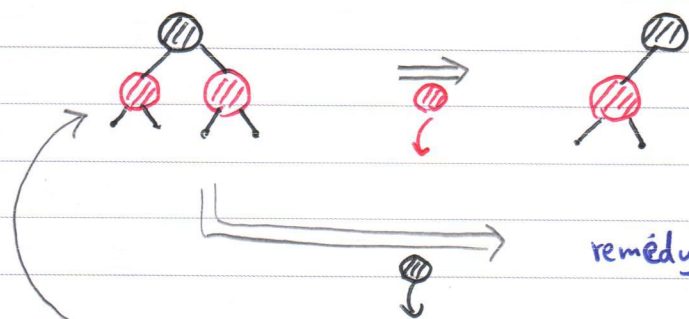
sibling of b
 is red
 (overflow cases)

* remove

① non-leaf in 2-3-4 \Rightarrow non-leaf in RB

replace w/ succeeding

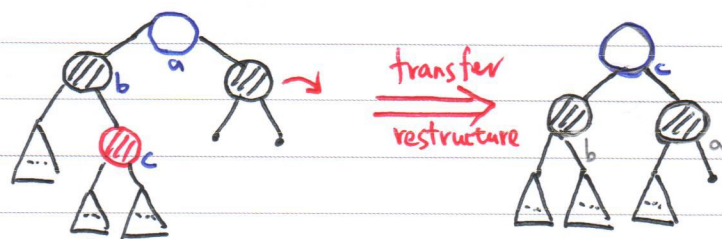
② remove 4-node at leaf



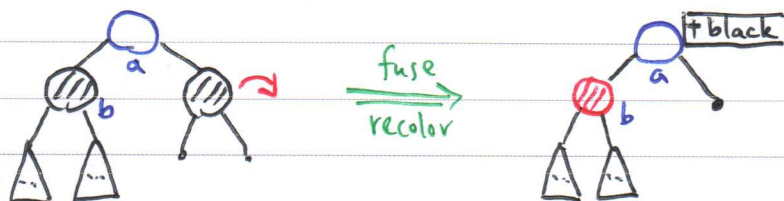
remedy by removing leaf only

③ remove 3-node at leaf

④ remove 2-node & transfer from sibling



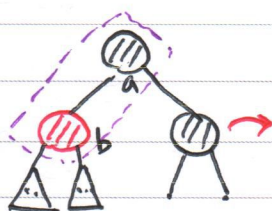
⑤ remove 2-node & fuse parent/sibling



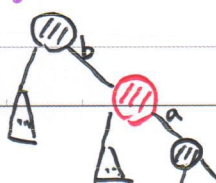
$\bigcirc + \text{black} = \text{black}$

$\bigcirc + \text{black} = \text{double black}$

(request removing upper level)
until \bigcirc or root



adjust to



*	2-3-4	R-B
	insert - 2	+ (red circle)
	insert - 3	+ (red circle) & restructure if RR
	insert - 4	+ (red circle) & recolors if RR
	remove - 4	- (red circle) (- (black circle) handled by finding leaf)
	remove - 3	- (red circle)
	remove - 2	or - (red circle) & recolor (red circle) adjust if necessary transfer by restructure or fuse-remove by recolors

complicated operations
(insert-4, remove-2)
(fuse)
mostly handled by recolors

* RB properties

* (red circle) has (black circle) child
lower-level 2-3-4 node

- root is (black circle) : representative of 2-3-4 root
- external nodes • (null node) w/ same black depth
depth in 2-3-4

