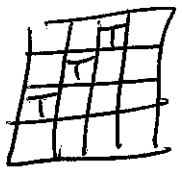
E	E	M	P
W	O	U	E
C	A	T	R
A	M	Y	X

prefix "COM" used



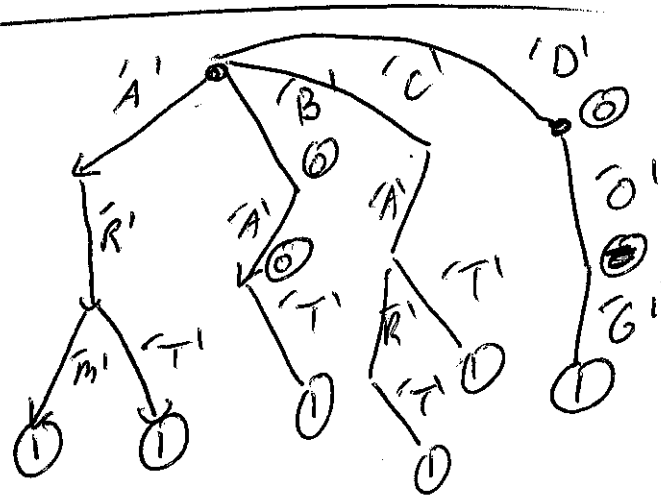
curX [0]
curY [2]

from here go in all 8 directions skipp (4) out of bounds
(5) used tiles

from E ← M → P
↓
U E

if no word starts with "COMU", then DON'T make the recursive call OR cut off that branch in the beginning of the recursive function

Tris



How to store words efficiently

"CAT"
"CART"
"BAT"
"MOP"
"MINT"
"ARM"
"ACT"
"ART"

```
Struct tri {
    int isWord;
    struct tri* next[26];
};
```

How to INSERT

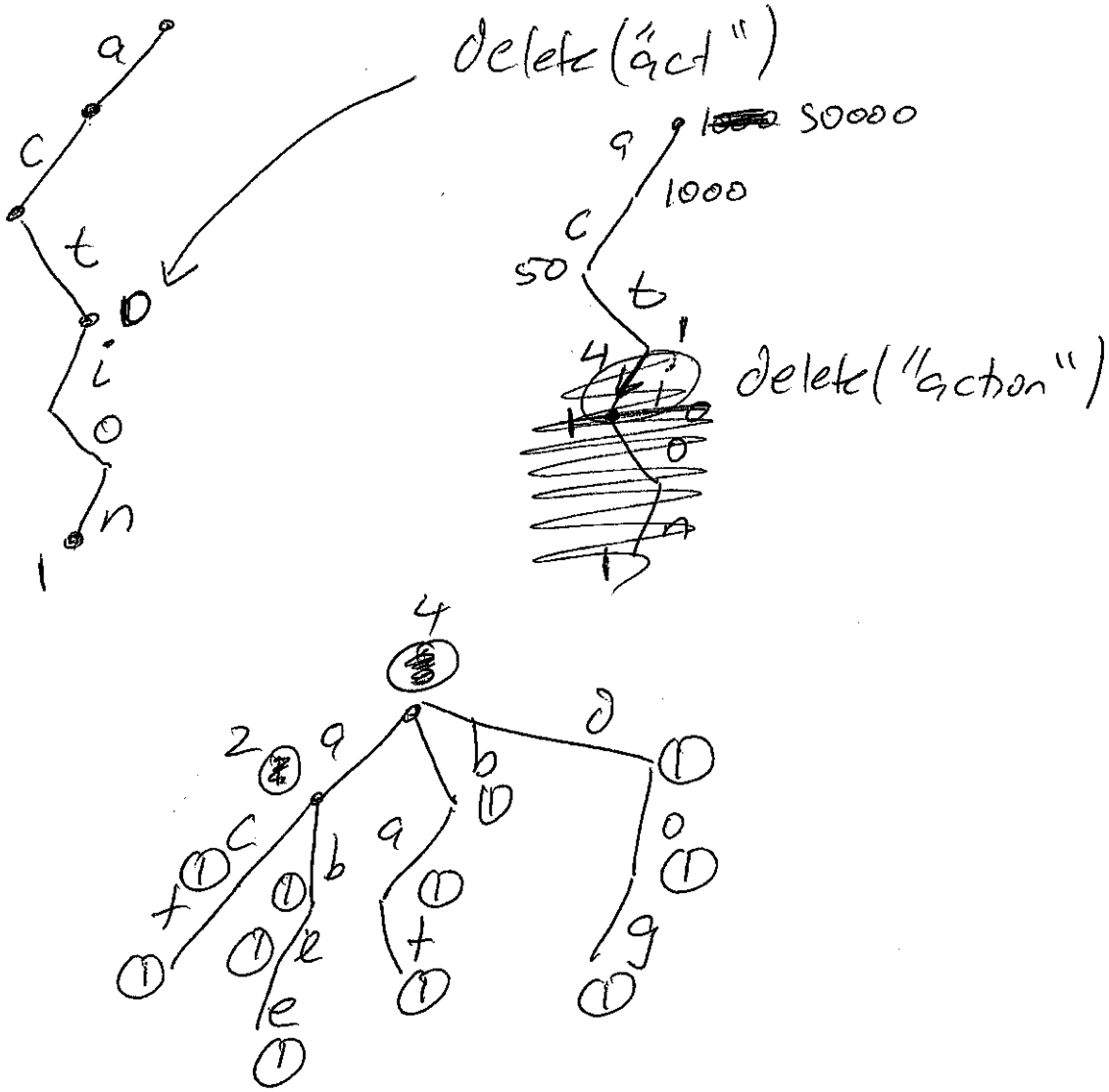
"DOG" → go down specified path.
"DO" if no "next" node exists, form it. When we get to the end of the word, put a 1 in its spot.

How to SEARCH

Go down tree. If you get stuck, return 0. If you get to the right node return its isWord component.

```
int search( struct tri* myTrie, char word[], int k){
    if (myTrie == NULL) return 0;
    if (k == strlen(word))
        return myTrie -> isWord;
    return search(myTrie -> next[ word[k] - 'a' ],
                  word, k+1
}
```

Delete of a word



prefix("zz") → 0

prefix("adj") → 1

AVL Trees

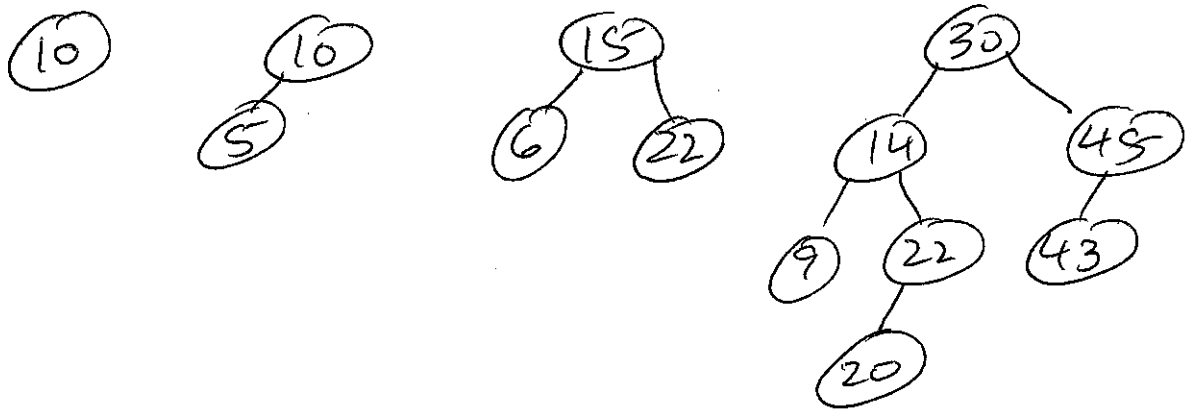
1st balanced binary tree

meaning that height = $O(\lg n)$ where n = # nodes so insert, delete take $O(\lg n)$ time.

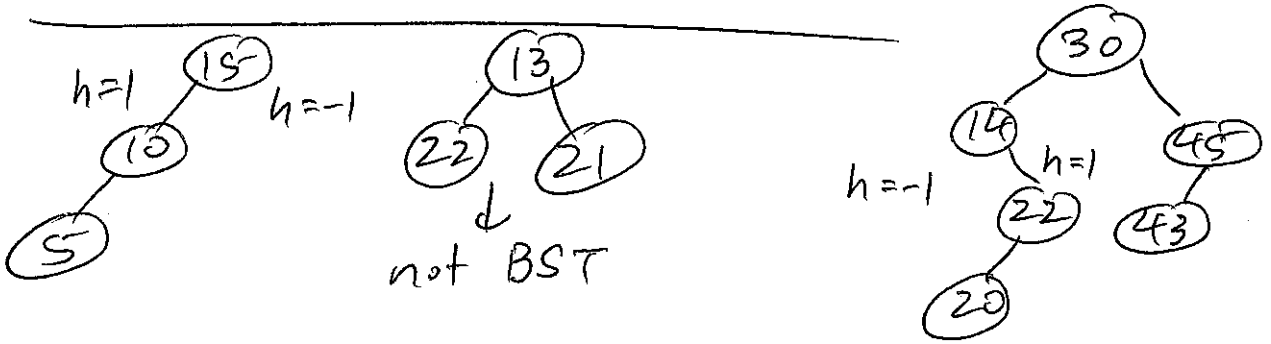
(1) Binary Search Tree

(2) for any node the height of its left and right subtrees can not differ by more than 1.

Valid AUL Trees

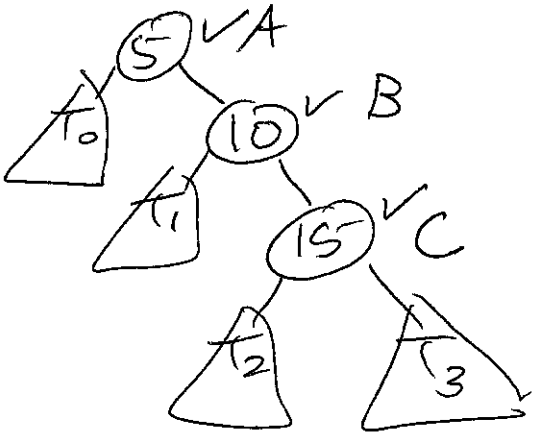


Invalid (NOT AUL)



AVL Tree Insert

Insert 15

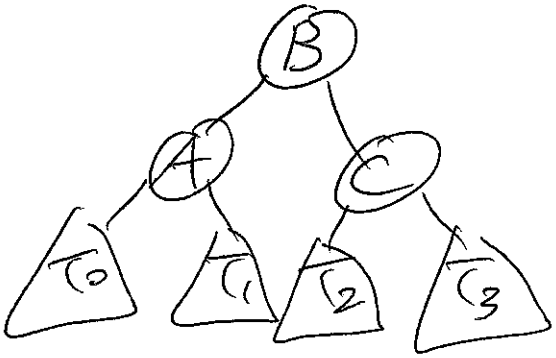


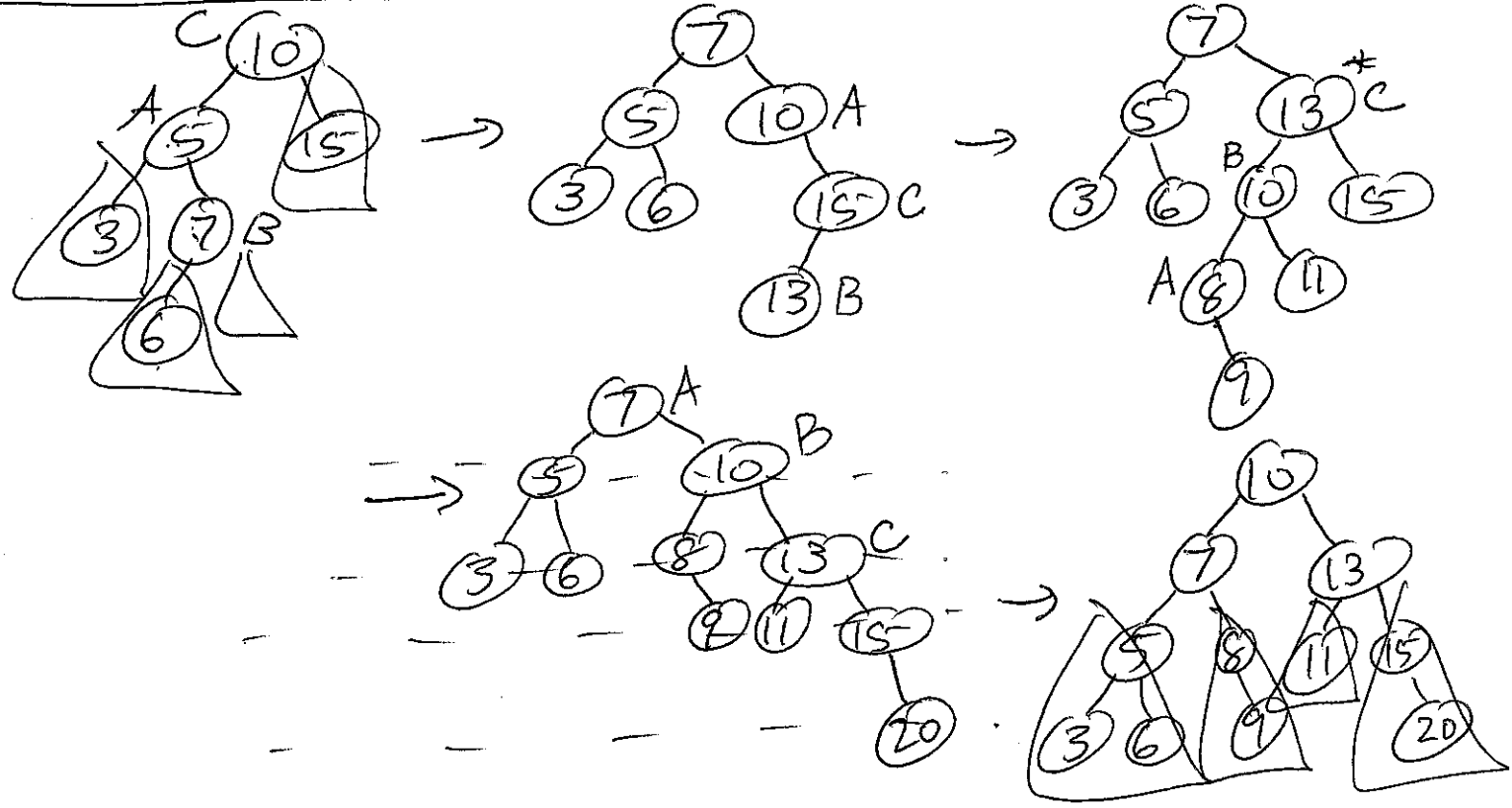
(1) Regular Insert

(2) Start marching up the tree. At each node check for imbalance

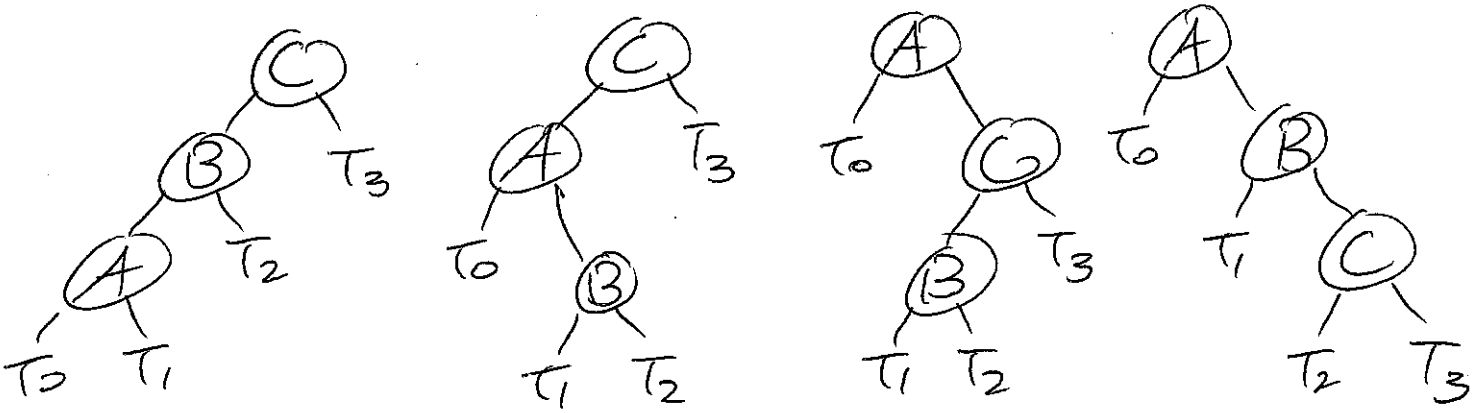
(3) If you find one, "rebalance".

$T_0 < A < T_1 < B < T_2 < C < T_3$





When you start tracing up, you may find one of 4 problems:

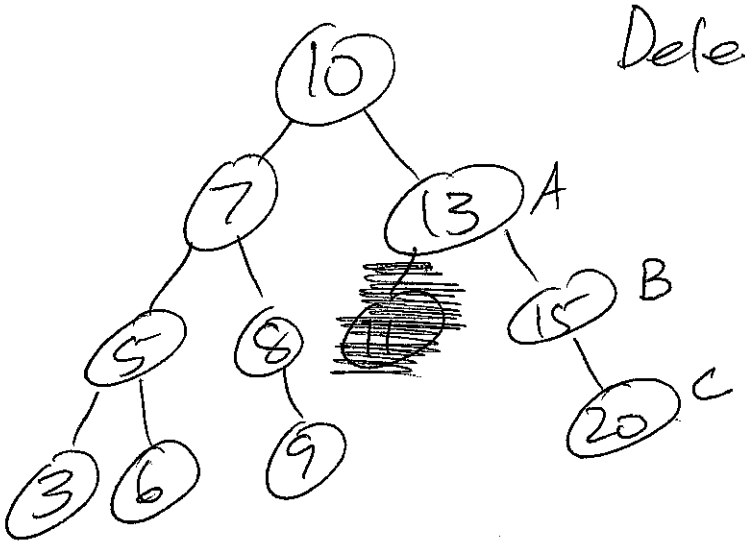


MOST COMMON ERROR

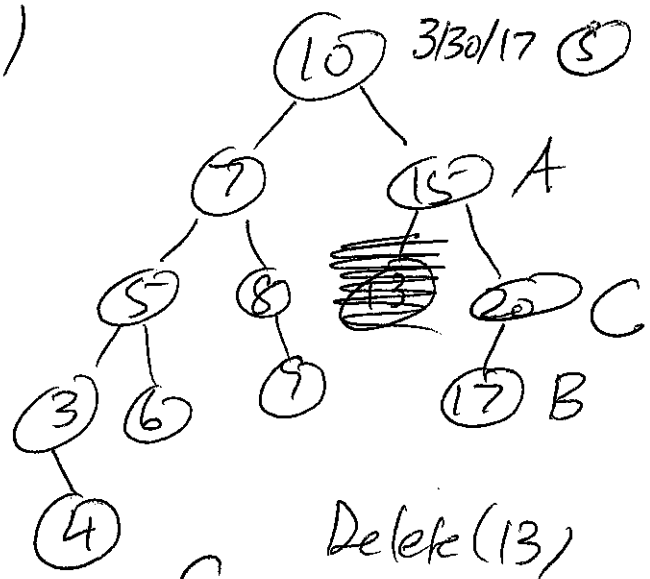
① Not rebalancing at "lowest" node (or 1st node) with imbalance.

② Answer that isn't a valid binary search tree!

Delete (11)

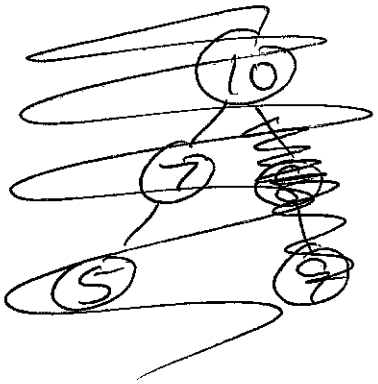


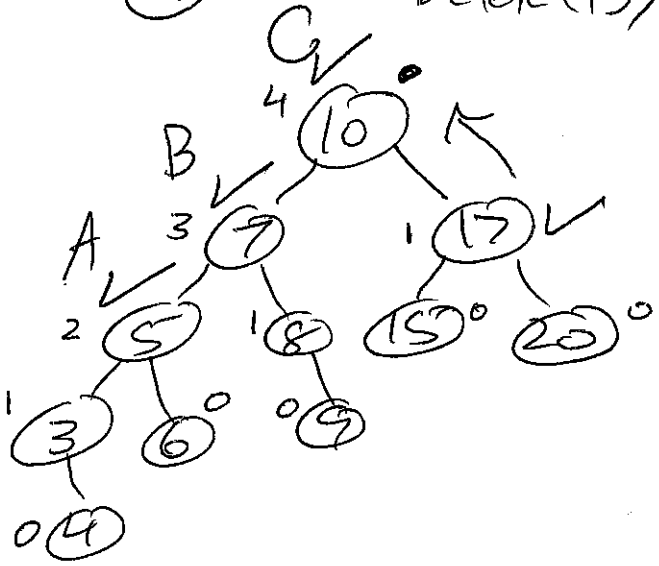
→



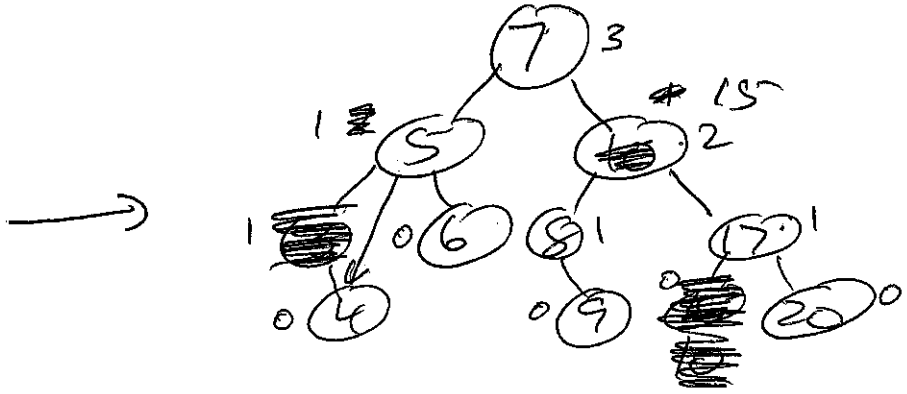
Delete (13)

→

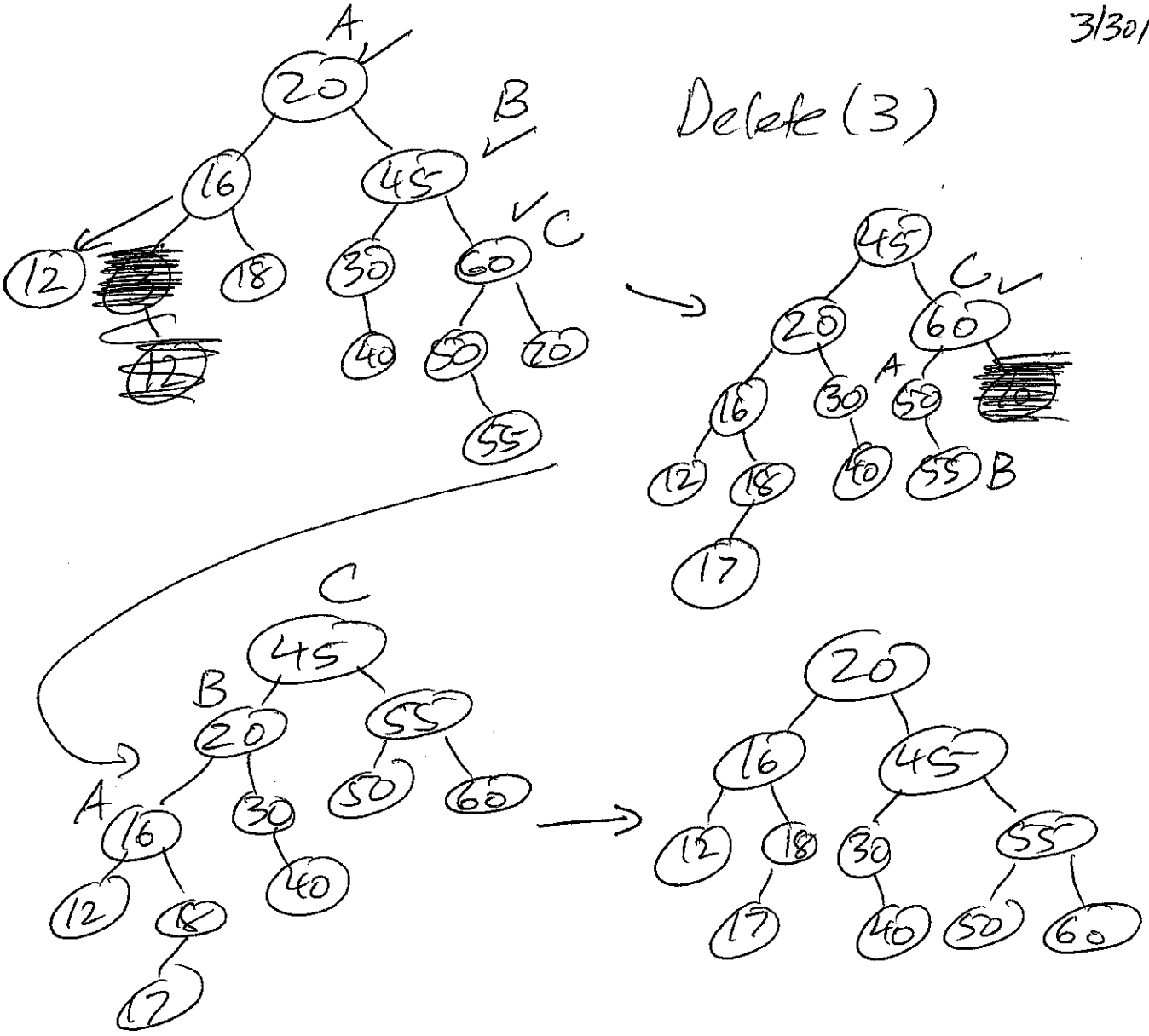




→



Delete(3)


A
B
C
20
16
45
12
18
30
60
40
50
70
55
45
20
60
16
30
50
70
12
18
40
55
17
C
45
B
20
55
A
16
30
50
60
12
18
40
17
20
16
45
12
18
30
55
17
40
50
60

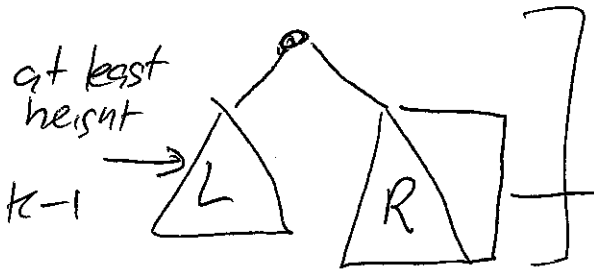
For an AVL tree of height h, it contains at least $F_{h+3} - 1$ nodes where $F_n = n^{th}$ Fibonacci number.

$h = 0$ 1 node $\quad F_{0+3} - 1 = F_3 - 1 = 2 - 1 = 1$ ✓

$h = 1$ 2 node $\quad F_{1+3} - 1 = F_4 - 1 = 3 - 1 = 2$ ✓

Assume for all non-neg ints $h \leq k$, where k is a arbitrarily chosen positive integer that AVL trees of height h have at least $F_{h+3} - 1$ nodes.

Prove for $h = k+1$ that AVL tree of height $k+1$ has at least $F_{k+4} - 1$ nodes.



$$\#\text{ nodes} \geq (\#\text{ nodes tree height } k-1) + (\#\text{ nodes tree height } k) + 1$$

$$\geq (F_{k+2} - 1) + (F_{k+3} - 1) + 1$$

$$= F_{k+2} + F_{k+3} - 1$$

$$= F_{k+4} - 1$$ ✓

An AVL tree of height h has at least $F_{h+3} - 1$ nodes

$$F_n = \frac{1}{\sqrt{5}}\left(\left(\frac{1+\sqrt{5}}{2}\right)^n - \left(\frac{1-\sqrt{5}}{2}\right)^n\right)$$

ϕ

Doesn't affect the answer much

$$F_n = \frac{1}{\sqrt{5}}\phi^n$$

n = # of nodes then

$$n \geq \frac{1}{\sqrt{5}}\phi^{h+3} - 1$$

$$n+1 \geq \frac{1}{\sqrt{5}}\phi^{h+3}$$

$$\lg_\phi \sqrt{5}(n+1) \geq \lg_\phi \phi^{h+3}$$

$$h+3 \leq \lg_\phi(\sqrt{5}(n+1))$$

$$h \leq \lg_\phi(\sqrt{5}(n+1)) - 3$$

$$h = O(\lg n)$$

3/30/17 (9)

Red - Black Trees
B - Trees
Spky Tree