COP 3502 10/29/21

(1) DON'T USE VARIABLE LENGTH ARRAY!

int nums[n]; → variable → bad

instead int* nums = malloc(sizeof(int)*n);

(2) finish bin trees

(3) AVL trees

Run Times for bin trees

~~Search~~, insert, delete

Search - goes down single path $O(h)$

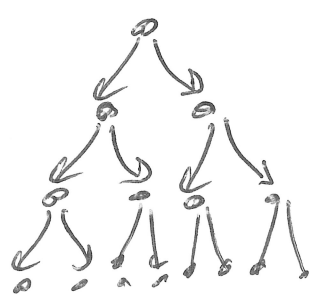
h = height tree

Visit each node once (traversals, height) - $O(n)$

relationship btw h, n.

best case for search

h=3



Complete binary tree - all "filled in" except possibly for the last row, here most left → right.

nodes = $1+2+4+8$ (max) $1+2+4\cdots+2^h = 2^{h+1}-1$

$= 1+2+4+1$ (min) $1+2+4\cdots+2^{h-1}+1 = 2^h$

$n = 2^{h+1}-1$

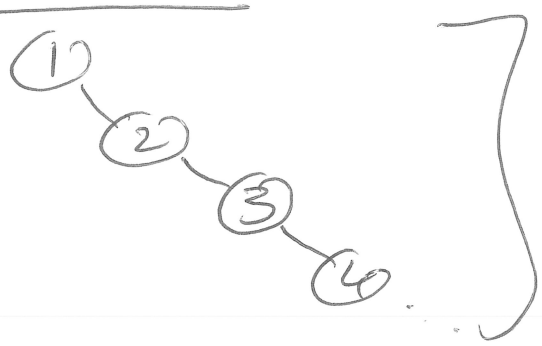
$n+1 = 2^{h+1}$

$\log_2(n+1) = h+1$

→ $h = \log_2(n+1) - 1$

Search $O(\log n)$.

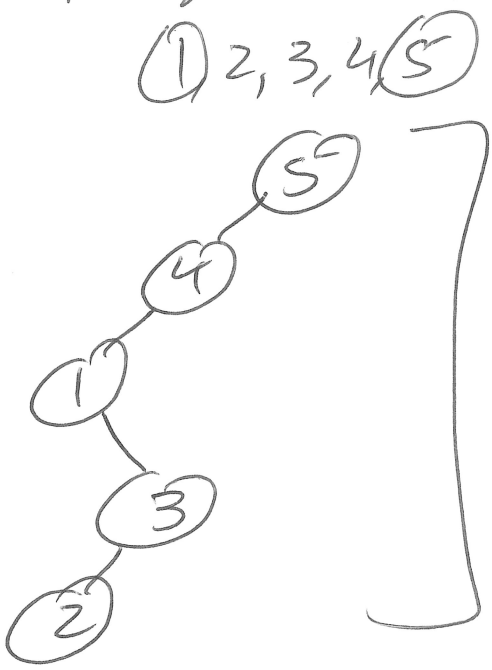
Worst Case



n nodes
$h = n-1$
$h = O(n)$

Randomly designed binary tree $h = O(\lg n)$ on avg.

How many ways can I insert $1, 2, \ldots, n$ in a BST to get a tree w/height $n-1$?



every level except last 2 choices

2^{n-1}.

Q: Can we obtain worst case $O(\lg n)$ insert, delete, search?

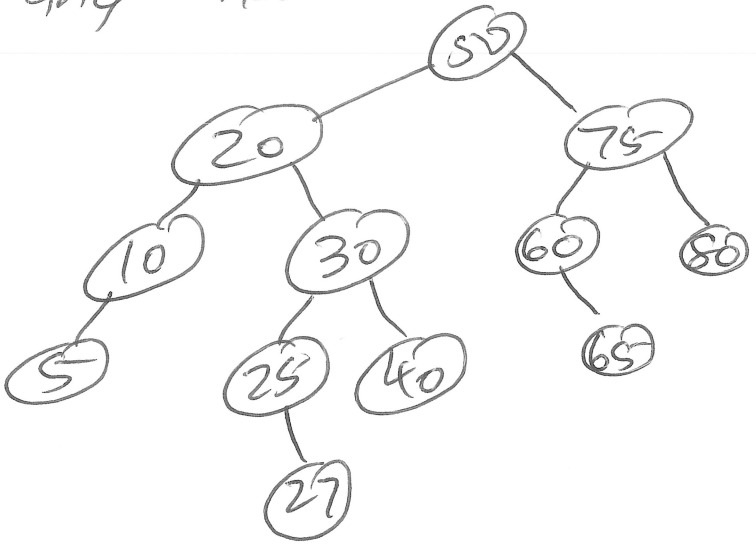
Ans: Balanced Binary Search Trees

- AVL Tree

AVL Trees

① Valid BST

② AVL node property: the heights of Left + Right subtrees of any node can't differ by more than 1.



1. Prove $h = O(\lg n)$

Given an AVL tree of height h, it must have AT LEAST $F_{h+3} - 1$ nodes, where F_n denotes the n^{th} Fibonacci #.

let $T(h)$ = min # nodes in an AVL tree height h

$$T(h) \geq T(h-1) + T(h-2) + 1$$

$$T(h) = T(h-1) + T(h-2) + 1$$

$$T(0) = 1, \quad T(1) = 2$$

$$F_n = O(\phi^n)$$

$$\phi = \frac{1+\sqrt{5}}{2}$$

$$T(h+1) = T(h) + T(h-1) + 1$$

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

$$= F_{h+4} - 1 \quad \checkmark$$

Sketch Induction ↓

num nodes $n = F_{h+3} - 1$

height

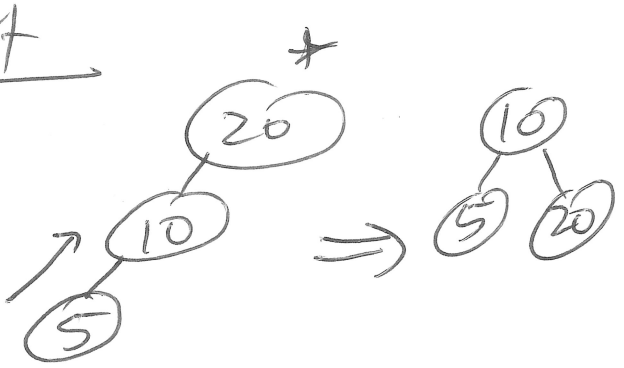
$n = \phi^{h+3}$

$\log_\phi n = h+3 \rightarrow h = \log_\phi n - 3 = O(\lg n)$

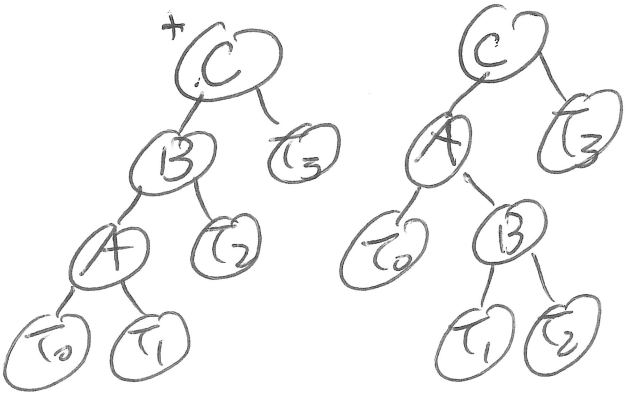
$\phi \approx 1.6$

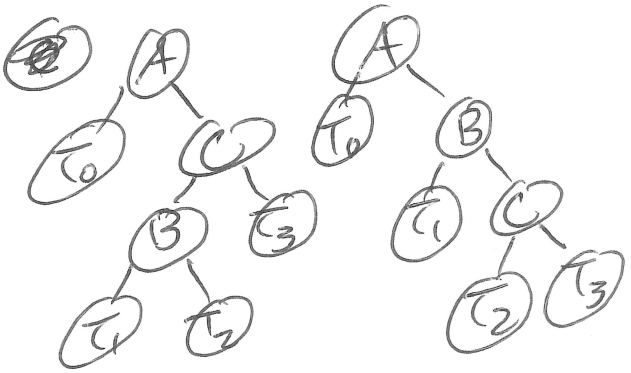
If we can maintain these properties, insert, search, delete $O(\lg n)$ worst case.

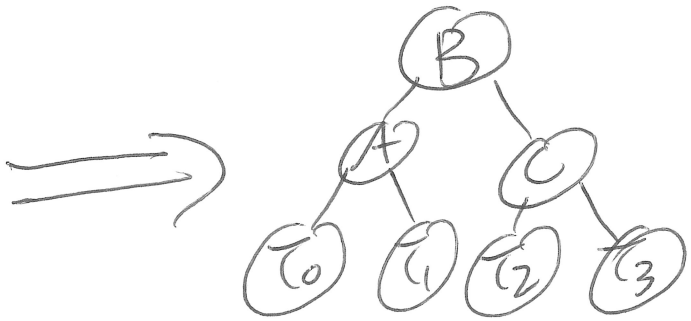
Insert



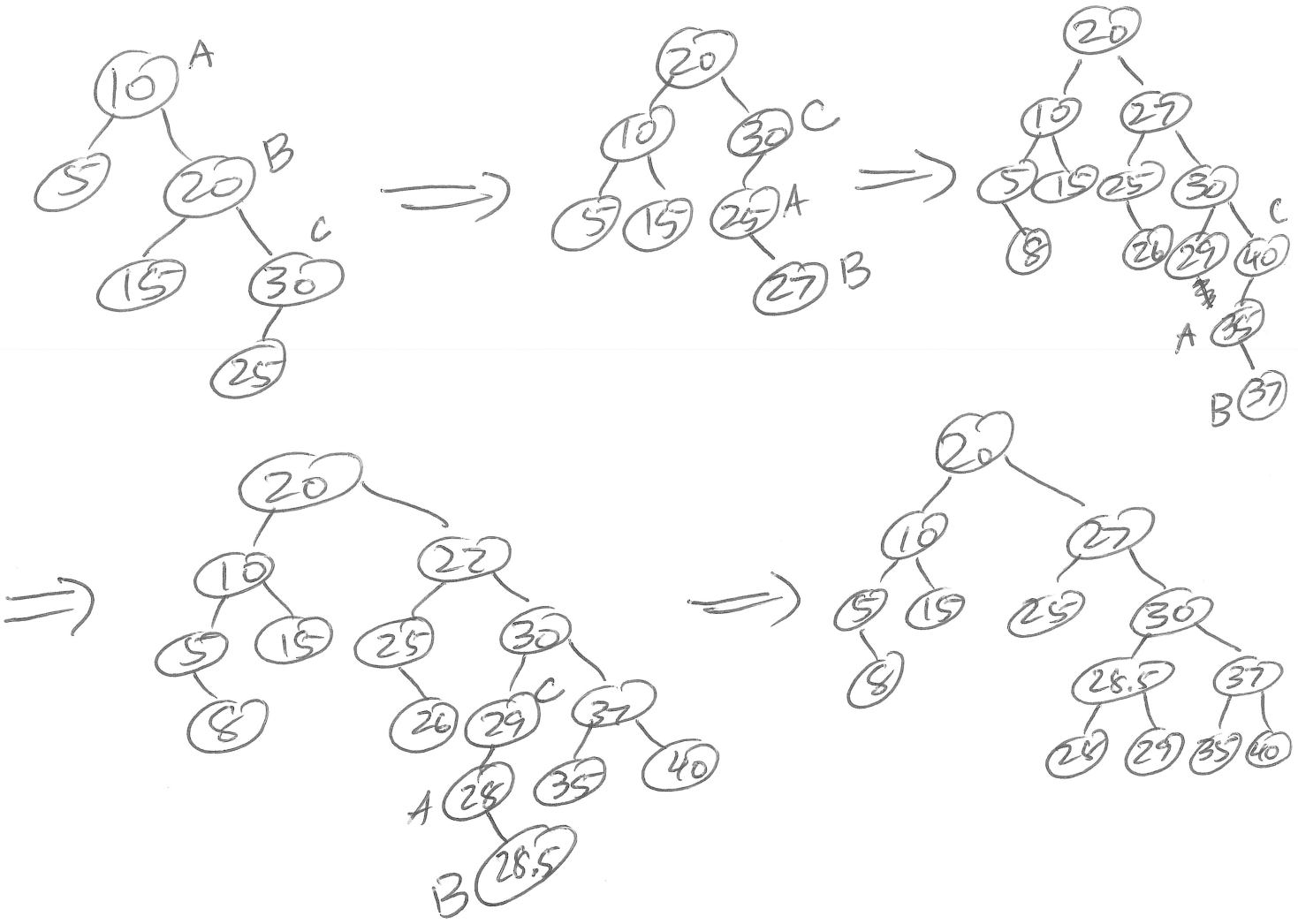
1) Do reg bst insert
2) Go back up ancestral path see if the node is balanced.
3) If not, do a rebalance

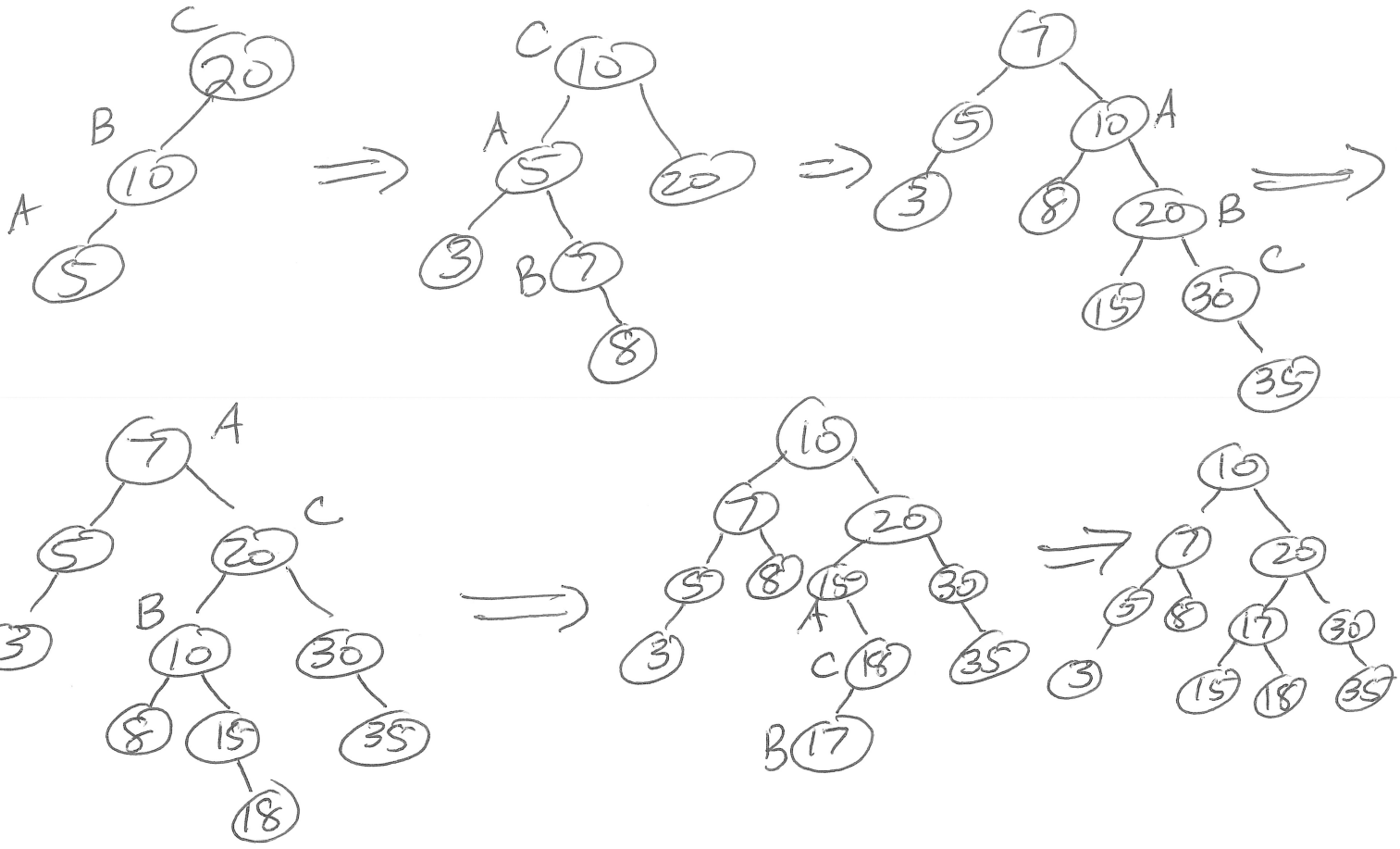






↳ Single/double rotation alternate


10
A
5
20
B
15
30
C
25
20
10
30
C
5
15
25
A
27
B
20
10
27
5
15
25
30
C
8
26
29
40
A
35
B
37
20
10
27
5
15
25
30
8
26
29
37
A
28
35
40
B
28.5
20
10
27
5
15
25
30
8
28.5
37
26
29
35
40


C
20
B
10
A
5
C
10
A
5
20
3
B
7
8
7
5
10
A
3
8
20
B
15
30
C
35
7
A
5
20
C
3
B
10
30
8
15
35
18
10
7
20
5
8
15
30
3
A
C
18
35
B
17
10
7
20
5
8
17
30
3
15
18
35