CS180 - AVL Trees (part 2), treaps

Note Title 11/16/2009

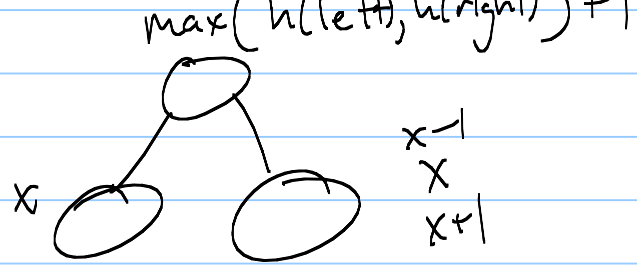
Announcements:

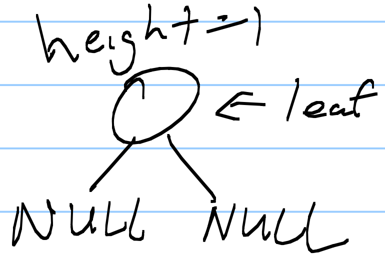
- Program coming today:
  implement insert method in AVL tree code

AVL trees :

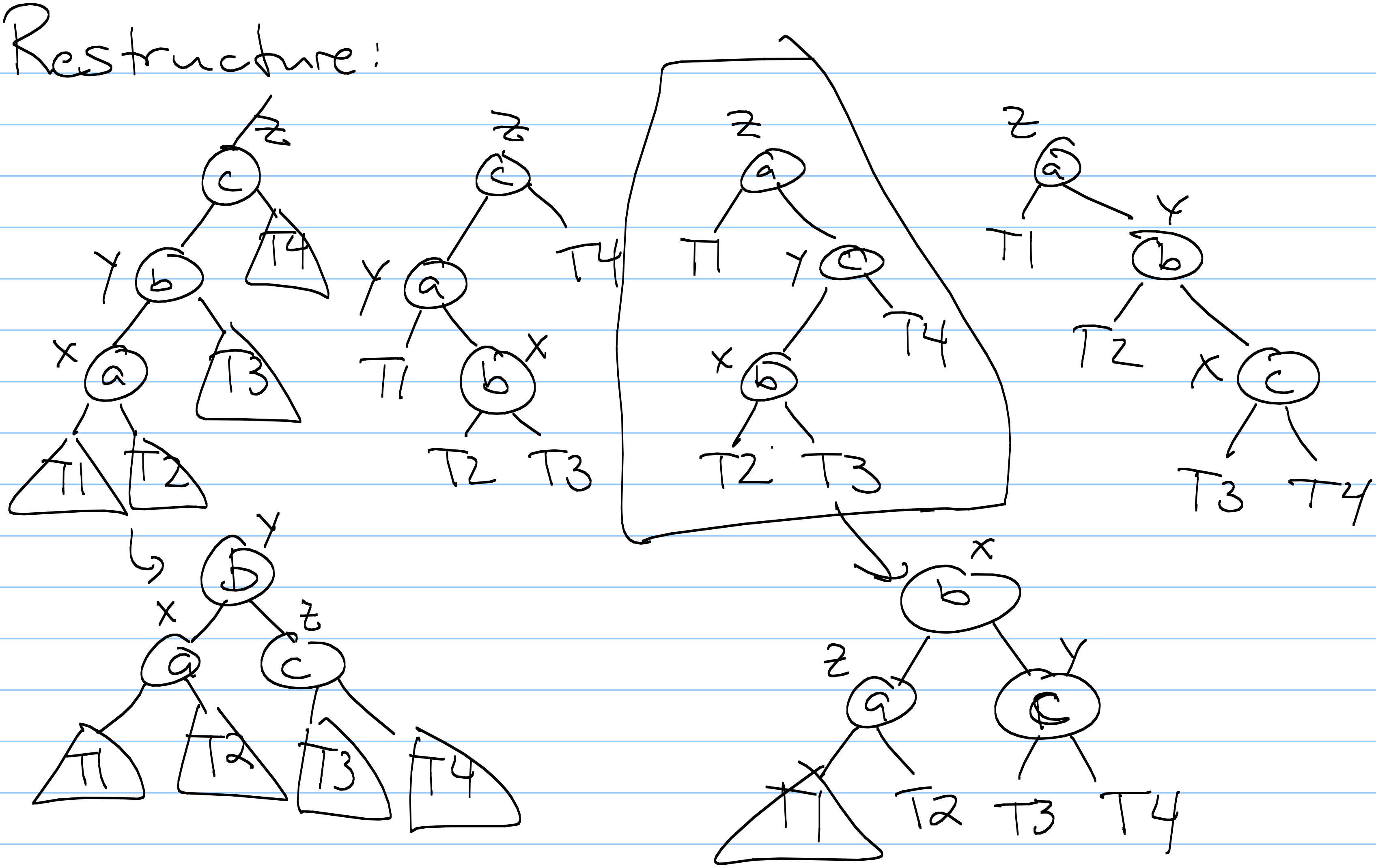
Height-Balance property:
For every (internal) node of T, the heights of the children differ by at most 1.

↳ height of tree ≤ $2 \cdot \log_2 n$





Restructure:



What was the running time?

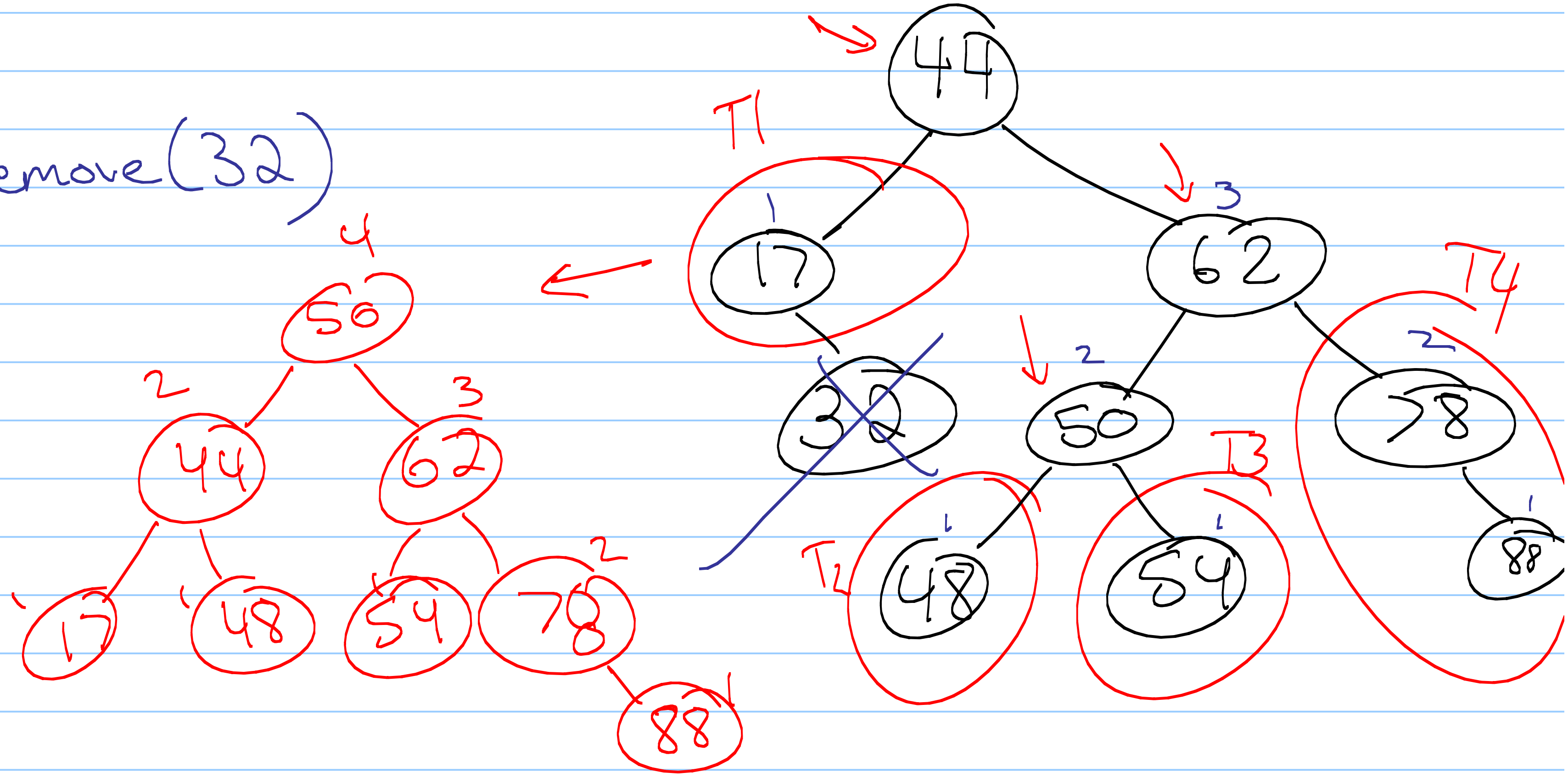
$O(\log n)$ – Why?

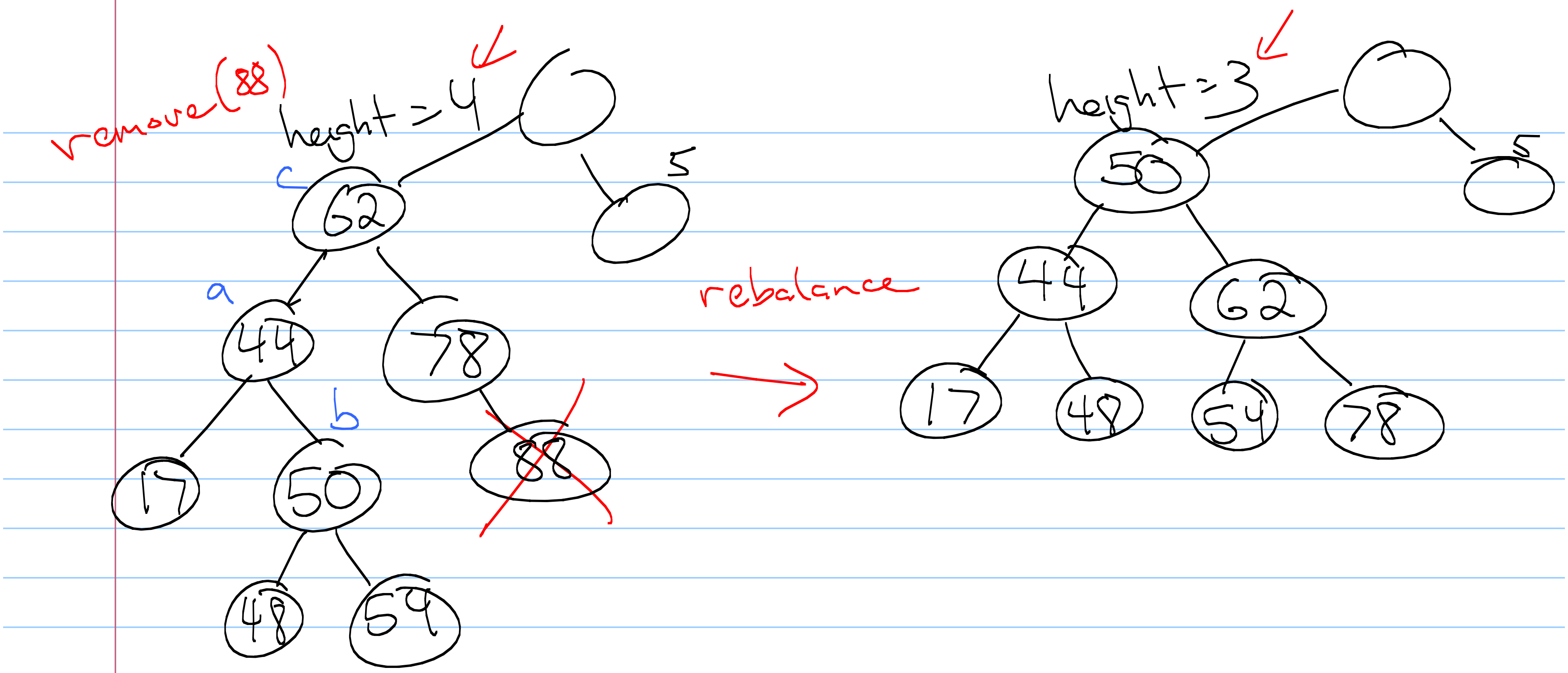
insert:

- search where new element <u>should</u> go
  $O(h)$
- go back up + find lowest place where height-balance is broken
  $O(h)$
- rotate + fix heights – $O(1)$
  (only 1 rotation is necessary!)

Remove is similar
(but a bit more complicated.)

remove(32)





What can this do to parent of this subtree?

Run time of remove:

$O(\log n)$ – might do $O(\log n)$ rebalances
(one at each level)

Since fixing the subtree can reduce the height by 1.

Some search tree alternatives:

- AVL: $O(\log n)$ to seach & insert & delete
  ($\text{max height} = 2 \cdot \log_2 n$)

  downside: delete may rotate $O(\log n)$ times, which is slower in practice

- Red Black: same worst case height $O(\log n)$ for search, insert, delete

  Pro: only $O(1)$ rotations per operation

  ↳ so faster, but rules are more complicated

A new data structure: treaps

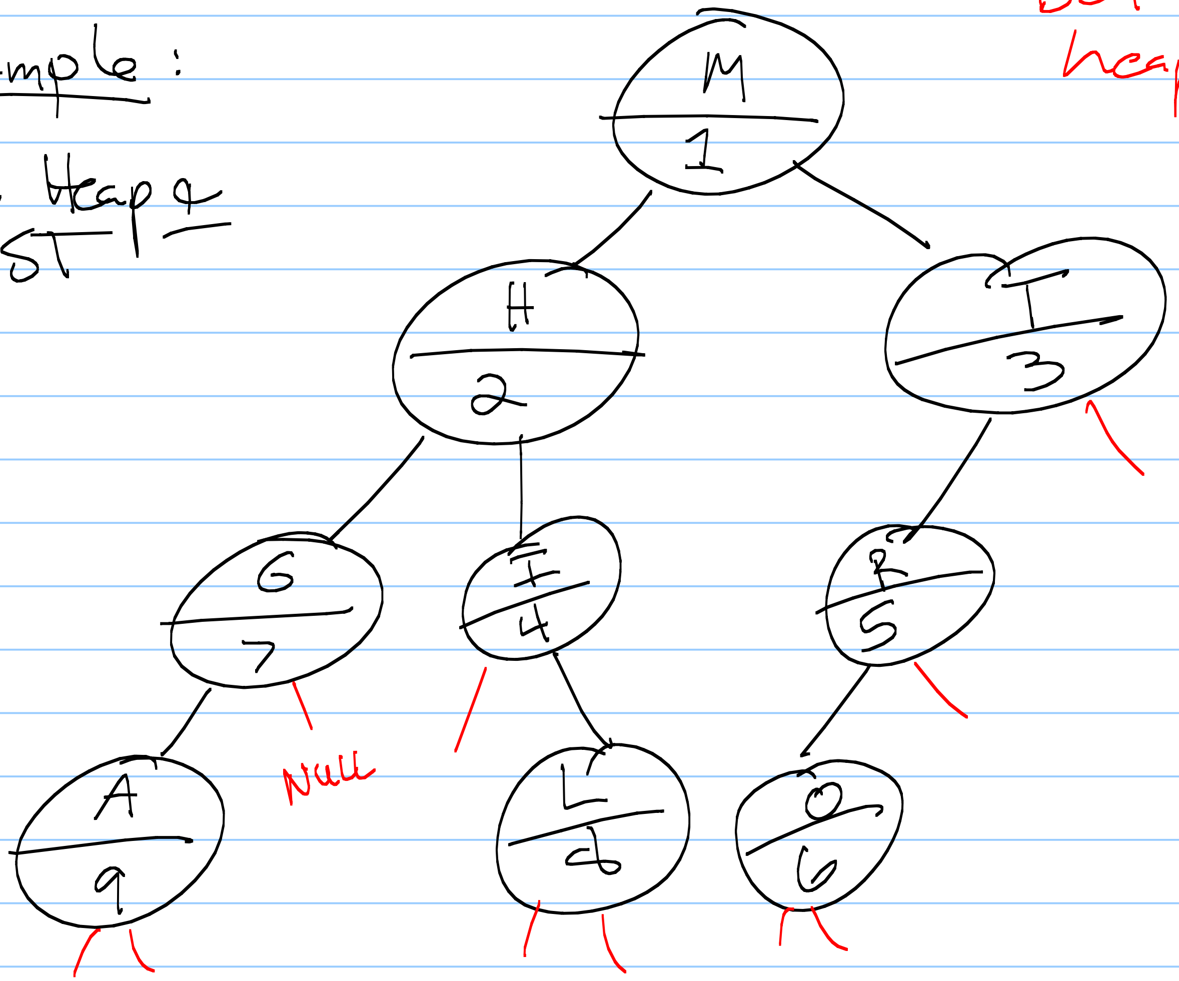
- Nodes will contain both values and "priorities".
- A treap is a BST over the values, & a heap over the priorities.

tree heaps

Example :

Min Heap &
BST +

BST over letters
heap over numbers

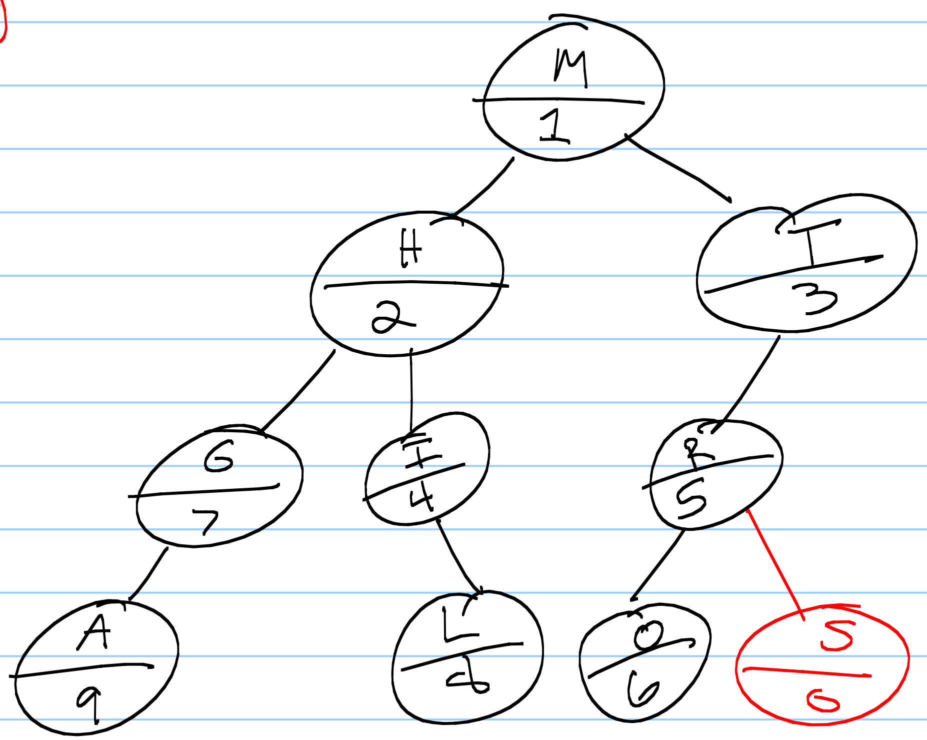


Insert:

insert(S, 0) priority

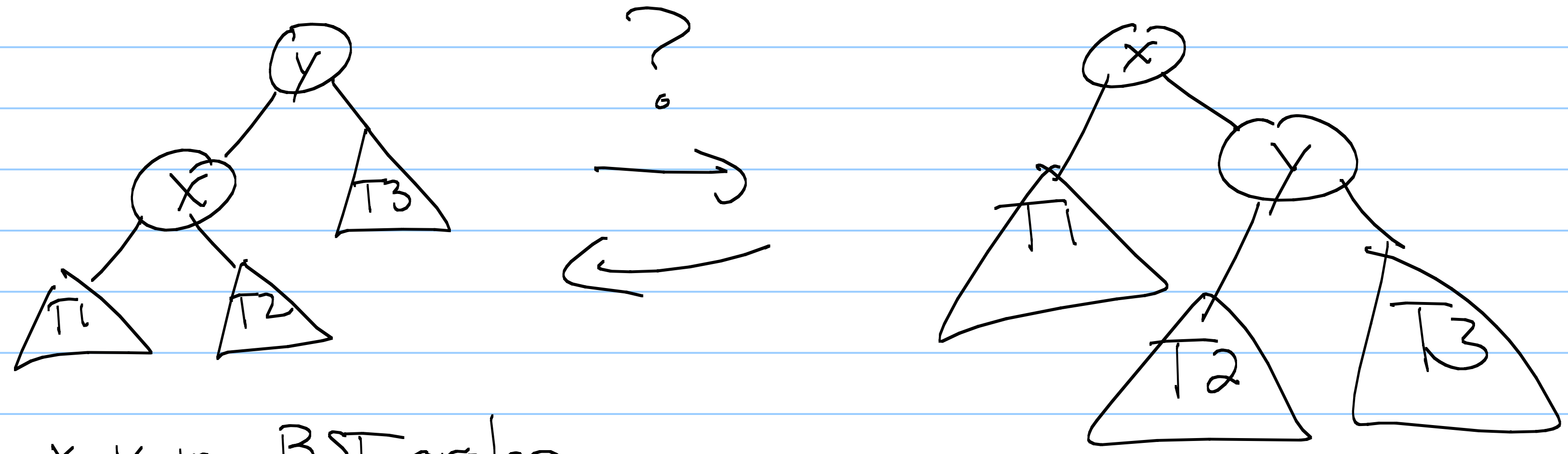
satisfy BST

violate heap

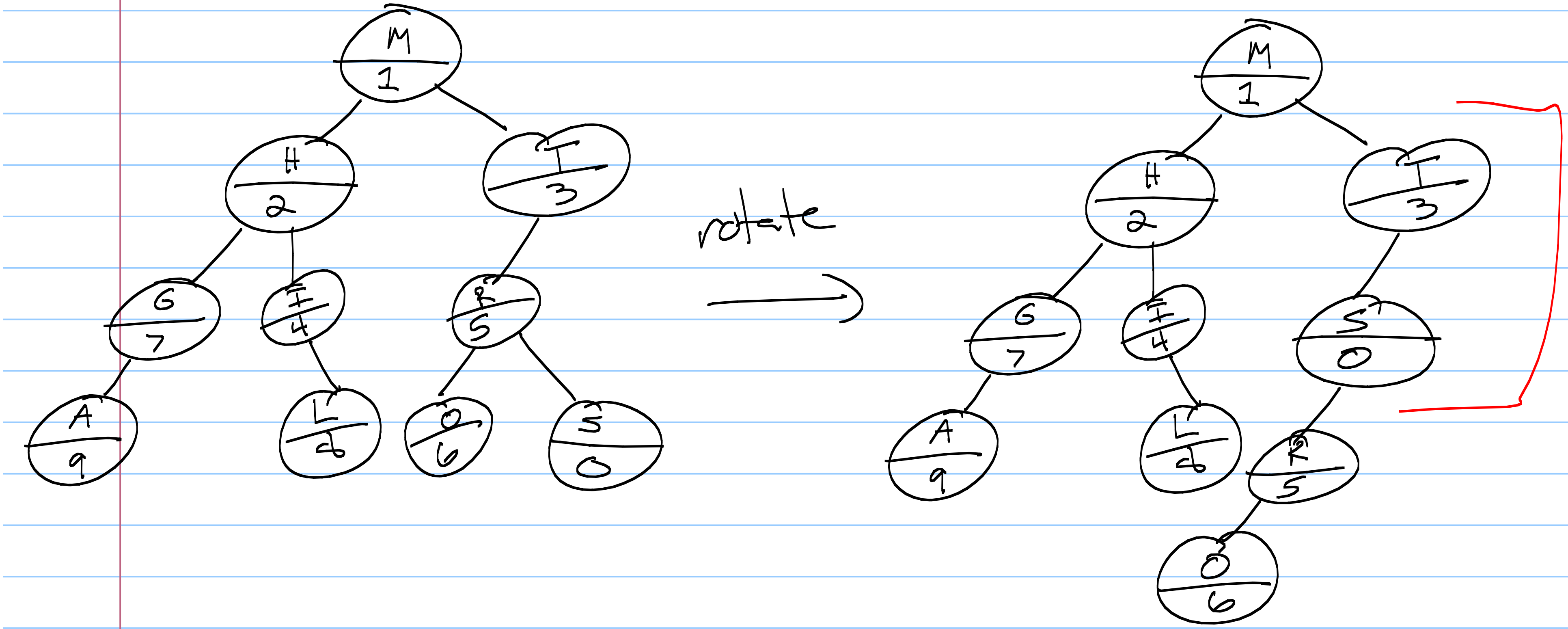


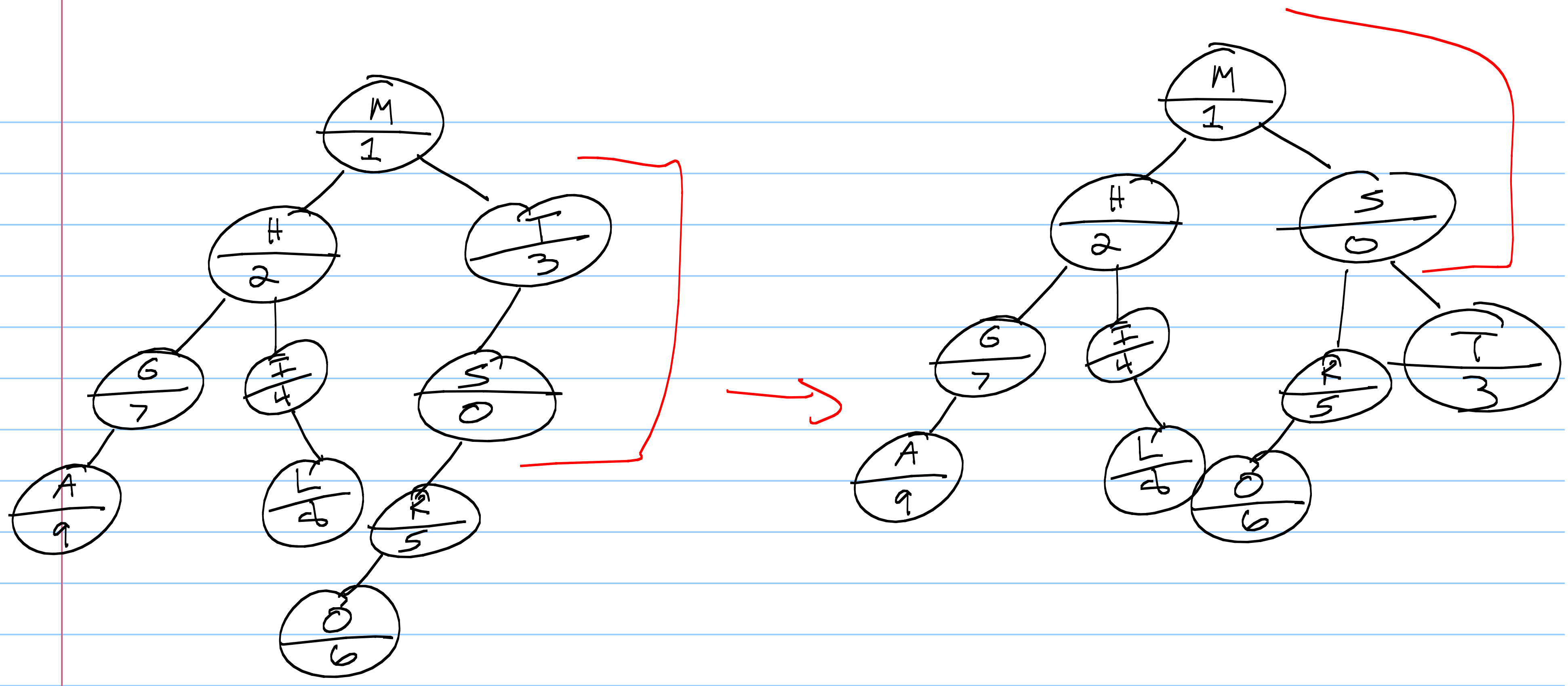
Can't just swap values like in heap, since need a BST.

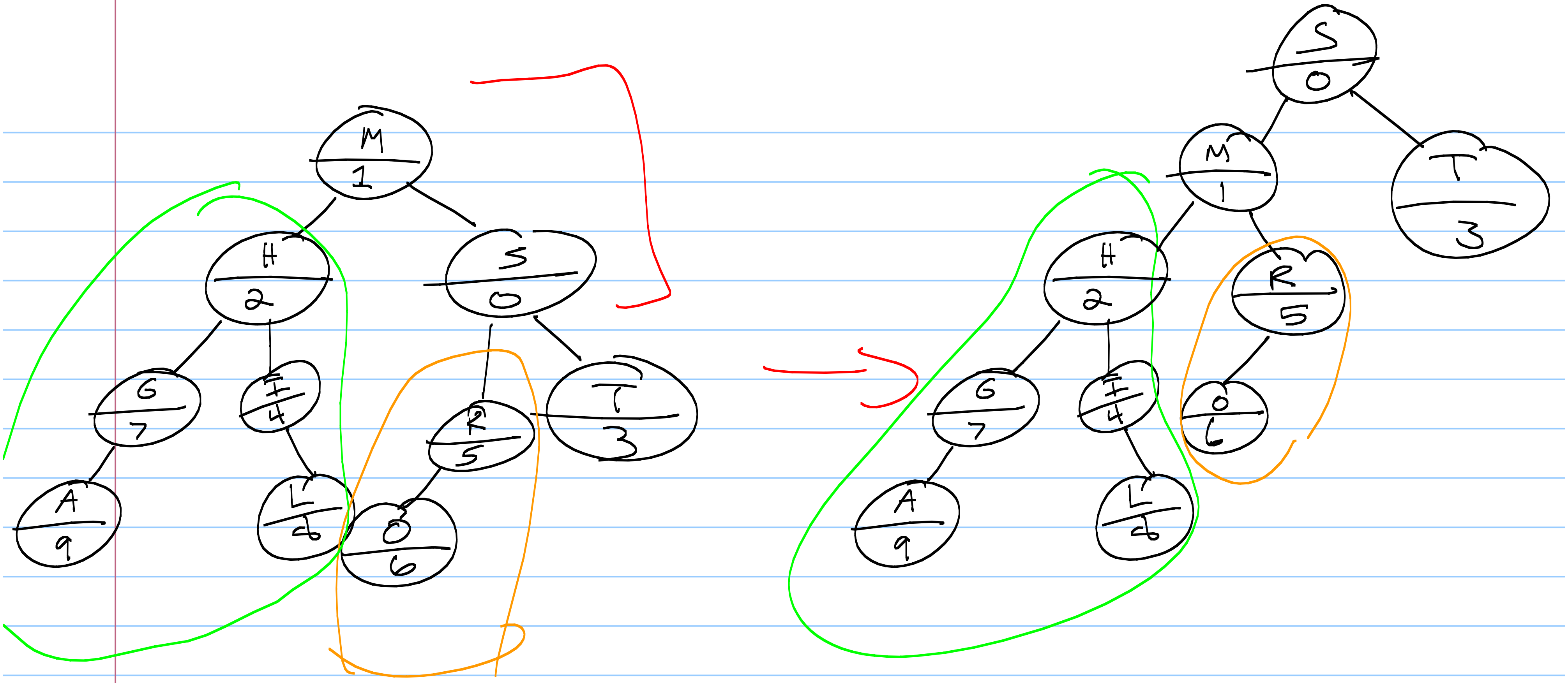
Rotations:



x, y in BST order, but priorities are wrong insert(S, 0)



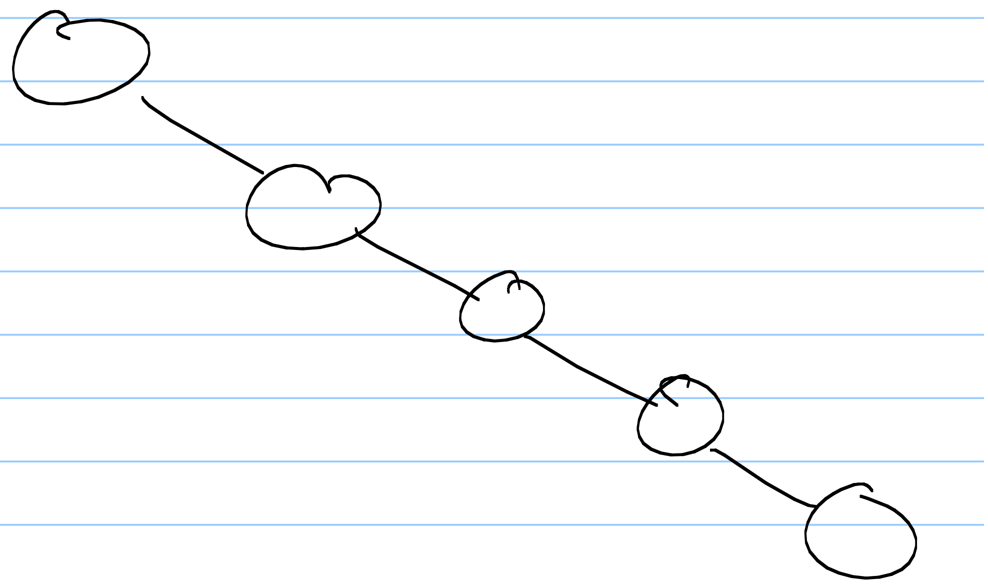

M
1
H
2
T
3
G
7
I
4
S
O
A
9
L
d
R
5
O
6
M
1
H
2
S
O
G
7
I
4
R
5
T
3
A
9
L
d
O
6



M 1
H 2
S 0
G 7
F 4
R 5
T 3
A 9
L 8
O 6
S 0
M 1
T 3
H 2
R 5
G 7
F 4
O 6
A 9
L 8

Downside:

height can be $O(n)$



Randomized treaps

Every element will get a random priority assigned to it.

Expected value: number that is expected in a series of random events

Expected height of random treap:

$O(\log n)$