Recap

- HW due
- Next HW: up
- Reading due Monday

Note: For BST homework:

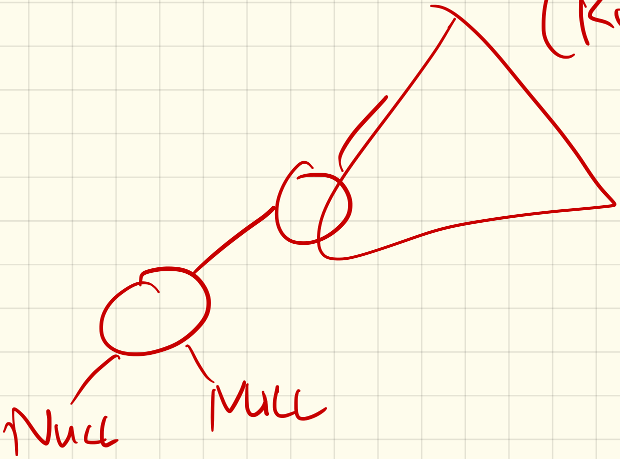
do not use

_recursivelyDelete

use

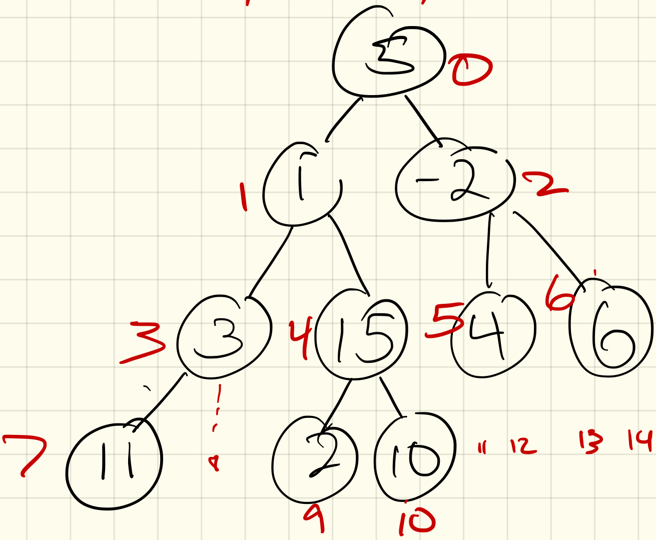
deleteAndPromoteLeftChild

(Right)



Array-based tree storage:

(for any binary tree)



A:

5	1	-2	3	15	4	6	11	x	2	10	x	x	x	x	x
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

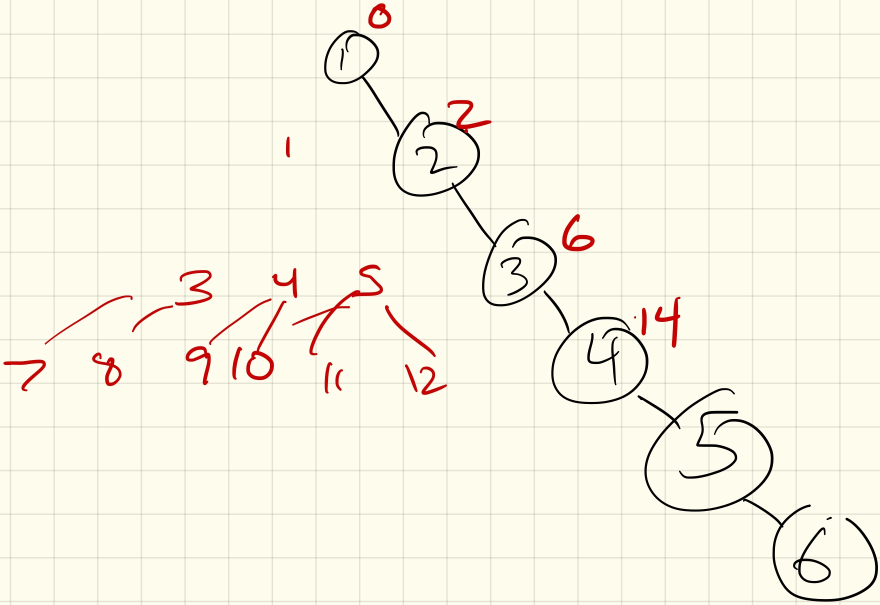
$$\text{left}(v) = 2v+1$$

$$\text{right}(v) = 2v+2$$

$$\text{parent}(v) = \left\lfloor \frac{v-1}{2} \right\rfloor$$

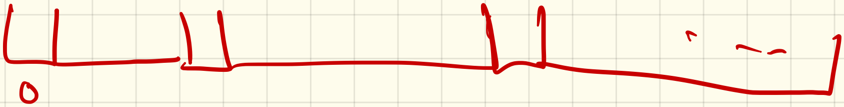
Potential downside of array:

Space!



Array:

A:

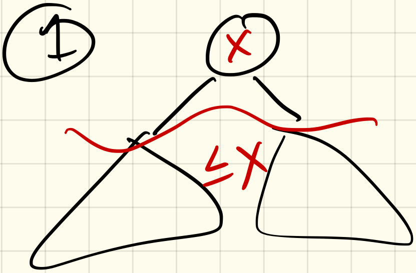


(Max)
Heap: A binary tree where:
(not BST)

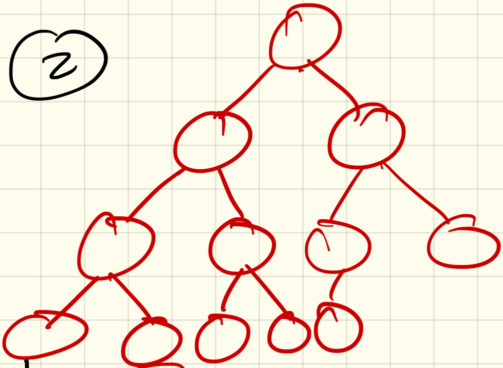
1. For every node v (other than r) the key stored at v is ≤ key stored at v's parent

2. The tree is complete: levels 0... h-1 are full & h is filled in left to right.

1



2

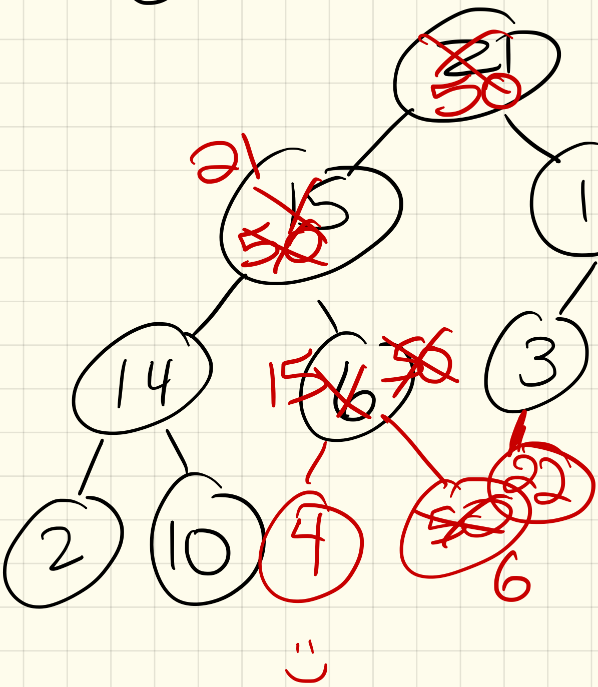


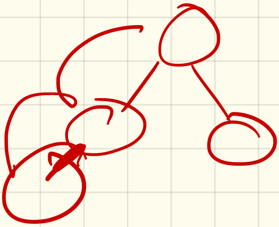
Operations:
- insert
- getMax
- removeMax

} limited, but fast

Bigger example:

NOT a BST

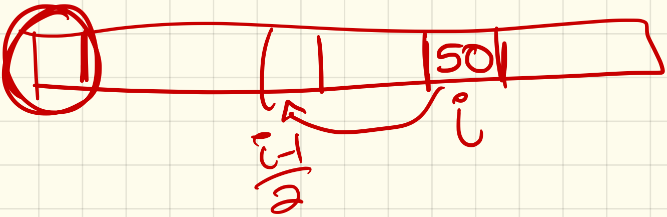




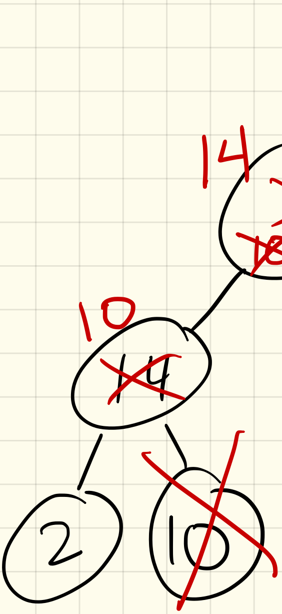
Insert(4)

Insert(50)

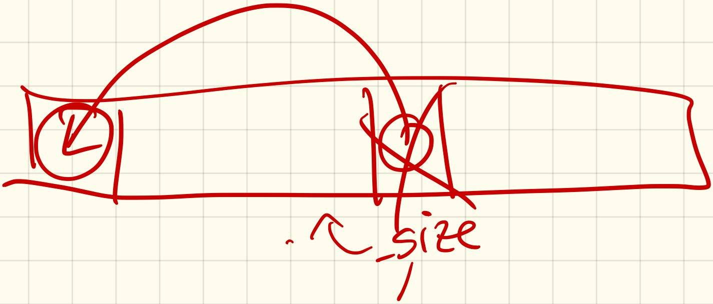
Insert(22)



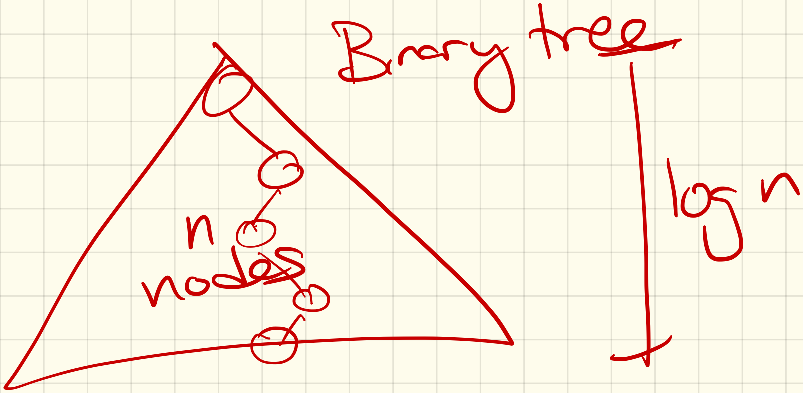
Harder: removeMax



How?



Runtimes



$O(\log n)$

(faster in practice than AVL)

The abstract data type:

priority queue (in reading)

Note: Could implement PQ with lists, too!

(How?)

$O(1)$
$O(n)$

Now: Implementation!
(on web page)