COP 3502 11/10/21

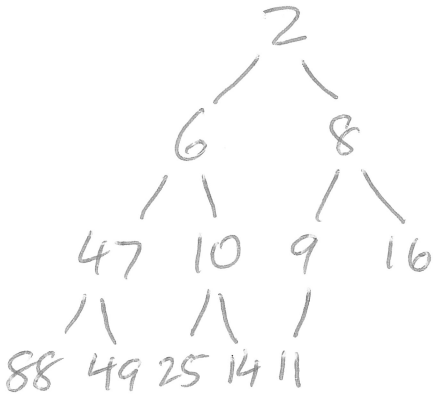
# Binary Heaps

- Insert
- ~~Delete~~
- Remove Minimum

] O(lg n)

Can achieve w/ balanced BST.
But if this all we need, a heap is nice.
(hidden constant is a bit less).

heap - complete binary tree that satisfies the heap property.




heap property - for each node, its children are bigger than it. (min of a subtree is at the root of the subtree.

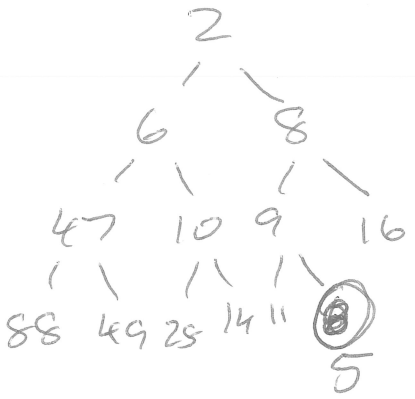
complete bin tree - each "level" filled in. Last level can be incomplete, but must be filled in left to right.

Storage:

| | 2 | 6 | 8 | 47 | 10 | 9 | 16 | 88 | 49 | 25 | 14 | 11 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |

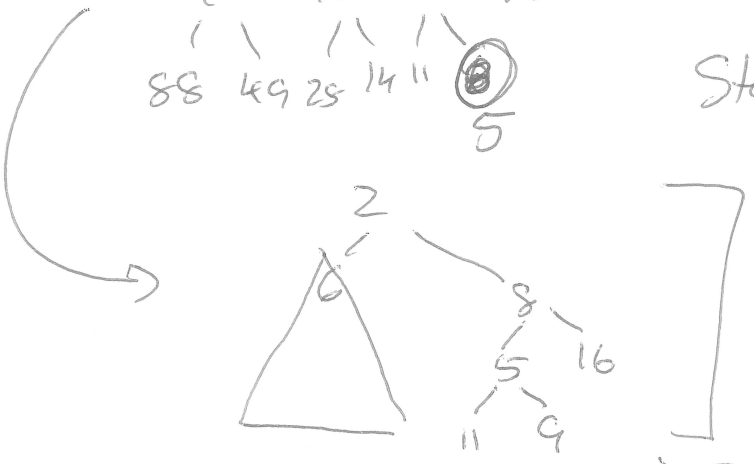
A node in index $i$, its left child is in index $2i$.
its right child is in index $2i+1$.
its parent is in index $i/2$.

## Insert (new hire)

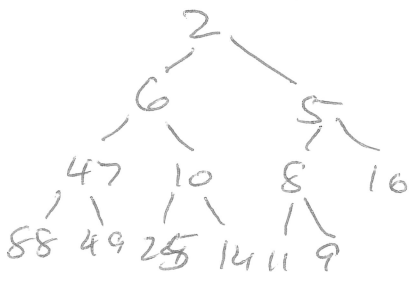



Step 1: Place in next open spot in tree structure.

Step 2: Percolate Up
(keep on challenging your boss until you lose.)



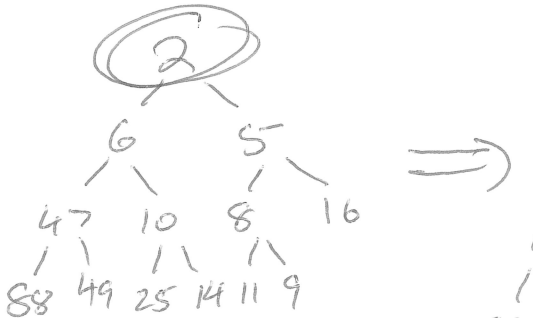

5 beats boss 9 swaps



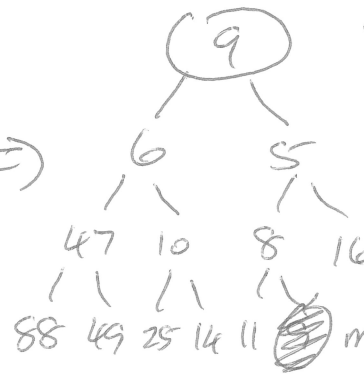

Final position - CEO 2 puts #5 in his place!

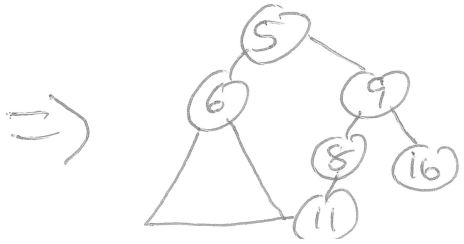
---

## Delete Min




1. Remove root
2. Move last item to root
3. Percolate Down





Heap

height = $O(\lg n)$, $n$ = # nodes

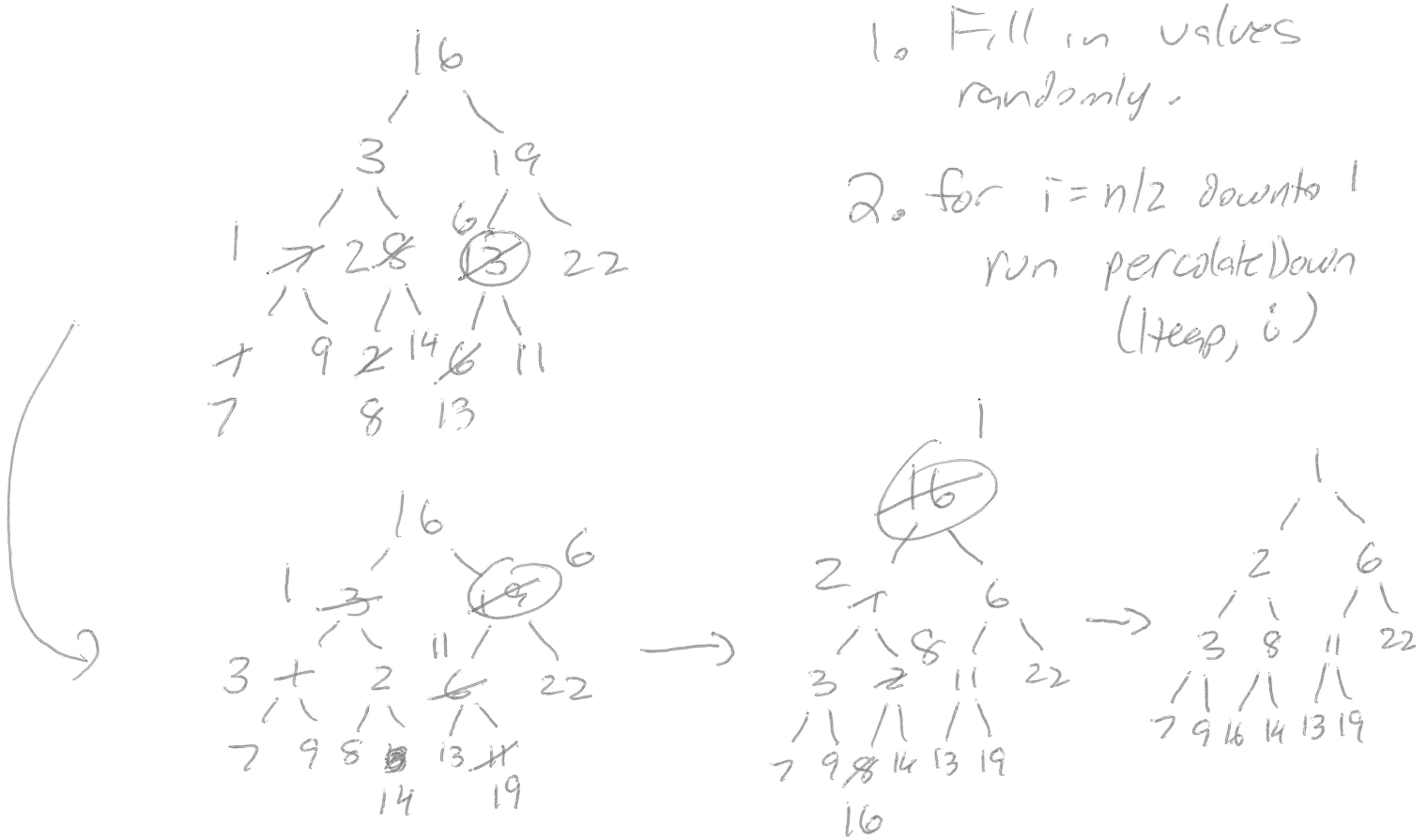
run-time is $O(\lg n)$ for both operations

## Make Heaps

Obvious way - do $n$ insertions takes ~~$O$~~ $O(n \lg n)$ time. [The last $\frac{n}{2}$ insertions in the worst case take at least $(\lg n - 1)$ steps so # steps $\geq \frac{n}{2}(\lg n - 1) = O(n \lg n)$. [$\Omega(n \lg n)$ is accurate notation]]

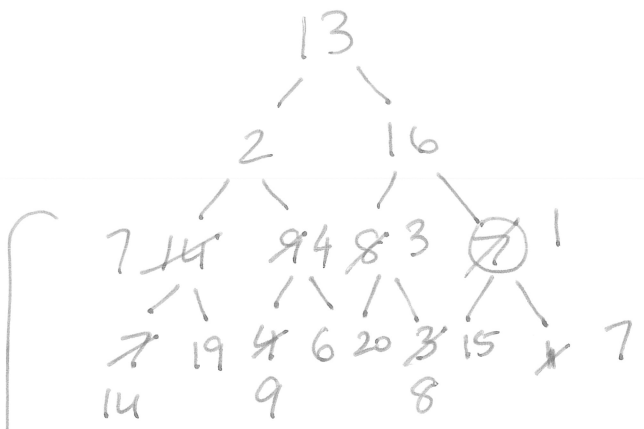
Is there a better way?

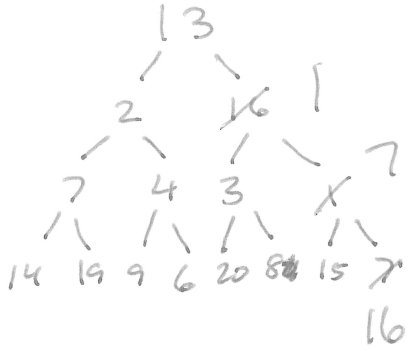
1. Fill in values randomly.
2. for i = n/2 downto 1
   run percolateDown (Heap, i)



# Make Heap Example (Aft. class)

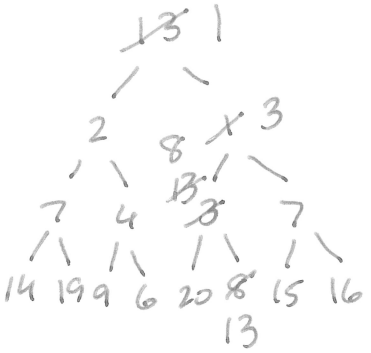
1. Put in randomly
2. for $i = n/2$ downto 1:
   percolateDown(heap, i)



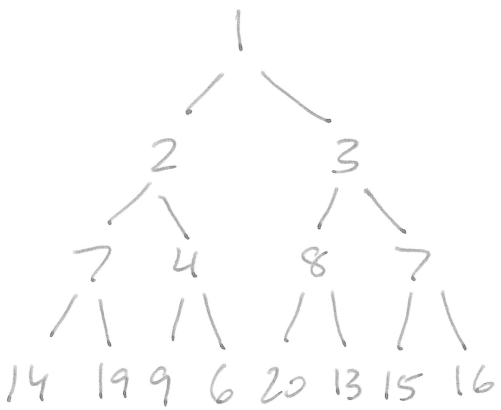

after 1st 4 percolateDowns

these 4 subtrees are valid heaps




after next 2 percolateDowns

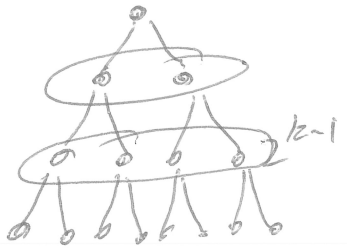
these 2 subtrees are valid heaps




Heap!

# Make Heap Run Time Analysis

height $k$




$n = 1 + 2 + 4 + \cdots + 2^k = 2^{k+1} - 1$

Let $S$ = max # swaps

$$S = 1 \times 2^{k-1} + 2 \times 2^{k-2} + 3 \times 2^{k-3} + 4 \times 2^{k-4} + \cdots k \times 2^0$$

$$-\frac{S}{2} = \qquad 1 \times 2^{k-2} + 2 \times 2^{k-3} + 3 \times 2^{k-4} + (k-1) \times 2^0 + \frac{k}{2}$$

---

$$\frac{S}{2} = 2^{k-1} + 2^{k-2} + 2^{k-3} + \cdots + 2^0 - \frac{k}{2}$$

$$S = 2\left(2^{k-1} + 2^{k-2} + \cdots + 2^0\right) - k$$

$$S = 2\left(2^k - 1\right) - k = 2^{k+1} - 2 - k$$

$$= n - 1 - k$$

$O(n)$

## Heap Sort

1) Run Make Heap $O(n)$
2) for $i = 0$ to $n-1$, array[$i$] = removeMin(heap); $O(n \lg n)$