More on Heaps

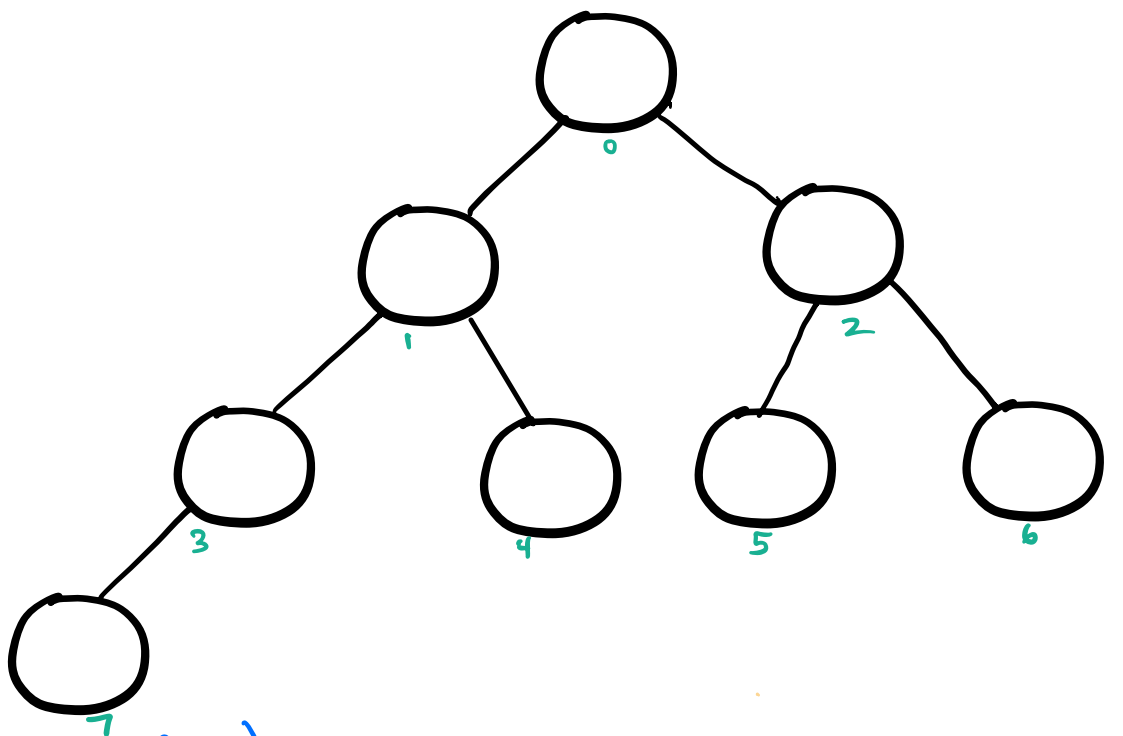
Suppose we are to initialize a new, empty heap and insert {40, 88, 106, 14, 3, 16, 60, 8} into it.

We could start with an empty heap and do 8 inserts.

It is more efficient, however, to simply write those values into our array and

Heapify

88	106	3	14	40	8	60	16



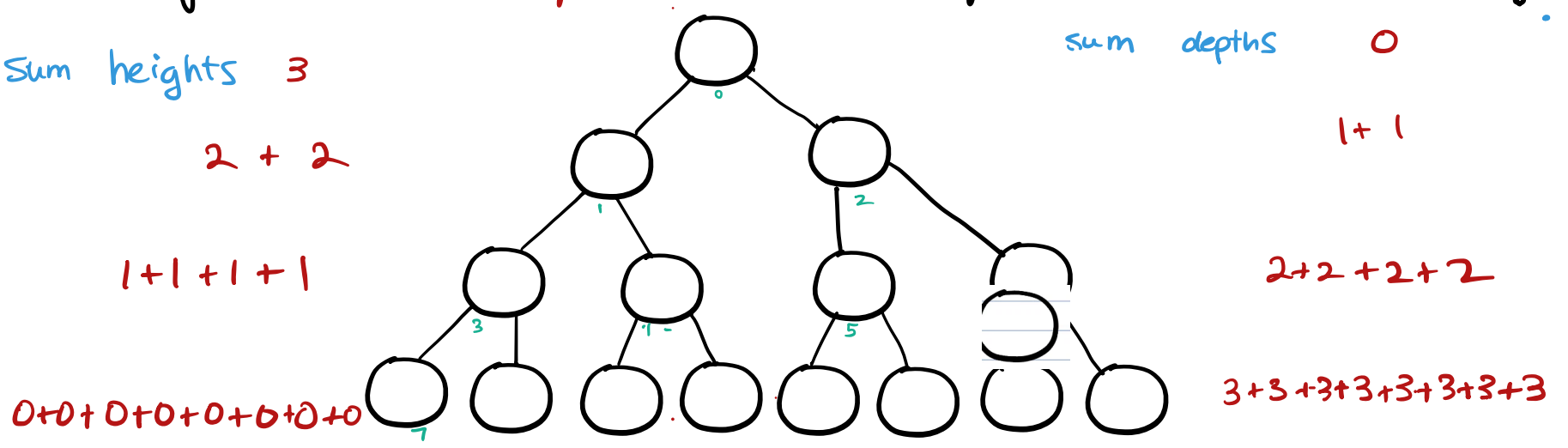
Heapify (1):

- assume the two subtrees rooted at 1 are heapified.

- if value at node 1 is out of heap-order with its children, swap down
  make sure swap with smallest child

Clearly, this is a bottom-up approach.

Why is bottom-up better than top-down for this DS?



Heapify worst-case run time is

$$\frac{n}{2}\cdot 0 + \frac{n}{4}\cdot 1 + \cdots + \frac{n}{2^i}\cdot(i-1) \cdots + \frac{n}{2^{\lg n}}\cdot(\lg n - 1)$$

$$= n\left(\frac{1}{4} + \frac{2}{8} + \cdots + \frac{i-1}{2^i} + \cdots + \frac{\lg n - 1}{n}\right)$$

half the elements have the max depth which is $\lg n$ so heapify would be $O(\lg n)$

$$S = \frac{1}{4} + \frac{2}{8} + \frac{3}{16} + \frac{4}{32} + \frac{5}{64} + \cdots$$

$$2S = \frac{1}{2} + \frac{2}{4} + \frac{3}{8} + \frac{4}{16} + \frac{5}{32} + \cdots$$

$$2S - S = \frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \frac{1}{16} + \frac{1}{32} + \frac{1}{64} + \cdots$$

$$= 1$$

∴ max # swaps if heapify downward, all elements, bottom up: $1 \cdot n = n$.

If we to heapify upwards, what is worst case?

$\frac{n}{2} \cdot \lg n + \frac{n}{4}$

MakeHeap (A, n)

// A an array elements, A[0..n-1]

O(n) for i = [ ] downto 0

Heapify (A, i)

