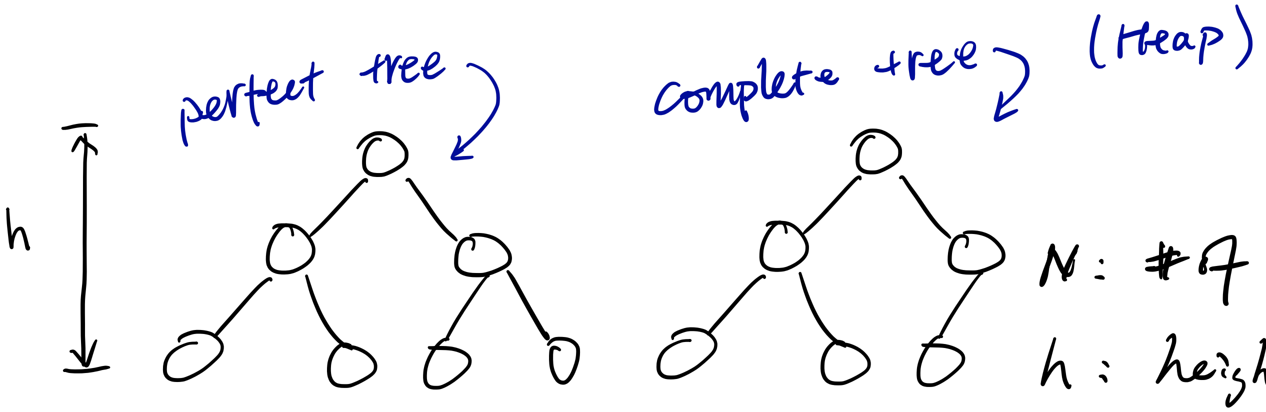
About height: $h = \log N$



N : # of nodes

h : height

- height of a perfect tree :

$$N = 2^{h+1} - 1 \Rightarrow h = \Theta(\log N)$$

- height of a complete tree :

$$\underbrace{2^{(h-1)+1} - 1}_{\text{\# perfect tree of height } (h-1)} \underbrace{+\, 1}_{+\,1} \leq N \leq 2^{h+1} - 1$$

$$\Rightarrow \quad 2^h \leq N \leq 2^{h+1} - 1$$

$h \leq \log N$

$\Rightarrow h = O(\log N)$

$h+1 \geq \log(N+1)$

$\Rightarrow h = \Omega(\log N)$

$$h = \Theta(\log N)$$

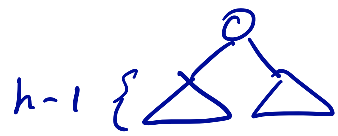
Build Heap : O(N)

Let's prove the asymptotic complexity of build_heap() using heapify_down is:

$$O(\underline{2^{h+1}-h-2}) = O(N)$$

Worst-case : work done by all heapify_down is sum of heights of all subtrees.

thus, we have the following recursion:



Base case: $S(0) = 0$

Recursion: $S(h) = 2 \cdot S(h-1) + h$

数学归纳法

↳ I.H.: $\forall k < h,\ S(k) = \underline{2^{k+1} - k - 2}$

$$\Rightarrow S(h) = 2 \cdot S(h-1) + h$$
$$= 2 \cdot (2^{(h-1)+1} - (h-1) - 2) + h$$
$$= \underline{2^{h+1} - h - 2}$$

∴ the complexity is:

$$O(2^{h+1} - h - 2) = O(N) \quad \blacksquare$$