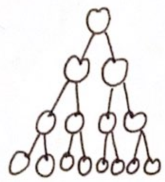
Buid - Max - Heap Laufzeit Analyse $O(n)$, why

1 ~~Ref~~ Pre Knowledge

Number of nodes at height $h \simeq \left\lceil \frac{n}{2^{h+1}} \right\rceil$

Beispiel:



$h = 3$	node $= 1$	$15/2^{3+1} = 15/16 \approx 1$
$h = 2$	node $= 2$	$15/2^{2+1} = 15/8 \approx 2$
$h = 1$	node $= 4$	$15/2^{1+1} = 15/4 \approx 4$
$h = 0$	node $= 8$	$15/2^{0+1} = 15/2 \approx 8$

Beweisen:

Naive intuition : Build-Max-Heap calls Max-Heapify about $n/2$ times. Each Max-Heapify can take up to $O(\log n)$ time (the height of the tree). So ~~people~~ at first glance, the Laufzeit seems to be $O(n/2 \cdot \log n) = O(n \log n)$

BUT IT IS WRONG !

Because

Node near the bottom of the tree have very small height, so their Max-Heapify runs quickly.

Only a few nodes (those near the root) can take $O(\log n)$ time.

e.g

In a binary heap of size n:

About $n/2$ nodes are leaves. height $= 0 \Rightarrow$ no work

About $n/4$ nodes are one level above leaves. height $= 1$

About $n/8$ nodes ~~are~~ have heigh $= 2$

⋮

Only one node (the root) is at height $\log n$.

$$\Rightarrow T(n) = \sum_{h=0}^{\lfloor \log n \rfloor} \frac{n}{2^{h+1}} \cdot O(h)$$

$$= O\left(n \cdot \sum_{h=0}^{\log n} \frac{h}{2^{h+1}}\right)$$

$$= O(2 \cdot n)$$

$$= O(n)$$

* Geometrische Reihe mit $|q| < 1$

$$\sum_{n=0}^{n} q^n = \frac{1}{1-q}$$

$$\sum_{h=0}^{\log n} \frac{h}{2^{h+1}} = \sum_{h=0}^{\log n} \frac{1}{2^h} = 2$$