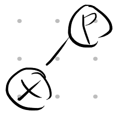
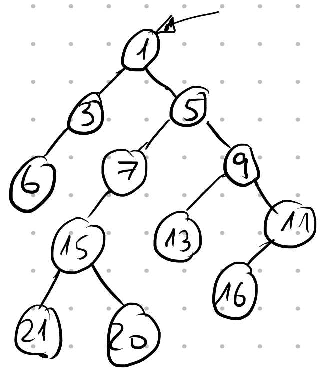
Self-adjusting heap

- binary tree
- efficient meld
  (meldable heap)
- no assumption on tree-structure
- each node stores a key, in heap-order

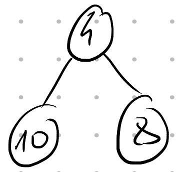




H_1

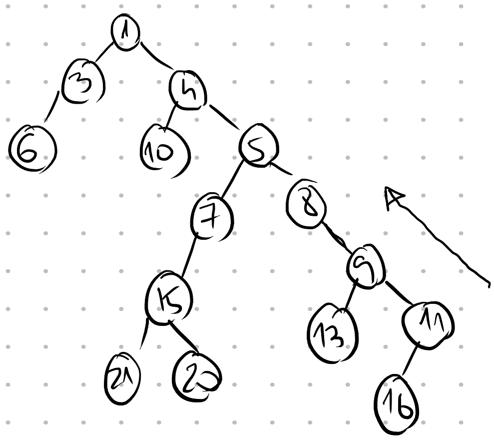
$x \geq p$

- make-heap = create empty tree (heap)
- find-min
- meld (H_1, H_2)

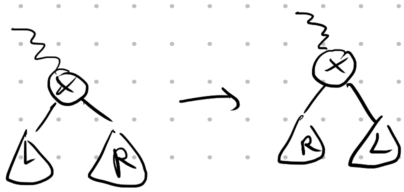


H_2

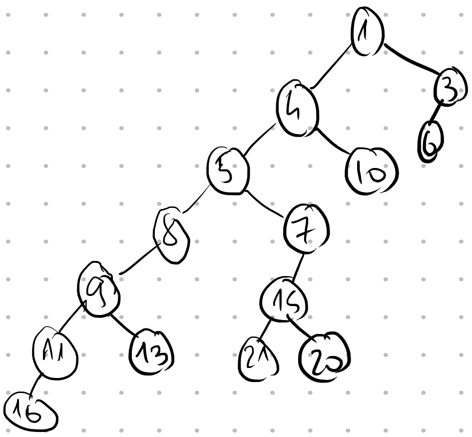
1. merge the right-spines of H_1, H_2
   (like in mergesort)



2. going on new right spine bottom-to-top
   for all nodes x, <u>except bottom-most</u>
   flip left right children

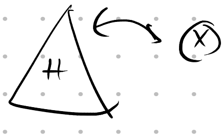


meld(H_1, H_2)

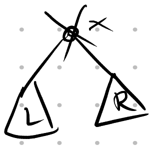


(can also implement
1, 2 together
top-bottom)

- insert(H, x):
  - create a new heap with node x
  - meld x with H

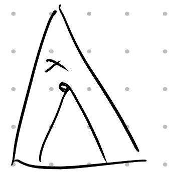


- extract_min (H)



- cut out root
- meld (L, R)

- decreasekey (x, k)

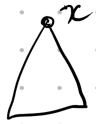


- cut out x with its subtree
- update x.key
- meld x and subtree with rest of heap

Thm. All operations in self-adj. heap take $O(\log n)$ time amortized (enough to analyze merge) (meld)

Proof

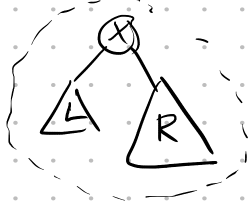
$s(x) = |\text{subtree}(x)|$



\# nodes, incl. x

node x is right-heavy if $s(x.\text{right}) > \frac{s(x)}{2}$

or. light

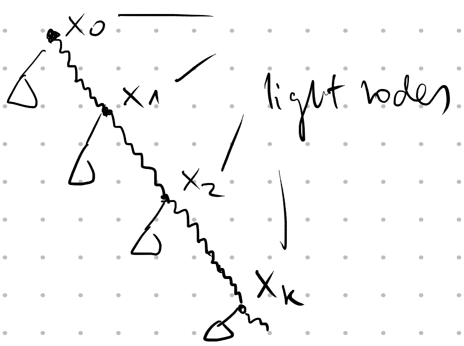


ϕ = # of right-heavy nodes.

$\phi_0 = 0$, $\phi \geq 0$

Obs. On the right-spine of tree there are at most $\log_2 n^{+1}$ light nodes.

Proof



$$n \geq s(x_0) \geq 2 \cdot s(x_1) \geq 2^2 s(x_2) \geq \dots \geq 2^k s(x_k)$$

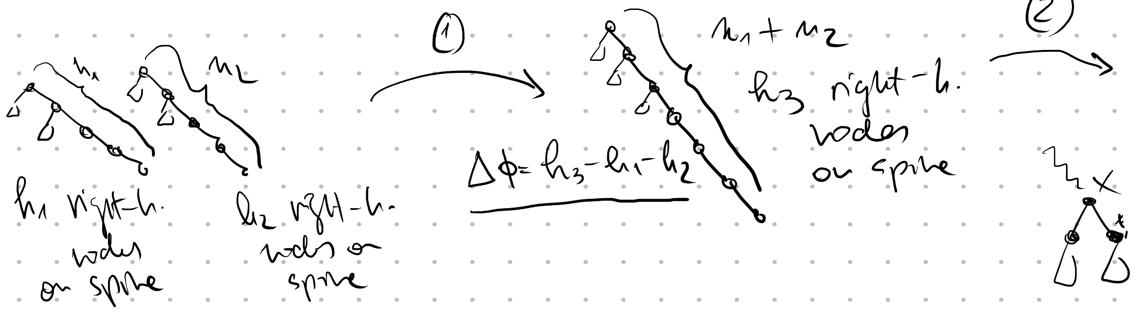
$$\geq 2^k$$

$$k \leq \log_2 n$$

Analysis (amortized cost of meld (H_1, H_2))

(Obs)

$$\Delta\phi \leq -h_3 + \log n$$



$\Delta\phi = h_3 - h_1 - h_2$

x right-heavy, then it becomes light, & vice versa

(due to ①, ②)

$$\Delta\Phi \le h_3 - h_1 - h_2 - h_3 + \log n$$

$h_1 \geq n_1 - \log(n)$

$h_2 \geq n_2 - \log(n)$

$$= \log n - h_1 - h_2$$

$$\le \log n - n_1 - n_2 + 2\log n$$

$$\le -n_1 - n_2 + 3\log n.$$

amortized cost = actual cost + $\Delta\phi$

$$= n_1 + n_2 - n_1 - h_2 + 3\log n$$

$$= 3\log n \in \underline{O(\log n)}.$$