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

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

1. merge the
   right-spines of H_1, H_2
   (like a mergesort)
   
2. grow a new right spine
   bottom-to-top
   for all nodes x, except bottom-most
   flip left-right children

   (can also implement
   @ @ 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)
- decrease $\lambda (x, k)$
  - cut at $x$ with its subtree
  - update $x$-key
  - weld $x$-adj subtree with rest of heap

Then, all operations in self-adj. heap take $O(\log n)$ time amortized (except to analyze merge)

Analysis (amortized cost of weld (HI, TH))

1. $\Delta f = h_3 + h_2$
2. $\Delta f = -h_3 + \log m$

Proof

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

$$\lambda(x) = \lambda(\text{moder}, \text{incl. } x)$$

mode $x$ is right-heavy

$$\phi = \# \text{ of right-heavy nodes}$$

$\phi = 0 \rightarrow \phi > 0$

Obv: On the right spine of tree there are at most $\log_{m+1}$$ right heavy nodes.

Proof

$$n \geq \lambda(x_0) \geq 2 \cdot \lambda(x_1) \geq 2^2 \cdot \lambda(x_2) \geq \cdots \geq 2^k \cdot \lambda(x_k)$$

$$k \leq \log_2 n$$

$\lambda(x)$ is right-heavy, then it becomes right-heavy very soon.
\[ \Delta \phi \leq h_3 - h_1 - h_2 - h_3 + \log n \]
\[ h_1 \geq n_1 - \log n \]
\[ h_2 \geq n_2 - \log n \]

\[ = \log n - h_1 - h_2 \]
\[ \leq \log n - m_1 - m_2 + 2 \log n \]
\[ \leq -m_1 - m_2 + 3 \log n. \]

\text{actual cost} = \text{actual cost} + \Delta \phi

\[ = m_1 + m_2 - m_1 - h_2 + 3 \log n \]
\[ = 3 \log n \in O(\log n). \]