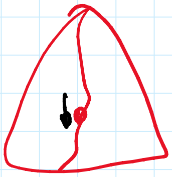


insert: $O(\log n)$ (up a path)
delete-min: $O(\log n)$ (down a path)
increase-key: $O(\log n)$ (up a path)
decrease-key: $O(\log n)$ (down a path)

preprocess: $O(n)$
(build-heap)

increase-key/find-min

Lower bd: n inserts/delete-mins require $\Omega(n \log n)$ time in comparison model by reduction from sorting.

Q: faster insert? decrease-key?

binomial heaps (Vuillemin '78):
- find-min $O(1)$
- insert $O(1)$ amortized
- delete-min $O(\log n)$

Fibonacci heaps (Fredman-Tarjan '85):
- decrease-key $O(1)$ amortized.

[appl: Dijkstra's algm
n delete-mins
m decrease-keys
$\Rightarrow O(n \log n + m)$]

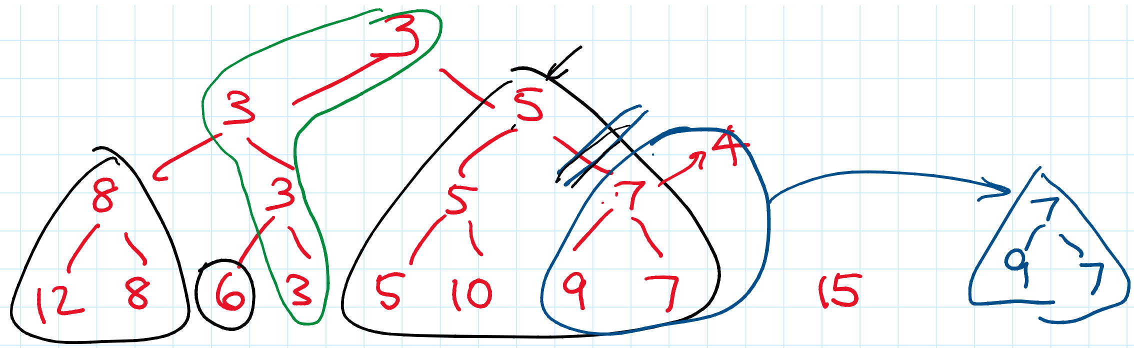
other alternatives:

$O(1)$ amort. decrease-key

- Takaoka '03: "2-3 heaps"
- C '09: "quake heaps"
- Hansen-Kaplan-Tarjan-Zwick '15: "hollow heaps"
- Brodal '96: worst case
- pairing heaps: simplified Fibonacci heaps but poorer guarantees

⋮

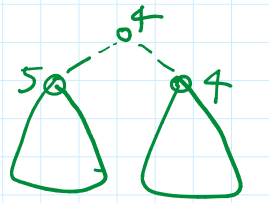
Tournament Tree Approach:



allow multiple tournament trees (forest)

insert(x): // be lazy!

just create a new tree for $\{x\}$



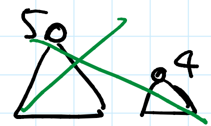
delete-min():

x = min of all the roots

remove path of x's nodes

whenever $\exists$ 2 trees of same height

link them



at end, $\leq \log n$ trees

Amortized Analysis:

define potential Φ = # trees

for each insert:

change in $\Phi = +1$

runtime $= O(1)$

$\leq O(2 - (\text{change in } \Phi))$

for each delete-min:

$t \leq \log n - (\text{change in } \Phi)$

let t = # trees before

change in $\Phi \leq (\log n) - t$

runtime $= O(t + \log n)$

$= O(2\log n - (\text{change in } \Phi))$

total time for n_I inserts & n_D delete-mins total time for n_I inserts & n_D delete-mins

$$\le O\Big(2n_I + (2\log n)\,n_D - \underbrace{\text{total change in } \Phi}_{(\underbrace{\text{final } \Phi}_{\ge 0} - \underbrace{\text{initial } \Phi}_{0})}\Big)$$

$$\boxed{\sum_{i=0}^{N-1} (\Phi_{i+1} - \Phi_i) = \Phi_N - \Phi_0 \quad \text{telescoping sum}}$$

$$\le O(n_I + (\log n)\,n_D)$$

$\Rightarrow$ $\left.\begin{array}{ll}\text{insert} & O(1) \\ \text{delete-min} & O(\log n)\end{array}\right\}$ amortized

---

What about decrease-key(x)?

basic idea - cut subtree at x

x is now a root

can decrease key trivially!

but tree may have deg-1 nodes

& unbalanced.

Solution 1 - 2-3 Heaps (Takaoka)

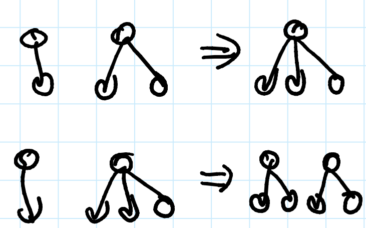
invariant - all nodes have deg 2 or 3

all leaves in same tree have same depth

whenever $\exists$ deg-1 node v

fuse v with its sibling

& split if deg is 4



Amortized Analysis:

new potential Φ = # trees + # nodes

insert : $O(3 - \Delta\Phi)$

delete-min : $O(2\log n - \Delta\Phi)$

decrease-key:

let f = # fuses

$$\Delta\Phi = \underbrace{-(f-1)}_{\Delta(\#\text{ nodes})} + \underbrace{1}_{\Delta(\#\text{ trees})}$$

$$\text{time} = O(1+f)$$
$$\leq O(3 - \Delta\Phi)$$

total time for n_I insert, n_D delete-min, n_{DK} decrease-key

$$\leq O(3n_I + (2\log n)n_D + 3n_{DK} - \underbrace{\text{total } \Delta\Phi}_{(\Phi_{\text{final}} - \Phi_{\text{init}})})$$

$\Phi_{\text{final}} \geq 0$, $\Phi_{\text{init}} = 0$

$$\leq O(n_I + (\log n)n_D + n_{DK})$$

$$\left.\begin{array}{ll} \text{insert} & O(1) \\ \text{delete-min} & O(\log n) \\ \text{decrease-key} & O(1) \end{array}\right\} \text{amort.}$$

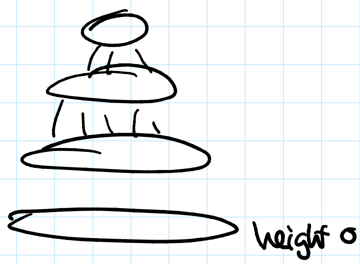
Solution 2 - Fibonacci heaps ⇒ cascading cuts

skipped

Solution 3 - Quake heaps

idea - be lazy + clean up once in a while invariant: - all nodes have deg 1 or 2

- all leaves in same tree have same depth



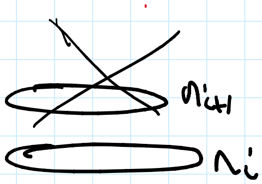
- $n_{i+1} \leq \alpha n_i$ ← global balance constraint

$\left(\frac{1}{2} < \alpha < 1\right)$

nodes at height i+1 — # nodes at height i.

$\Rightarrow$ height $O\left(\log_{\frac{1}{\alpha}} n\right)$.

no fuse etc.



whenever $n_{i+1} > \alpha n_i$ ← seismic event

just remove all nodes at height $> i$

clean-up

$n_i \geq 2 n_{i+1} - \left(\begin{array}{c}\text{\# deg-1} \\ \text{nodes at} \\ \text{height } i+1\end{array}\right)$

Amortized Analysis:

new potential Φ = #trees + #nodes + $\frac{1}{2\alpha-1}$(#deg-1 nodes)

deg-1 nodes at height i+1 $\geq 2n_{i+1} - n_i$

insert: $O(3 - \Delta\Phi)$

delete-min: $O(\,O(\log n) - \Delta\Phi\,)$
ignoring clean-up

~~decrease-key~~ clean-up: say $n_{i+1} > \alpha n_i$.

$\Delta(\text{\#trees}) \leq +n_i$

$\Delta(\text{\#nodes}) \leq -n_{i+1} - n_{i+2} - \cdots$

$\Delta(\text{\#deg-1 nodes}) \leq +1 - (2n_{i+1} - n_i)$

$\leq +1 - (2\alpha - 1)\, n_i$.

$\Rightarrow \Delta\Phi \leq +1 - n_{i+1} - n_{i+2} - \cdots$

runtime $O(1 + n_{i+1} + n_{i+2} + \cdots)$

$$\leq O(2 - \underline{\underline{\Delta\Phi}})$$

$$\Rightarrow \left.\begin{array}{ll} \text{decrease-key} & O(1) \\ \text{delete-min} & O(\log n) \\ \text{insert} & O(1) \end{array}\right\} \text{amortized.} \qquad \square$$