★ Fibonacci Heap

→ extract max, delete takes a lot of time

getmax : O(1)

meld : O(1)

insert : O(1)

extract max : O(log n)

increase key : O(1)

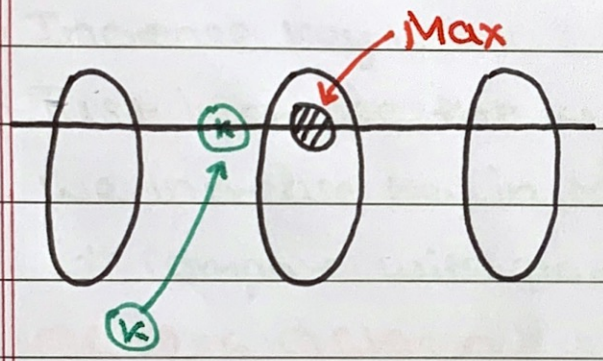
Delete key : O(log n)

Build heap

★ All are amortized times

★ The pointer (head) points to the max

rootlist stored in doubly circular linked list



∴ getmax : O(1)

Insert : O(1) (compare k with max & adjust pointers)

★ $\phi(H) = t(H)$: no. of trees in the heap

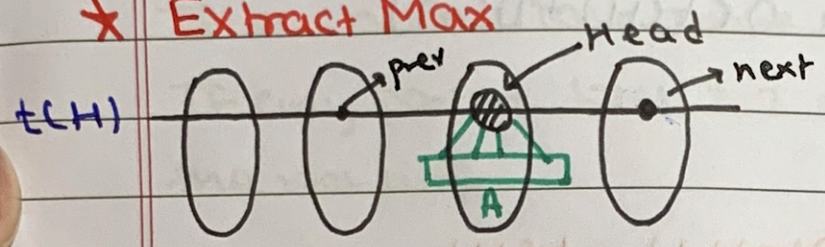
$\phi_0(0) = 0$ (# empty heap)

Insert : $\Delta\phi = t(H_{new}) - t(H_{old}) = O(1)$

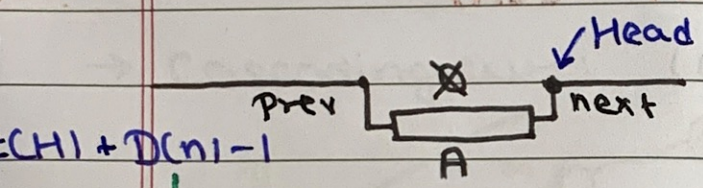
∴ Insert : $O(1)$

GetMax : $O(1)$

★ Extract Max



★ Remove the head & maintain the children of head in a circular doubly linked list



★ Then merge all siblings with root list in $O(1)$

$t(H) + D(n) - 1$

↓

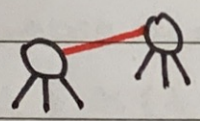
max Degree of a tree in a heap

★ Consolidate Fibonacci Heap

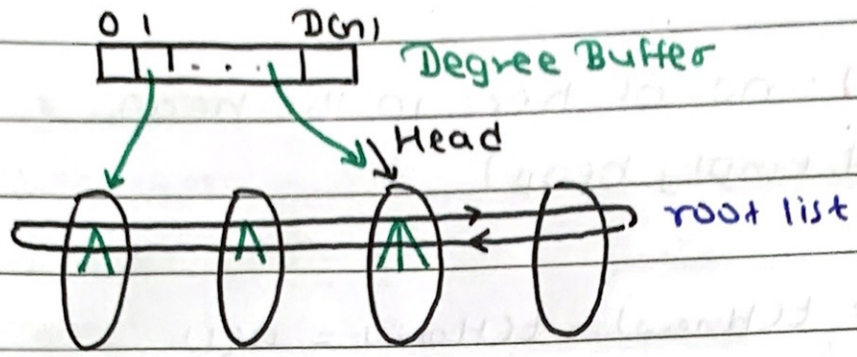
To have atmost $D(n)+1$ trees in the heap

PHP $0, 1, 2, \ldots, D(n)$: Degrees

→ If 2 trees have the same degrees



check root values of the two trees



Each ele. of buffer points to atmost 1 tree

If clash in buffer merge the two trees

So Consolidate takes $O(t(H) + D(n))$

$\therefore$ Extract max:

$O(t(H) + D(n)) + (D(n)+1) - t(H)$

$= O(t(H)) - t(H) + O(D(n))$

Let $O(t(H)) \leq c\,t(H)$

Let $\phi(H) = \alpha\, t(H)$ where $(\alpha > c)$

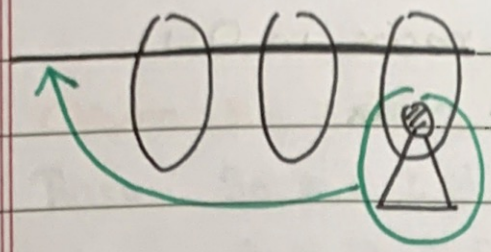
$\therefore$ Extract max:

$O(t(H) + D(n)) - \alpha t(H) + O(D(n)).\alpha$

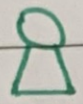
$= c\,t(H) - \alpha\, t(H) + O(D(n))$

$= \underline{\underline{O(D(n))}} \rightarrow$ extract max

★ Increase key:

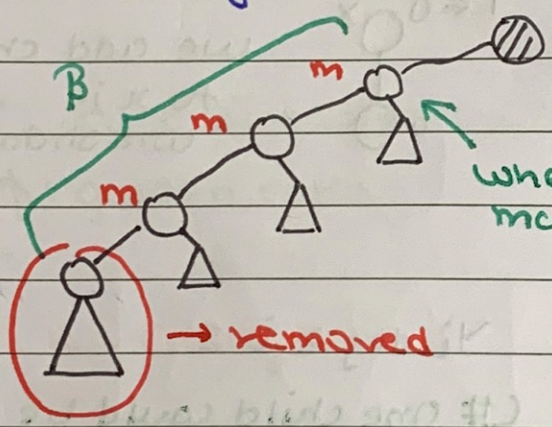


★ Donot use check parent & swap # worst case $O(h) = O(n)$

add  to the root list

★ If a child is lost mark the parent (by m)
If any parent loses ≥ 2 children move the parent to the root list.

→ Cascading cuts:



★ Now consider $\phi(H) = \alpha(t(H) + 2m(H))$ → why no 1? Left as ex to reader.
↓ no. of marked nodes

$m(H) \geq 0 \therefore \phi(H_0) = 0$

★ Increase key:

node incr not marked

$$O(\beta) + \alpha(t(H) + \beta + 2\times(m(H) - (\beta - 1) + 1)) - \alpha(t(H) + 2m(H))$$

$$= O(\beta) + 4\alpha - \alpha\beta = c'\beta - \alpha\beta + 4\alpha$$

Set $\alpha = \max(c', c)$ ∴ Insert : $O(1)$

★ **Meld:**

Join 2 linked list & update max in $O(1)$

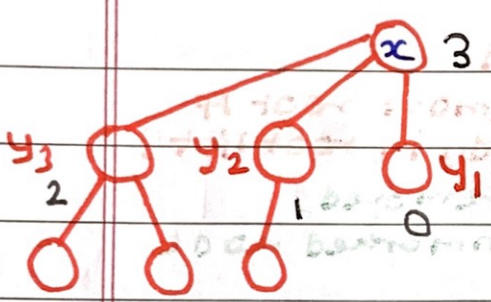
→ Consolidate done in Extract max to save the cost of meld.

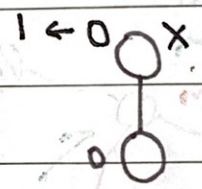
★ **Delete:**

Increase the key to ∞ & then extract max

$O(D(n))$

★ If $x.degree = k$ then $size(x) = S_k$





we add child to x in consolidate.

$y_i.deg \geq i-2$

(# One child could be lost # marked)

$$\therefore size(x) \geq \underbrace{2}_{y_1 \text{ \& root}} + \sum_{i=2}^{d} S_{y_i.degree}$$

$\because S_k$ is monotonically ↑ w.r.t k

$\therefore S_{y_i.deg} \geq S_{i-2}$

$\therefore \text{size}(x) \geq 2 + \sum_{i=2}^{d} S_{i-2}$ (# d is deg(x))

Claim: $S_k \geq f_{k+2}$

Base $S_0 = 1 \geq f_2 = 1$

$S_1 = 2 \geq f_2 = 2$

TPT $S_k \geq f_{k+2}$

$S_k \geq 2 + \sum_{i=2}^{k} S_{i-2}$

$\Rightarrow S_k \geq 2 + \sum_{i=2}^{k} f_i$

$\geq 1 + \sum_{i=0}^{k} f_i = f_{k+2}$ (# Property of Fibo)

$\therefore n \geq \text{size}(x) = S_k \geq f_{k+2} \geq \phi^k$

ϕ: golden ratio

$\therefore k = O(\log n)$