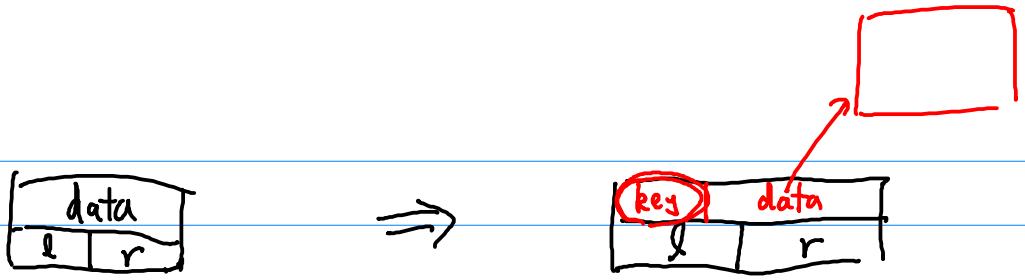
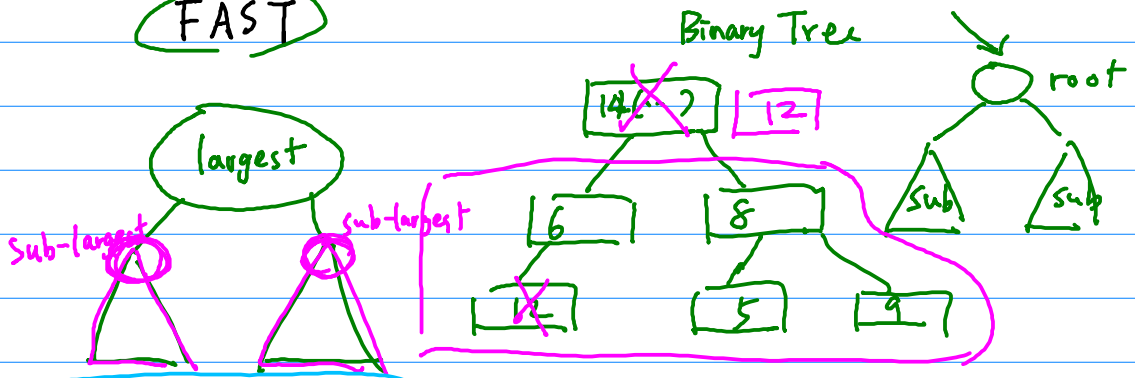


* task: find the node w/ largest

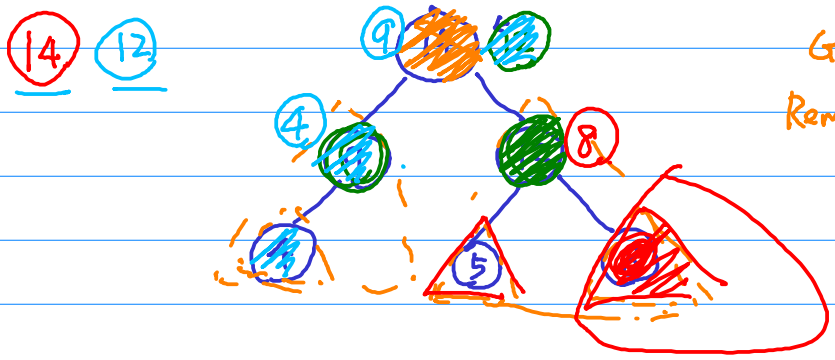
node (key to mean priority
data to mean todo

priority queue

FAST



* binary max-tree
① root key ≥ other node's key)
② every sub-tree is bin-max tree ←



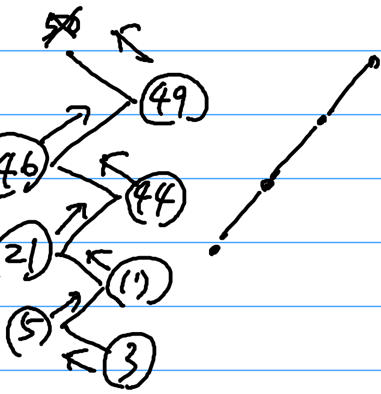
GetLargest ? root

Remove Largest ?
replace w/
larger of sub-root
recursively

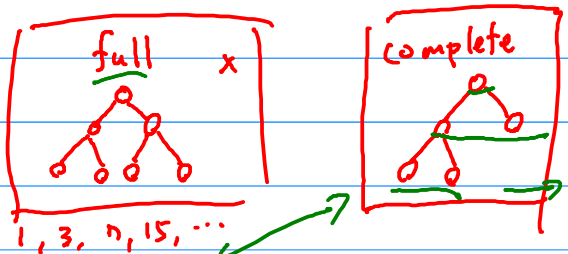
* worst time complexity : $O(h)$ (smaller than $O(n)$)

$\Downarrow$

$O(n)$

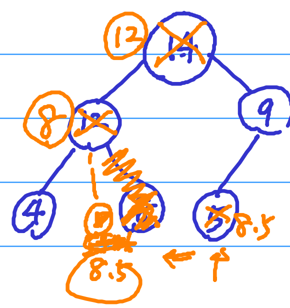


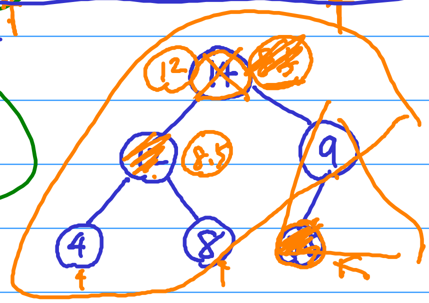
$h = O(\log n)$



* complete binary max-tree

max-heap





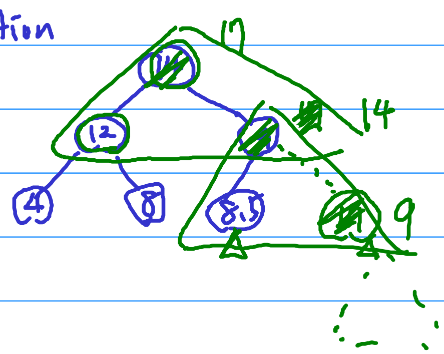
Remove Largest

Remove Largest

① last → root (complete)

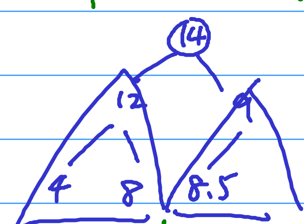
② sink down (max) ? $O(h)$ $O(\log n)$ ← complete

* insertion



① insert at last

② float up ($O(h)$)



14	12	9	4	8	8.5

* complete bin. tree ⟷ array

heap ⟷ partially ordered array

ordered array

heap array

sel. sort on (unordered) array [array] $O(n \cdot O(n)) = O(n^2)$

heap sort $O(n \lg n)$ (n insertion $O(h)$ heap "4 ҟ" [heap] → $O(n \; O(1) + O(\log n)) = O(n \lg n)$

array

		insert	removeLargest
PQ	heap	O(log n)	O(log n)
	max-tree	O(h)	O(h)
unordered	linked-list	O(1)	O(n)
ordered	linked-list	O(n)	O(1)

max-heap

min-heap