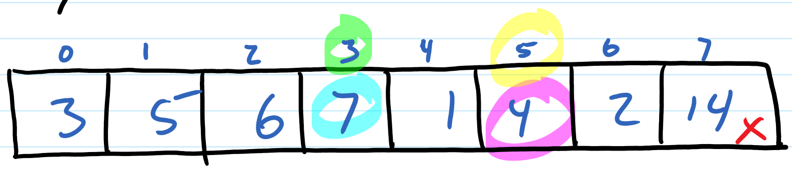
Priority Queue

	unsorted array	sorted array	balanced BST	heap
enqueue (elt, pri) ↳ adds elt w/ priority	$O(1)$	$O(n)$	$O(\log n)$	$O(\log n)$
dequeue ( ) ↳ removes elt w/ highest priority (max queue) (or lowest) (min queue)	$O(n)$	$O(1)$	$O(\log n)$	$O(\log n)$
change-priority (elt, pri) change priority of existing item	$O(1)$	$O(n)$	$O(\log n)$	$O(\log n)$

unsorted array



elts are indices 0, ..., n-1

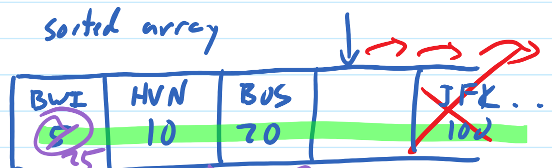
enqueue (3, 7)

enqueue (BWI, 4)

key BWI

value 5

sorted array



enqueue (ORH, 1)

Priority Queue Sort

$O(1)\cdot n$ 1) add each element to priority queue — priority = key to sort on — balanced BST or heap $O(\log n)\cdot n$

$O(n)\cdot n$ 2) while p.q. not empty — $O(\log n)\cdot n$

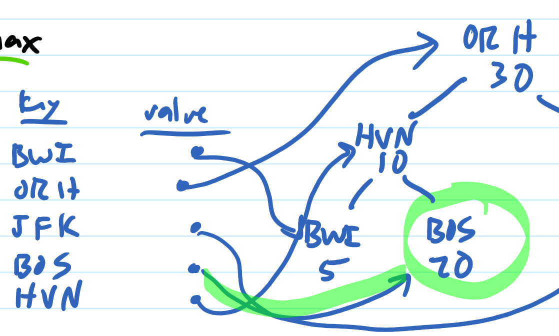
remove min/max

$O(n^2)$ — $O(n \log n)$

↑

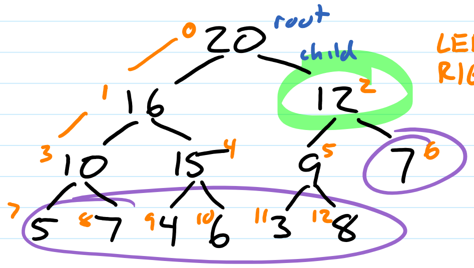
when queue is unsorted array ||| selection sort

Tree Sort or Heap Sort



change-priority (BOS, 50)

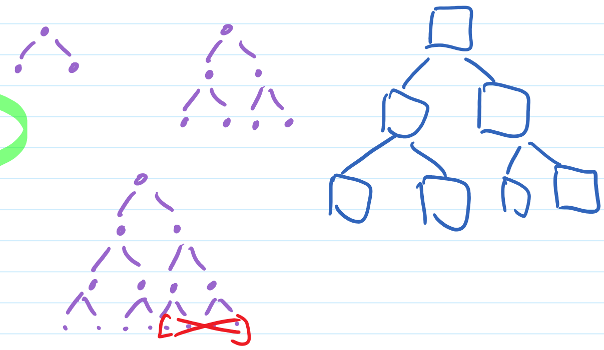
Heap



$LEFT(n) = 2n+1$

$RIGHT(n) = 2n+2$

$PARENT(n) = \left\lfloor \frac{n-1}{2} \right\rfloor$

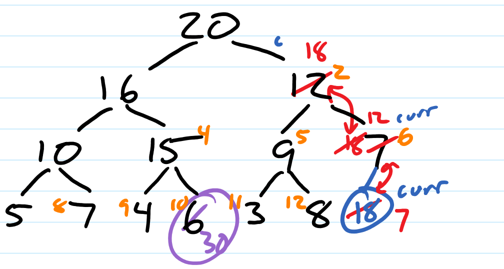


0	1	2	3	4	5	6	7	8	9	10	11	12
20	16	12	10	15	9	7	5	7	4	6	3	8

binary tree s.t.

shape tree is nearly complete
↳ leaves as far down/left as possible – leaves on last 1 or 2 levels and as far left as possible on bottom level

order value in node is ≥ values in children max-heap
≤ min-heap



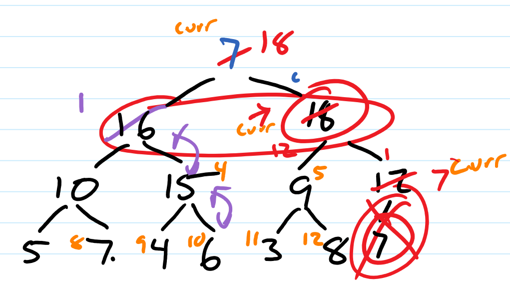
enqueue(18,18)

enqueue: add at next available spot

$O(\log n)$ → while curr not at root and curr > parent
$O(1)$ [swap (curr, parent)
move curr up one level

$O(\log n)$

heap-reheap-up



dequeue

dequeue: get max from root $O(1)$

move last leaf to root $O(1)$

$O(\log n)$ while curr not leaf and curr < one child
$O(1)$ [swap w/ largest child
move curr down to where swap was

$O(\log n)$ total heap-reheap-down

Heapsort

1) build heap from array

2) extract max repeatedly from array