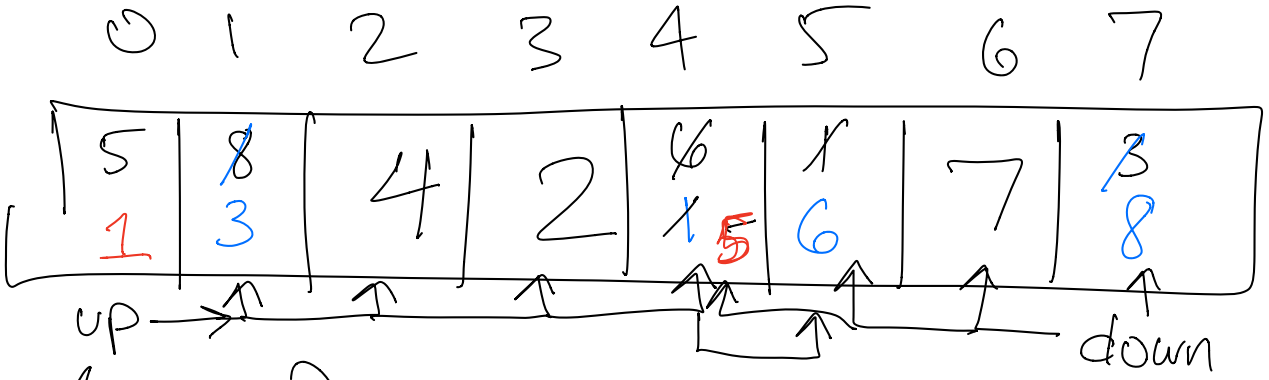
Finish Quicksort



qs(array)

qs(array, 0, 7)

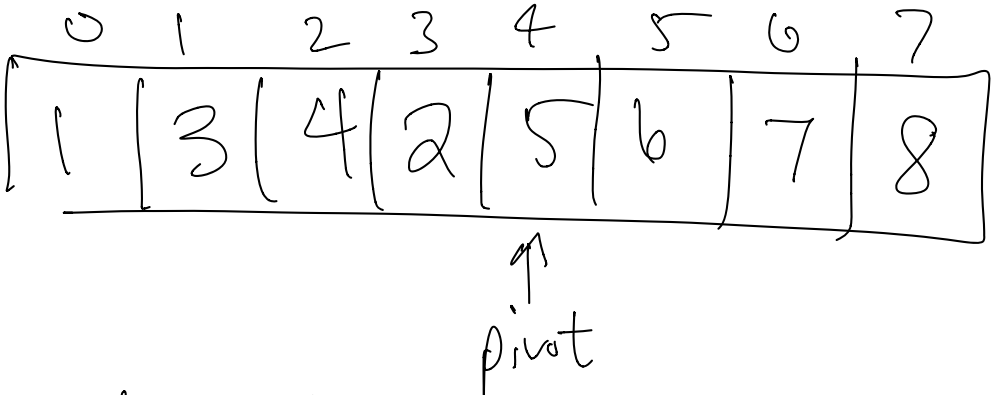
- partition(array, 0, 7)

  pivot = 5

  Swap pos 1 & 7

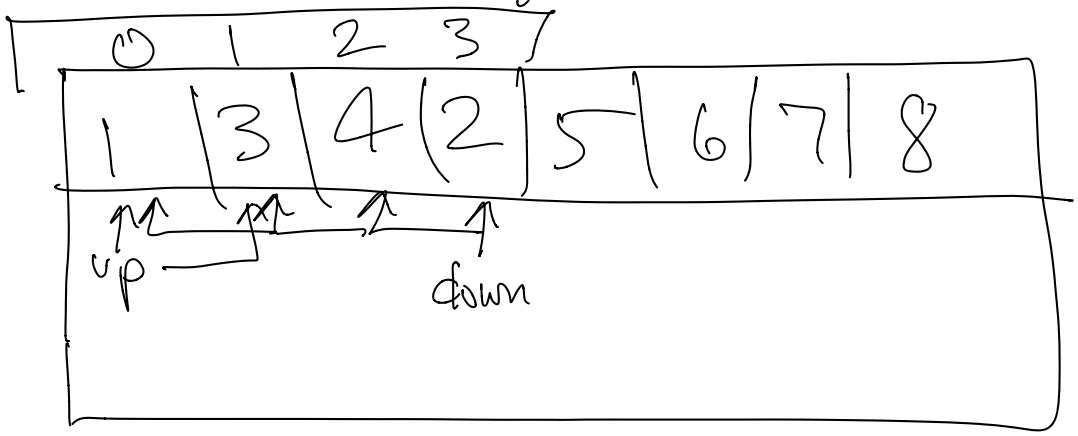
  Swap pos 4 & 5

  Swap pivot (pos 0) & pos 4



  return dum = 4

- qs(array, 0, 3)
  - partition(array, 0, 3)



pivot = 1

Swap pivot (pos 0)
with pos 0

return down (0)

  - qs(array, 0, -1)
  - qs(array, 1, 3)

Priority Queue

Cross between a stack & a queue.

Stack – LIFO (last in, first out)

Queue – FIFO (first in, first out)

Priority queue – order that items come out in controlled by a numerical priority.

~~A, priority = 10~~
~~B, priority = 15~~
~~C, pri = 12~~
D, pri = 5
~~E, pri = 20~~
~~F, pri = 7~~

Assume that higher #s = false priority

order that they are helped

B (15)
E (20)
C (12)
A (10)
F (7)
D (5)

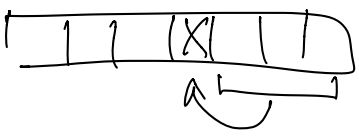
Priority queue (ADT)

insert (item, priority) → Adds a new item to our priority queue w/ the given priority.

deleteMax () → Retrieves the highest priority item in the PQ. & deletes it.

Ex

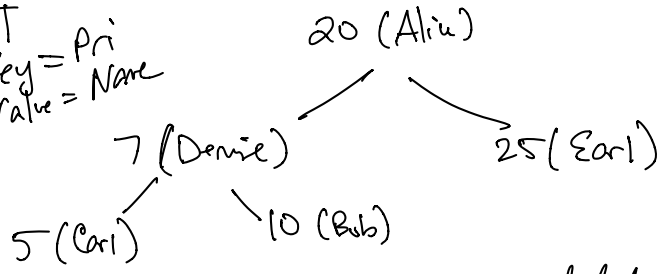
→ ArrayList (unsorted)
  - ↳ insert O(n)
  - ↳ deleteMax O(n)



→ Hashtable Key = Name, Value = Priority — insert → O(1), deleteMax → O(n)

Key = Priority, Value = Name — insert → O(1), deleteMax → O(n)

BST
Key = Pri
Value = Name



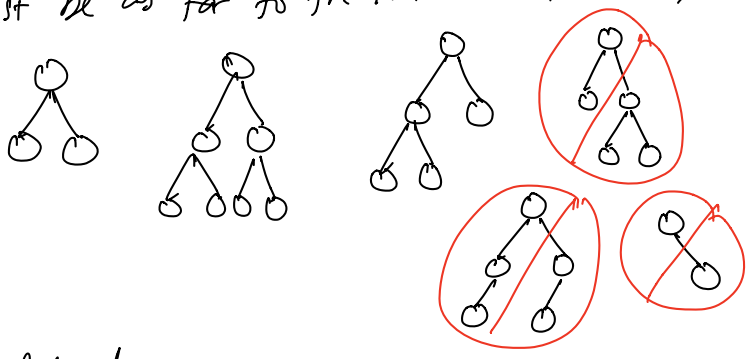
20 → Alice
7 → Denise
10 → Bob
5 → Carl
25 → Earl

insert
$O(\log n)$

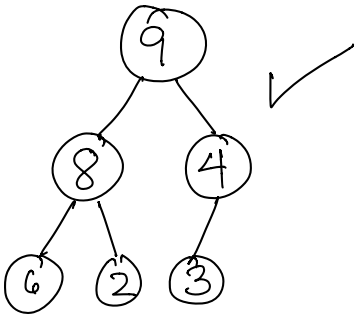
delete Max
$O(\log n)$

Heap

- A heap is a binary tree.
  w/ some rules:
- It is "complete" (every level of the tree is completely full, except for possibly the bottom level & if the bottom level is not full, all the items must be as far to the left as possible)



- The item stored at each node must be >= the items at both children. (Heap-order property)



✓

delete Max
Always @ the root. → $O(\log n)$

insert → $O(\log n)$