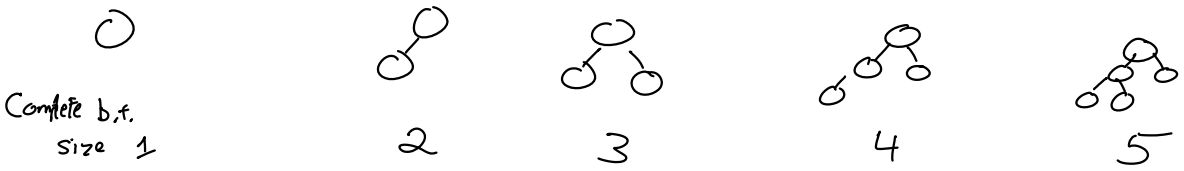
Heap

What is a max-heap?

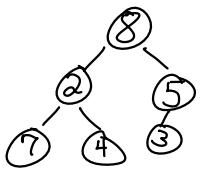
A complete binary tree s.t. all parents' values are greater than their childrens'



ADT: Priority Queue

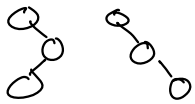
Heap can implement PQ

Not heap



8 2 5 7 4 3

not complete b.t. of size 3

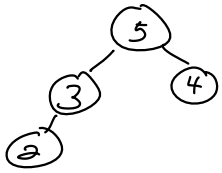


Heap ops

insert —— add to "end" of tree and then bubble up end

remove —— swap root w/ end, remove end, bubble down root

peek —— get root



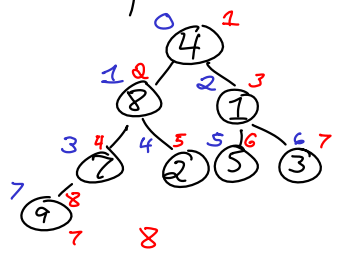
6

ArrayTree: store complete binary tree as ArrayList of its level-order traversal

i * 2 + 1 = left(i)

i * 2 + 2 = right(i)

⌊(i+1)/2⌋ - 1 = parent(i)



i * 2 = left(i)

i * 2 + 1 = right(i)

⌊i/2⌋ = parent(i)

1	2	3	4	5	6	7	8
0	1	2	3	4	5	6	7
4	8	1	7	2	5	3	9

(i+1) * 2

2i + 2 - 1

Method bubbleUp(index): Assume obj has an ArrayList reprsnt. tree

If index == 0 or this→tree→get(index) ≤ this→tree→get(parent(index)):

Return

swap this→tree→get(index) and this→tree→get(parent(index))

bubbleUp(parent(index))

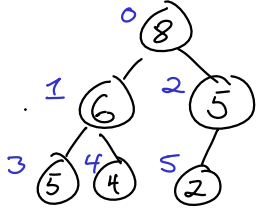
End Method

Method insert(prio, value):

this→tree→insertBack(pair(prio, value))

this→bubbleUp(this→tree→getSize()-1)

End Method



Method bubbleDown(index):

If left(index) ≥ this→tree→getSize():

Return

If right(index) ≥ this→tree→getSize():

If this→tree→get(index) < this→tree→get(left(index)):

swap index value & left(index) value

~~bubbleDown(left(value))~~

Else:

swap & bubble down whichever child is larger of all 3 nodes

End Method

Heap Sort : $O(n \log n)$

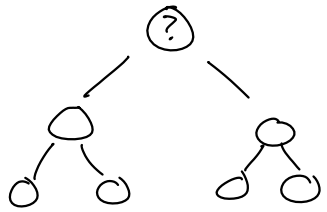
$O(n \log n)$ - add all elements to min-heap

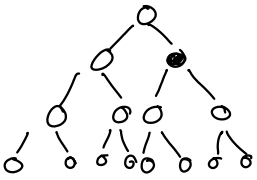
$O(n \log n)$ - remove all elements one at a time and put them in list ✓

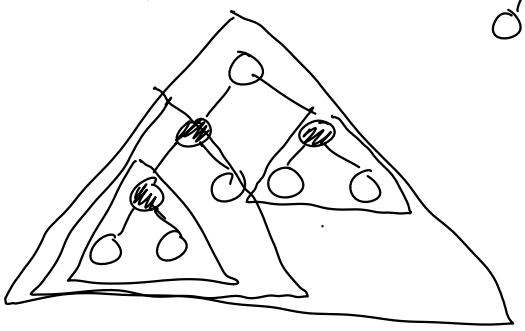
7 2 5 2 8 9 1 4

Heapify():

For each index from end to root:
bubbleDown(index)



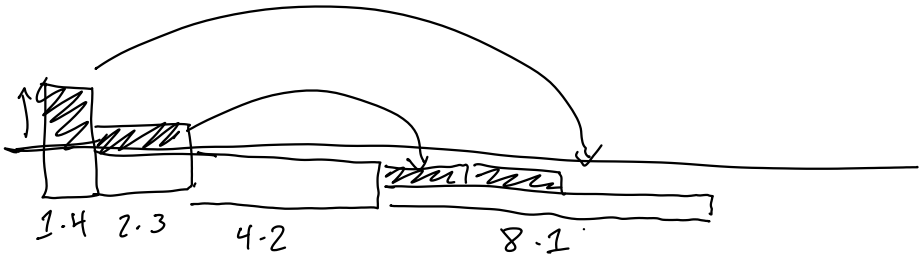




Most bubble downs are small.

$$8 \cdot 1 + 4 \cdot 2 + 2 \cdot 3 + 1 \cdot 4$$

$$\sum_{i=1}^{\log n} 2^{i-1} \cdot (\log n - i) \leq \sum_{i=1}^{\infty} \ldots \leq n$$



heapify $O(n)$

Top-k : $O(n)$