Discussion 10

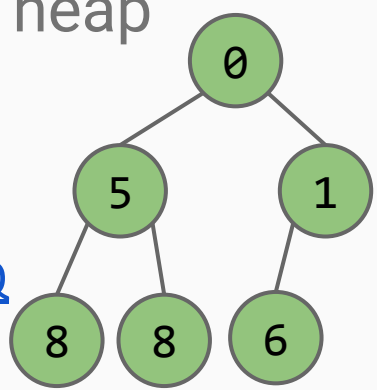
Heaps, Advanced Trees, Exam Review

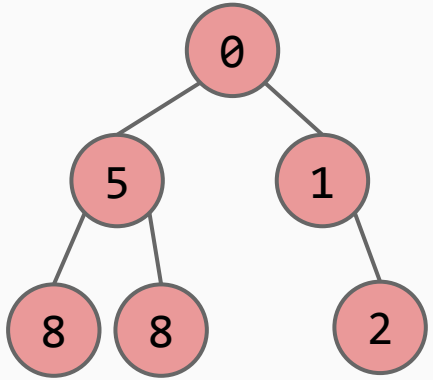
Priority Queues

- Abstract Data Type
  - add(Item x)
  - getSmallest()
  - removeSmallest()

Heaps

- Heaps are one implementation of a Priority Queue
- 2 important properties for min heap
  - Every node is ≤ its children
  - Heap is “complete”
- Heap Demo: https://goo.gl/wBKdFQ

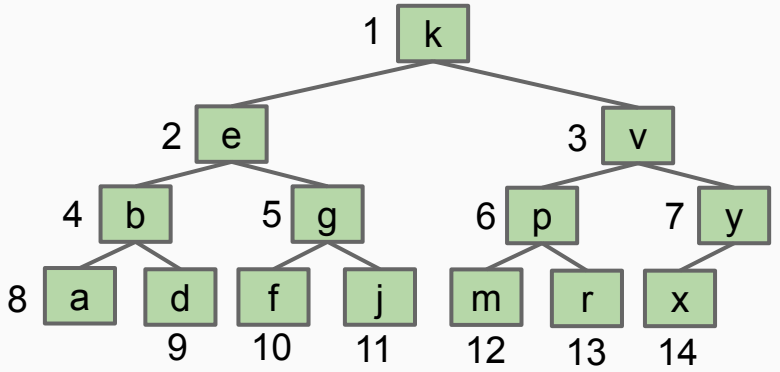




Incomplete

Heaps as Arrays

- leftChild(k) = k*2
- rightChild(k) = k*2 + 1
- parent(k) = k/2



Key[] keys

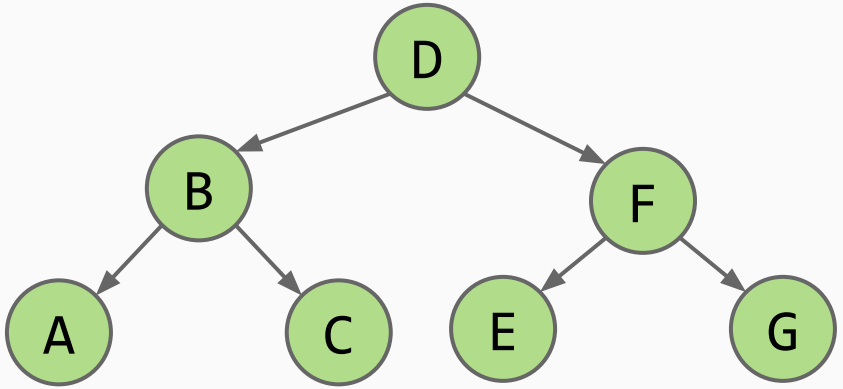
-	k	e	v	b	g	p	y	a	d	f	j	m	r	x
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

Heaps (Runtime)

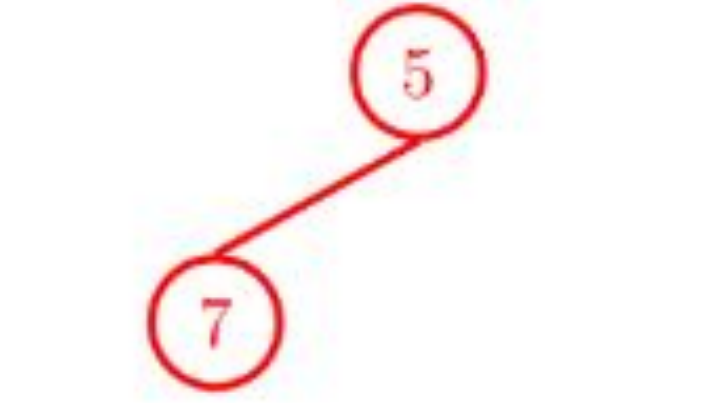
Operation	**Runtime**
add(Item x)	Θ(logN)
getSmallest()	Θ(1)
removeSmallest()	Θ(logN)

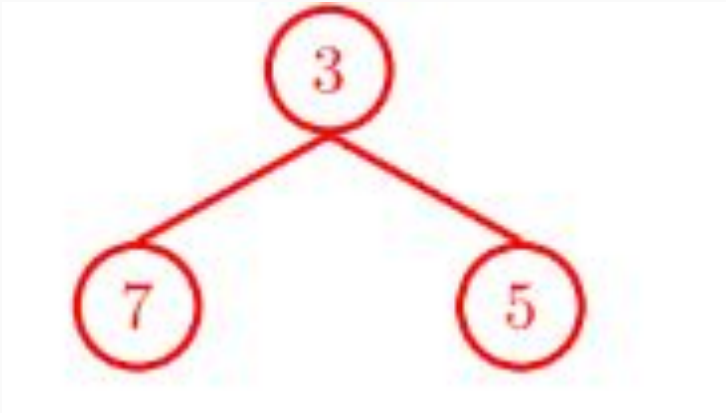
Tree Traversals

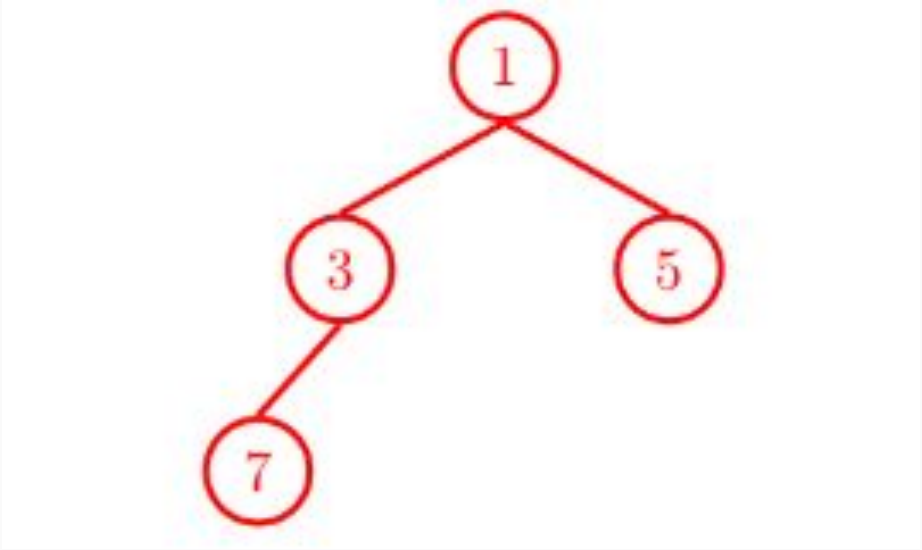
- Level Order (DBFACEG)
- Depth First
  - Preorder (DBACFEG)
  - Inorder (ABCDEFG)
  - Postorder (ACBEGFD)

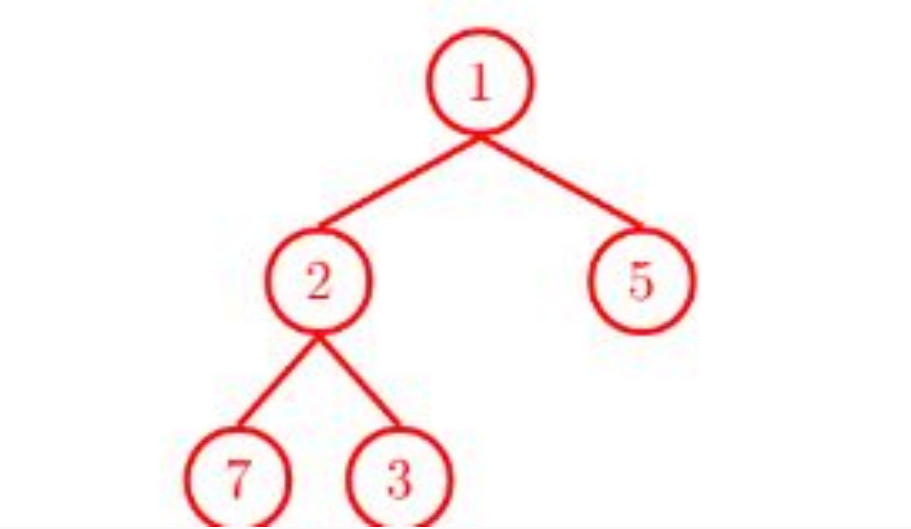


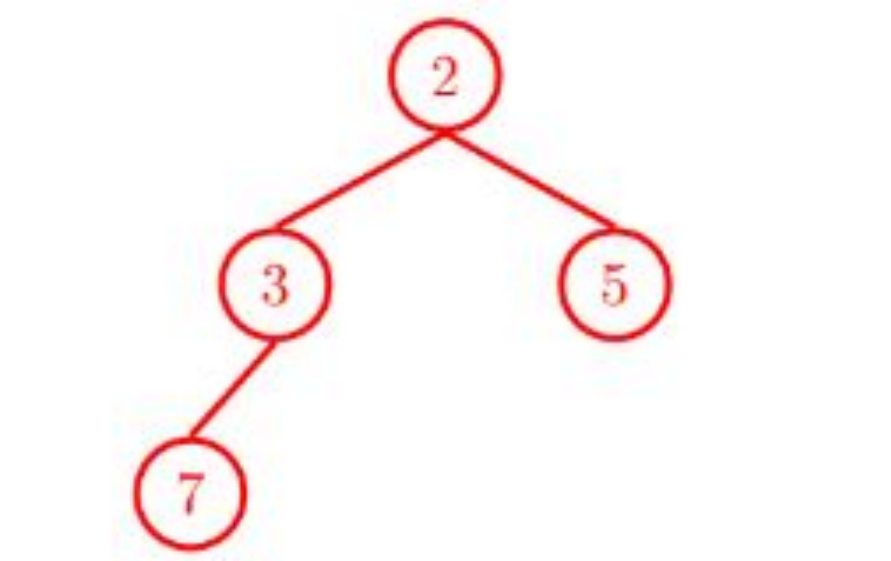
```
1  Heap h = new Heap();
2  h.insert(5);
3  h.insert(7);
4  h.insert(3);
5  h.insert(1);
6  h.insert(2);
7  h.removeMin();
8  h.removeMin();
```

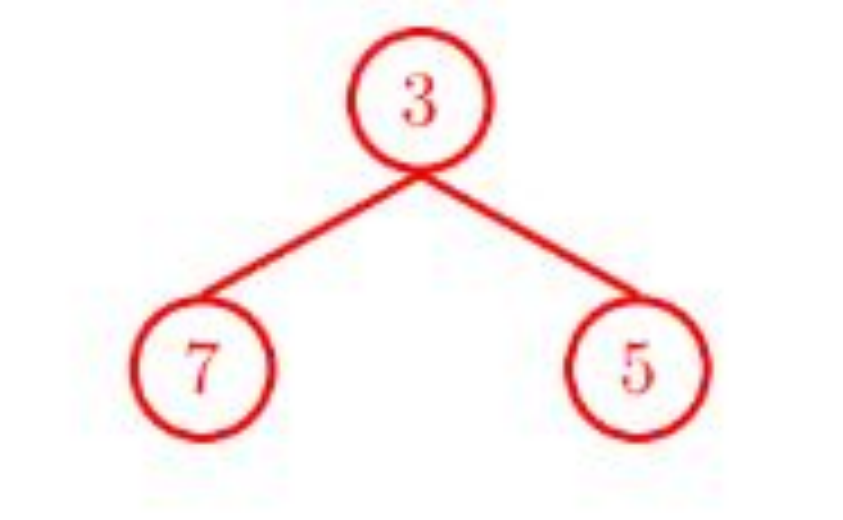

5
7


3
7
5


1
3
5
7


1
2
5
7
3


2
3
5
7


3
7
5

Your friend Sahil Finn-Garng challenges you to quickly implement an integer max-heap data structure. "Hah! I'll just use my min-heap implementation as a template to write MaxHeap.java," you think to yourself. Unfortunately, two Destroyer Penguins manage to delete your `MinHeap.java` file. You notice that you still have `MinHeap.class`. Can you still complete the challenge before time runs out?

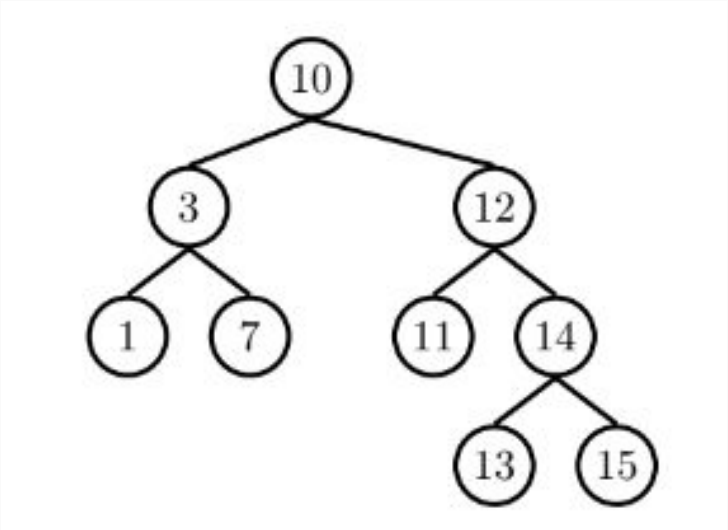
Hint: You can still use methods from `MinHeap`.

Pre-Order?

In-Order?

Post-Order?

Level-Order (BFS)?



Pre-order: 10 3 1 7 12 11 14 13 15

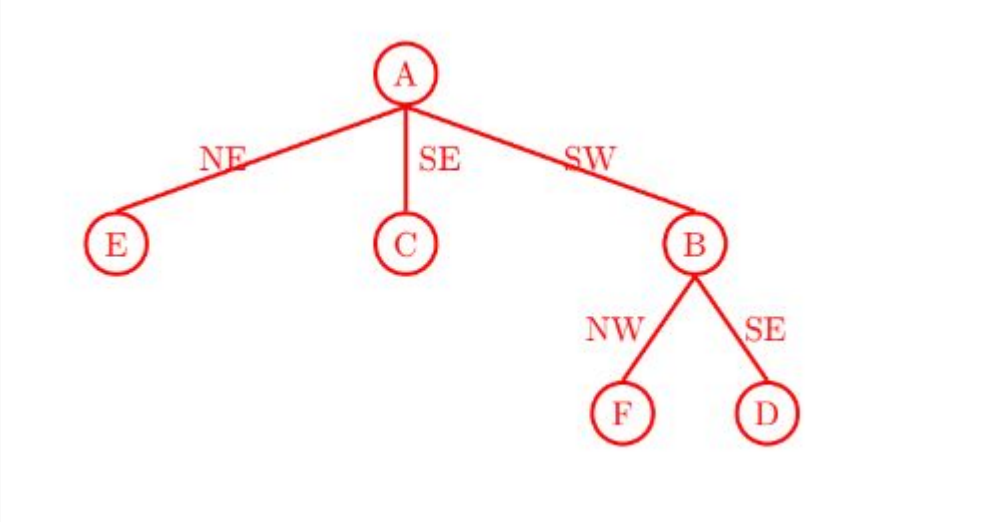
In-order: 1 3 7 10 11 12 13 14 15

Post-order: 1 7 3 11 13 15 14 12 10

Level-order (BFS): 10 3 12 1 7 11 14 13 15

Draw the quadtree built by inserting the following nodes with the given coordinates.

```
insert A (2, 3);
insert B (-1, 1);
insert C (3, 2);
insert D (0, 0);
insert E (4, 4);
insert F (-3, 2);
```


A
NE
SE
SW
E
C
B
NW
SE
F
D