Java priority queue: Like heap?

* Add/offer → Add an element to PQ, O(logn).
* Remove/poll → remove and return the minimum → O(logn)
* Peek() → Return the minimum element without removing → O(1)

Offer/poll comes from the queue class, add/remove comes from another class. Only in priorityqueue that they behave the same

Make minimum priority queue into maximum priority queue → Put collections.reverseOrder();

Patient implements Comparable<Patient> → Allow us to compare them

* Implement compareTo method

@Override

Public int compareTo(Patient o) {

If (this.age < 0.age) {

Return -1;

} else if (this.age > o.age) {

Return 1;

} else return 0

}

OR: just return this.age – o.age

Hashmap → Contains() is O(1)

If increment the value, shift up

If decrement the value, shift down

HashSet & TreeSet → implement set interface

Add: HashSet(O(1)) TreeSet(O(lg n))

Remove: HashSet(O(1)) TreeSet(O(lg n))

Contains: HashSet(O(1)) TreeSet(O(lg n))

TreeSet implements Navigableset and sorted set

* Has last, first, floor, lower, subset(startIdx, lastelementindex+1), headset, descendingset, pollFirst, pollLast → Use TreeSet if you need **ordering**, else use hashset
* Cannot print hashmap (Has no order/ random order)
* Successor = ts.tailset(11, false).first();
* Use subset to query in between → Be careful of the corner case (Not very efficient so use binary tree AVL)