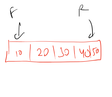
Lec-35

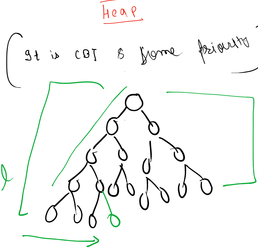
22/09/16 4:54 PM



queue

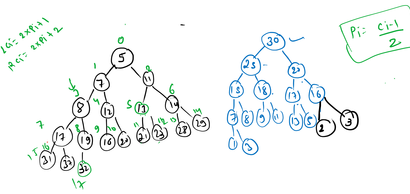
priority queue

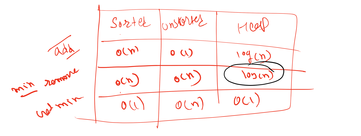
Heap

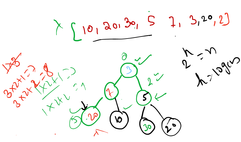


Rank → min heap

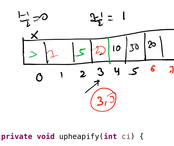
max → max heap







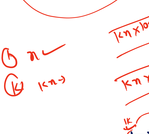
```
private void downheapify(int pi) {
    // TODO Auto-generated method stub
    int lci = 2 * pi + 1;
    int rci = 2 * pi + 2;
    int mini = pi;
    if (ll.get(lci) < ll.get(mini)) {
        mini = lci;
    }
    if (ll.get(rci) < ll.get(mini)) {
        mini = rci;
    }
    if (mini != pi) {
        swap(pi, mini);
        downheapify(mini);
    }

}
```

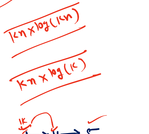


```
private void upheapify(int ci) {
    int pi = (ci - 1) / 2;
    if (ll.get(pi) > ll.get(ci)) {
        swap(pi, ci);
        upheapify(pi);
    }
}
```

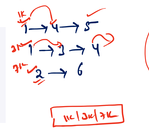
Input: nums = [3,2,3,1,2,4,5,5,6], k = 4

Output: 4

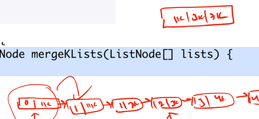




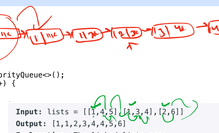
```
Input: lists = [[1,4,5],[1,3,4],[2,6]]
Output: [1,1,2,3,4,4,5,6]
Explanation: The linked-lists are:
[
  1->4->5,
  1->3->4,
  2->6
]
```



```
public ListNode mergeKLists(ListNode[] lists) {

}
```



```
public ListNode mergeKLists(ListNode[] lists) {
    PriorityQueue<ListNode> pq = new PriorityQueue<>();
    for (int i = 0; i < lists.length; i++) {
        if (lists[i] != null) {
            pq.add(lists[i]);
        }
    }
    ListNode Dummy = new ListNode();
    ListNode temp = Dummy;
    while (pq.isEmpty()) {
        ListNode rv = pq.poll();
        Dummy.next = rv;
        Dummy = Dummy.next;
        if (rv.next != null) {
            pq.add(rv.next);
        }
    }
}
```



```
6
7 9
0 10
4 5
8 9
4 10
5 7
```