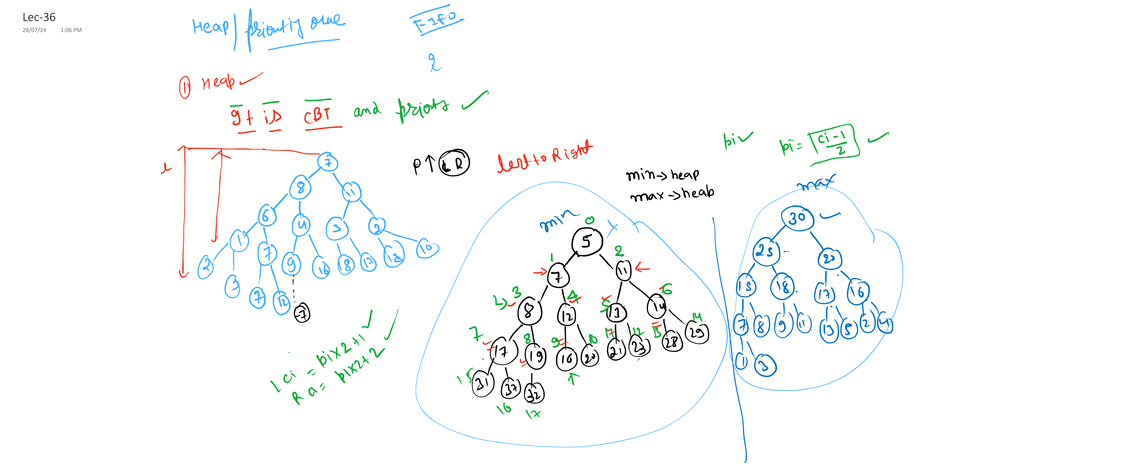
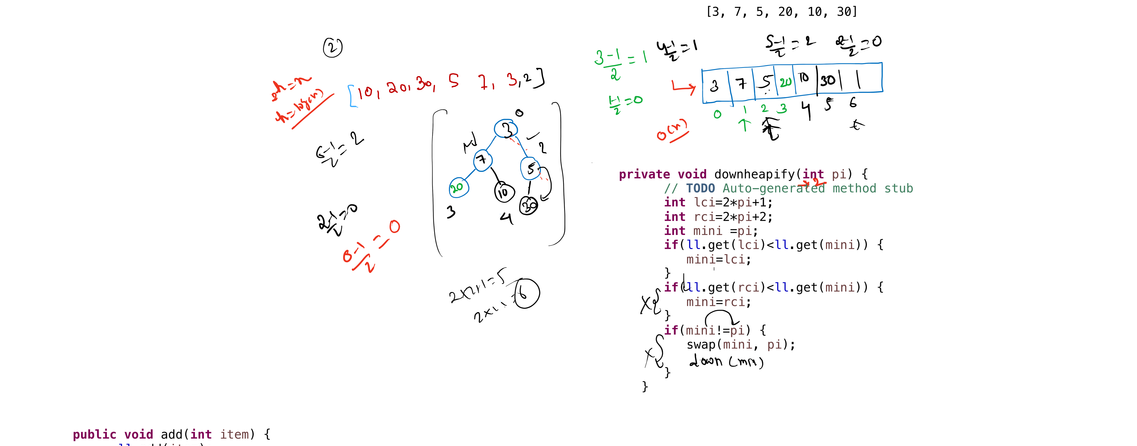
Lec-36

28/07/24 1:06 PM



	sorted ll	unsort ll	Heap
Get min	O(1)	n	O(1)
remove min	O(n)	n	log(n)
Add	O(n)	O(1)	log(n)

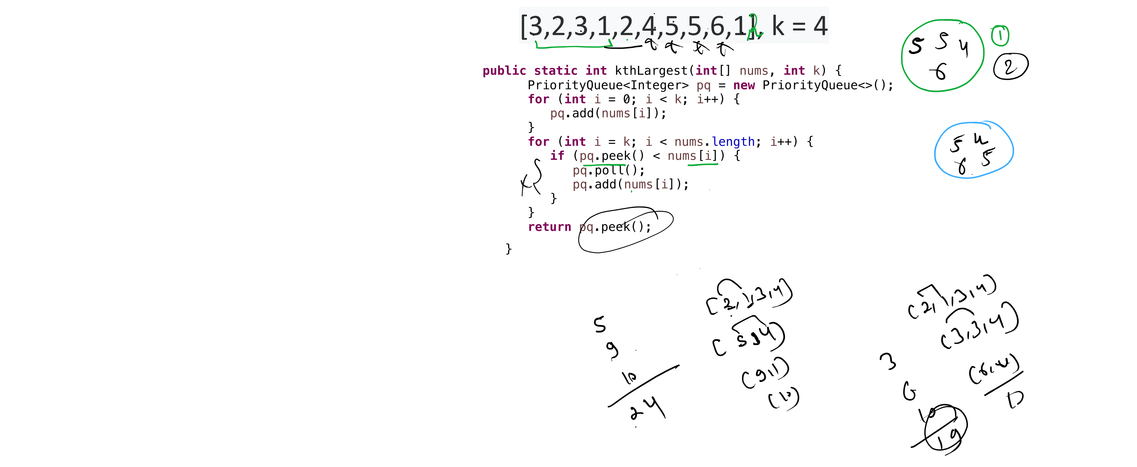



```
private void downheapify(int pi) {
	// TODO Auto-generated method stub
	int lci=2*pi+1;
	int rci=2*pi+2;
	int mini =pi;
	if(ll.get(lci)<ll.get(mini)) {
	   mini=lci;
	}
	if(ll.get(rci)<ll.get(mini)) {
	   mini=rci;
	}
	if(mini!=pi) {
	   swap(mini, pi);
	   doun (mn)
	}
}
```

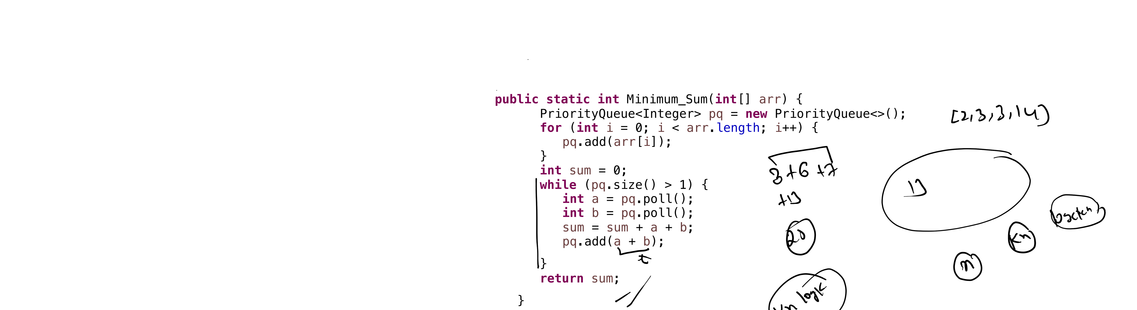
```
public void add(int item) {
	ll.add(item);
	upheapify(ll.size() - 1);
}
int pi = (ci - 1) / 2;
	if(ll.get(pi)>ll.get(ci)) {
public void upheapify(int ci) {
	swap(pi,ci);


	}


}
```




```
public static int kthLargest(int[] nums, int k) {
	PriorityQueue<Integer> pq = new PriorityQueue<>();
	for (int i = 0; i < k; i++) {
	   pq.add(nums[i]);
	}
	for (int i = k; i < nums.length; i++) {
	   if (pq.peek() < nums[i]) {
	      pq.poll();
	      pq.add(nums[i]);
	   }
	}
	return pq.peek();
}
```



```
public static int Minimum_Sum(int[] arr) {
	PriorityQueue<Integer> pq = new PriorityQueue<>();
	for (int i = 0; i < arr.length; i++) {
	   pq.add(arr[i]);
	}
	int sum = 0;
	while (pq.size() > 1) {
	   int a = pq.poll();
	   int b = pq.poll();
	   sum = sum + a + b;
	   pq.add(a + b);
	}
	return sum;
}
```


```
public ListNode mergeKLists(ListNode[] lists) {
	PriorityQueue<ListNode> pq = new PriorityQueue<>();
	for (ListNode head : lists) {
	   if (head != null) {
	      pq.add(head);
	   }
	}
	ListNode Dummy = new ListNode();
	ListNode temp = Dummy;
	while (!pq.isEmpty()) {
	   ListNode rv = pq.poll();
	   Dummy.next = temp;
	   Dummy = Dummy.next;
	   if (rv.next != null) {
	      pq.add(rv.next);
	   }
	}
	return temp.next;
}
```

[
1->4->5,
1->3->4,
2->6
]